WEB PROXY WORKLOAD CHARACTERISATION AND MODELLING

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

Anirban Mahanti

September 1999

© Copyright Anirban Mahanti, 1999. All rights reserved.

Permission To Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

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

Abstract

Understanding WWW traffic characteristics is key to improving the performance and scalability of the Web. In the first part of this thesis, Web proxy workloads from different levels of a caching hierarchy are used to understand how the workload characteristics change across different levels of a caching hierarchy. The main observations of this study are: HTML and image documents account for 95% of the documents seen in the workload; the distribution of transfer sizes of documents is heavy-tailed, with the tails becoming heavier as one moves from the client side to the server side of the network; the popularity profile of documents does not precisely follow the Zipf distribution; one-timers account for approximately 70% of the documents referenced; concentration of references is less at proxy caches than at servers, and concentration of references is higher at lower-level proxies than at higher-level proxies; there appears to be no correlation between document modification rate and document popularity, but the modification rate is higher at higher-level proxies; a gradual drift in the "hot set" of active documents is observed, indicating the presence of some documents of enduring popularity; temporal locality is observed in the referencing behaviour of popular documents.

The objective in the second part of the thesis is to determine whether or not Web document references at proxy caches can be modelled as independent and identically distributed random events. Trace-driven simulations using empirical and synthetic traces (with varying degree of temporal locality) show that temporal locality is an important factor in cache performance. The caching simulation results also show that temporal locality arising out of short-term correlation between references is important only for small caches. For large caches, a synthetic workload generated by applying the Independent Reference Model on a day-to-day basis gives performance very similar to that obtained for empirical traces. Finally, a mechanism for introducing correlations between recent past and future references is proposed that can generate synthetic workloads which achieve cache hit ratios similar to those of empirical traces, for all cache sizes considered.

Acknowledgements

I would like to take this opportunity to thank some people who helped me in one way or the other during the course of my M.Sc. thesis. First of all, my supervisors, Professor Derek Eager and Professor Carey Williamson, for their invaluable guidance, supervision, and encouragement during the course of my study.

I would also like to thank members of my committee: Dr. Raj Srinivasan (external), Dr. Rick Bunt, and Dr. Eric Neufeld. Their comments and suggestions have improved the thesis.

Greg Oster deserves many thanks for bearing with my time consuming system administration concerns, and also for helping me with the traces, simulations, LaTeX formatting, and numerous other issues (the list is endless). Also, my fellow graduate students' suggestion, advice, and company is greatly appreciated. I would especially like to thank Jayakumar Srinivasan for many useful discussions, and also for bearing with my resource consuming experiments.

I am also grateful to Maureen Desjardins, Jan Thompson, and Corinne Fasthuber, office staff members of the Department of Computer Science, for answering my numerous queries. Their prompt answers made life much easier.

Funding for my first year of study was in the form of a Research Assistantship (R.A.) provided by Prof. Derek Eager. This funding enabled me to pursue graduate studies, and will always be appreciated. I also thank TRLabs Saskatoon for funding the second year of my study.

I would also like to thank the Kushwaha family for many wonderful suppers, movie nights, and also for throwing a surprise party the evening of my defense.

Finally, I would like to make special mention of my parents. I wouldn't be doing higher studies if they didn't teach me the value of hard work and dedication. Although they are thousands of miles away from me, I always feel their presence and blessings, which makes me stronger and even more determined to achieve my goals.

Table Of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The World-Wide Web (WWW or the "Web") [11, 15, 25, 81] is an Internet-based globally distributed information system developed at CERN (Conseil Europeen pour la Recherche Nucleaire). The Web was originally intended to allow researchers collaborating with CERN to share information with each other rapidly. The first complete system including a simple browser, an information server, and other essential software for developers to build their own tools, was released in 1991 [25]. Since then, the Web has experienced phenomenal growth. This growth can be attributed to many factors, such as, simple and quick access to a wide variety of information, fast and effective dissemination of knowledge, and user friendly interfaces for accessing the Web.

Currently, Web traffic accounts for a large fraction of the Internet traffic (e.g., as high as 80% on some wide-area backbone links [83]). This growth of the Web has contributed significantly to the network traffic on the Internet [5, 7, 16], and motivated much research into improving the performance and scalability of the Web.

In recent years, Web proxy caches have been deployed to reduce network traffic and provide better response time for Web accesses. A Web proxy consists of application level software that accepts document retrieval requests from a set of clients, forwards these requests to appropriate servers if the requested documents are not already present in the proxy's cache, and sends documents back to the clients [34]. The proxies reduce average access latencies and save network bandwidth by caching frequently requested documents. A logical extension to proxy caching is hierarchical caching. In this scheme, a group of institutional or organisational level proxies ("lower-level" proxies) share a "higher-level" proxy cache. Requests that cannot be satisfied from a lower-level cache are forwarded to the higher-level cache, with the expectation that some of these requests will be stored there.

Understanding WWW traffic characteristics is key to the design of techniques that save network bandwidth, reduce latency, and improve response time of Web accesses. This thesis is concerned with understanding factors that affect the performance of Web proxy caches. The study builds upon previous work done by Arlitt and Williamson [5] on Web server workloads, Cunha *et al.* [29] on Web client workloads, and Abdulla *et al.* [1] on Web proxy workloads. The present work differs from that in [1] as workloads from proxies at different levels of a caching hierarchy are considered, permitting a more complete study of how the workload characteristics change as one moves from the client side to the server side of the network.

The workload study described in Chapter 3 reveals some interesting characteristics of Web proxy caches operating in a hierarchy. The main observations are: (a) documents referenced only once account for a significant fraction of the total references, and this fraction increases as one moves from the client side to the server side of the network; (b) the popularity of Web documents does *not* follow Zipf's law as closely as it does in Web servers; (c) the concentration of references is lower at Web proxies in comparison to that at servers, and at higher-level proxies in comparison to lower-level proxies; (d) the rate of change of documents is higher at higher-level proxies, and is much higher than that seen at Web servers; and (e) references to popular documents exhibit temporal locality.

An important goal of this study is to determine whether or not temporal locality significantly impacts cache performance in Web proxy workloads. For this purpose, trace-driven caching simulations are carried out with empirical and synthetic workloads. The synthetic workloads are generated such that temporal locality is filtered to varying degrees. The results of the experiments show that temporal locality matters, with the impact depending on the workload under investigation and the cache replacement policy used at the Web proxy. Experiments also show that temporal locality arising out of short-term correlations between references is important only for small cache sizes. For large caches, a synthetic workload that models document accesses across a particular day as independent random events with stationary probabilities gives cache performance very similar to the corresponding empirical workload.

The rest of the thesis is organised as follows. Chapter 2 introduces the Web and reviews related research efforts. Chapter 3 presents a workload characterisation study of Web proxy workloads across different levels of a caching hierarchy. Chapter 4 describes the experiments carried out to determine the importance of temporal locality in Web proxy workloads and evaluates various approaches to modelling Web proxy document reference streams. Chapter 5 contains a summary of the thesis and some suggested future directions for research.

Chapter 2

The World-Wide Web

In recent years the World-Wide Web has experienced phenomenal growth. The proliferation of the Web has contributed significantly to the volume of network traffic on the Internet, resulting in much research interest in the performance and scalability of the Web.

This chapter is organised as follows. Section 2.1 introduces the Web and briefly describes how Web clients, servers, and proxies function. Section 2.2 reviews previous work on Web performance. The chapter is summarised in Section 2.3.

2.1 The World-Wide Web

The guiding principle behind the Web is that information should be accessible in a seamless fashion. Any network-connected computer should be able to access information residing on any other computer. The Web can be viewed as a large distributed system based on the client-server architecture (Figure 2.1) [79]. Information resides on *Web Servers*, which are accessed by the users via *Web Browsers*. The browsers act as the clients of the Web servers.

The Web uses HyperText for structuring information [11]. HyperText (or *hypermedia*) is text with links (called *hyperlinks*) to other information (e.g., text, image, audio, video), similar to references in a scientific paper or cross-references in a dictionary. HyperText documents are typically written in the HyperText Markup Language (HTML). When a user clicks on a hyperlink, the Web browser asks the

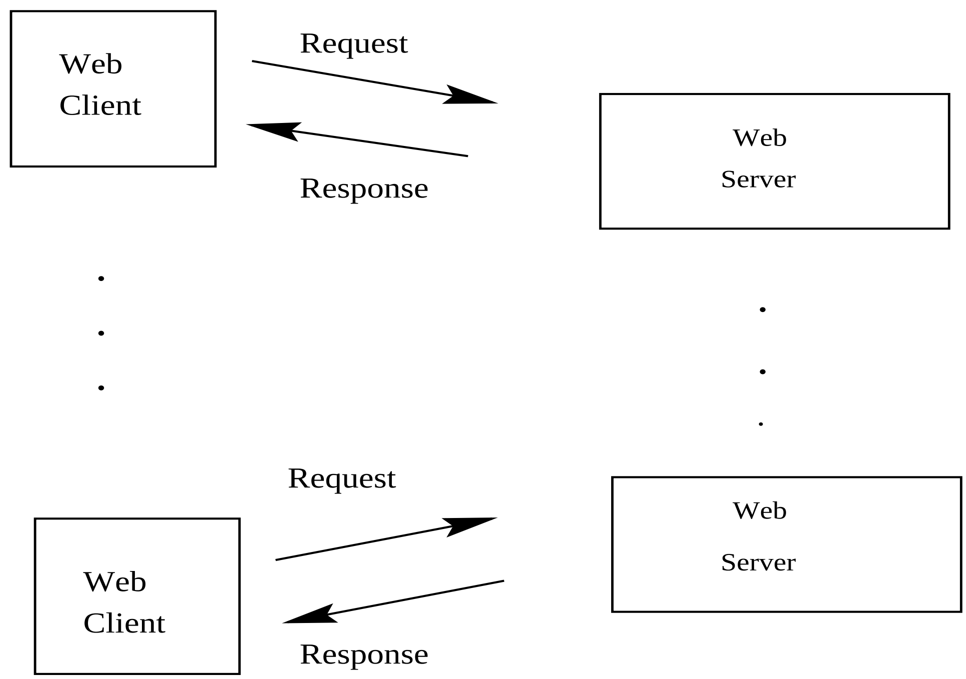


Figure 2.1: Client-Server Architecture of the Web

Web server to deliver the document referenced (Web page) to the user. Therefore, the Web provides an illusion of information residing on a single repository, where documents can be accessed at the click of a mouse.

Communication across a network requires a common "language", or *communication protocol*. The Web uses the HyperText Transfer Protocol (HTTP) [34] for communication between the clients and the servers on the Internet. HTTP is a simple, connection-oriented, application-level protocol that employs the Transmission Control Protocol/Internet Protocol (TCP/IP) [70] to manage connections and provide reliable bidirectional communication channels between the clients and the servers.

The HTTP protocol is a request/response protocol. Communication is initiated by a Web client in the form of a request to the Web server. After a communication channel has been established by TCP/IP, the client sends a request to the server. The server parses the request received from the client, retrieves the requested information, and returns it to the client. Every request results in a response from the server. After a response is received, the TCP/IP connection is usually closed.

2.1.1 The Web Client

A Web client is a program (known as a "browser") that sends requests for Web documents to the servers [34]. Examples of browsers include `Netscape Communicator` [60], `Internet Explorer` [52] , `Mosaic` [57] and `Lynx` [48]. When a user clicks on a hyperlink (or requests a particular Web page by an "open page" command from the browser), the browser creates a request to be sent to the server on which the document resides.

A typical request has the following format [34, 81]:

```
Request = Request-Line
          [Headers]
          CRLF
          [Message Body]
```

The `Request-Line` specifies the method (i.e., operation requested), the name of the Web document expressed as an Universal Resource Locator (URL) [12], and the version of HTTP in use. The methods supported by HTTP/1.0 (HTTP version 1.0) are described in Table 2.1. The `Header` field is optional, and can be used by the client to pass additional information about the request to the server (e.g., preferred document, referrer, user-agent). `CRLF` stands for Carriage Return and Line Feed.

A sample client request for the document `/people/grads.shtml` from the Web server `www.cs.usask.ca` is given below:

```
GET /people/grads.shtml HTTP/1.0
Accept:  */*
If-Modified-Since:  Wed, 23 Mar 1999 14:40:33 GMT
Referrer:  http://www.usask.ca
User-Agent:  Netscape/3.1
```

The `Request-Line` specifies that the client wants to obtain (since the method used is "GET") the document `/people/grads.shtml`, and is using HTTP version 1.0.

The `Accept` field indicates that the client is willing to accept any type of document. The `If-Modified-Since` header field tells the server to send the document only if it has been modified after 2:40 p.m. GMT on Wednesday, 23 March 1999. If the document has not been modified since the time specified in the `If-Modified-Since` header field, the server sends the client a response indicating that the client has an up-to-date copy of the document in its cache; otherwise, a full document transfer occurs. The `Referrer` header field allows the client to specify the URL of the resource from which the requested document's URL was obtained. This information can be used by the server to generate back-links to Web pages of interest, optimise caching, etc. [44]. The `User-Agent` field informs the server that the client is using the Netscape/3.1 Web browser. The system administrator at the server site may use this information for statistical purposes.

Table 2.1: Methods Supported by HTTP/1.0

Method	Function
GET	Obtain the requested document
HEAD	Obtain the header information of the requested document
POST	Send data to the server
PUT	Place document on the server
DELETE	Delete document from the server

After sending a request to the Web server, the client waits to receive a reply. Depending upon the reply received, the client may make another request to the Web server, or display the requested document.

2.1.2 Web Servers

A Web server is a program that provides clients with requested documents[1]. Example of servers are `Apache` [3], `CERN's ''httpd'' Server` [26], `Netscape's Enterprise Server` [61], and `Microsoft's Internet Information Server` [53].

[1]The term 'Web Server' is also used to refer to the machine which runs the Web server software. In this document, these two interpretations are used interchangeably depending on the context.

The server receives requests from clients on a specific port, the default port being 80 [34], to establish a TCP/IP connection with the client. A typical server response has the following format [34, 81]:

```
Response = Status-Line
           [Headers]
           CRLF
           [Message Body]
```

The `Status-Line` specifies the protocol version, a numeric status-code, and its associated textual interpretations. The status-code is a 3-digit integer describing the result of the server's attempt to fulfill a client's request. The first digit of the status-code defines the response class. The five response classes are [34]:

- 1xx: This is a provisional response indicating that the request has been received and is currently being processed.
- 2xx: This means that the request is successfully received, understood and processed by the server.
- 3xx: This implies that further action needs to be taken to complete the request.
- 4xx: This indicates that an invalid request was made by the client.
- 5xx: This indicates that although the server thought that the request made by the client was valid, it was unable to process it. That is, an error occurred at the server while processing the request.

Table 2.2 describes the commonly used status-codes [34, 81]. The headers are optional and can be used to provide additional information regarding the server and the document being returned.

Table 2.2: Commonly Returned Status-Codes and their Interpretations

Code	Interpretation
200	OK
204	No Content
206	Partial Content
302	Moved Temporarily
304	Not-Modified
400	Bad Request
403	Forbidden
404	Not Found
500	Internal Server Error

A possible response of the server to the sample request given earlier on page 6 is:

```
HTTP/1.0 200 OK
Date:  Sat, 27 March 1999 16:37:47 GMT
Server:  NCSA/1.4
Content-Length:  1192
Content-Type:  text/html
Last Modified:  Fri, 26 March 1999 10:30:21 GMT
```

The `Status-Line` of the server's response specifies that the server is HTTP/1.0 compliant, and that the request was successfully processed (status-code is 200). The date and time of the response, as well as the type of the server (NCSA server, version 1.4) is specified. `Content-Length` specifies the size of the document being transferred. `Content-Type`, specified using the Multipurpose Internet Mail Extension (MIME) [59] format, identifies the document to be of type HTML. The `Last Modified` field informs the client that the requested document was last updated on Friday, 26 March 1999 at 10:30 p.m. GMT[2]. This information is used by the client to issue requests using the "GET" method with an "If-Modified-Since" header field (typically referred to as an "If-Modified-Since", or "GET If-Modified-Since" request)

[2]Note that the date specified in the `Last Modified` header field is later than that specified in the `If-Modified-Since` header field of the client request. This implies that the client's copy of the document is stale.

to verify the validity of documents in the client's cache.

Usually, requests to the server are for documents stored on the server. However, it is possible for the Web server to dynamically create documents at the client or the server side in response to client requests. The Common Gateway Interface (CGI) is one such technology. In CGI, a Web server associates a program with some URLs. When a client requests such URLs, the server executes the associated program and sends the output of the program to the client. Sun Microsystems developed the Java technology, which allows clients to retrieve copies of programs residing on the Web server. Once the client has a copy of the program, the program runs on the client's local computer. Documents generated in the above described manner are known as *dynamic* documents.

2.1.3 Web Proxy

A Web proxy is an application program that accepts document retrieval requests from a set of clients, forwards these requests to the appropriate servers (if required), and sends the requested documents back to the clients [34] [3]. While receiving and serving requests from the clients, the proxy functions as a server. On the other hand, while forwarding requests to the appropriate servers, the proxy functions as a client. Commonly used proxy servers include the `` `httpd' `` Web server from CERN [26], which can also be used as a proxy, and `Squid` [78], which is a public domain successor of the `HARVEST` proxy cache [27].

Proxies were originally designed to allow the network administrators to be able to control access to the Internet from within an Intranet [7]. It was recognised, however, that proxies may also serve as repositories for frequently requested documents. This role of proxies has made them very popular. Caching documents at the proxy can save network bandwidth and reduce network latency for document accesses [36, 47].

Most browsers can be configured to use a proxy. Proxies can be deployed almost anywhere in the Internet. A second-level cache, the first-level being the browser

[3] The term 'Web Proxy' is also used to refer to the machine which runs the Web proxy software. In this document, these two interpretations are used interchangeably depending on the context.

cache, can be provided by a proxy cache that serves requests from a large community such as a large corporation, a university, or the customers of an Internet Service Provider. Higher-level proxies have also been deployed. These proxies typically have other second-level caches (i.e., lower-level proxies) as their clients.

Web caches connected in the above described tree-like fashion are said to form a cache hierarchy. Since the Internet topology is hierarchical, Web cache hierarchies are a natural choice for distributing load away from server "hot spots", saving wide-area network bandwidth, and for reducing access latencies. Lower-level caches typically configure a higher-level cache as their parent. In this parent-child relationship, requests that are not satisfied by the child cache (i.e., a miss occurred at the child cache) are forwarded to its parent cache. If the parent does not have the requested document in its cache, it forwards the request on behalf of the child cache. Caches at the same level can have a sibling relationship with each other. A sibling cache can ask its peer for a document. If the document is available at the peer, the peer delivers it to the requesting cache. However, the peer cannot forward requests on behalf of its siblings. Both `HARVEST` and `Squid` use the Internet Cache Protocol (ICP) [82] for quick and efficient inter-cache communication.

2.2 Related Work

The growth of the Web has motivated much research into understanding the Web traffic characteristics, and improving the performance and scalability of the Web. Careful analysis and understanding of the workload characteristics can lead to the design of techniques that save network bandwidth and reduce latency for Web accesses.

The following sections present an overview of previous work on Web performance. Section 2.2.1 discusses some recent workload characterisation studies. Section 2.2.2 describes research on Web caching. This is followed by a review of work on cache consistency in Section 2.2.3. In Section 2.2.4, recent efforts at improving Web cache hierarchies are outlined. Finally, Section 2.2.5 discuses some other efforts to reduce user perceived latency.

2.2.1 Web Client, Server and Proxy Workload Characterisation

One of the earliest attempts towards understanding Web client access patterns was made by researchers at Boston University [29]. Client traces were gathered by instrumenting the `Mosaic` Web browser. The collected traces provided information regarding all requests for documents, as well as relationships of documents with each other (i.e., requests directly made by users could be distinguished from those that were for embedded images in a particular Web page), time-stamps of all events, and durations of file transfers over the network [10, 29]. Their detailed statistical analysis showed that several characteristics of Web client usage can be modelled using power-law distributions (i.e., the shape of the distribution is a hyperbola). In particular, they showed that the distribution of document sizes, the popularity of documents as a function of size, and the frequency of reference to a document as a function of its rank, follow power-law distributions. In a more recent paper, Barford *et al.* [10] compared Web client traces from their previous study [29] with a more recent one, in an attempt to understand how workload characteristics have changed over time. The statistical analysis of the new data set shows that most of the Web characteristics outlined in their previous study can still be modelled using power-law distributions.

Many researchers have studied Web server workload characteristics [2, 4, 5, 16, 54]. In an extensive study, Arlitt and Williamson [5] identified ten common workload characteristics using six different Web server traces that spanned durations from one week to one year. The characteristics identified are: (a) approximately 90% of all requests result in successful transfer of documents; (b) HTML and image files account for more than 90% of the requests, similar to observations made by Braun *et al.* [16] for the NCSA server, and Cunha *et al.* [29] for their client traces; (c) distinct requests account for less than 3% of the total requests seen at the server;

(d) the mean transfer size is less than 21 KB (kilobytes); (e) approximately one-third of the distinct documents are accessed only once; (f) the file size distribution is heavy-tailed, implying that a small fraction of the documents transferred account for a large fraction of the bytes transferred; (g) document inter-reference times are independent and exponentially distributed; (h) 10% of the files accessed account for 90% of the requests seen and bytes transferred by the server; (i) remote sites account for at least 70% of the references to the servers; and (j) Web servers are accessed by thousands of domains, with 10% of the domains accounting for more than 75% of the usage.

Abdulla *et al.* [1] identified nine common workload characteristics for Web proxies, similar to the Web server study in [4, 5]. The characteristics identified are: (a) the mean file size is approximately 2 KB; (b) the median file size is less than 27 KB; (c) HTML, image and dynamic files account for 90-98% of the files accessed; (d) image files account for the most bytes transferred; (e) less than 11% of the accesses are to unique servers; (f) less than 5% of the servers are referenced only once; (g) 25% of the servers seen are responsible for 80-90% of the references; (h) 25% of the servers are responsible for 90% of the bytes accessed; and (i) a majority of the references seen at the proxy result in the successful transfer of documents to the clients.

2.2.2 Web Caching

Caching is a popular and effective technique for improving Web performance. Caching is effective because a small fraction of the total documents accessed on the Web often account for a large fraction of the activity. By identifying popular documents and moving them into Web caches closer to the requesting clients, repeated transfers of the same documents across the network can be avoided.

Web caching techniques have been influenced considerably by traditional caching studies (e.g., memory hierarchy caching, distributed file systems). Although the concepts are similar, there are three primary differences between Web caching and traditional caching. First, traditional caching is often concerned with fixed size transfer units [29, 87], whereas Web caching involves variable sized documents, due to the restrictions imposed by the HTTP protocol. Second, documents of the same size often have different download times (i.e., different associated costs). Third, the Web contains many more files and users than large-scale distributed file systems like the Andrew File System (AFS) [75].

There are many logical choices for locating caches in the Internet. These include: (a) at the client, or the client's network [13]; (b) at the server, or the server's network [5]; and (c) at various points in the network, such as the entry points to regional and backbone networks [30].

Bestavros *et al.* [13] consider caching at a location near the client. The study compared the performance of document caching at the session level (a single user), at the host level (several users using a single workstation), and at the network level (multiple users on a single LAN). The basic finding was that surprisingly high document hit ratios[4] are achievable, although byte hit ratios[5] are much lower. Session-level caching does reduce latency of access, but the volume of external traffic is not significantly reduced. Using trace-driven simulations, they show that shared caches significantly reduce host loads as compared to local-client caches, achieving similar performance using comparatively smaller cache space. Similar results have been reported in [7].

Main memory caching of Web documents at the servers can improve performance by reducing disk accesses [51]. Using trace-driven simulations, Markatos [51] concludes that a 512 KB main memory cache is enough to cache the most frequently accessed documents, achieving document hit ratios between 55-70%.

Caching at various network points can significantly reduce network traffic. Danzig *et al.* [30] consider caching at various switching points in the network within the NSFNET. Their study shows that caching of File Transfer Protocol (FTP) files can

[4]Document hit ratio is the fraction of requests seen at the cache that are satisfied by finding the requested document in the cache.

[5]Byte hit ratio is the fraction of the total volume of data transferred that is served from the cache because the requested documents were found in the cache.

reduce file transfers by about 50% over wide-area network links. At the application level, proxies can be deployed at various points in the network to reduce network traffic. Kroenger *et al.* [42] show that the maximum achievable proxy cache hit ratio can be as high as 50%, and the latency reduction is approximately half the observed hit ratio. This is because a majority of the cache hits are for documents smaller in size compared to the average size of documents. Similar observations have been made by Williams *et al.* [85]. This implies that the majority of cached documents are small in size [13, 42, 85].

A variety of cache replacement policies have been proposed for Web proxies. The Least Recently Used (LRU) replacement policy, although widely used in proxies (e.g., Squid), has been shown to be inferior to other policies in the WWW context [5, 20, 65, 85]. The main weakness of the LRU policy is its inability to quickly flush out documents that are referenced only once in the access logs. This limitation has significant impact on its performance because a high percentage of the documents seen are accessed only once [5, 72]. The Least Frequently Used (LFU) replacement policy discriminates against one-timers, keeping the most frequently referenced documents in the cache.

In recent years, researchers have turned their attention towards replacement policies that take Web dynamics (e.g., different download times for documents of the same size) into account [20, 65, 72, 85].

Cao *et al.* [20] have proposed the GD-Size replacement algorithm. GD-Size associates a value H with each document. When a document is brought into the cache, H is set to the value of the ratio of the cost of bringing a document into the cache and the size of the document. When a replacement has to be made, the document with the smallest H value (H_{min}) is removed and every document's H value is reduced by H_{min}. If a document is accessed again, then its H value is restored to its cost. Cao *et al.* show that setting the cost of document retrieval to one (i.e., all documents have the same cost of retrieval) yields an algorithm with superior document hit ratios compared to that of more traditional algorithms such as LFU and LRU. The main drawbacks of using such a simple cost function are that it does not consider delays associated with downloading documents, and it does not explicitly consider the frequency of accesses made to documents.

Niclausse *et al.* [65] propose the MIX algorithm, where documents that are costly to retrieve and/or are often requested, are cached, and those that have not been accessed for a long time and/or are large, are the first to be removed. Simulations show that MIX yields the best byte hit ratio and performs better than GD-Size in terms of document hit ratio when the cache sizes are small. The main disadvantage of the MIX algorithm is the difficulty in *a priori* determining the parameters of the cost function. Selection of proper parameters is crucial to the success of the algorithm. Another algorithm that considers cost and size of a document to assign a relative value to each document is the LRV algorithm [72]. The document with the lowest relative value is removed from the cache. The relative value of a document is defined to be a function of the probability of re-referencing a document after 'i' references ($P(i)$), the probability of requesting it again as a function of the time since last access ($D(t)$), the cost of retrieval, and the size of the document. Simulations show that this algorithm performs very well, but just like the MIX algorithm, *a priori* determination of $P(i)$ and $D(t)$ leaves its performance uncertain across different reference streams.

A performance study of proxy cache replacement policies done by Arlitt *et al.* [6] shows that size-based policies achieve higher document hit ratios, while frequency-based policies achieve higher byte hit ratios. This can be explained as follows: size-based policies discriminate against large objects, keeping many smaller documents in the cache, thereby increasing the hit ratio. On the other hand, frequency-based policies discriminate against one-timers, while keeping frequently referenced large documents. Thus, they can achieve higher byte hit ratios and reduce network traffic.

Gwertzman *et al.* [38] propose *geographic push caching*, where the server decides what, when, and where to cache. In this scheme, whenever the popularity of a document at a server exceeds a specified *push threshold*, the document is replicated to other servers located in the geographic region responsible for the most requests. These other servers can remove or refuse to store document replicas. The clients contact the original server on the first request to a document. The server then sends the client a "redirect message". This message contains information regarding the location of the server at which a copy of the document can be found. This has direct bearing on client caching since storing "redirect messages" consumes much less space than storing the corresponding documents. Push caching works best when the network topology is known. Thus, it can be implemented easily on an Intranet. However, for the Internet as a whole, obtaining an accurate description of the network topology is a complicated research problem.

An extreme case of push caching is *mirroring*, in which the storage of document replicas is guaranteed at some fixed number of other "mirror" sites. Mirroring is motivated by the relatively high volume of client-driven validation messages seen in proxy logs [7]. This high volume of cache-validation requests can be reduced if the server maintains copies of documents at the mirror sites that are guaranteed to be up-to-date. Mirroring has certain advantages. It is one step towards providing more fault tolerant service since documents can be accessed from the mirror sites if the main server is down. Further, if certain documents are likely to become popular in the near future (e.g., new software releases), or are frequently updated, these can be sent to mirror sites in advance. Mirroring is well suited to the replication of popular FTP sites, as they usually contain relatively static documents that are frequently accessed by many users. In this manner initial misses of popular data in caches can be anticipated and document transfers can be done outside the critical path of document retrieval initiated by the single user.

2.2.3 Web Cache Consistency

For caching to be effective, cache consistency must be maintained (i.e., cached copies should be updated when changes are made to the original document). Existing Web caches mostly provide *weak* cache consistency, which infrequently allows stale documents to be returned to the clients. Two widely used weak consistency mechanisms are: *Time-To-Live (TTL)* fields [34, 44], used by the `CERN ''httpd'' Web server`

[26]; and *client polling*, used by the `HARVEST` proxy cache [27].

TTL is an *a priori* estimate of a document's life-time, during which the cached document is always considered to be valid. This type of consistency is implemented in the HTTP protocol using the *expires* header field [34, 44], which informs the cache how long the document should be considered to be up-to-date. If a client request results in a hit to a document that is not expired, then the document is served from the cache without contacting the server to check the validity of the cached document. However, if a client request results in a hit to an expired document, an If-Modified-Since request is issued to the server to verify whether or not the document has been modified. If the document has been modified, the server returns the new document with status-code 200; otherwise, the server returns status-code 304, which informs the cache that the document has not been modified.

The client polling approach requires the clients to periodically check the validity of the cached documents with the server. This approach was first proposed in [24] for the Alex file system. This method takes advantage of the observation that documents not modified for a long time, are less likely to be modified compared to documents which have been recently modified [8]. Thus, the cache manager assigns a TTL field to a document expressed as a percentage (known as the update threshold) of the document's current age. For example, consider a cached document that is ten days old. If the update threshold is set to 50%, then the cached document should be considered to be valid for 5 days, and possibly invalid after this. Since this approach allows the TTL to be set according to a document's age, it is also referred to as *adaptive TTL* [22].

The weak cache consistency model does not guarantee that the document returned to the client is always valid. In the TTL approach, selecting appropriate TTL values is challenging. In the client polling approach, selecting a proper update threshold is crucial because a low update threshold value can invalidate valid documents, causing more If-Modified-Since requests to be generated. On the other hand, a high update threshold might cause stale documents to be returned to the clients.

Strong cache consistency can be provided by *invalidation protocols* and should be used where weak consistency is not sufficient. In this approach, the server notifies caches when a document is modified. To enable invalidation, the server needs to keep track of the clients that cache its documents. Upon receiving an invalidation notice, a client should delete the stale document. This method eliminates the possibility of clients receiving stale copies of documents. However, the current HTTP protocol has no facility for implementing invalidation messages.

An excellent comparison of cache consistency approaches using trace-driven simulations was conducted by Gwertzman and Seltzer [37]. They conclude that although invalidation protocols are required for providing perfect cache consistency, weak cache consistency approaches, like client polling, can reduce bandwidth consumption considerably. However, this study used network bandwidth as the main metric and did not consider other important issues, such as server loads, and client response times. In a more recent and exhaustive study, Cao and Liu [22] argue that strong cache consistency can be maintained with little or no extra cost.

2.2.4 Web Cache Hierarchies

Although caching is increasingly being used in the Internet [58, 76] to reduce network traffic and distribute load away from server "hot spots", it has been less successful in reducing access latencies [80]. Cache hits at the higher levels require the document to be sent down the hierarchy, with each cache along the path storing a copy of the document being fetched. In a multi-level hierarchy, this round trip delay (one delay for locating the document and another for percolating the document through the cache hierarchy) can be significant [49]. Also, this "store and forward" policy results in large scale duplication of documents between child caches and their parents, effectively reducing the number of distinct documents that can be stored in the hierarchy. Finally, it has been observed that cache sizes at higher-levels will have to increase at an exponential rate with the increase in Web traffic in order to maintain maximal hit ratios at the higher-level caches [71].

Povey and Harrison [71] describe a caching technique, which they call *distributed* *caching*. This technique caches documents only at the leaf-level caches and uses the higher-level caches to maintain location hints regarding where documents are cached. On a cache miss, the leaf-level cache queries its parent for the document. If the parent does not know where the document is cached, it queries its parent for the document. This process of recursively querying continues until a cached copy of the document is located. Otherwise, the document is retrieved from its Web server and the higher-level caches are informed of the location of this new cached document.

Tewari *et al.* [80] also describe a distributed caching strategy similar to the one proposed in [71]. However, in addition to caching at the leaf nodes and using the higher-level nodes to provide location hints, they also incorporate "push" caching to move documents towards clients in anticipation of the document being referenced in the near future. Using simulations, they show that the proposed scheme provides speedups of 1.27 to 2.43 compared to traditional cache hierarchies.

2.2.5 Other Approaches to Web Latency Reduction

The delay perceived by the user depends on factors such as network bandwidth, network latency, and server and client processing speeds. Although there are substantial benefits of caching, caching does not help with *compulsory misses* (i.e., misses that occur because the document requested is being accessed for the first time), *coherence misses* (i.e., misses because of document modifications), and misses that occur for documents that are not cacheable. In this section, some other latency reduction techniques are reviewed.

A technique helpful in reducing retrieval latency is "pre-fetching", in which a client or proxy pre-fetches Web pages that a user is likely to reference in the near future, and keeps them in its cache. Kroenger *et al.* [42] categorise pre-fetching into two types: local-hint and server-hint pre-fetching. In local-hint pre-fetching, the agent doing the pre-fetching (proxy or client) uses local information (from a history of prior reference patterns it maintains) to make predictions regarding documents that might be accessed in the future. In the server-hint approach, the server uses its global knowledge of client access patterns to determine which objects to pre-fetch. Using trace-driven simulations, Kroenger *et al.* show that, at best, 26% latency reduction is possible using caching, while 57% latency reduction can be achieved by pre-fetching, and 60% latency reduction is possible using a combination of caching and pre-fetching. In related research, Padmanabhan and Mogul [68] consider pre-fetching between servers and clients. In their model, the prediction is done at the server, although the actual pre-fetching is done by the clients. Their simulations show that pre-fetching is useful in reducing average latency over both low and high bandwidth connections, at the cost of an increase in network traffic. Cao *et al.* [21] apply pre-fetching between low-bandwidth clients and a proxy. The idea is to exploit user idle times to pre-fetch documents into the client's cache. Results of their simulations show that the latency of Web access can be reduced by about 10% [21].

Several researchers are working towards improving the performance of the HTTP protocol. Usually, most Web pages contain inline images (i.e., URLs referring to image files which should be displayed along with the HTML text of the Web page, by the Web browser). In its current form, the HTTP protocol requires a separate TCP connection to be established for each document transfer [44]. For example, if a Web page contains three inline images, then the client makes four HTTP requests to the server: one for the HTML text, and one request for each of the three inline image files.

The requirement of establishing separate TCP connections for fetching each URL introduces round-trip delays due to the time required to open and close TCP connections [34, 55]. Furthermore, TCP does not fully utilise the available network bandwidth for the first few round-trips of a connection because of TCP's congestion control algorithm called *slow-start*. The *slow-start* algorithm is good for congestion control, but increases latency for short connections [55]. To address these inefficiencies, Persistent-Connection HTTP (P-HTTP) was proposed by Mogul *et al.* [55]. In P-HTTP, a client uses one TCP connection to send multiple requests, reducing network latency because less time is spent in TCP's connection-opening handshake

[34]. In addition to reducing latency, P-HTTP reduces load on the Web servers since extra resources (e.g., ports, data structures, and other resources) are not required, and also reduces the number of packets created due to the opening and closing of TCP connections. Presently, P-HTTP is implemented in HTTP/1.1 [34].

Current HTTP implementations use If-Modified-Since requests to ascertain the validity of documents cached at the client. Such requests result in document transfers between the client and the server if the copy of the document at the client side is not up-to-date. However, it is obvious that documents seldom change substantially. Therefore, bandwidth savings can be achieved if the difference (referred to as "delta") between the stale copy and the most recent copy of the document is transmitted to the client [56]. Mogul *et al.* [56] have explored the benefits of delta encoding using trace-driven simulations. Their results show that an overall bandwidth saving of 22% and transfer time saving of 11% can be achieved without introducing significant overhead to the HTTP protocol.

2.3 Summary

The first part of this chapter introduced the World-Wide Web. The World-Wide Web is a globally distributed communication system that uses hypertext for structuring information. It is based on the client-server architecture. Information residing on Web servers is accessed by users via Web clients. Web proxies receive requests from groups of clients and service these requests on behalf of its clients. By caching frequently accessed documents, proxies can reduce the volume of network traffic, and the latency of accesses.

The continued explosive growth of the Web has motivated researchers to explore techniques for improving performance of the Web. In the second part of this chapter, several recent efforts in this direction were reviewed. The rest of the thesis focuses on Web proxy workload characteristics, and modelling Web document accesses.

Chapter 3

Workload Characterisation

This chapter presents the results from a workload characterisation study that was carried out for Web proxy servers at three different levels of a caching hierarchy. The purpose of the workload characterisation is two-fold. First, the study identifies common characteristics in the workloads presented to the Web proxy servers. Understanding these characteristics is an important step towards improving the performance of caching hierarchies. Second, the study identifies changes observed in the workload characteristics across different levels of a caching hierarchy.

This chapter is organised as follows. Section 3.1 describes the data collection, reduction, and analysis of access logs that was performed. Section 3.2 presents the results of the workload characterisation study, followed by conclusions in Section 3.3.

3.1 Data Collection, Reduction, and Analysis

This section presents an overview of the data sets used in the workload characterisation study, and describes how they were processed. Section 3.1.1 describes the format of the access logs used in this study. Section 3.1.2 describes the data collection sites. Section 3.1.3 presents the analysis of the "raw" access logs, while Section 3.1.4 describes the reduction of the raw data sets to only contain useful information for the analysis performed in this chapter.

3.1.1 Access Log Format

The workload characterisation study is conducted by analysing access logs from Web proxy servers. An entry in the access log has the following format:

```
Timestamp Elapsed Client Action/Code Size Method Hierarchy/From Content
```

with:

- Timestamp: The time when the request is completed, specified in seconds since January 1, 1970 with millisecond resolution.
- Elapsed: This field notes the elapsed time of the request in milliseconds (i.e., the time between the accept() and close() of the proxy socket).
- Client: The IP address of the connecting client.
- Action: This field describes how the request was treated locally at the proxy server (e.g., hit, miss).
- Code: The status-code in the HTTP reply sent to the client in response to its request.
- Size: This field records the number of bytes transferred from the proxy to the client for a particular document retrieval request.
- Method: The HTTP method requested by the client.
- URL: The URL of the requested document.
- Hierarchy: This field describes how and where the requested document was fetched.
- From: This is the host name of the machine from which the requested document was obtained.
- Content: The content type of the document specified using the MIME format.

The information provided by the access logs has some limitations. For example, the access logs record the number of bytes written to clients for document transfers, not the actual document size. Also, the timestamp records when a request is processed, and not when it arrived at the proxy. Furthermore, there is no distinction between document transfers caused by humans and document transfers caused by software (e.g., search engines). These issues are outside the scope of the present study. However, the access log entries are still sufficient to produce useful summary statistics about workload volume, document types and sizes, document popularity, proxy cache performance, as well as other statistics.

3.1.2 Data Collection Sites

The access logs for this study were obtained from three World-Wide Web proxy servers:

- University of Saskatchewan
- CANARIE [23]
- NLANR [63]

Each of these sites continuously records access logs, which were obtained on a daily basis using FTP. Further detail on each of these sites is provided below.

The Web proxy server at the University of Saskatchewan represents an institution-level Web proxy in the CA*net II caching hierarchy. It serves several hundred users on the University of Saskatchewan campus who have configured their browsers to use the proxy cache. This proxy server is operated by the Department of Computing Services at the University of Saskatchewan. The proxy uses a Digital AlphaServer 1200 5/400 with two 400 MHz processors running `Squid` version 1.2.beta20 [35]. The proxy is configured to use the CANARIE cache as its parent. That is, cache misses at the University of Saskatchewan cache result in requests to the CANARIE cache for the missing document.

The CANARIE proxy cache is the root of the CA*net II caching hierarchy. This machine (called *roadrunner*) is a SPARC Ultra 180 MHz machine, running `Squid` version 1.2.beta22 [14]. It has 512 MB RAM and a 30 GB hard disk. The CANARIE proxy server is physically located at Bell Canada in Toronto, though it is administratively controlled by the CANARIE ARDNOC (Advanced Research and Development Network Operations Centre). This cache currently functions as a parent for several first-level proxies, including proxies at the University of Saskatchewan, the University of Alberta, Dalhousie University, and McMaster University. There are also a small number of users who configure their browsers to directly use the CANARIE proxy cache. The CANARIE proxy has parent links to two nodes in the NLANR (National Laboratory for Applied Networking Research) caching hierarchy, namely the Pittsburgh NLANR node, and the NLANR node at the University of Illinois, Urbana-Champaign (UIUC).

The NLANR traces in this study come from the NCSA (National Center for Supercomputing Applications) proxy server at the University of Illinois, Urbana-Champaign (UIUC). This site represents one of several top-level nodes in the NLANR Web caching hierarchy; it receives requests from sibling caches at the top level, as well as from lower-level caches that use it as a parent. The NCSA proxy server is a Digital AlphaServer 1000 266 MHz machine [62] running `Squid` version 1.2.beta17 [64].

The access logs from the six root caches in the United States can be obtained by FTP from `ftp://ircache.nlanr.net/Traces/`. The access logs for the Urbana-Champaign cache were downloaded regularly since the access logs are updated on a weekly basis. The access logs from the CA*net II cache were obtained by FTP from `http://ardnoc41.canet2.net/cache/squid/rawlogs/`. The CA*net II logs were first made available in December 1998, and are updated on a regular basis. The access logs for the Saskatchewan cache were available directly from the operators of the cache.

The three data sets described above will be referred to as USask, CANARIE, and NLANR in the rest of the thesis. The access logs are presented in this relative order to reflect the progression from an institution-level Web proxy cache to a top-level

Web proxy cache.

3.1.3 Raw Data Analysis

The first step towards analysing the access logs was to concatenate the access log files of individual days together to obtain longer data sets for each site. A 45-day trace was created for the University of Saskatchewan proxy server by concatenating access logs from February 5 to March 30, 1999 [1]. Similarly, a 45-day CANARIE trace spanning February 2 to March 24, 1999 [2], and a 30-day NLANR trace spanning February 3 to March 4, 1999 were created. The University of Saskatchewan proxy server recorded a total of 30,330,149 requests in 45 days of activity. The CANARIE proxy server recorded a total of 20,725,429 requests in 45 days of activity. The NLANR access logs recorded 35,631,782 requests in the 30 days of activity. Table 3.1 provides a summary of the access logs for all three proxy servers.

Table 3.1: Summary of Web Proxy Access Log Characteristics (Raw Data)

Item	USask	CANARIE	NLANR
Access Log Duration	45 days	45 days	30 days
Start Date	Feb 5, 1999	Feb 2, 1999	Feb 3, 1999
End Date	Mar 30, 1999	Mar 24, 1999	Mar 4, 1999
Total Requests	30,330,149	20,725,429	35,631,782
Avg Requests/Day	674,003	460,565	1,187,726
Total Bytes Transferred (GB)	181	91	350
Avg Bytes/Day (MB)	4,026	2,044	11,675

The access logs provide information on proxy servers with different workloads. The clients of the USask proxy server are mostly individual users. The CANARIE cache is configured as a parent for the USask cache. Its clients are mostly institution-level proxies (University of Saskatchewan, University of Alberta, Dalhousie University, McMaster University, CA*net II Networks Operations Centre, Communications

[1] The access logs for 9 days were not available, because of down time for server upgrades and network outages.

[2] The access logs for 6 days were not available.

Research Centre) [14]. The CANARIE proxy is also a child to two NLANR proxies, namely the PSC (Pittsburgh Supercomputing Center) cache at Pittsburgh, Pennsylvania, and the NCSA cache at Urbana-Champaign, Illinois. Most of the clients of the NCSA cache at Urbana-Champaign are institutional caches located in the United States.

Table 3.1 provides insight into the activity of each of these proxy servers. In terms of requests received per day, the NLANR cache is the busiest, followed by the USask and CANARIE caches. Similar observations can be made regarding the average volume of bytes transferred. Upon further analysis of the CANARIE access log, it was found that approximately 52% of the requests were inter-cache queries made using ICP, which do not result in any transfer of documents (i.e., the access log records a transfer of zero bytes for them). Thus the actual workload in terms of document transfers for the CANARIE proxy is much less than that suggested in Table 3.1. For the USask proxy, 17% of the recorded requests were ICP queries, while for the NLANR proxy, none of the requests were ICP queries, since the NLANR access log was not configured to record activity at the ICP port. These results are summarised in Table 3.2.

Table 3.2: Breakdown of Request Methods

Method	USask	CANARIE	NLANR
GET	81.91	48.27	99.90
ICP QUERY	17.40	51.72	-
Others	1.69	0.01	0.10
Total	100.0	100.0	100.0

Since requests to the ICP port do not result in transfers of documents, the remaining analysis focuses on requests received at the TCP port. The breakdown of the HTTP reply codes as a percentage of the total number of HTTP requests seen, is given in Table 3.3.

The HTTP reply codes (outlined in Table 2.2) in the traces describe how a client's request is serviced. A reply code of 200 (OK) means that a valid document

Table 3.3: Breakdown of HTTP Response Codes

Response Code	USask	CANARIE	NLANR
200 (OK)	84.25	72.99	68.74
206 (Partial Contents)	0.22	0.07	0.18
302 (Found)	5.89	6.29	3.70
304 (Not Modified)	6.35	17.98	24.29
Others	3.29	2.67	3.09
Total	100.00	100.00	100.00

was made available to the client, either directly from the proxy cache (TCP_HIT), or by retrieving the document from another proxy cache or from the originating server (TCP_MISS). A reply code of 304 (Not Modified) implies that a client had issued a GET If-Modified-Since request to determine whether or not the client's cached copy of a document was up-to-date, and the server (or some intermediate proxy) replied indicating that the client has a valid copy of the document. An HTTP response code of 206 (Partial Content) implies partial transfer of the document to the client in response to a partial GET request. Partial GETs are used to recover efficiently from partial failed transfers in HTTP/1.1. An HTTP response of 302 (Found) means that the requested document is known to reside in a different location than that specified by the URL. From Table 3.3 it is clear that about 90% of the requests (HTTP 200 and HTTP 304) made to the proxy result in the client successfully obtaining the document.

3.1.4 Data Reduction and Analysis

The access logs record the number of bytes transferred to the clients, regardless of where an up-to-date copy of the requested document was found (i.e., at the proxy cache, at some other cache in the hierarchy, at the originating server, or at the client). The objective is to characterise all requests that would result in the document being accessed from the origin server in the absence of intermediate proxies. Therefore, only HTTP 200 (OK) and HTTP 206 (Partial Contents) responses are considered in the remaining analyses. The raw access logs are reduced to consider only these cases, which reflect transfers of documents to the requesting clients.

Table 3.4 summarises the reduced access logs for the three Web proxy sites. Based on the average number of requests seen per day at each proxy server, the NLANR proxy server has the highest activity, while the CANARIE proxy server has the least activity. This is not surprising since very few institutional caches currently use the CANARIE proxy cache[3].

Table 3.4: Summary of Web Proxy Access Log Characteristics (Reduced Data)

Item	USask	CANARIE	NLANR
Total Requests	21,070,330	7,310,038	24,560,611
Avg Requests/Day	468,229	162,445	818,687
Total Bytes Transferred (GB)	177	89	345
Avg Bytes/Day (MB)	3,943	1,954	11,516
Distinct Documents	5,510,561	4,571,539	8,482,661
Distinct Servers	133,692	117,433	272,509
Distinct Clients	1117	13	-
Mean Transfer Size (Bytes)	8,422	12,297	14,066
Median Transfer Size (Bytes)	2,500	3,455	3,128
Coefficient of Variation	13.79	13.28	17.60

Table 3.4 also indicates the number of distinct documents, servers, and clients recorded in the access logs. Each proxy handles requests for millions of distinct documents located at thousands of different Web servers. The number of clients (i.e., distinct client IP addresses seen) varies quite significantly in the three traces. This number is not known precisely for the NLANR log, since the client IP addresses in the NLANR access logs are randomised every day (for privacy concerns). However, on any particular day, about 700 clients generate requests to the NLANR cache.

Overall, the mean and median document transfer sizes are quite small, as has been reported in previous studies of Web servers [4, 5, 16] and Web proxy servers [1]. In these data sets, the mean size of the documents transferred ranges from 8-13 KB,

[3]The majority (approximately 50%) of the requests at the CANARIE proxy are from the USask proxy server.

while the median is in the range of 2-3 KB. The mean transfer size is larger than the median transfer size because there are some very large documents that skew the mean of the transfer size distribution. There is also high variability in the sizes of documents transferred, as indicated by the coefficient of variation (COV) reported in Table 3.4.

These reduced data sets are used in the analyses in the rest of the chapter.

3.2 Proxy Workload Characterisation

The following sections present the results from a detailed workload characterisation study of Web proxy workloads. The following characteristics are studied: document types and sizes, proportion of documents accessed only once in the access logs, distribution of transfer sizes, popularity of documents, rate of document modifications, distribution of inter-reference times, and temporal locality.

3.2.1 Document Types and Sizes

The next step in the analysis classifies document requests (based on URL extensions[4]) into the following categories:

- HTML: ".html",".shtml", ".htm", and ".map".
- Image: ".giff", ".tiff", ".jpeg", ".xwd", ".rgb", ".xbm", ".tif", ".gif89", ".bmp", ".ief", ".jpe", ".ras", ".pgm", ".ppm", ".pcx", ".pic" and ".jpg".
- Audio: ".au", ".aiff", ".aif", ".aifc", ".mid", ".snd", ".wav" and ".lha".
- Video: ".mov", ".qt", ".avi", ".mpe", ".movie", ".mpeg", ".mpg" and ".mp2".
- Compressed: ".z", ".gz", ".zip" and ".zoo".
- Formatted: ".ps", ".pdf", ".dvi", ".ppt", ".tex", ".rtf", ".src" and ".wsrc".

[4]Note that all documents with a "cgi-bin" or "?" in the URL string are classified as Dynamic documents. The content type information available in the access logs is used only if URL extensions do not classify the documents into one of the above specified generic types.

- Dynamic: ".cgi", ".pl" and ".perl".

Any document that could not be classified under one of the above categories was placed in the Others category.

The results of this analysis for the three data sets are summarised in Table 3.5. The table shows that HTML and image documents account for close to 95% of the total requests. Similar results were reported for client traces by Cunha *et al.* [29] and for server traces by Arlitt and Williamson [4, 5].

Unlike Web server workloads, however, Table 3.5 shows that documents of image type are consistently the most requested document type (65-78%), followed by HTML documents (about 20%). Similar observations were made by Abdulla *et al.* [1] in their characterisation of Web proxy traffic. It is also observed that image type documents are responsible for the highest percentage of bytes transferred (40-52%), followed by HTML documents (18-28%).

Table 3.5: Breakdown of Document Types and Sizes

USASK								
Item	HTML	Image	Audio	Video	Compressed	Formatted	Dynamic	Others
% of Requests	19.47	77.52	0.29	0.06	0.06	0.45	1.91	0.24
% of Bytes	23.25	52.15	3.39	10.07	3.20	5.80	0.82	1.32
Mean Transfer Size	10,060	5,665	98,687	1,322,156	458,526	109,657	3,621	46,141
Median Transfer Size	4,710	2,230	3,641	366,890	46,632	5,728	804	5,052
COV of Transfer Size	5.80	2.98	5.47	2.38	3.80	5.95	23.90	7.49
CANARIE								
Item	HTML	Image	Audio	Video	Compressed	Formatted	Dynamic	Others
% of Requests	20.58	76.38	0.44	0.12	0.10	0.56	1.64	0.19
% of Bytes	17.86	49.01	5.74	12.36	5.12	7.89	0.24	1.78
Mean Transfer Size	10,668	7,890	160,373	1,294,966	656,479	173,987	1,794	115,717
Median Transfer Size	4,982	3,199	10,541	445,901	100,447	6458	1,279	11,148
COV of Transfer Size	6.28	2.30	4.48	2.32	3.15	5.38	2.17	8.93
NLANR								
Item	HTML	Image	Audio	Video	Compressed	Formatted	Dynamic	Others
% of Requests	21.51	68.31	0.26	0.08	0.19	0.62	1.69	0.48
% of Bytes	18.11	39.99	3.32	6.36	7.83	14.58	1.00	8.81
Mean Transfer Size	11,845	8,234	176,737	1,071,632	566,495	330,509	8,330	260,419
Median Transfer Size	4,686	2,435	15,978	438,584	104,275	10,038	1,979	16,244
COV of Transfer Size	26.65	2.98	4.67	2.30	3.17	5.38	24.20	3.30

A substantial portion of the byte transfer volume is accounted for by audio, video, compressed, and formatted document types, which, despite their relatively infrequent access (typically close to 1% of the requests), are large enough to generate 20-30% of the bytes transferred. This observation is substantiated by the large mean and median transfer sizes indicated for these document types, compared to the other document types. In general, the COV values within each document type category are much lower than the COV of transfer sizes for the aggregate data set (see Table 3.4). The large variation in the mean transfer sizes across the diverse document types helps explain the high COV reported in the aggregate data sets.

3.2.2 One-Time Referencing

A surprising observation made in previous analyses of Web server workloads [4] was that (regardless of the duration of the access log studied) typically 15-30% of the documents accessed in the log were accessed only once in the log. This so-called "one-time" referencing behaviour is of concern for Web caching, since there is clearly no point caching something that will be accessed only once.

The precise cause of this one-time referencing behaviour is not fully understood. Several explanations have been proposed. First, it might indicate the vastness of the World-Wide Web, and the low signal-to-noise ratio for much of its content. Second, it might reflect human nature in browsing habits (e.g., once a site has been visited, there is no need to visit it again). Third, it might reflect the behaviour of content providers, who might use date-based URL names, or who might redesign or modify Web pages on a regular basis to keep them current, while possibly removing or renaming old pages. Fourth, it may reflect the presence of search engines or Web robots that traverse many pages to construct an index. Fifth, the presence of a high percentage of one-timers might indicate that caching hierarchies are actually working well. Finally, it may be the consequence of document pre-fetching (i.e., prediction) algorithms in some proxies and/or client browsers. All of these hypotheses seem plausible. If any (or all) of them are true, then they indicate a challenging workload environment for Web caching algorithms.

Several other explanations seem less plausible. For example, attributing this one-time referencing to the presence of dynamic requests (e.g., CGI) is not possible, since dynamic documents typically account for much less than 5% of the workload in Web server and Web proxy access logs. Attributing one-time referencing to typographical errors made by clients in requested URL names does not make sense, since the access log analyses usually focus on *successful* requests, not errors.

Table 3.6 summarises the contribution of one-timers with respect to the number of distinct documents, the total requests, and the total bytes transferred, for the three data sets. One-timers account for a substantial portion of the total requests (17.8-45.8%) and total bytes transferred (27.1-48.8%). Also, approximately 70% of the documents referenced (67.8-74.9%) are one-timers. The latter number is significantly higher than that for Web server workloads [4]. Presumably this is because requests made to a proxy are for a much larger population of Web documents, compared to those made to a single server.

Table 3.6: One-Time Referencing Behaviour in Web Proxy Workloads

Item	USask	CANARIE	NLANR
Distinct Documents	5,510,561	4,571,539	8,491,008
One-Timer Documents	3,753,468	3,470,920	6,056,302
One-Timer/Distinct Documents (%)	68.11	76.87	71.32
Total Requests	21,070,330	7,310,038	25,942,786
One-Timer Requests	3,753,468	3,514,589	6,056,302
One-Timer/Total Requests (%)	17.81	48.07	23.34
Total Bytes Transferred (GB)	177	89	345
Total One-Timer Bytes (GB)	48	43	111
One-Timer/Total Bytes (%)	27.1	48.3	32.2

Table 3.7 presents a more detailed analysis of one-timer documents based on document type. This analysis shows that HTML and image type documents constitute 95% of the one-timers seen, with each type occurring in the same proportion as they do in the request stream.

The predominance of one-time referencing for Web documents highlights the need for novel Web proxy caching policies that can effectively discriminate against one-timers. For example, frequency-based algorithms, such as Least Frequently Used (LFU), tend to perform better than recency-based algorithms, such as Least Recently Used (LRU) [6]. Similar observations have been made for Web server caching algorithms [5, 85].

Table 3.7: Breakdown of Document Types for One-Timers

Item	USask	CANARIE	NLANR
HTML	22.76	21.07	23.52
Image	74.23	76.11	73.37
Others	3.01	2.82	3.11
Total	100.0	100.0	100.0

3.2.3 Transfer Size Distribution

The next analysis focuses on the transfer size distribution for the documents returned to the requesting clients (either directly by the proxy, or after obtaining the document from a higher-level proxy or the originating server). Of particular interest are, the shape of this distribution, the presence of a heavy tail in the distribution, and the impact of the heavy-tailed distribution on Web and network performance.

Figure 3.1 shows the cumulative distribution function of the transfer size for each proxy server, using a logarithmic scale on the horizontal axis. Almost all the transfers are in the range from 100 to 100,000 bytes, with very few small transfers (say, less than 100 bytes) and very few large transfers (say, more than 100,000 bytes). This distribution is similar to the document size distribution reported for Web clients [29] and for Web servers [4, 5, 9, 16, 28].

The transfer size distribution in Figure 3.1 is *heavy-tailed*. In simple terms, a "heavy tail" in a Web document size distribution means that the very large "elephants" (i.e., outliers) in the tail of the distribution, although relatively few in number, are large enough to contribute significantly to the overall traffic volume observed (e.g., skew the mean transfer size distribution, as observed in Table 3.4).

Figure 3.2(a) illustrates the "heavy-tail" transfer size distribution for the USask data set. Figure 3.2(a) shows that in the USask data set 75% of all the distinct documents requested are for transfers less than 10,000 bytes in size[5]. About 80% of

[5]Since multiple references to a document can be associated with different transfer sizes (because of aborted transfers, partial transfers, or document modifications), the average transfer size is considered for all documents referenced more than once.

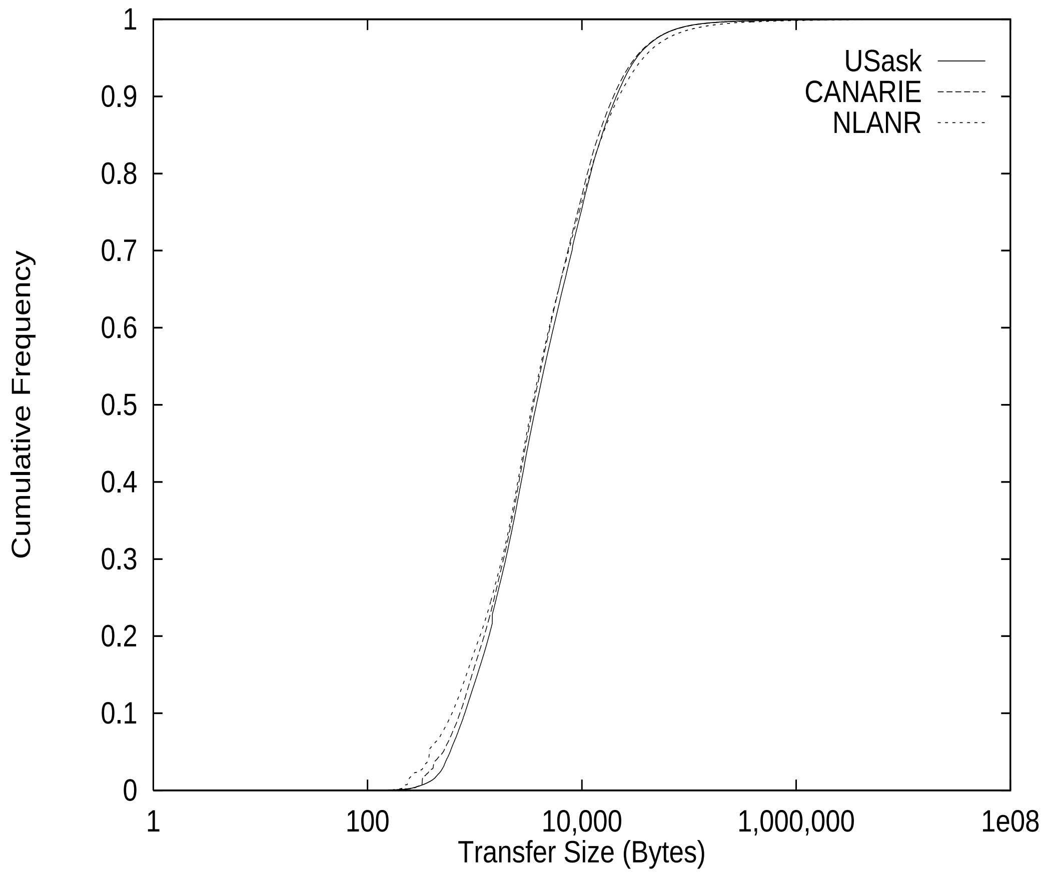
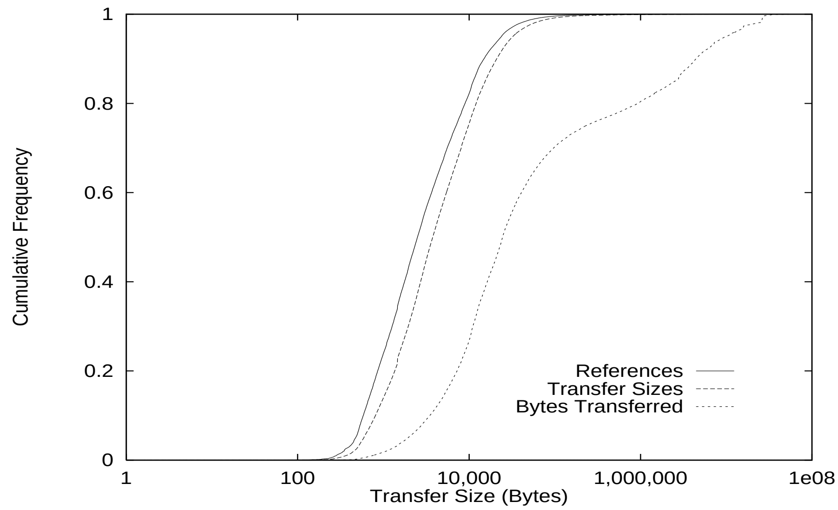


Figure 3.1: Cumulative Distribution Function for Transfer Sizes, by Proxy

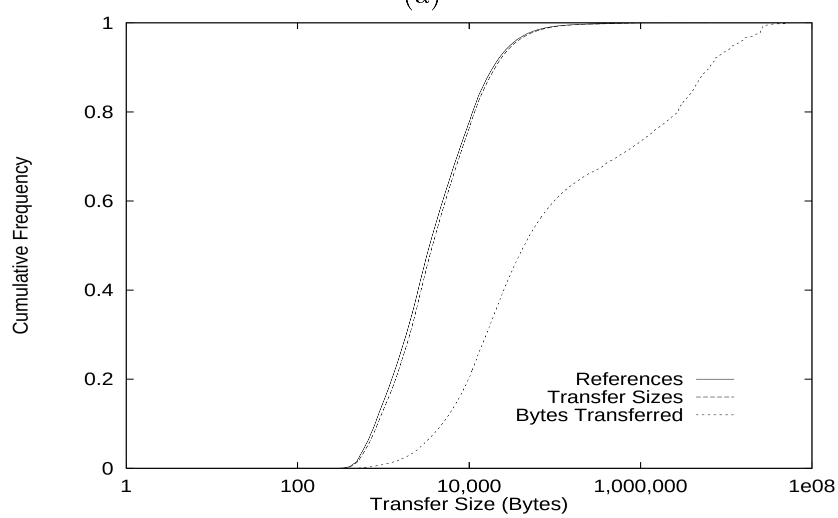
the references are for documents in this category. However, transfers of documents smaller than 10,000 bytes account for only 27% of the total bytes transferred by the proxy to the clients. The tail of the distribution (transfers over 100,000 bytes) accounts for a significant 30% of the total bytes transferred. Similar observations can be made for the other data sets. Therefore, caching smaller documents can increase the hit ratio at the cost of more bytes transferred over the network. To reduce the network traffic volume, proxies can cache larger documents, thus sacrificing the cache hit ratio.

The simplest example of a heavy-tailed distribution is the doubly-exponential Pareto distribution; its probability density function is:

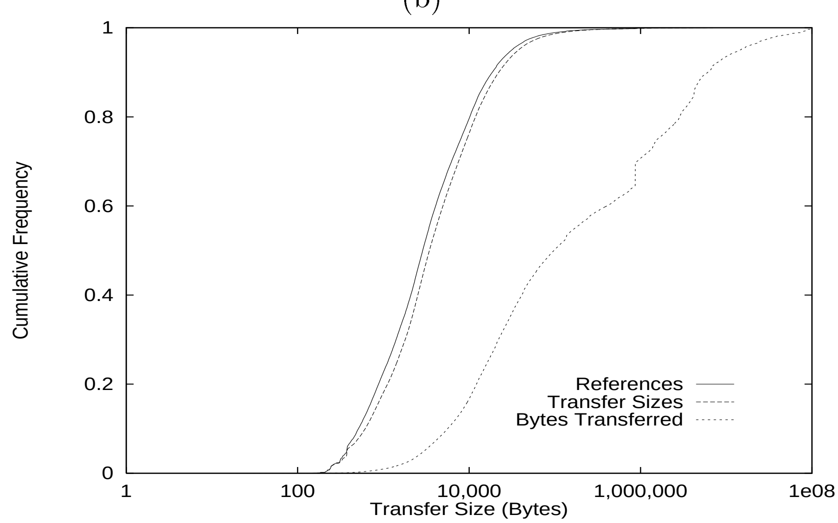
$$P(x) = \alpha k^{\alpha} x^{-\alpha-1}, \alpha, k > 0, x \geq k$$



(a)



(b)



(c)

Figure 3.2: Illustrating *heavy-tail* in Transfer Size Distribution: (a) USask (b) CANARIE (c) NLANR

The cumulative distribution function for the Pareto distribution is:

$$F(x) = P[X \leq x] = 1 - (k/x)^{\alpha}$$

The Pareto distribution has been applied to phenomena observed in social sciences, such as the distribution of the length of books on a library shelf [50], and the distribution of income [40]. In computing applications, this distribution has been used to model FTP data bursts [66, 67] and Web traffic [4, 28, 29].

The α parameter is known as the tail index [28], and k defines the beginning of the "tail" of the distribution (i.e., it represents the smallest possible value of the random variable in the heavy-tailed distribution). As α decreases, the tail of the distribution becomes heavier [28]. In other words, an arbitrarily large portion of the probability mass may be present in the tail of the distribution as α decreases.

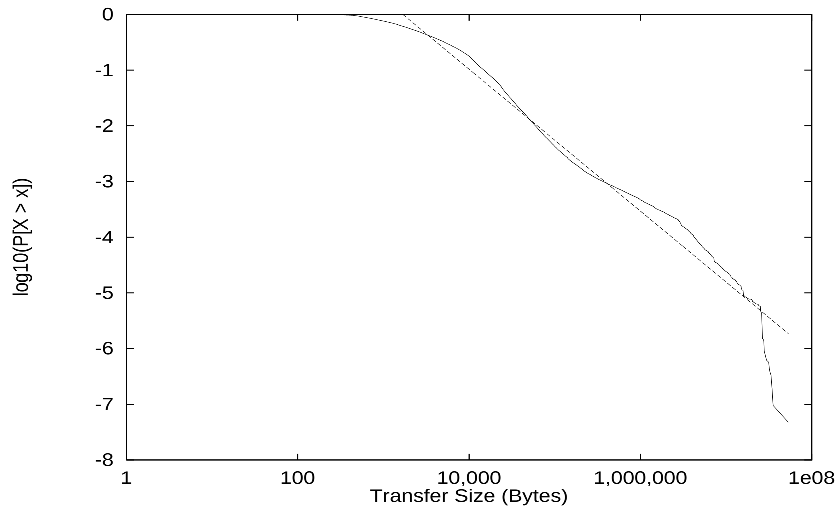
To estimate the tail index α for the transfer size data sets, the approach outlined in [10] and [28] is followed. First, a log-log complementary distribution (LLCD) is plotted for the transfer sizes in the data sets. A LLCD plot graphs $log\bar{\mathrm{F}}(x) = log(1 - F(x))$ versus $logx$, for large x [10]. Heavy-tailed distributions have the following property:

$$\frac{dlog\bar{\mathrm{F}}(x)}{dlogx} = -\alpha, x > k$$

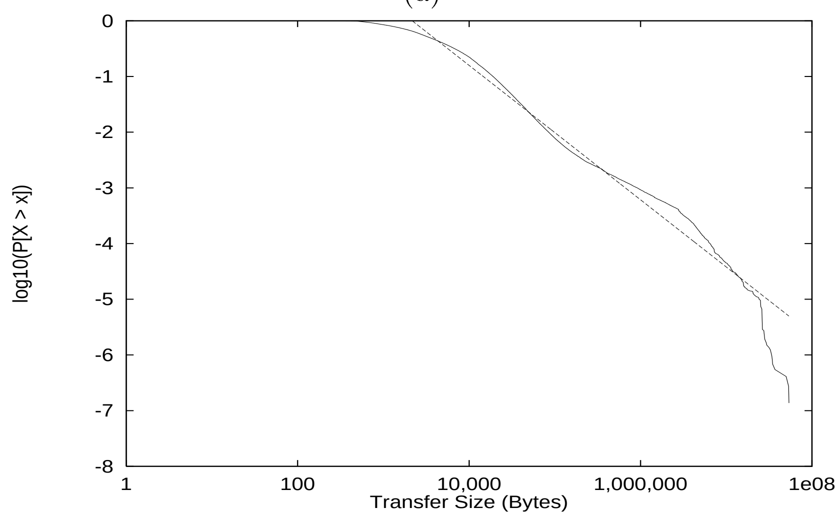
An estimate of the tail index α is obtained by determining the slope of the LLCD plot for values of x greater than k, using least-squares linear regression [39, 66].

Figure 3.3 shows the LLCD plot for all the data sets, along with the least-squares regression fit for the heavy tail ($k = 10,000$ bytes). Table 3.8 summarises the estimated α value for each data set, along with the coefficient of determination (R^2), which assesses the "goodness of fit" for the linear regression. These results show a very strong fit (R^2 is close to 1.0), and α values ranging from 1.07 to 1.27.

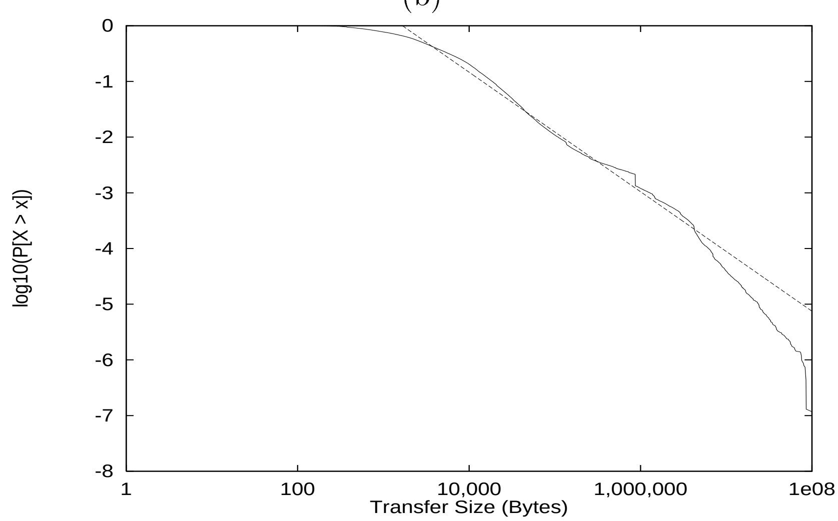
It can be concluded that proxy transfer sizes are heavy-tailed, just like server transfer sizes ($0.93 \leq \alpha \leq 1.33$) [5]. Among the three data sets considered, the NLANR data set exhibited the heaviest tail ($\alpha = 1.07$), while the USask data set shows the least heavy tail ($\alpha = 1.27$). This implies that lower-level proxies tend to



(a)



(b)



(c)

Figure 3.3: Log-Log Complementary Distribution (LLCD) Plot of Transfer Sizes: (a) USask (b) CANARIE (c) NLANR

Table 3.8: Estimates of α for Heavy-Tailed Transfer Size Distributions

Item	USask	CANARIE	NLANR
α	1.27	1.20	1.07
R^2	0.96	0.97	0.98

favour caching small documents over large ones, resulting in heavier tails at higher-level caches.

3.2.4 Document Popularity

Many researchers have noted that the popularity of Web documents is highly uneven [2, 5, 9, 10, 16, 17, 29]. The objective of this analysis is to determine whether the Zipf distribution applies to the proxy document reference streams across different levels of a caching hierarchy.

Zipf's Law [89] states that if items are ranked (r) according to their popularity (P) measured by the frequency of reference for indiviual items, then the popularity of an item is given by,

$$P \sim \left(\frac{1}{r}\right)^{-\beta}$$

where the exponent β is often close to unity [41].

In the special case where the exponent β is equal to one, the N^{th} most popular item is exactly twice as popular as the $2N^{th}$ item, and so on. The Zipf distribution has been widely used to model many aspects of social and economic behaviour (e.g., salaries of executives, distribution of religious castes [41], popularity of books borrowed from a public library [50, 89], and popularity of words used in the English language [41]). It has also been applied to model memory referencing behaviour of computer programs [18, 69], and, most recently, to model Web referencing behaviour [17].

To determine whether or not Web proxy document references follow the Zipf distribution, the documents are first ranked, with the most referenced document being assigned a rank of one, followed by the next most referenced document with a rank of two, and so on.

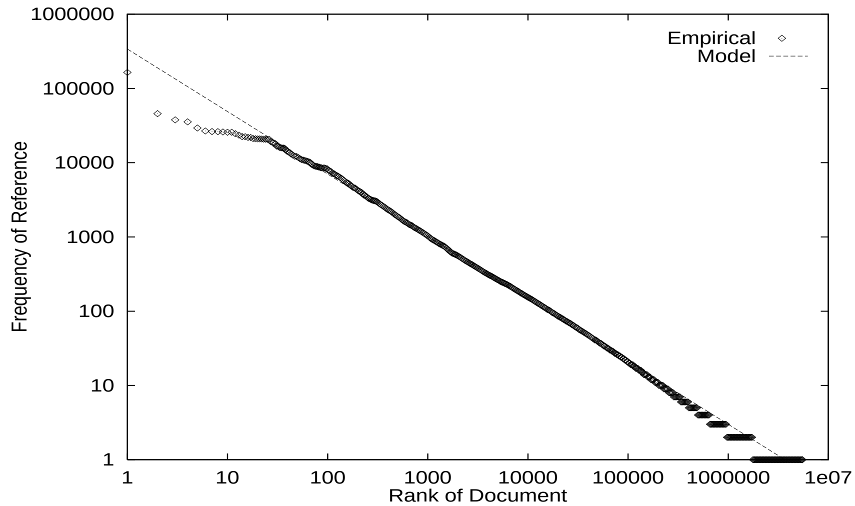
The frequency versus rank plots for the three data sets (on log-log scale) appear in Figure 3.4. Visual inspection of the graphs suggests that the distributions follow Zipf's law, albeit not as strictly as has been observed for Web servers. There appears to be some flattening out at the most popular end and the least popular end of the plots. This might be due to caching of the frequently requested "hot" documents at lower-level caches. The middle portion appears to follow the Zipf distribution more accurately. Similar observations are made by Breslau *et al.* [17].

To determine the best-fit exponent (β) of the Zipf distribution, the method in [73] is used. This involves grouping all documents that have been referenced an identical number of times and assigning them the same rank. For example, if three documents are referenced 100 times each and are ranked 1000, 1001 and 1002, then the modified data set considers them to be a single data point with a rank of 1001 (i.e.., the average of the three ranks) and popularity equal to 100. After obtaining the modified data set, a least-squares fit over the (log-transformed) modified data set is performed. The calculated values for β and the goodness of fit (R^2) values are shown in Table 3.9. Figure 3.4 also shows that the Zipf distribution plots obtained with the calculated β values are very similar to their corresponding frequency versus rank plots. These results suggest that a Zipf-like model can capture the distribution of references seen at a proxy. Note that the exponent β decreases for higher-level proxies. This is most likely due to filtering of popular documents at the lower-level proxies. Roadknight *et al.* [73] came to similar conclusions after a detailed study of the document popularity characteristics.

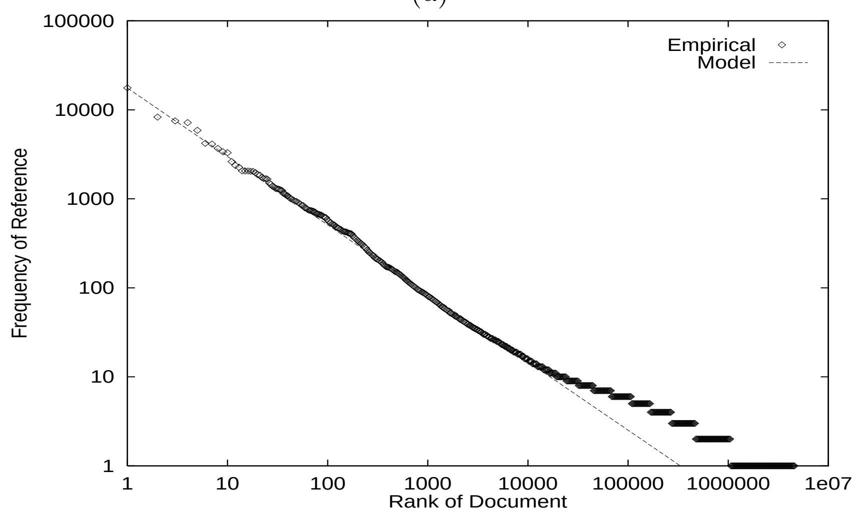
Table 3.9: Estimated Slopes for Zipf-Like Referencing Distribution

Item	USask	CANARIE	NLANR
β	0.84	0.77	0.74
R^2	0.96	0.90	0.93

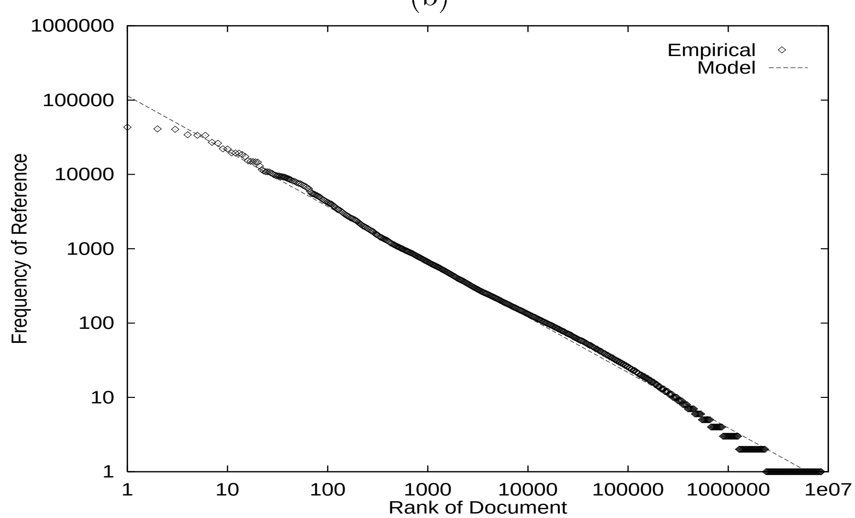
Another way of characterising the uneven document access patterns is to de-



(a)



(b)



(c)

Figure 3.4: Reference Count versus Rank: (a) USask; (b) CANARIE; (c) NLANR

termine the extent to which references are skewed towards certain documents. A measure of skewness, referred to as *concentration*, was applied to memory and file referencing behaviour [18, 86], and later to Web server document accesses [5, 16, 43].

The concentration phenomenon, as applied to documents sorted by their reference counts, is illustrated in Figure 3.5. Non-uniform referencing of Web documents is clearly reflected in this figure, with 30% of the documents accounting for 60-80% of the references. The remaining 70% of the documents (primarily the one-timers) account for the remaining requests.

Concentration of references is also observed when documents are sorted according to the volume of bytes transferred in response to requests for them. Figure 3.6 shows that 10% of the documents account for approximately 90% of the bytes transferred by the proxy to clients.

Higher-level proxies (i.e., CANARIE, NLANR) receive requests from clients that are typically lower-level proxies. Therefore, the concentration of requests at these proxies might be due to the overlap in requests from different clients, as frequent requests from users generally get captured by the browsers and lower-level proxy caches. Among the three data sets, the USask data set shows the most concentration, while the CANARIE data set shows the least. The lower concentration at the CANARIE proxy might be due to its small client population (about 13).

These results suggest that the concentration of references is lower at the Web proxy servers than at the Web servers [4, 5]. This observation makes sense intuitively, since the clients of a Web proxy can effectively access any available document in the Web (i.e., the document set is very large). For Web servers, the requests are restricted to a limited set of documents (i.e., the documents present at the Web server).

However, some recent studies have shown that many static documents are never cached [19, 88]. There are three main reasons for these documents not being cached. First, a significant fraction of the responses received from the Web servers contain no `Last Modified` dates, disabling caching of these requests [88]. Second, many

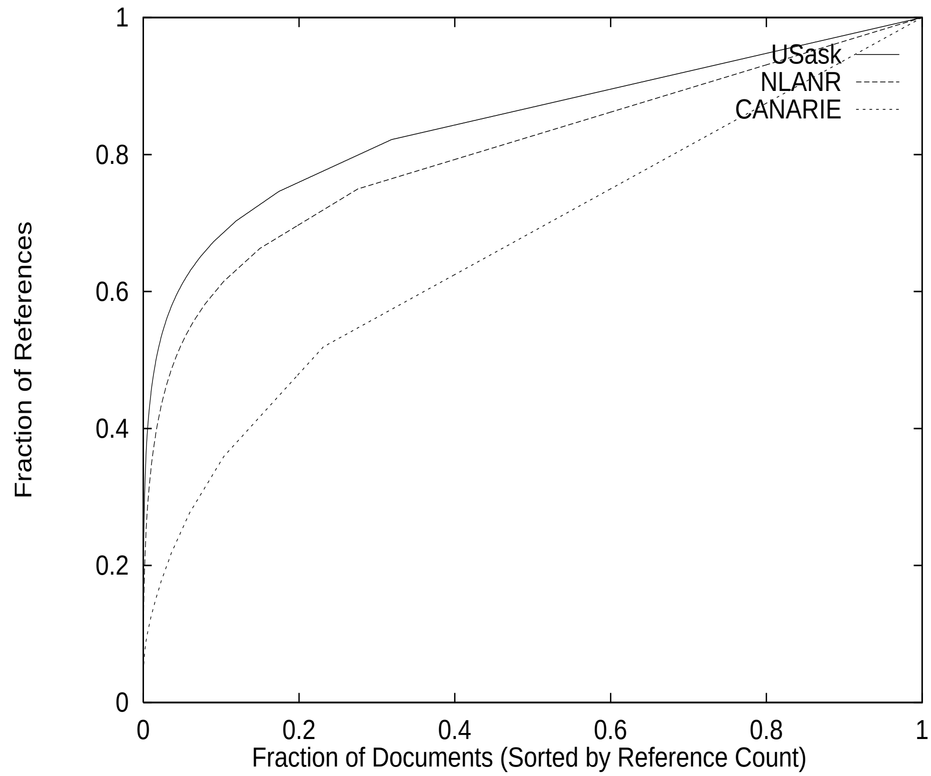


Figure 3.5: Concentration of References (Documents Sorted by Reference Count)

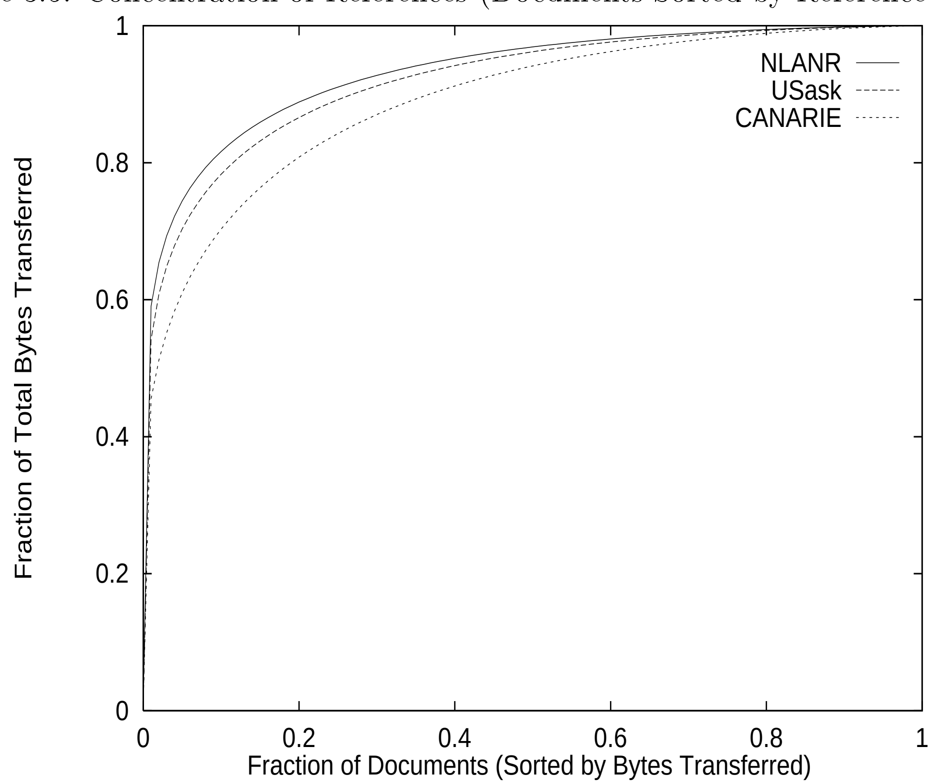


Figure 3.6: Concentration of References (Documents Sorted by Bytes Transferred)

requests in the proxy traces are associated with "cookies" [6], and most proxies do not cache responses that are cookie-specific. Third, some static documents are noticed to be associated with an HTTP directive ("no-cache" pragma in HTTP/1.0) that informs the clients that the document should not be cached.

A cursory analysis of the access logs revealed that there are many documents that are never being cached (at the browser, and/or at the proxy). Therefore, part of the concentration of references seen is definitely due to the uncacheability of the documents. Since the access logs used in this analysis did not have the necessary information to distinguish clearly between cacheable and uncacheable documents, no further attempt was made to understand the impact of cacheability of documents in the rest of the thesis.

3.2.5 Correlation between Document Popularity and Document Size

It was observed earlier that most Web transfers involve small documents, as reflected by a median value lower than the mean value of the transfer size. A natural question that arises is whether there is any relationship between the frequency of reference to a document and the document size.

To answer this question, the correlation coefficient between frequency of access and transfer size is calculated for each of the data sets. The correlation coefficient always lies between -1 and +1: a correlation coefficient close to +1 would indicate that large transfers occur more often, whereas a correlation coefficient close to -1 would indicate that large transfers occur less often. A correlation coefficient close to zero indicates no linear relationship between number of requests and transfer size.

The correlation coefficients for each of the data sets are shown in Table 3.10. The results indicate little or no correlation between document popularity and document size. However, the fact that all three correlation coefficients computed are negative

[6] A cookie is a small piece of information sent by Web servers to the browsers, and are stored at the browsers. This information is used by the Web server the next time the user visits the Web site.

may imply that transfers of smaller sized documents are slightly favoured over larger documents. Similar observations have been made by Arlitt *et al.* [4] for Web server workloads and more recently by Breslau *et al.* [17] for Web proxy workloads.

Table 3.10: Correlation Analysis: Frequency of Reference versus Transfer Size

Item	USask	CANARIE	NLANR
Correlation Coefficient	-0.04	-0.06	-0.03

3.2.6 Rate of Change of Documents

As already seen, there are multiple references to many Web documents. However, some instances of multiple references may represent accesses to different versions of a document that has been updated. This type of referencing behaviour would yield quite different cache behaviour than would multiple accesses to an unmodified document. Therefore, it is important to know if there is any correlation between frequency of access and document modifications. Since Web caches do not cache dynamic files, the following analysis focuses on static [7] documents only.

A document is assumed to be modified when the size associated with a reference to the document is different from the size associated with the document when it was last referenced. There is some error in this assumption, as the access logs report the bytes transferred from the proxy to the clients, which includes HTTP headers. Since bytes transferred is approximated as document size, modifications are reported when header size changes occur, and when clients aborts requests. The *change ratio* of a document is defined to be the ratio of the number of modifications seen to the number of references made since the document was first referenced.

Figure 3.7 shows the average change ratio of documents as a function of the midpoints of reference count intervals. The change ratio for all documents with

[7] As outlined in Section 3.2.4, it is not necessary that all static documents are cacheable. However, for simplicity of analysis, all static documents are considered to be cacheable.

reference counts in the interval $[x_i, x_{i+1}]$, where $x_{i+1} = 1.1 * x_i$, were averaged to produce each point in the figure.

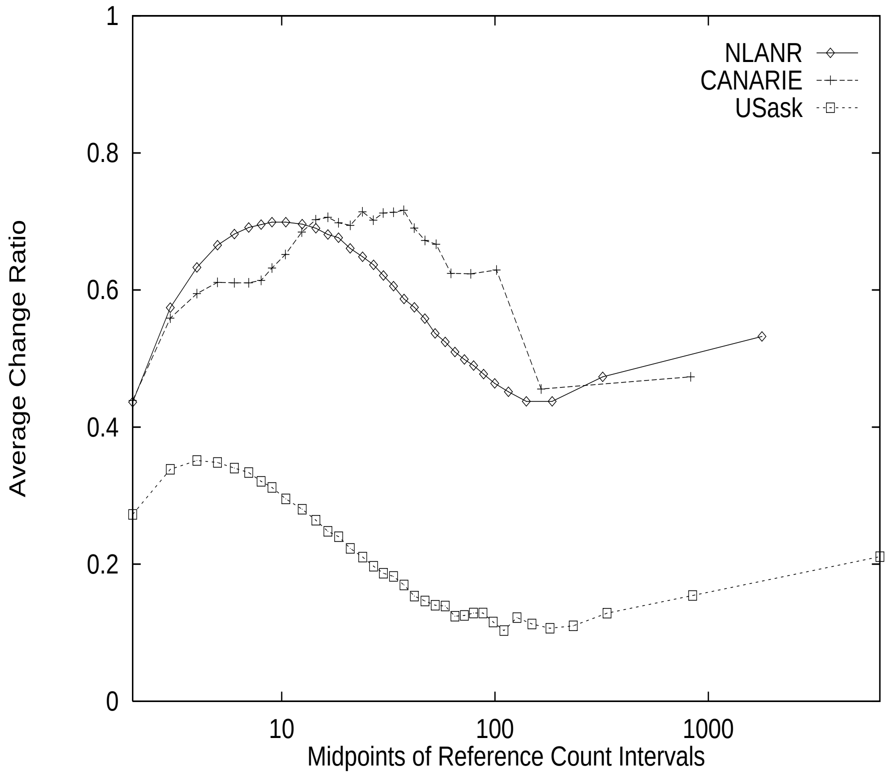


Figure 3.7: Average Change Ratio as a Function of the Midpoints of Reference Count Intervals

The average change ratio appears to increase initially, then decrease, and finally increase again as the reference count increases. The documents seen in the traces tend to be more frequently modified as one moves higher up in the caching hierarchy because requests from lower-level proxies are forwarded to higher-level proxies when document modifications occur.

3.2.7 Inter-Reference Times

The next analysis focuses on inter-reference times for documents that are referenced more than once. The inter-reference time is measured using the timestamps in the access logs. The first seven days in each trace are analysed [8].

Figure 3.8 shows the inter-reference time distributions for the USask, CANARIE, and NLANR data sets. In this section, the discussion focuses on the USask data set. Similar results were found in the other data sets.

Figure 3.8 (a) shows that the probability of re-referencing a document decreases with time, with marked peaks around 24-hour intervals, indicating re-access of documents on a daily basis. Similar observations were made by Breslau *et al.* [17] and Rizzo *et al.* [72]. The peaks at 24-hour intervals can also be seen in the CANARIE data set, but not in the NLANR data set. This suggest that time-of-day effects are visible only when the clients of a proxy lie in the same time zone.

Figure 3.8 (b) shows the cumulative distribution function of the inter-reference times for the USask data set. About 30% of the re-references occur in the first minute. Careful investigation of the USask access log reveals that the re-references are caused by a number of factors. Certain popular Web documents, such as the hotmail Web page, are accessed by different clients at almost the same time. This type of document sharing can also be caused by class-related activity. Some of the re-references seen in the access logs are caused by dynamic documents, which could not be recognised by the software used for the document inter-reference time analysis.

A number of re-references were observed for documents which "expire" immediately. Some proxies (e.g., `Squid`) allow the "expire" time specified by the HTTP header to be overridden by a configurable "min age" parameter. The proxy administrator can use this parameter to set a lower bound on the life-time of a document. The current "age" of a document is calculated as the time elapsed since the document was last retrieved. If the "age" of a document is less than or equal to the "min age", then the proxy considers the document to be fresh (irrespective of its "expire" status). Sometimes Web page designers make Web documents "expire" immediately. When such a document is currently being viewed by a client, it would

[8] Longer durations could not be analysed because of discontinuities in some of the traces.

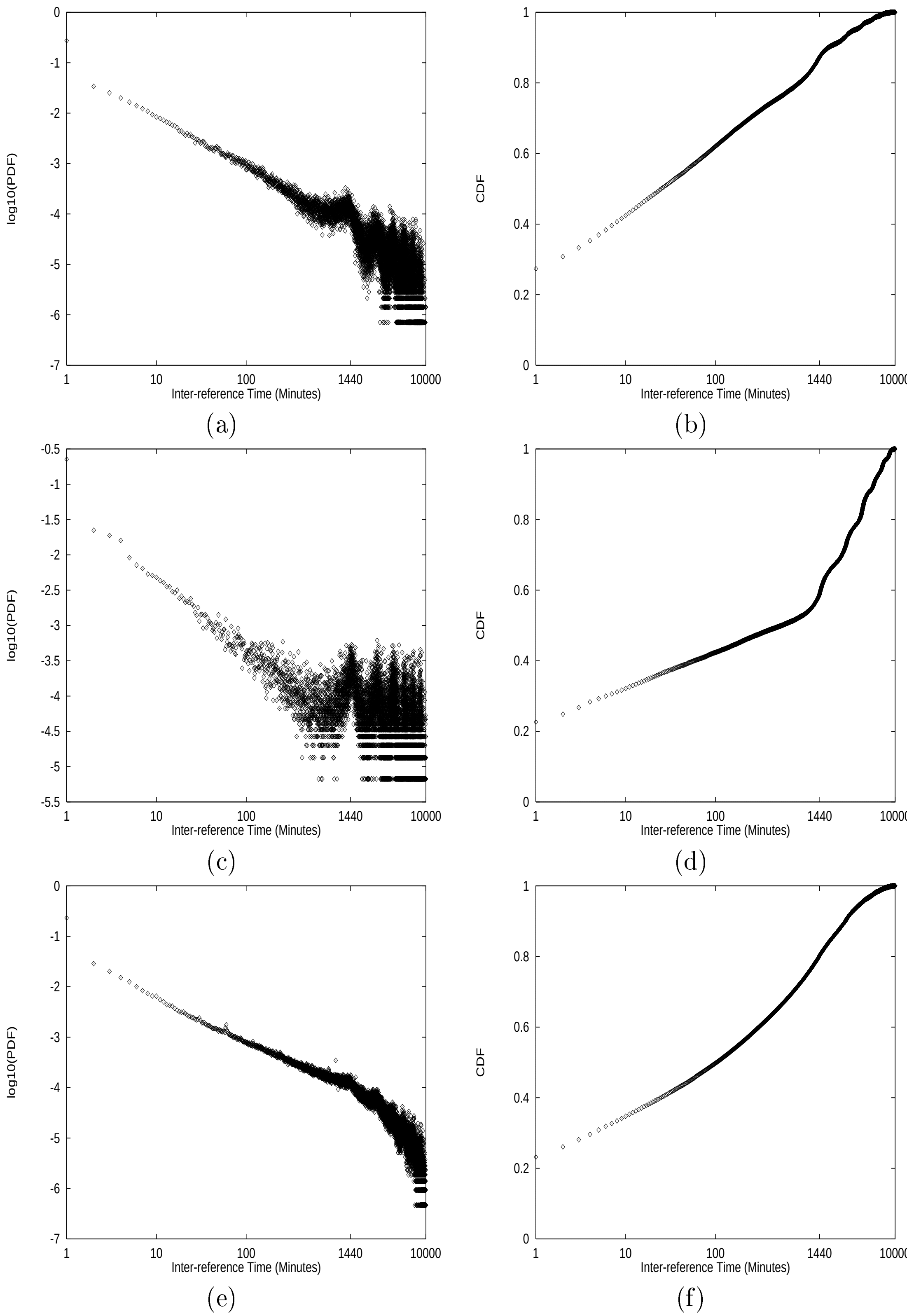


Figure 3.8: Distribution of Inter-reference Times: (a) PDF (USask); (b) CDF (USask); (c) PDF (CANARIE); (d) CDF (CANARIE); (e) PDF (NLANR); (f) CDF (NLANR)

send out If-Modified-Since requests to the proxy to verify whether or not the document has been updated at the proxy. The proxy considers the document to be fresh as long as the "min age" parameter is greater than or equal to the current "age" of the document in the cache. Since the request was an If-Modified-Since, ideally a not modified (i.e., HTTP 304) response should be sent to the proxy. However, there are many such requests which result in hits at the proxy cache followed by a full document transfer (i.e., HTTP 200) instead of a not modified response. In HTTP 1.1, an If-Modified-Since request is treated as an ordinary GET if the time specified in the HTTP header of a request is ahead of the server clock [34]. Since full document transfers happened for the client and the proxy, it is possible that a clock skew problem caused the repeated transfers. However, this explanation could not be verified.

3.2.8 Temporal Locality

Intuitively, temporal locality refers to the property that referencing behaviour in the recent past is a good predictor of the referencing behaviour to be seen in the near future. For example, it is known that the correlation between immediate past and immediate future page reference patterns (of a program) tends to be high, whereas the correlation between disjoint reference patterns tends to zero [31]. While the temporal locality property is traditionally used to describe the referencing behaviour in an aggregate reference stream, it can also be considered on a per-item basis. In the latter context, temporal locality refers to the property that the probability of referencing a particular item decreases with the increasing time of last reference to that item. It is also worth noting that the notion of temporal locality is orthogonal to the concentration behaviour analysed in Section 3.2.4 in the sense that the presence of concentration need not necessarily imply the presence of temporal locality (i.e., the concentration metric does not consider the correlation between a reference to a document and the time since it was last accessed). The following analysis focuses on the presence of "long-term" and "short-term" temporal locality in the workloads.

3.2.8.1 “Hot Set” Drift Analysis

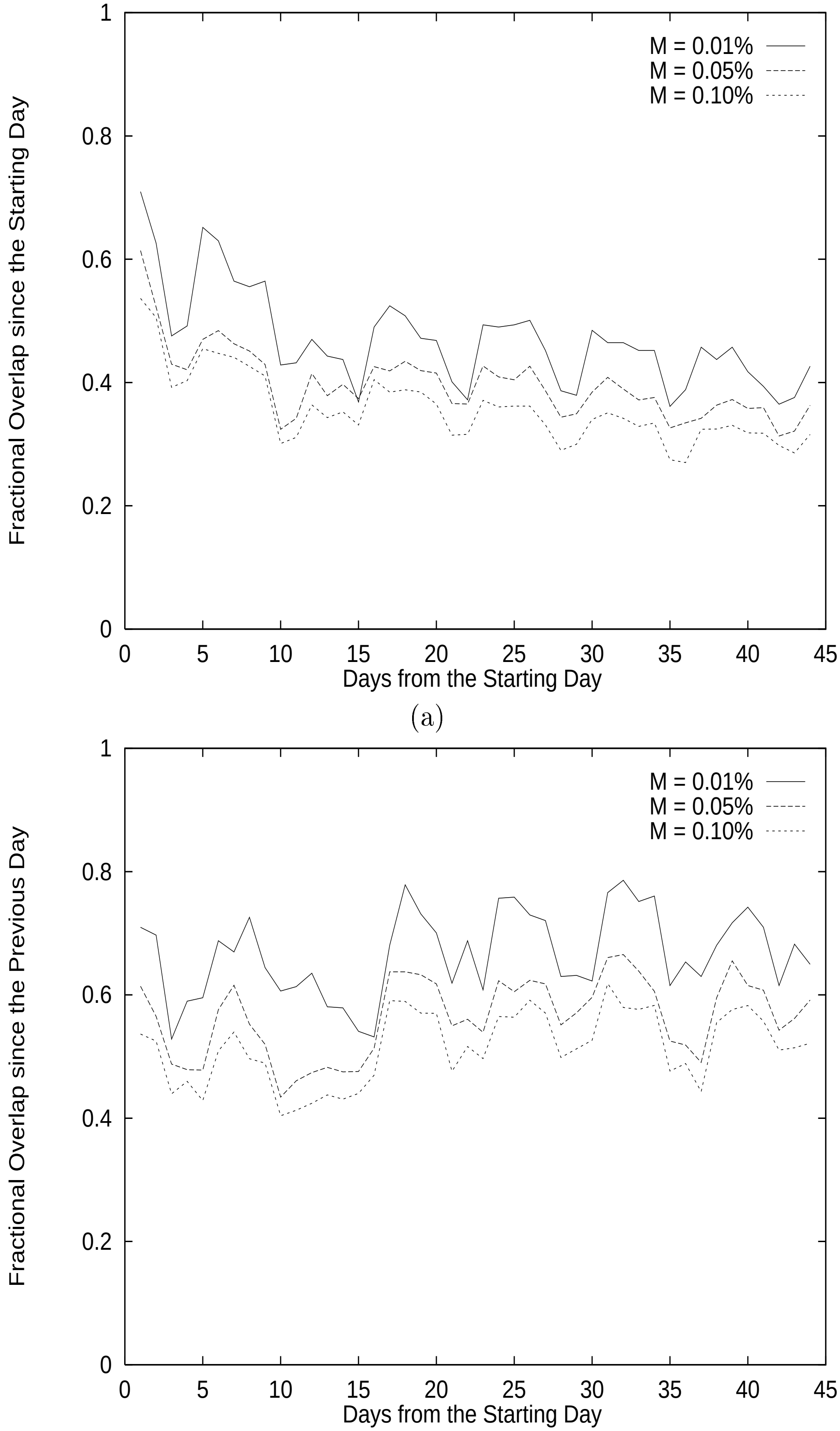
A simple analysis was performed to understand how the set of most popular documents (referred to as the “hot set”) changes with time for each Web proxy. The objective of this analysis is to determine whether or not long-term temporal locality is present in Web proxy references.

Two different measures called *absolute* and *relative* drift were developed. *Absolute drift* measures the fractional overlap with the “hot set” of size “M” on day i of the trace with the “hot set” from the initial starting day (day 0), while *relative drift* measures the fractional overlap with the “hot set” of size “M” from the previous day (i.e., on a day-to-day basis). To understand the impact of the size of the “hot set” on the drift, three values for “M” are considered: 0.01%, 0.05%, and 0.10% of the total unique documents in the trace.

In these experiments, a granularity of one day is considered. Finer time scales (such as hourly) are strongly influenced by time-of-day variations.

The results of the “hot set” drift for the three data sets appear in Figures 3.9-3.11. Several general characteristics can be identified. Ignoring the obvious non-stationarities because of “week-end” effects, the relative drift in the “hot set” appears to be fairly constant for a particular value of “M”. As the size of the “hot set” increases, the fraction of documents common decreases. This is also observed for the absolute drift in the “hot set”. The gradual absolute drift of the “hot set” reflects the enduring popularity of some documents. The fractional overlap of documents decreases as the size of the “hot set” increases. This implies that a small but significant number of documents have enduring popularity.

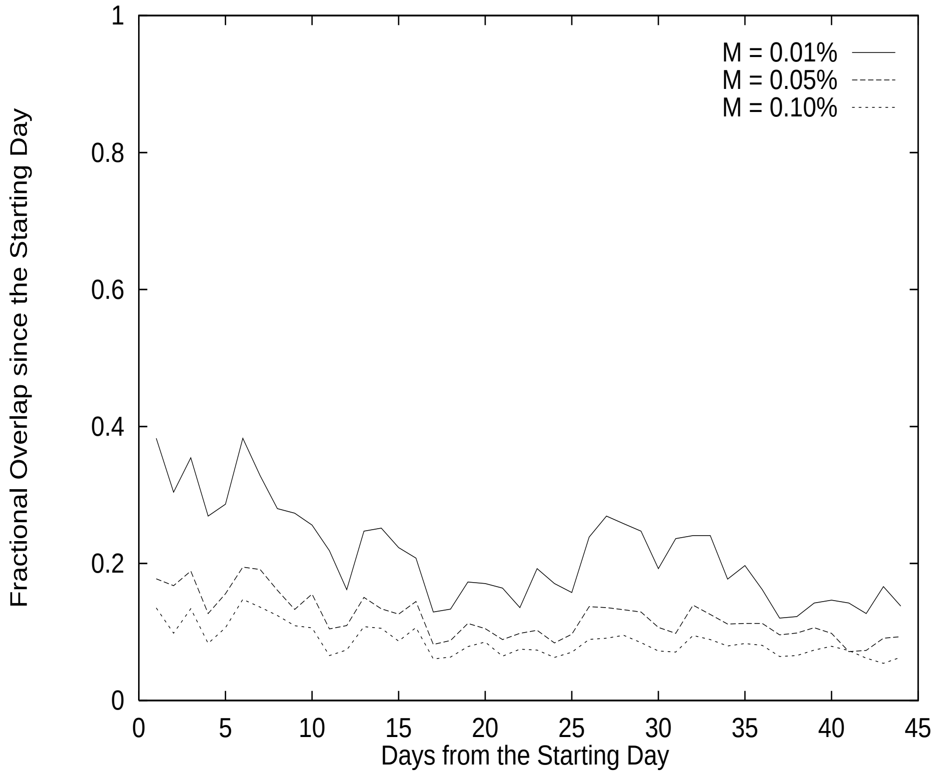
It can also be observed that the fractional overlap since the start date decreases more rapidly for the NLANR data set compared to the USask or CANARIE data set. A possible explanation for this behaviour is that references to “hot” documents get captured at the lower-level caches, causing quicker “hot set” decay at the higher-level.



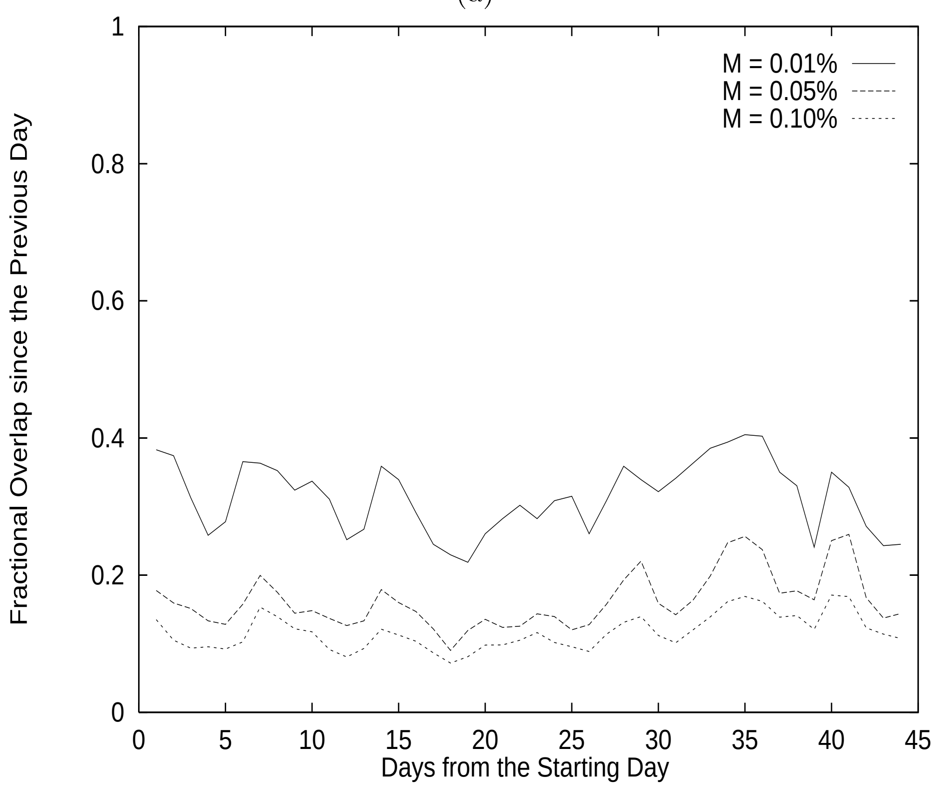
(a)

(b)

Figure 3.9: "Hot Set" Drift for the USask Data Set: (a) Absolute Drift (b) Relative Drift

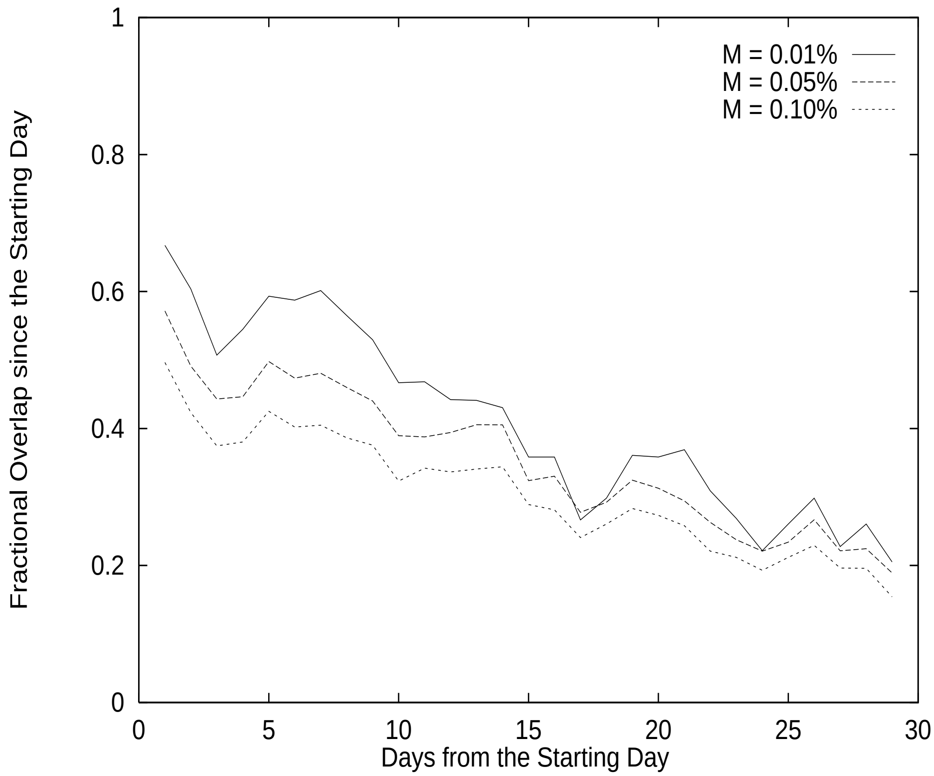


(a)

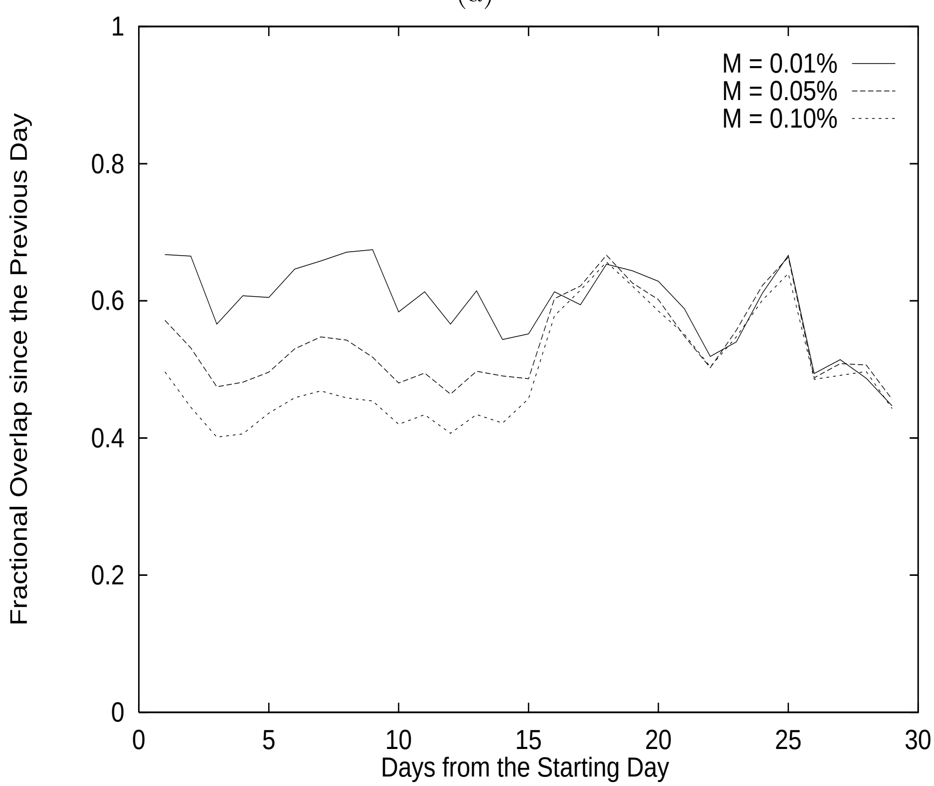


(b)

Figure 3.10: "Hot Set" Drift for the CANARIE Data Set: (a) Absolute Drift (b) Relative Drift



(a)



(b)

Figure 3.11: "Hot Set" Drift for the NLANR Data Set: (a) Absolute Drift (b) Relative Drift

3.2.8.2 Short-Term Measure of Temporal Locality

The *Least Recently Used Stack Model* (LRUSM) [77] has been widely used [2, 5, 86, 87] to measure temporal locality. The LRUSM is a stack-based ordering of referenced objects, according to their recency of reference (i.e., most recently referenced item on top (position 1), and the least recently referenced item on the bottom). For each reference in the request stream, the stack is searched from the top to the bottom until the requested object is found, or the bottom of the stack is reached. If found (i.e., a hit), the object is moved from its present position in the stack (say, d) to the top of the stack, pushing the other $d-1$ items that used to be above it down one position. For an item that is not found in the stack (i.e., a miss), it is simply added to the top of the stack, pushing all other stack items down one position. Of interest is the probability of referencing a particular position in the stack (and not a particular document).

The main drawback of the LRUSM model is its inability to distinguish "hot set" effects from temporal locality. That is, a "hot" document will cause many references to near the top of the stack, even if there is no correlation between the probability of referencing a particular document and the time since the document was last referenced.

To address this drawback, a quantitative measure of temporal locality (T_{ji}) for a stack depth i and a particular document D_j that separates "hot set" effects from temporal locality is developed. This measure is defined as follows:

$$T_{ji} = \frac{P\left(Reference\,to\,D_j \mid D_j\;is\;at\;stack\;depth\;i\right)}{P\left(Reference\,to\,D_j\right)}$$

This is a measure of short-term temporal locality. Note that T itself is not a probability; rather, it is an unbounded unit-less ratio of two probabilities. Values of T greater than 1 at low stack depths suggest the presence of temporal locality in the referencing characteristics; lower values of T do not.

The temporal locality measure T was computed for the 25 most popular documents in each of the traces. The top 50 positions in the LRU stack were considered and grouped into buckets of size 5 (i.e., stack depth 1-5 is bucket 1, stack depth 6-10 is bucket 2, and so on). Simple tests showed that a bucket size of 5 was adequate to filter out noise.

Tables 3.11-3.13 show the values of T for the top 25 documents in each of the data sets. The results indicate that there is considerable temporal locality in the referencing behaviour for some (but not all) of the most popular documents (e.g., documents 1-5, 9, and 11-14 in the USask data set). The referencing pattern of a document causes its T measure to increase or decrease at particular stack positions. For example, the T measure for document 1 in the USask data set increases as you move down the stack. This implies that the document does not stay for a long duration at the lower stack positions (beyond 50 documents) without being referenced again. In other words, references to this document are closely spaced in time. The cumulative document inter-reference distribution for this document reveals that 77% of the re-references occurred at an inter-reference distance less than or equal to 50 documents, while 95% of the re-references happen at an inter-reference distance of less than or equal to 100 documents. This type of referencing behaviour can be caused by documents that have associated pages (e.g., embedded images). On the other hand, document 2 exhibits a decrease in the T measure as the stack is traversed. This implies that documents stay well-down in the LRU stack for a long duration before being re-referenced. This is confirmed by the cumulative document inter-reference distribution, which shows that 33% of the total re-references occur at distances up to 50 documents, 39% of the total re-references occur at distances up to 100 documents, and 82% of the total re-references occur at distances up to 1000 documents. Similar observations can be made for the other two data sets.

3.3 Summary

This chapter presented a workload characterisation study for Web proxy servers. Three different Web proxies were studied: the Web proxy cache at the University of Saskatchewan; the CANARIE Web proxy cache; and an NLANR Web proxy cache

Table 3.11: Short-Term Temporal Locality Measure for Top 25 Documents (USask Data Set)

Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	8.90	9.75	11.14	12.51	14.09	14.09	14.82	15.49	15.72	16.08
2	18.49	9.65	5.06	3.43	2.50	1.79	1.31	1.11	1.04	1.09
3	6.79	5.87	5.37	4.52	4.00	3.93	3.66	3.22	3.41	3.28
4	99.76	34.68	16.49	11.25	7.72	5.76	4.93	4.10	3.62	3.95
5	1.90	1.18	0.96	1.03	1.06	1.20	1.11	1.31	1.16	1.02
6	0.98	0.98	1.12	1.48	1.45	1.44	1.45	1.49	1.52	1.51
7	0.78	1.01	1.27	1.56	1.59	1.41	1.55	1.33	1.53	1.34
8	0.93	0.95	1.19	1.43	1.40	1.56	1.51	1.55	1.47	1.37
9	2.25	1.83	1.83	1.55	1.54	1.53	1.15	1.28	1.18	1.23
10	1.38	1.38	1.29	1.43	1.43	1.22	1.22	1.28	1.11	1.09
11	2.23	1.63	1.72	1.64	1.58	1.71	1.84	1.70	1.69	1.84
12	2.86	2.09	2.04	2.04	1.83	1.98	1.94	2.04	2.33	2.26
13	46.01	11.78	7.14	4.70	4.05	3.11	3.00	2.86	2.62	2.18
14	3.64	2.24	2.13	1.89	1.97	1.97	1.97	1.89	1.85	1.77
15	0.78	1.06	0.98	1.04	1.28	1.22	1.00	1.07	1.22	1.05
16	1.01	0.98	1.12	1.16	1.04	1.14	1.04	1.19	1.24	1.11
17	0.85	0.97	1.04	1.33	1.61	1.69	1.46	1.43	1.46	1.46
18	1.02	0.79	1.15	1.47	1.56	1.46	1.45	1.36	1.72	1.69
19	0.84	0.78	1.13	1.74	1.54	1.57	1.49	1.41	1.54	1.60
20	0.88	0.89	1.02	1.54	1.41	1.65	1.53	1.44	1.62	1.58
21	1.11	1.00	1.17	1.25	1.45	1.46	1.53	1.42	1.62	1.81
22	0.76	1.01	1.17	1.65	1.46	1.58	1.52	1.41	1.59	1.64
23	0.69	0.99	1.34	1.68	1.49	1.64	1.54	1.34	1.40	1.54
24	0.75	0.90	1.22	1.78	1.44	1.61	1.36	1.50	1.60	1.44
25	0.96	1.03	1.21	1.45	1.57	1.55	1.54	1.37	1.42	1.47

Table 3.12: Short-Term Temporal Locality Measure for Top 25 Documents (CANARIE Data Set)

Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	2.95	2.19	2.05	2.00	1.85	1.94	1.70	1.87	1.88	1.89
2	2.81	1.77	1.76	1.69	1.40	1.54	1.31	1.43	1.38	1.19
3	14.74	21.57	32.19	37.13	41.34	44.09	43.00	49.74	46.84	46.62
4	1.80	2.00	1.71	1.69	1.33	1.94	1.38	1.74	1.20	1.55
5	1.82	1.70	1.16	1.42	1.80	1.69	1.49	1.28	1.33	1.33
6	6.34	4.52	4.41	4.02	4.23	2.92	4.42	3.32	3.97	3.40
7	6.61	2.13	1.96	1.89	2.25	1.82	2.69	1.40	2.72	1.86
8	14.33	10.20	10.23	8.39	8.83	8.28	9.20	8.92	8.74	8.52
9	8.49	4.46	3.73	4.14	3.64	2.62	3.17	3.32	1.47	1.47
10	13.60	11.19	11.60	10.30	9.67	8.66	12.23	8.77	10.53	9.96
11	7.79	6.55	4.48	7.51	6.29	5.47	4.62	6.26	5.18	5.22
12	12.56	10.64	9.75	9.59	9.21	12.07	8.09	11.35	14.71	13.58
13	117.56	79.06	78.89	92.19	88.51	78.66	76.74	77.54	64.56	67.78
14	113.18	4.31	1.95	1.96	1.18	0.39	1.16	1.18	0.39	1.56
15	21.26	6.59	8.39	4.93	3.90	3.92	2.87	3.96	1.44	2.16
16	27.41	57.56	54.47	48.04	50.45	64.89	52.65	51.40	55.37	52.87
17	1.37	2.06	3.42	3.09	2.41	2.42	2.42	2.42	4.52	2.44
18	1.73	3.10	2.41	2.41	2.42	2.77	1.74	2.43	4.87	2.80
19	13.75	4.87	4.49	3.76	2.27	3.02	4.17	1.90	1.52	2.28
20	492.30	60.31	44.45	56.68	62.64	63.69	61.74	41.34	38.84	58.27
21	29.25	23.86	32.37	28.51	33.03	28.35	27.82	28.20	26.59	26.88
22	4.75	4.75	3.34	6.23	2.88	2.40	5.32	6.80	3.42	2.94
23	12.63	3.54	2.03	6.61	3.07	2.05	3.09	1.03	1.54	1.55
24	0.50	0.50	1.00	0.00	0.50	0.00	0.00	0.50	0.50	0.00
25	15.65	6.24	5.25	3.68	5.82	1.60	4.25	3.75	2.69	3.22

Table 3.13: Short-Term Temporal Locality Measure for Top 25 Documents (NLANR Data Set)

Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	3.99	2.04	2.09	1.94	2.08	2.11	1.93	2.21	2.03	2.03
2	4.87	3.22	3.04	2.71	2.71	2.63	2.56	2.47	2.47	2.17
3	5.29	1.87	1.55	1.63	1.60	1.52	1.48	1.57	1.47	1.44
4	1.94	1.50	1.60	1.48	1.54	1.50	1.65	1.58	1.66	1.56
5	1.90	1.44	1.60	1.48	1.50	1.56	1.48	1.54	1.54	1.61
6	0.03	0.77	3.82	10.14	17.90	25.88	32.21	37.49	38.21	37.23
7	4.81	2.97	3.20	3.19	3.33	3.44	3.41	3.53	3.29	3.37
8	3.02	2.23	1.93	2.32	2.03	2.28	2.47	2.00	2.41	2.16
9	10.31	8.62	7.67	7.26	7.32	6.69	7.39	6.96	7.22	6.65
10	1.92	1.11	1.18	1.27	1.26	1.33	1.07	1.31	1.12	1.13
11	3.39	2.45	1.74	1.80	1.73	1.85	1.38	2.10	1.91	1.60
12	18.39	8.81	8.33	6.96	6.94	7.11	6.37	6.81	6.46	6.85
13	2.79	2.20	2.30	1.94	2.02	1.90	2.06	2.13	2.33	2.09
14	40.22	10.05	5.86	4.40	3.58	3.41	3.03	3.52	3.41	2.96
15	8.09	4.04	3.31	2.97	3.69	3.07	3.61	3.19	3.40	3.50
16	8.46	4.78	5.78	5.04	4.80	4.94	5.20	4.81	4.64	4.66
17	4.10	2.41	2.47	2.75	3.23	3.14	2.41	2.81	3.04	3.13
18	37.39	13.73	10.01	8.53	7.14	6.14	6.10	5.47	5.38	4.46
19	11.10	7.61	7.14	6.21	6.73	6.14	6.66	6.10	6.51	6.80
20	5.11	2.72	2.76	2.76	3.08	3.08	3.39	2.60	3.49	3.43
21	4.97	3.55	3.57	3.87	3.81	3.75	3.66	3.12	2.92	2.94
22	2.29	1.86	1.72	1.30	1.30	1.55	1.30	1.74	1.60	1.46
22	2.29	1.86	1.72	1.30	1.30	1.55	1.30	1.74	1.60	1.46
23	4.39	3.07	2.09	1.82	1.31	1.79	1.79	1.48	1.88	1.65
24	3.60	2.42	3.13	2.69	2.87	2.34	2.77	2.54	2.12	3.07
25	17.59	13.32	11.98	13.20	12.76	13.08	12.42	14.36	12.44	12.43

at the University of Illinois, Urbana-Champaign. These Web proxies reflect the progression from institutional-level Web proxy cache to a top level Web cache.

The main observations of the workload characterisation study are:

- Approximately 90% of the requests made to the proxy result in the client successfully obtaining the document.

- HTML and Image documents account for 95% of the total requests. Image type documents are consistently the most requested document type, followed by HTML documents.

- Most Web document transfers are small. The mean document transfer size is 7-15 KB, while the median transfer size is 2-3 KB.

- One-timers account for approximately 70% of the documents referenced and between 17-45% of the total requests seen. The percentage of one-timers in Web proxy workloads is higher than that reported for Web server workloads.

- Transfer size distributions are heavy-tailed. The tail of the distribution is heavier at the higher-level proxies than at lower-level proxies.

- The popularity of Web documents does *not* strictly follow Zipf's law as has been reported in Web server workloads, but it does follow a Zipf-like referencing distribution.

- Concentration of references is lower at Web proxy servers that has been reported for Web servers. 30% of the documents account for 60-80% of the references, while 10% of the documents account for about 90% of the bytes transferred. The concentration of references is higher at lower-level Web proxies than at higher-level proxies.

- There appears to be no direct correlation between a document's modification rate and its popularity. The rate of change of documents is higher at higher level proxies.

- The distribution of inter-access times (over all documents) decays rapidly with time. A significant fraction of the re-references occurs within a short interval of time. Periodicity in access patterns is also observed.

- A gradual drift in the "hot set" is observed, indicating the presence of some documents with enduring popularity.

- A short-term measure of temporal locality is developed that can distinguish "hot set" effects from temporal locality effects much more effectively than the LRU Stack Model. Analysis of Web proxy logs suggests that popular documents exhibit temporal locality.

Chapter 4

Impact of Temporal Locality on Web Proxy Workloads

The presence of temporal locality in computer systems workloads, and its impact on (for example) memory management policies, has been well-studied, and is fairly well-understood [18, 31, 32]. In the Web context, however, the presence and importance of locality is still the subject of some debate [2, 5, 20, 72].

From the workload characterisation study in Chapter 3, it is evident that temporal locality does exist in Web references, at least for popular documents. This observation raises an important question: "Does temporal locality really matter in Web proxy cache performance?". If temporal locality is unimportant for Web proxy cache performance, then generating synthetic traces would be rather straightforward. Otherwise, synthetic workload generation needs to incorporate temporal locality in the synthetic traces. These issues are central to this chapter.

The remainder of this chapter is organised as follows. Section 4.1 describes the methodology adopted for addressing this question. Section 4.2 describes the essential elements of simulation design, such as, the simulation model, performance metrics, factors and levels considered in the experiments and validation of the simulation model. Section 4.3 presents the simulation results aimed at understanding the impact of temporal locality on Web cache replacement policies. Section 4.4 describes a technique that introduces temporal locality in synthetic workloads. This is followed by a summary of the chapter in Section 4.5.

4.1 Methodology for Generating Synthetic Workloads

To determine the impact of temporal locality on the cache performance achieved by different replacement policies, trace-driven caching simulations using empirical and synthetic traces are carried out. The empirical traces are derived from the Web proxy access logs used for the workload characterisation study in Chapter 3. The empirical trace is a time-ordered two-column file identifying all static documents referenced along with the associated transfer size[1], for a particular duration of activity seen at the Web proxy.

The following sections are organised as follows. Section 4.1.1 describes the basic model for the synthetic traces. Section 4.1.2 describes the synthetic workload generation process. This is followed by the validation of the synthetic traces in Section 4.1.3.

4.1.1 The Independent Reference Model

Consider the document references seen at a Web proxy to consititute a n-document set $N = \{1, 2, ..., n\}$. Then, the referencing behaviour observed at a Web proxy can be modelled as a reference stream $D_1, D_2, ..., D_t, ...$, with each document D_t being a document in the set N (i.e., if $D_t = i$, then the t^{th} reference is to the document i).

The synthetic traces used in this study are based on the *Independent Reference Model* (IRM) [31, 32]. This model considers the above-described document reference stream to be an unending sequence of independent random variables with stationary probabilities:

$$P\left[D_t = i\right] = \lambda_i, 1 \leq i \leq n, t > 0$$

[1]Since individual document sizes are not recorded in the access logs, the transfer sizes are used as document sizes in the simulation.

where the probability of referencing document i is:

$$\lambda_i = \lim_{x \to \infty} \left[\frac{no.\ of\ references\ to\ document\ i\ in\ D_1 ... D_x}{x} \right]$$

In the IRM model, the distribution of the inter-reference distance is geometric. That is, the probability $d_i(k)$ of an inter-reference distance k is [31]:

$$d_i\left(k\right) = \lambda_i (1 - \lambda_i)^{k-1}$$

The overall density function $d(k)$ (i.e., probability that another request for a document i is separated by $k - 1$ requests for documents other than document i) is:

$$d\left(k\right) = \sum_{i=1}^{n} \lambda_i d_i\left(k\right)$$

4.1.2 Synthetic Workloads

A synthetic trace generator was developed that takes an empirical trace file as input, and generates the following two synthetic traces:

- `Synthetic1`: This trace preserves the concentration characteristics of the original trace, but filters out any long-term and short-term temporal locality present in the empirical trace. This is achieved by considering the document reference stream (in the empirical trace file) to be a sequence of independent random events with stationary probabilities.

- `Synthetic2`: In addition to preserving the concentration characteristics of the original trace, this trace also preserves any long-term temporal locality observed in the original trace, but filters out short-term temporal locality present in the empirical trace. While generating this trace, the synthetic trace generator applies the IRM model on individual days (i.e., IRM is applied on a day-to-day basis but not on the entire trace).

While generating the synthetic traces, a reference is defined to be a “two-tuple” consisting of a document and its associated transfer size. This definition of a *unique* reference assumes significance in the Web context because a considerable fraction of the documents experience modifications (Section 3.2.6). The synthetic trace generator *does not* change the order in which the transfer size changes occur in the empirical traces. In this manner, the pattern of assumed document modifications is preserved. Table 4.1 summarises the empirical and the corresponding synthetic traces generated by the trace generator.

Table 4.1: Summary of the Traces used for the Simulations

Item	USask	CANARIE	NLANR
Trace Duration	45 days	45 days	30 days
Start Date	Feb 5, 1999	Feb 2, 1999	Feb 3, 1999
End Date	Mar 30, 1999	Mar 24, 1999	Mar 4, 1999
Total Requests	20,667,174	7,190,498	24,123,673
Avg Requests/Day	459,270	159,789	804,122
Total Bytes Transferred (GB)	164	83.52	318.41
Avg Bytes/Day (GB)	3.64	1.86	10.61
Distinct Documents	5,428,481	4,507,730	8,388,907
Distinct Documents/Total Reqs. (%)	26.27	62.69	34.77
One-Timer Documents	3,695,041	3,467,473	6,080,721
One-Timers/Requests (%)	17.89	48.22	25.21
One-Timers/Distinct Documents (%)	76.92	72.97	72.49
Distinct Document Size (GB)	64.54	54.60	142
Avg. Distinct Document Set Size (GB)	2.21	1.59	6.89
Total One-Timer Bytes (GB)	47	43.59	109.10
One-Timers/Distinct Bytes (%)	72.82	79.84	76.83

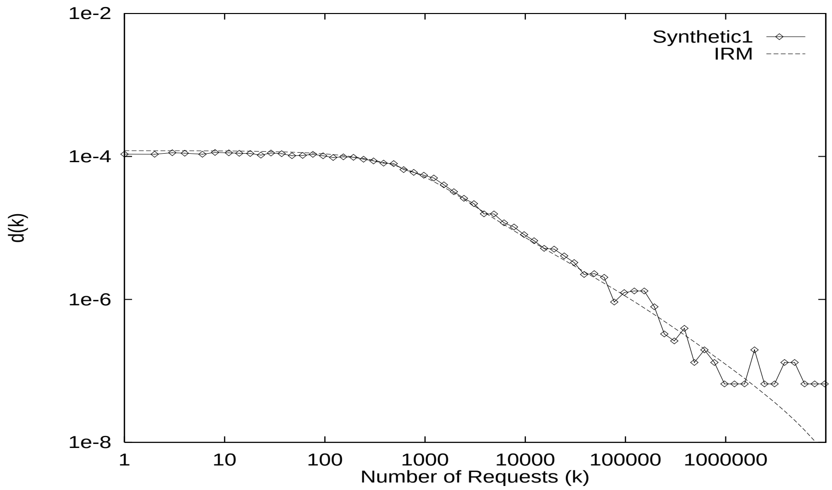
Almeida *et al.* [2] generated a synthetic workload similar to the `Synthetic1` trace to understand the applicability of the IRM model in the Web server environment. However, the approach used here is different from that in [2] in two ways. First, document modifications are explicitly modelled, since these can affect the hit ratios achieved by the cache. That is, the synthetic workload is *not* generated by applying random permutations to the original trace, as in [2]. Second, caches of fixed size (in bytes) are simulated, instead of caches with storage capacity of a fixed number of documents (as in [2]).

4.1.3 Synthetic Trace Validation

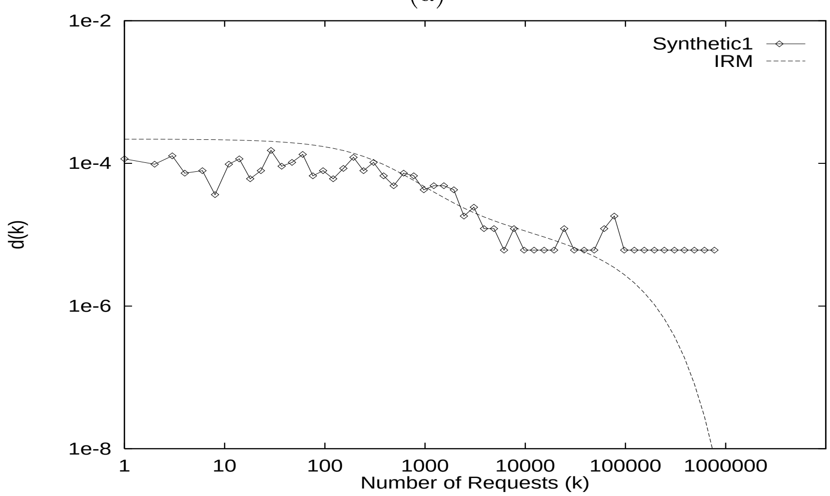
The validity of the synthetic traces was established by performing the following tests: comparing the document inter-reference distribution observed in the synthetic traces with that predicted by the IRM model, and computing the short-term temporal locality measure T for some popular documents. Note that in the asymptotic case, one-timers make no contribution to the probability density function $d(k)$ (i.e., $\lambda_i = 0$, for one-timers in the asymptotic case). Therefore, in order to reasonably approximate $d(k)$, one-timers are not considered while calculating the λ_i's.

Figure 4.1 shows that the probability density function for the document inter-reference distribution for the `Synthetic1` traces closely follows the IRM model. The document inter-reference density function for the IRM model assumes an unending sequence of random events with stationary probabilities. However, real traces are of finite length, therefore, a limit is placed on the maximum inter-reference distance. A direct artifact of finite trace duration is a "flattening" of the distribution for large inter-reference distances in the figures. Since the IRM model is applied on a day-to-day basis while generating the `Synthetic2` traces, the document inter-reference distribution of a particular day in the synthetic trace is compared with that predicted by the IRM model for the same day. Figure 4.2 shows that the probability density function for document inter-reference distribution for the first day of the USask and NLANR synthetic traces reasonably follows the IRM model. The document inter-reference distribution for the `Synthetic2` CANARIE trace does not follow the IRM model mainly because the number of requests per day is low and there is a high percentage of one-timers in the request stream.

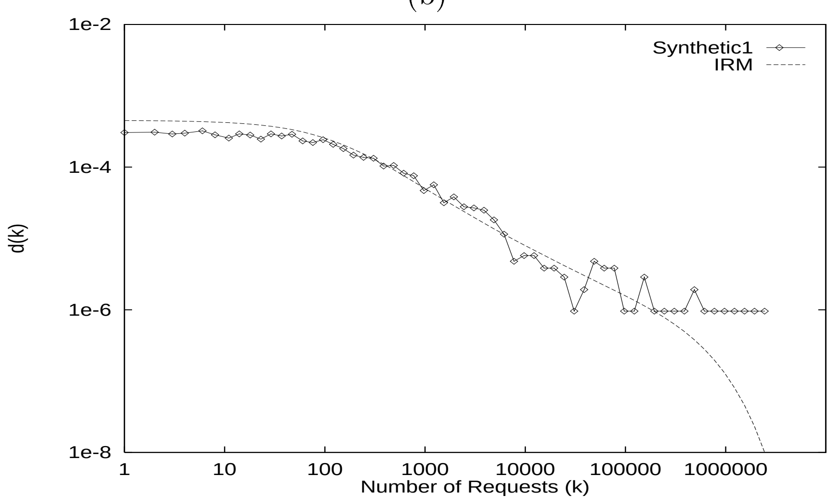
Table 4.2 shows the T values computed using the short-term temporal locality measure T for the ten most popular documents in the `Synthetic1` trace. These values are statistically close to 1.0, further confirming that temporal locality is not



(a)

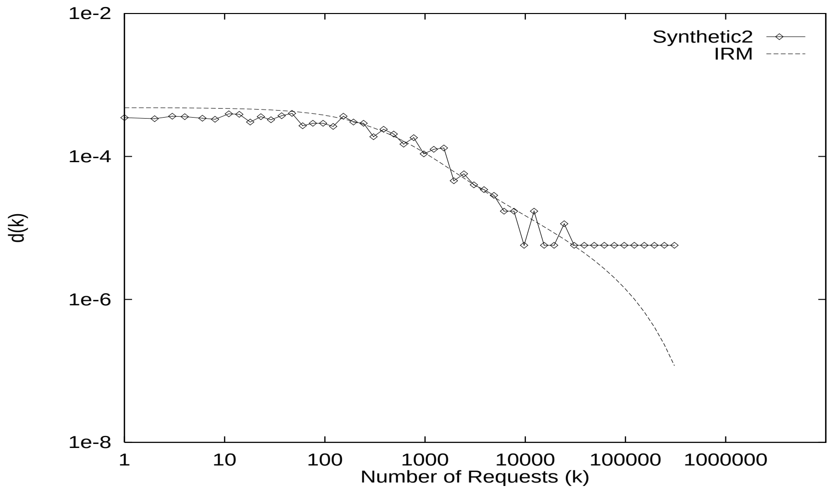


(b)

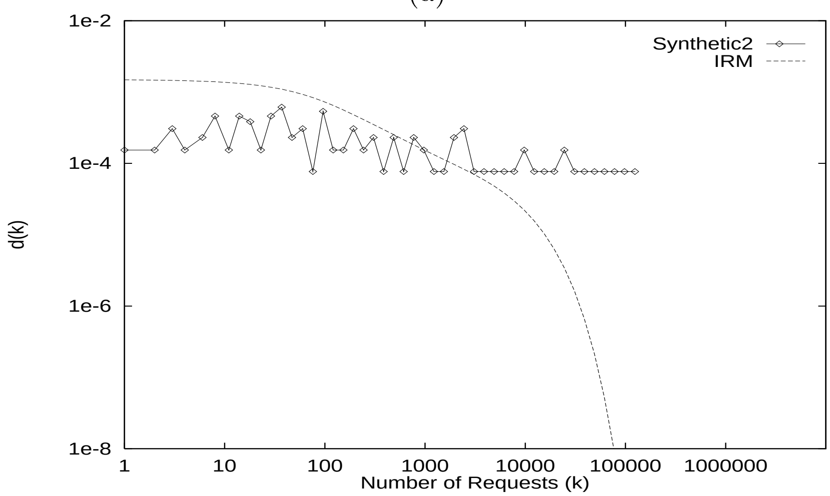


(c)

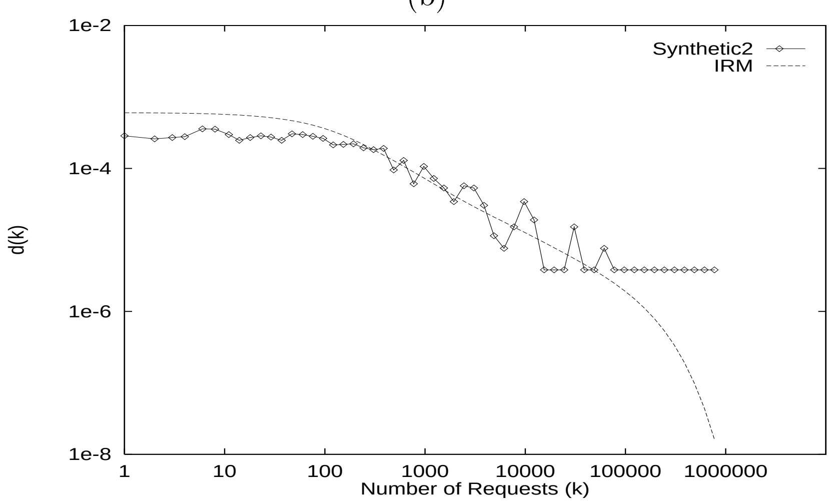
Figure 4.1: Document Inter-reference Density Function, Synthetic1 Trace versus IRM model: (a) USask; (b) CANARIE; (c) NLANR



(a)



(b)



(c)

Figure 4.2: Document Inter-reference Density Function, Synthetic2 Trace versus IRM model: (a) USask; (b) CANARIE; (c) NLANR

Table 4.2: Short-Term Temporal Locality Measure for the `Synthetic1` Trace

USask										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	0.94	1.06	0.95	1.01	1.08	0.99	0.93	1.01	1.04	0.96
2	1.12	1.01	1.10	0.99	1.00	1.01	0.95	0.96	1.03	0.98
3	0.96	0.97	1.09	1.07	1.13	1.02	1.03	0.99	0.89	0.99
4	0.97	0.92	0.97	1.11	1.05	0.88	0.92	1.07	1.04	0.96
5	0.96	1.04	1.13	1.01	0.99	0.97	1.01	1.12	1.23	1.12
6	0.90	0.89	1.08	0.89	1.14	1.23	0.87	1.01	1.07	0.92
7	1.01	0.92	0.96	0.91	0.88	0.97	0.98	1.03	0.84	1.14
8	0.95	0.97	1.06	1.01	1.07	0.93	1.01	0.92	1.15	1.04
9	1.06	1.12	1.09	1.00	1.06	1.01	0.92	1.01	0.95	1.01
10	0.95	1.07	1.06	1.10	1.08	1.14	1.08	1.00	0.92	0.78
CANARIE										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	1.04	1.02	1.04	0.95	1.01	0.98	0.99	1.03	1.01	0.92
2	0.99	1.12	1.15	1.22	0.97	1.17	0.96	1.12	0.88	1.15
3	1.13	0.95	1.09	1.15	0.94	0.97	1.16	1.06	0.88	0.81
4	1.00	1.06	1.04	1.02	0.85	1.37	0.86	0.98	0.93	0.99
5	1.12	1.21	0.55	1.22	0.80	1.36	0.90	0.51	0.73	0.73
6	1.14	0.98	0.74	0.99	0.82	0.74	0.99	1.08	1.17	1.09
7	0.51	1.36	1.45	0.60	0.69	1.12	1.12	1.04	0.95	1.48
8	1.06	0.85	1.28	0.96	1.07	0.97	0.75	1.19	1.19	1.63
9	1.14	0.76	0.89	1.27	1.27	1.02	0.64	0.90	0.77	1.03
10	0.53	1.20	0.27	1.07	0.67	0.40	1.34	1.48	1.48	0.81
NLANR										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	1.05	1.03	1.05	1.04	1.02	1.01	0.90	0.95	0.95	0.96
2	0.99	0.98	0.90	0.94	1.00	0.97	1.08	0.92	0.99	1.02
3	1.06	1.05	1.10	1.00	1.00	0.95	0.97	1.05	1.05	1.02
4	0.88	1.01	1.00	1.04	0.96	1.05	1.06	1.07	0.99	1.00
5	1.07	0.97	0.95	0.97	1.07	1.08	1.00	1.07	0.97	0.85
6	1.00	0.94	1.11	1.03	0.90	1.03	1.06	1.05	1.07	0.98
7	0.95	0.99	0.93	0.92	0.97	1.16	0.92	0.93	0.98	1.06
8	1.17	0.95	1.04	0.90	0.97	0.99	0.87	0.84	0.98	0.91
9	1.20	0.94	0.99	0.93	0.89	0.84	0.96	1.12	1.23	1.05
10	0.91	0.95	0.93	0.98	0.97	1.04	0.87	1.12	0.87	0.72

Table 4.3: Short-Term Temporal Locality Measure for the first day of the `Synthetic2` Trace

USask										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	1.20	0.74	1.21	1.20	0.83	0.96	0.82	1.29	0.55	0.86
2	1.13	1.05	0.61	0.97	0.89	1.22	0.65	0.87	0.61	0.84
3	0.95	1.18	0.78	1.05	0.75	0.98	1.38	1.13	0.97	1.46
4	1.15	1.49	0.86	0.76	0.66	1.55	1.13	0.91	1.15	0.93
5	1.21	0.89	1.23	0.91	0.69	1.04	0.94	0.71	0.95	0.48
6	0.76	0.51	1.29	1.05	1.19	0.67	1.07	1.22	0.96	0.55
7	1.96	1.69	1.72	0.32	1.27	1.45	0.49	1.31	0.49	0.83
8	0.84	1.06	0.43	1.71	0.87	1.31	0.88	0.89	0.89	2.48
9	0.65	0.87	0.44	0.87	0.22	1.32	1.34	0.90	2.05	1.38
10	0.69	1.62	1.17	0.71	1.19	1.20	1.21	2.95	1.00	1.26
CANARIE										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	0.26	0.78	1.11	0.81	0.81	1.10	1.67	0.28	1.72	0.88
2	1.69	1.28	0.86	0.87	1.32	0.45	1.79	1.83	0.46	0.46
3	0.98	0.99	1.00	0.51	0.51	1.53	0.52	1.56	0.53	0.00
4	1.85	1.25	0.00	0.63	1.90	1.28	1.30	3.97	0.00	0.00
5	0.00	2.27	0.00	0.00	0.00	0.00	0.00	4.59	2.34	2.36
6	0.00	2.56	2.57	0.00	2.61	2.64	5.32	0.00	0.00	0.00
7	0.00	0.00	2.88	2.93	0.00	0.00	0.00	0.00	5.93	0.00
8	0.00	0.00	0.00	3.14	0.00	3.19	0.00	0.00	0.00	0.00
9	3.63	0.00	3.67	3.71	0.00	0.00	0.00	0.00	3.74	0.00
10	0.00	0.00	0.00	0.00	0.00	3.90	0.00	0.00	3.93	3.96
NLANR										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	0.86	1.09	1.04	1.01	1.12	1.17	1.19	1.06	1.05	0.93
2	1.16	0.97	0.82	0.96	1.04	1.08	1.08	0.94	1.05	1.02
3	0.92	0.93	1.05	0.99	1.09	1.27	0.91	0.88	0.82	0.78
4	0.91	0.79	0.80	1.15	1.25	0.83	1.33	0.81	1.04	1.01
5	0.79	0.95	0.96	0.81	0.65	0.90	1.15	1.33	0.67	1.52
6	0.61	0.87	1.40	0.62	0.89	1.35	0.90	0.91	1.47	1.59
7	0.74	1.48	0.85	0.75	1.30	1.20	1.32	1.11	1.91	1.02
8	0.86	1.52	0.76	1.10	0.55	1.78	1.01	1.36	1.15	0.58
9	0.57	1.15	1.16	1.17	1.06	1.07	0.96	1.08	0.85	0.73
10	0.72	0.72	1.16	1.17	0.29	0.29	0.74	0.59	0.75	1.50

present in these versions of the synthetic traces. Table 4.3 tabulates the T values for the ten most popular documents in the first day of the `Synthetic2` traces. It is observed that the T values are close to one. However, statistical fluctuations are responsible for some of the popular documents showing temporal locality in the `Synthetic2` traces (e.g., document 7 in the USask data set, documents 4, 5, and 9 in the CANARIE data set).

4.2 Experimental Design

Proper experimental design is crucial in order to obtain maximum information with the minimum number of experiments. This section describes the design of the trace-driven simulation experiments. Section 4.2.1 describes the system to be simulated. Section 4.2.2 defines the performance metrics used in this study. Section 4.2.3 outlines the factors and the associated levels considered for the simulations. Section 4.2.4 briefly describes how document updates are simulated. Section 4.2.5 describes the procedure adopted for validating the simulation model.

4.2.1 Simulation Model

Trace-driven simulations have been widely used to evaluate the performance of Web proxy cache replacement policies [6, 20, 42, 65]. Such simulations incorporate a model of the Web proxy cache and its associated cache management policy. The input to such simulations is traces of Web document references. A variety of cache management strategies can be evaluated by simply altering the simulation module that manages the cache.

The simulation model used in this study is similar to the ones used by other researchers to evaluate the performance of Web proxy cache replacement policies. The simulated system operates in the following manner. The input to the system is a set of time-ordered document retrieval requests from a set of Web clients (i.e., requests that could not be served from the browser caches). Upon receiving a request, the proxy searches its cache for the requested document. A *cache hit* occurs if the requested document is served from the cache; otherwise, a *cache miss* occurs. A cache miss can occur for the following reasons: (a) the requested document is not found in the cache; and (b) the cache has a *stale* copy of the requested document in its cache (i.e., document has been modified since it was last cached). In the simulations, a cached copy of a document is defined as stale if the document size associated with the request (in the trace file) is different from the size of the document in the cache (see Section 4.2.4). On a cache miss due to document modification, the stale copy of the document is purged from the cache, and a copy of the updated document is added to the cache. In order to create sufficient space to add a new document to the cache, the removal of zero or more documents from the cache may be required. Documents larger than the specified cache size are never cached.

Typically, a proxy uses disk space for cache storage. The meta-object database, which stores information on the documents currently cached, is kept in the main memory of the system. Spare main memory is used to keep copies of frequently requested objects (so that disk accesses are reduced), and objects that are currently being fetched from remote servers. These details are important if response time is used as a metric to evaluate the performance of the proxy. For simplicity, and to permit focus on the impact of temporal locality, the simulator used in this thesis does not separately model a disk cache and a memory cache. Instead the simulator assumes a single document cache (i.e., main memory and disk cache together).

4.2.2 Performance Metrics

The most common performance metrics in previous work have been the *document hit ratio* and the *byte hit ratio*. The document hit ratio is the fraction of the total requests to the proxy that are satisfied by searching the cache for a copy of the requested document. The byte hit ratio refers to the number of bytes served to the clients from the proxy cache as a fraction of the total volume of data transferred between the proxy and its clients. These metrics are used to understand how temporal locality impacts Web cache replacement policies by comparing the hit ratios observed for the empirical and the synthetic traces.

4.2.3 Experimental Factors and Parameters

The number of factors (system and workload parameters), and the levels associated with each factor, have a great impact on the number of experiments that must be performed. Two system parameters are considered in this study: cache size and the cache replacement policy. These parameters are described in Sections 4.2.3.1 and 4.2.3.2 respectively. The workload parameters in the experiments and related issues are considered in Section 4.2.3.3.

4.2.3.1 Cache Size

The cache size indicates the amount of space available to store the documents in the cache. A total of 12 levels are considered in this study: 1 MB (megabyte), 4 MB, 16 MB, 64 MB, 256 MB, 1 GB (gigabyte), 4 GB, 8 GB, 16 GB, 32 GB, 64 GB, and 256 GB. The largest cache size of 256 GB is large enough to hold the entire document set for any of the data sets considered. Therefore, results for this size indicate performance for a cache of unbounded size.

4.2.3.2 Cache Replacement Policy

Another obvious factor is the replacement policy. The replacement policy determines which documents should be evicted from the cache when there is insufficient space for adding the most recently referenced document (i.e., a cache miss occured, and the size of the requested document is greater than the available amount of free space in the cache).

In recent years, many proxy cache replacement policies have been proposed. To keep the number of experiments manageable, only three different replacement policies are considered: LRU, LFU-Aging, and GD-Size(1) [21]. The rationale behind selecting these caching algorithms is to reflect a broad range of cache replacement approaches in a small number of policies.

The LRU policy is widely used in popular proxy caching software (e.g., `Squid`). According to this policy, documents not referenced for the longest period of time are replaced. This policy works well for workloads exhibiting good temporal locality.

The LFU-Aging policy is derived from the LFU replacement policy and is used to understand the impact of using a frequency-based replacement policy. In the LFU policy, a reference count is maintained for each document in the cache. The document with the lowest reference count is removed from the cache. In case of a tie, the least recently used document (among those with the lowest reference count) is selected for replacement. A major drawback of this policy is that some temporarily "hot" documents may build up high reference counts such that they rarely become candidates for replacement, although they are no longer popular. This problem can be addressed by introducing reference count "aging" [74]. In this approach, a limit (A_{max}) is placed on the average reference count of all documents in the cache. Once this limit is reached, the reference count for each document in the cache is reduced by a certain factor. (In this particular implementation, reference counts are reduced by a factor of two.) In this way, temporarily "hot" documents which become unpopular can eventually be flushed out of the cache.

The GD-Size [20] policy is used to understand the impact of a size-based cache replacement policy. A value H is associated with each document, which is set to $cost/size$, where $cost$ may be set according to various objectives (see below), and $size$ is the size of the document. When a document is brought into the cache, H is set to $cost/size$. If a document is to be evicted from the cache because of a cache miss, the document with the lowest H value (H_{min}) is selected, and the H values of all documents in the cache are reduced by H_{min}. Ties are resolved by selecting the least recently referenced (among those with the lowest H values) document for eviction from the cache. If a cache hit occurs, the H value is restored to its original cost.

Cao *et al.* [20] show that by simply altering the definition of $cost$, different goals can be achieved. For example, the hit ratio is maximised if $cost$ is set to 1. This version of the algorithm, referred to as GD-Size(1), is used in this study. Other values can be assigned to *cost*. For example, *cost* could be set to download latency if the objective was to minimise average latency. To keep the number of experiments limited, other versions of the GD-Size algorithm are not considered in this study.

In order to implement the GD-Size algorithm efficiently, instead of subtracting H_{min} from the H values of all documents in the cache every time a document is replaced, an inflation value L is kept, and all future values of H are offset by L. This implementation of the GD-Size algorithm is given as follows [20]:

```
Algorithm GD-Size:
c(p) :  The cost of bringing document p into the cache
s(p) :  The size of document p
M : The number of documents in the cache at a particular instant of time
Initialise L ← 0.
For each reference to a document p:
(1) if p is already in cache:
(2)          H(p) ← L + c(p)/s(p)
(3) if p is not in cache
(4)          while free cache space less than size of the document p
(5)             Let L ← min_{q∈M} H(q)
(6)             Evict a document q such that H(q) = L
(7)          Put p in cache and set H(p) ← L + c(p)/s(p)
```

4.2.3.3 Workload Parameters

The workload parameters considered in this study are the length of the "warm-up" period and the set of client requests. As outlined in Section 4.1.2, access logs from the three Web proxy sites are used to create the empirical and the synthetic trace files. In this study, the effect of varying the client population making requests to the proxy (as in [33]) is not considered. Instead, the entire client set is used for the simulation experiments.

Most cache simulation studies focus on the steady-state behaviour of the system.

The idea is to neglect the cache misses due to an empty cache, as the replacement policy has no role to play in the empty cache scenario. The period during which the cache operates in its usual manner, without collecting statistics, is known as the "warm-up" period. Since the objective is to compare cache performance with the synthetic trace and the empirical trace as input, the cache is not warmed-up. Thus, in particular, an infinite cache simulation will achieve the same hit ratios for both the synthetic and the empirical traces.

4.2.4 Simulation of Document Updates

An important aspect of proxy caching is to ensure that cached copies of documents are up-to-date. Web proxies (like `CERN` and `Squid`) typically assign a "life-time" to cached documents in cache, after which they are marked "stale". Documents may be retrieved from cache and returned to the requesting clients without any consultation with higher-level proxies or the origin server, unless they are marked "stale". When a "stale" document is requested, the proxy issues a GET request with an If-Modified-Since header to determine whether or not it still has a valid copy of the document. If the document is found to have changed, the old copy is purged from the cache and a new copy of the document is brought into the cache. To precisely implement the above described cache consistency mechanism in a simulation model, it is necessary to model the document modifications at the higher-level proxies and the origin servers. Such a model was considered to be outside the scope of the thesis. In the simulations, a cache hit occurs when the size associated with the document in the proxy cache is the same as the size of the document seen in the trace file. However, it is possible that document updates do not get reflected in the size of the document (i.e., the size of the document remains unchanged). Therefore, the simulation results here are likely to be more optimistic when compared with real proxy cache implementations.

4.2.5 Validation of the Simulator

An important step in any simulation study is to validate the simulator. To ensure that the simulator performed properly, test runs were carried out using two short traces (100 - 150 requests). One was a hand-made trace while the other was part of the empirical trace. The results of the simulator were then verified by hand. Also, obtaining the same hit ratios for infinite cache simulations with different replacement policies enhanced confidence in the simulator. Furthermore, the hit ratios reported here are similar to those obtained by other researchers [6, 21, 72].

4.3 Temporal Locality and its Impact on Web Proxy Caching Results

This section presents the simulation results of the experiments performed to understand the impact of temporal locality on cache replacement policies. Among the three replacement policies considered, the LFU-Aging policy requires parameterisation to function properly. Section 4.3.1 examines the effects of parameterisation on the performance of the LFU replacement policy. Section 4.3.2 compares the performance of the three replacement algorithms considered, and evaluates the applicability of the IRM model in the design of synthetic Web proxy reference streams.

4.3.1 Determining the Aging Parameter A_{max}

This section describes how the aging parameter A_{max}, needed in the LFU-Aging policy, was determined for the workloads under investigation. In this study, the method described in [4] was used. Different values of A_{max} were tested to determine which gave the best document hit ratio for each cache and trace; these values were then used to compare the performance of LFU-Aging with the two other replacement policies. Table 4.4 shows the hit ratios obtained with different A_{max} values for each empirical trace. The results for the USask data set indicate that the influence of the aging parameter is only significant for cache sizes in the range of 256 MB - 4 GB;

the aging parameter has little or no impact on smaller and larger caches. When the cache size is small documents are frequently evicted from the cache, preventing cache pollution. For larger caches, cache pollution is unimportant because documents are seldom evicted form the cache. The aging parameter has little or no impact for the CANARIE and the NLANR data sets because a large fraction of the total requests seen at these proxies are one-timers and many document size modifications occur, causing frequent evictions from the cache. Finally, note that the relationship between the A_{max} value and the hit ratios is not strictly monotonic. For example, in Table 4.4, at a cache size of 1 GB, $A_{max} = 10$ performs worse than $A_{max} = 5$, but better than $A_{max} = 3$, for the USask data set.

Table 4.4: Effect of A_{max} on Cache Hit Ratio (Empirical Traces)

USask										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	9.70	14.36	20.09	27.14	35.44	44.73	52.76	55.72	56.30	56.30
5	10.06	14.75	20.68	28.30	37.14	45.86	52.26	55.71	56.30	56.30
10	10.00	14.68	20.52	28.09	35.92	42.35	47.69	55.71	56.30	56.30
20	9.98	14.63	20.47	27.87	33.21	37.27	47.69	55.71	56.30	56.30
40	9.96	14.60	20.44	27.53	31.67	37.27	47.69	55.71	56.30	56.30
CANARIE										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	1.92	2.54	3.06	3.47	3.71	4.21	5.49	9.15	9.93	9.93
5	1.92	2.54	3.08	3.44	3.71	4.21	5.49	9.15	9.93	9.93
10	1.92	2.53	3.08	3.44	3.71	4.21	5.49	9.15	9.93	9.93
20	1.92	2.53	3.07	3.44	3.71	4.21	5.49	9.15	9.93	9.93
40	1.92	2.53	3.07	3.44	3.71	4.21	5.49	9.15	9.93	9.93
NLANR										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	2.30	3.58	5.57	8.04	11.19	15.20	19.47	21.92	22.79	22.86
5	2.34	3.61	5.57	7.74	9.26	12.10	19.47	21.92	22.79	22.86
10	2.36	3.61	5.55	7.64	9.02	12.10	19.47	21.92	22.79	22.86
20	2.37	3.61	5.54	7.61	9.02	12.10	19.47	21.92	22.79	22.86
40	2.37	3.61	5.54	7.60	9.02	12.10	19.47	21.92	22.79	22.86

Similar iterative tests were performed to determine suitable A_{max} values for the `Synthetic1` and `Synthetic2` traces. For the `Synthetic1` traces it was found that the relationship between the A_{max} value and the hit ratio is monotonically increasing, and the best performance is achieved with $Amax = \infty$. In the `Synthetic1` traces, documents have stationary popularity, and popular documents always remain popular. Therefore, the performance of the cache does not deteriorate because of documents building up high reference counts. Hence, the `Synthetic1` trace simulations are run with aging parameter $A_{max} = \infty$. The cache hit ratio results with different A_{max} values for the `Synthetic2` traces are given in Table 4.5.

Table 4.5: Effect of A_{max} on Cache Hit Ratio (`Synthetic2` Traces)

USask										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	4.94	8.37	13.74	21.23	31.36	43.24	52.59	55.73	56.30	56.30
5	5.53	9.13	14.99	23.12	33.31	44.10	51.99	55.71	56.30	56.30
10	5.81	9.62	16.07	25.31	34.30	42.15	48.51	55.71	56.30	56.30
20	5.87	9.72	16.29	25.44	32.11	36.77	48.51	55.71	56.30	56.30
40	5.88	9.73	16.28	25.04	30.10	36.77	48.51	55.71	56.30	56.30
CANARIE										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	0.55	0.96	1.60	2.29	3.02	4.03	5.52	9.14	9.92	9.92
5	0.59	1.01	1.64	2.28	3.02	4.03	5.52	9.14	9.92	9.92
10	0.61	1.03	1.65	2.28	3.02	4.03	5.52	9.14	9.92	9.92
20	0.62	1.03	1.64	2.28	3.02	4.03	5.52	9.14	9.92	9.92
40	0.62	1.03	1.64	2.28	3.02	4.03	5.52	9.14	9.92	9.92
NLANR										
	Cache Size									
A_{max}	1 MB	4 MB	16 MB	64 MB	256 MB	1 GB	4 GB	16 GB	64 GB	256 GB
3	0.88	1.56	3.20	5.86	9.85	14.34	19.05	21.92	22.79	22.86
5	0.96	1.63	3.39	6.04	9.34	13.84	19.05	21.92	22.79	22.86
10	1.01	1.67	3.46	6.02	8.97	13.84	19.05	21.92	22.79	22.86
20	1.02	1.68	3.47	5.99	8.97	13.84	19.05	21.92	22.79	22.86
40	1.02	1.68	3.46	5.98	8.97	13.84	19.05	21.92	22.79	22.86

4.3.2 Performance of Cache Replacement Algorithms

The cache performance of the three replacement policies considered in this study is shown in Figures 4.3-4.5. The upper graphs are for document hit ratios; the bottom graphs are for byte hit ratios.

Figure 4.3(a) shows that the maximum possible document hit ratio for the USask data set is 56.30%; the remaining 43.70% of the requests are for documents requested for the first time, or for documents that have been updated. Since compulsory misses account for 26.27% of the total requests (Table 4.1), the percentage of coherence misses should be 17.43%. Figure 4.3(b) shows that the maximum possible byte hit ratio for the USask data set is 35.93%.

The higher-level proxies achieve much lower cache hit ratios compared to the

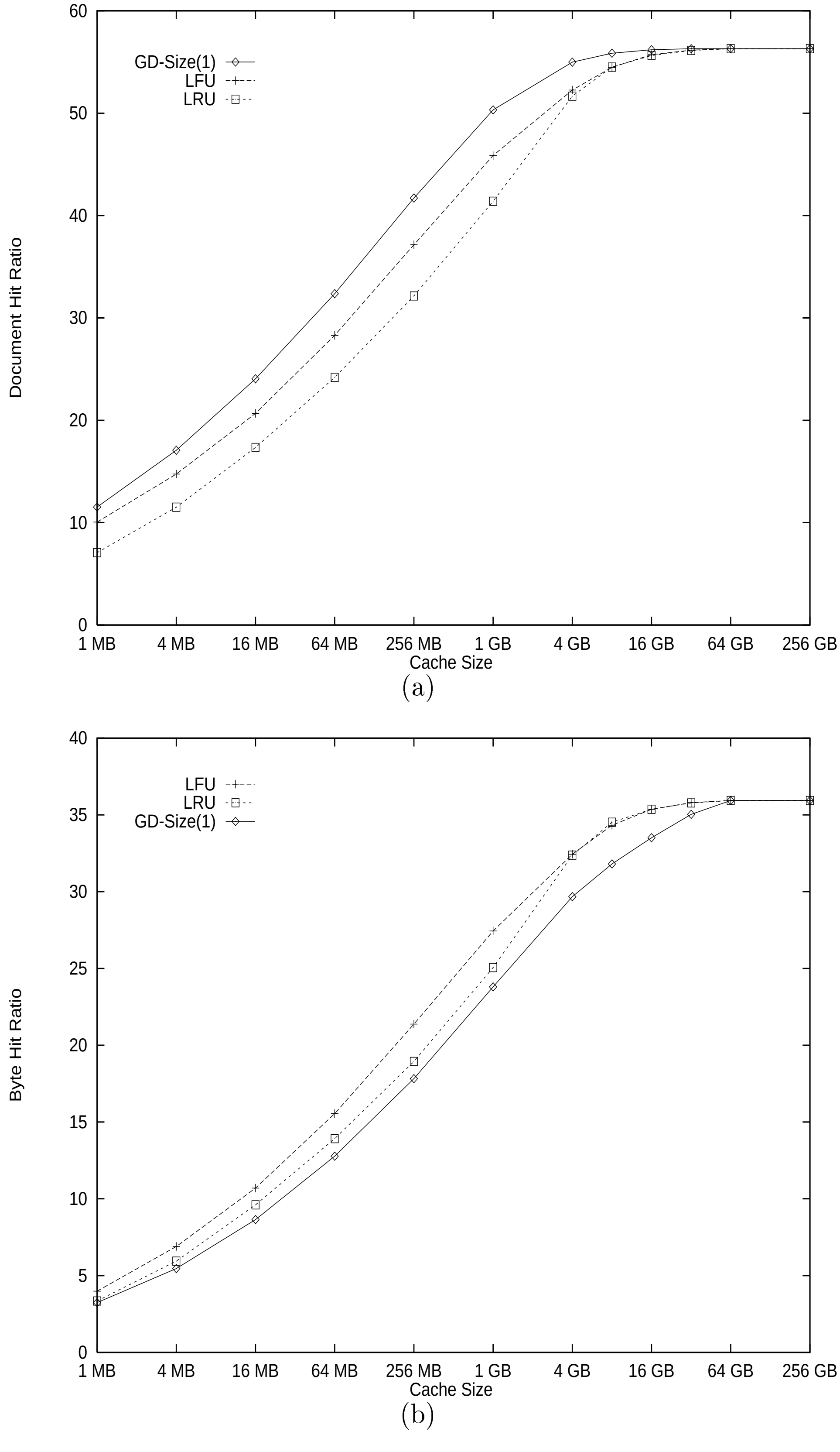


Figure 4.3: Comparison of Cache Hit Ratios for the USask data set

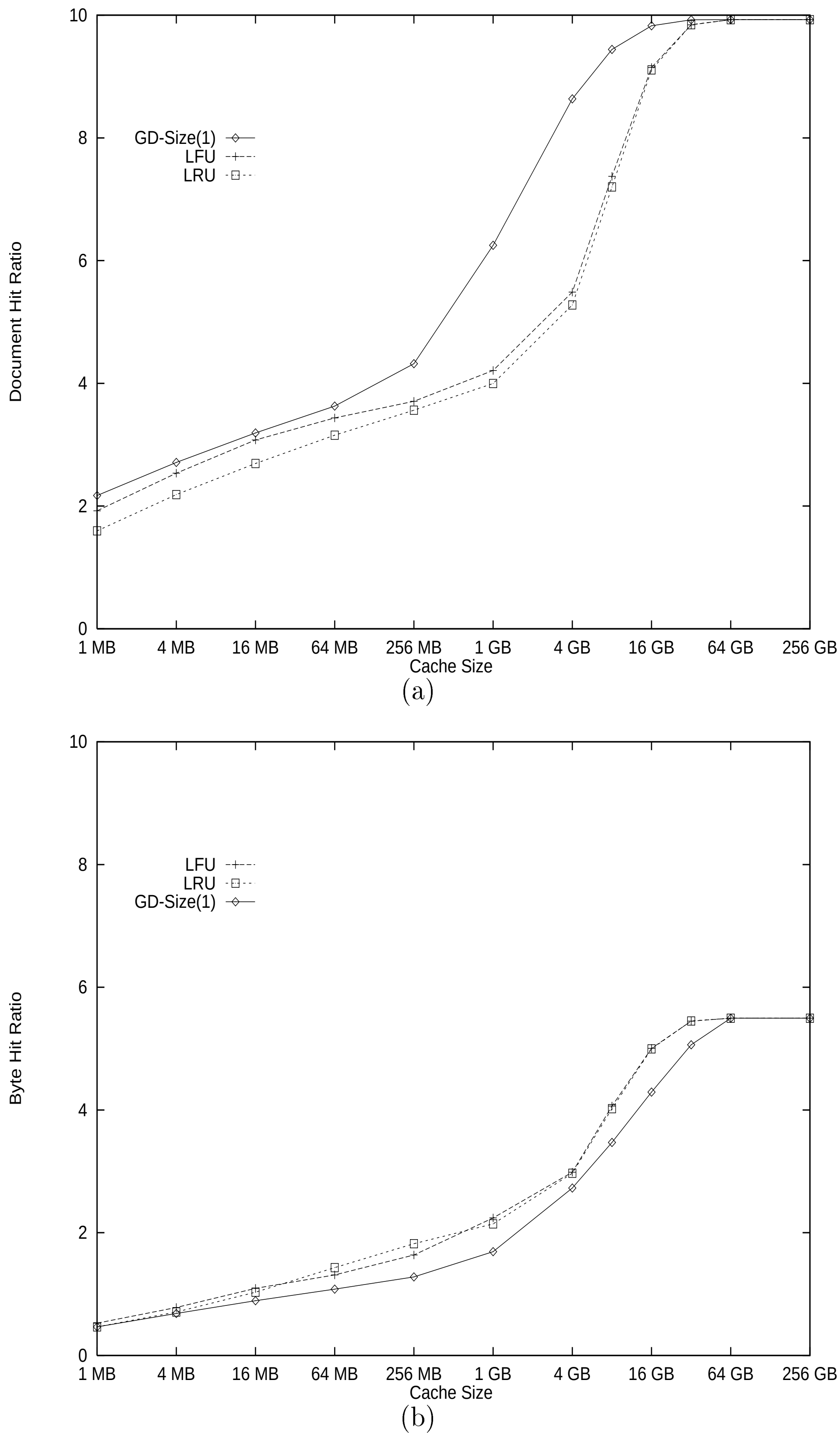
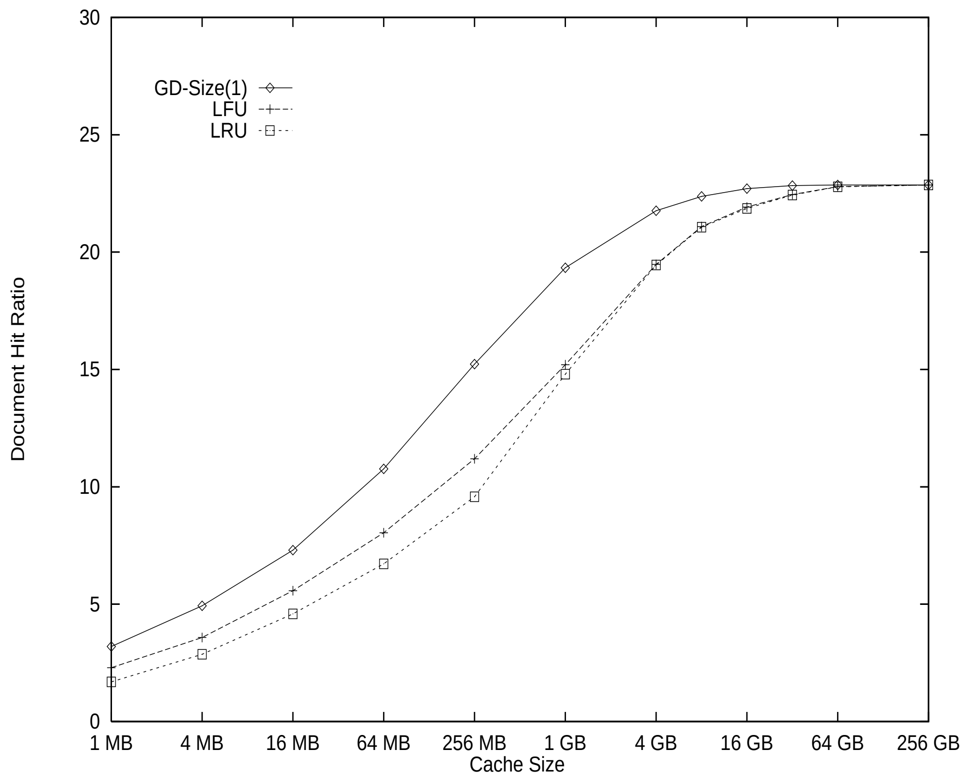
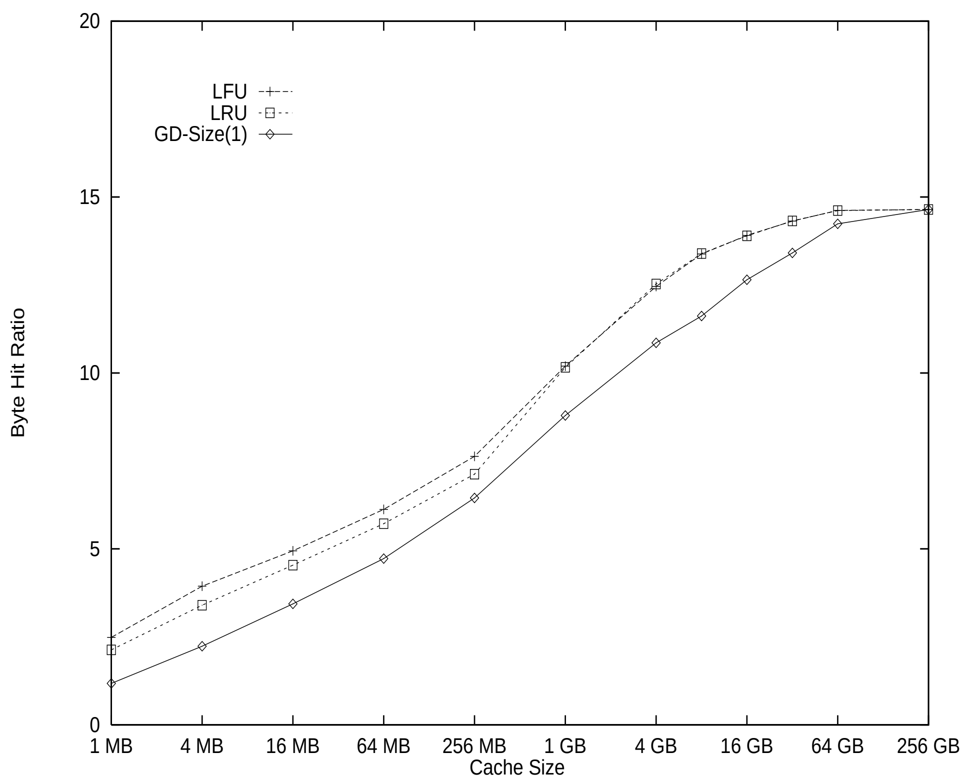


Figure 4.4: Comparison of Cache Hit Ratios for the CANARIE data set



(a)



(b)

Figure 4.5: Comparison of Cache Hit Ratios for the NLANR data set

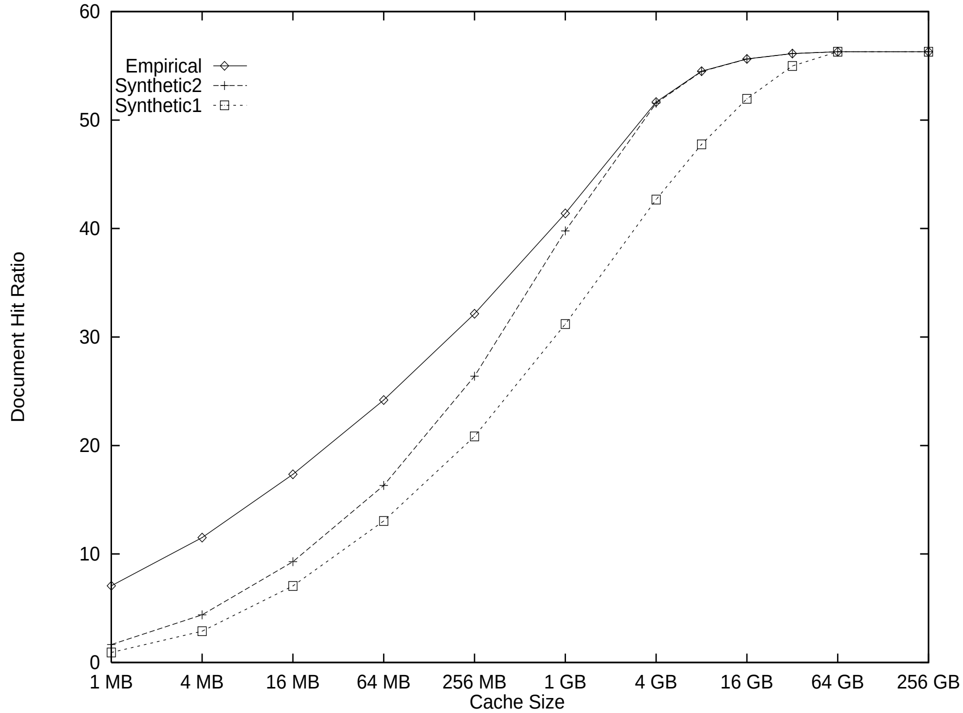
USask proxy. Figure 4.4 shows that the infinite cache document hit ratio and byte hit ratio for the CANARIE data set is 9.92% and 5.49% respectively. The low hit ratio for the CANARIE data set is due mainly to the much lower volume of requests, which causes many compulsory misses (i.e., the compulsory misses account for 62.69% of the total requests). The maximum achievable document hit ratio and byte hit ratio for the NLANR proxy are 22.86% and 14.64% respectively.

Several general trends can be seen across all the data sets. The GD-Size(1) policy achieves higher document hit ratios by favouring smaller documents over large ones, and by aging documents not referenced for a long duration of time. LFU-Aging performs better than LRU, which obtained the lowest document hit ratios among the three policies considered, by keeping frequently requested popular documents in the cache and quickly removing infrequently referenced documents. For example, the results for the USask data set in Figure 4.3 show that at a cache size of 256 MB, the document hit ratios achieved by GD-Size(1), LFU-Aging, and LRU are 41.70%, 37.14%, and 32.15% respectively (i.e., GD-Size(1) achieves 4.56% points and 9.55% points higher document hit ratios compared to LFU-Aging and LRU replacement policies respectively). It is also observed that the LFU-Aging policy maximises byte hit ratios and performs much better than both LRU and GD-Size(1) on this measure. LFU-Aging attains higher byte hit ratios because it retains popular documents in its cache for a longer time and does not discriminate against larger documents. Since GD-Size(1) discriminates against large documents, its performance is the worst among the three replacement policies considered, with respect to byte hit ratio.

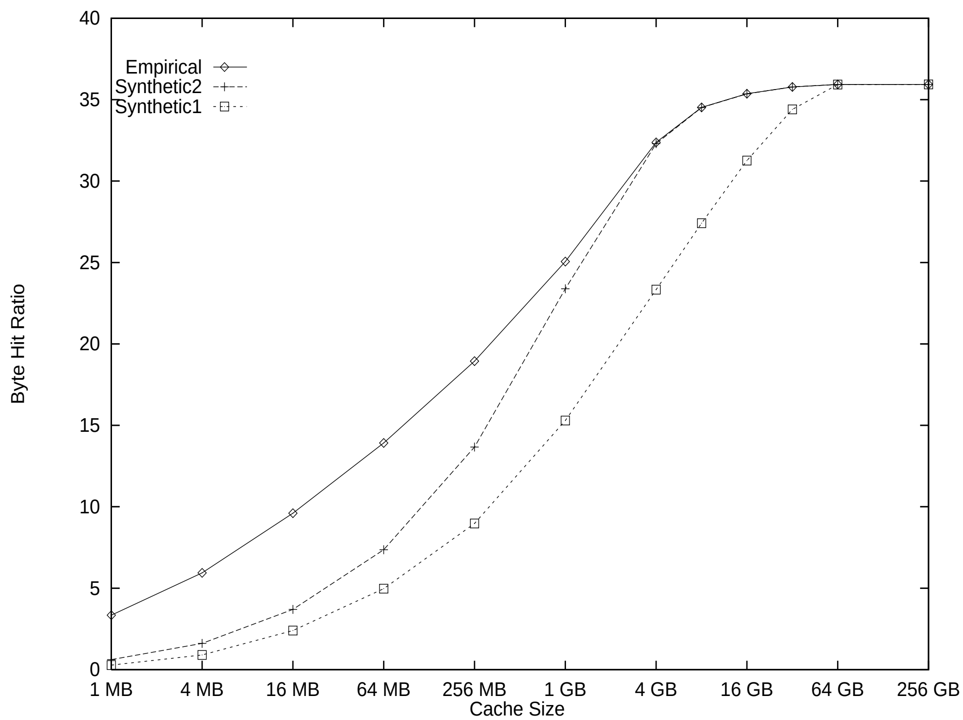
The LRU policy is known to work well when there is strong correlation between immediate past and immediate future references. Since LRU performed the worst (in terms of document hit ratio) among the replacement policies considered, perhaps temporal locality is not a major factor in Web proxy workloads. If that were the case, then it would be possible to employ very simple synthetic workload models in Web performance studies, in which document references were modelled as independent random events with stationary probabilities. This question is answered by comparing the hit ratios for the empirical traces with those for the synthetic traces. Figures 4.6- 4.14 present the simulation results for each data set. The graphs on the upper half are for document hit ratios; the graphs on the lower half are for byte hit ratios. In each figure, the y-axis of the graph is truncated to show clearly the caching performance differences between the empirical and synthetic traces. For each trace, results are presented for LRU, LFU-Aging, and GD-Size(1), in that order.

In general, it is observed that the cache hit ratios for the synthetic traces underestimate the hit ratios obtained with the empirical trace. It is also observed that as the size of the cache increases, the hit ratios obtained by the `Synthetic2` traces get closer to those obtained by the empirical trace. For example, consider the cache performance results obtained by the empirical, `Synthetic2`, and `Synthetic1` traces for the NLANR data set under the LFU-Aging cache replacement policy (see Figure 4.10) at cache sizes of 16 MB, 256 MB, and 4 GB. At a cache size of 16 MB, the document hit ratios obtained are 3.57%, 3.20%, and 2.38% respectively, at a cache size of 256 MB the document hit ratios obtained are 11.19%, 9.85%, and 7.45% respectively, while at a cache size of 4 GB, the document hit ratios are 19.47%, 19.05%, and 16.22% respectively. Note that for cache sizes close to the average unique document set size (i.e., the average total size of all unique documents accessed in a single day), the hit ratios obtained with the `Synthetic2` traces are close to those obtained for the empirical traces.

Some interesting observations can be made regarding the influence of temporal locality on the performance of the three caching policies considered. The results for the USask data set are used to illustrate these observations. First, among the three cache replacement policies considered, the LRU policy shows the greatest sensitivity to the absence of temporal locality. Second, the LFU-Aging policy is observed to be the least affected by the absence of temporal locality. For cache sizes up to 4 GB, the `Synthetic1` trace obtains hit ratios that are 5 to 6 percentage points lower than the corresponding empirical trace. Finally, the impact of temporal locality on GD-Size(1) decreases much more rapidly compared to LRU and LFU-Aging, with an increase in cache size. As an example, consider the cache performance results

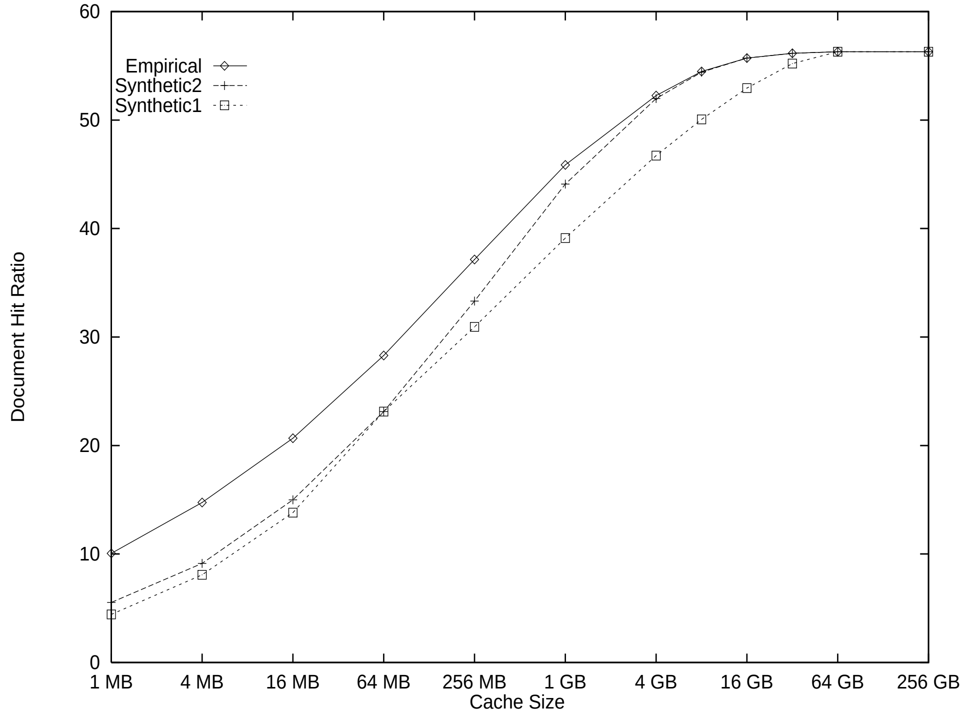


(a)

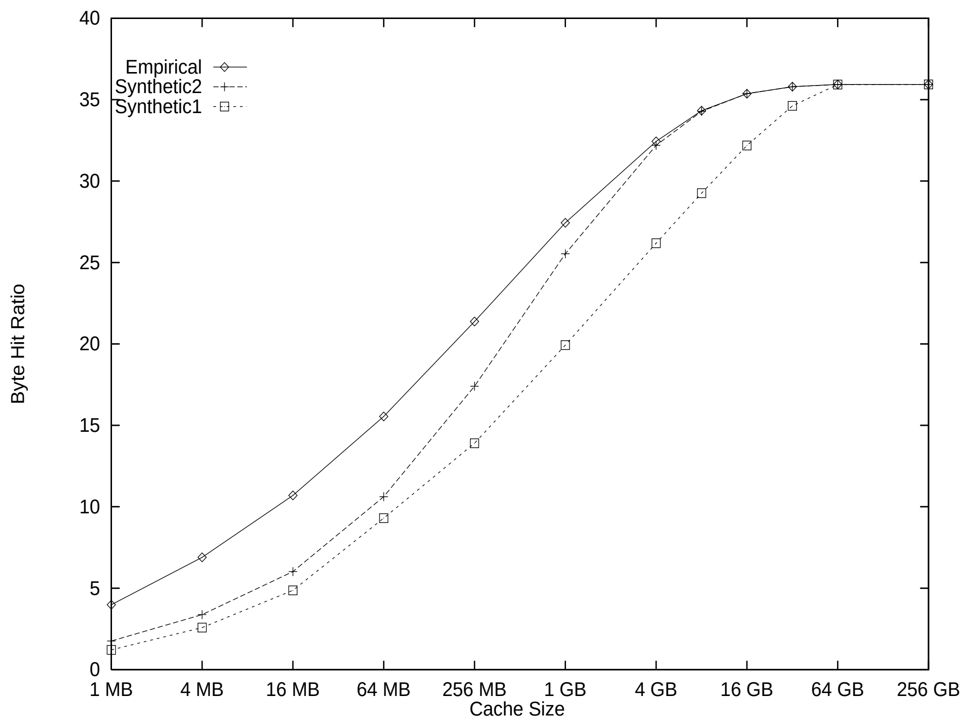


(b)

Figure 4.6: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU



(a)



(b)

Figure 4.7: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging

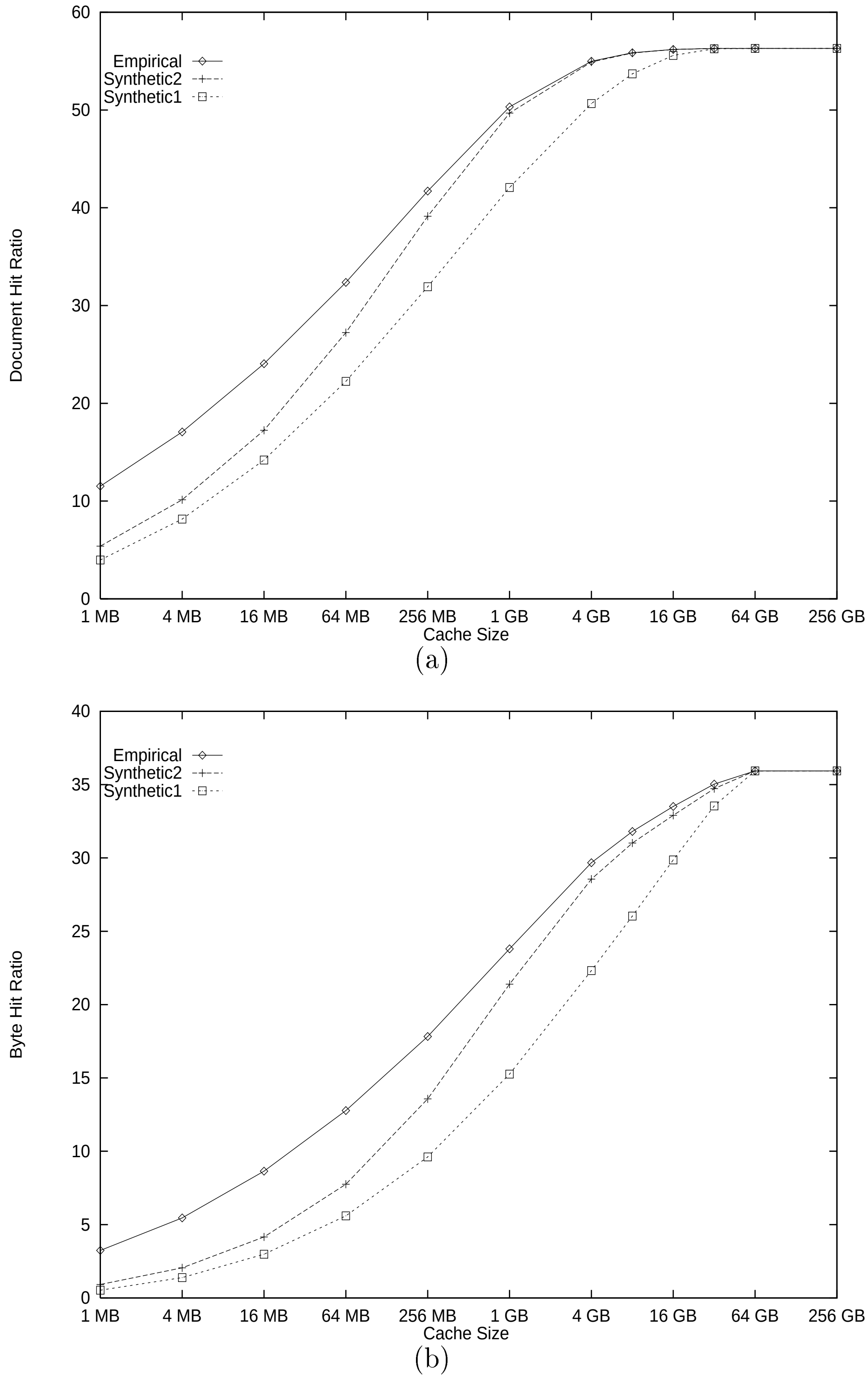
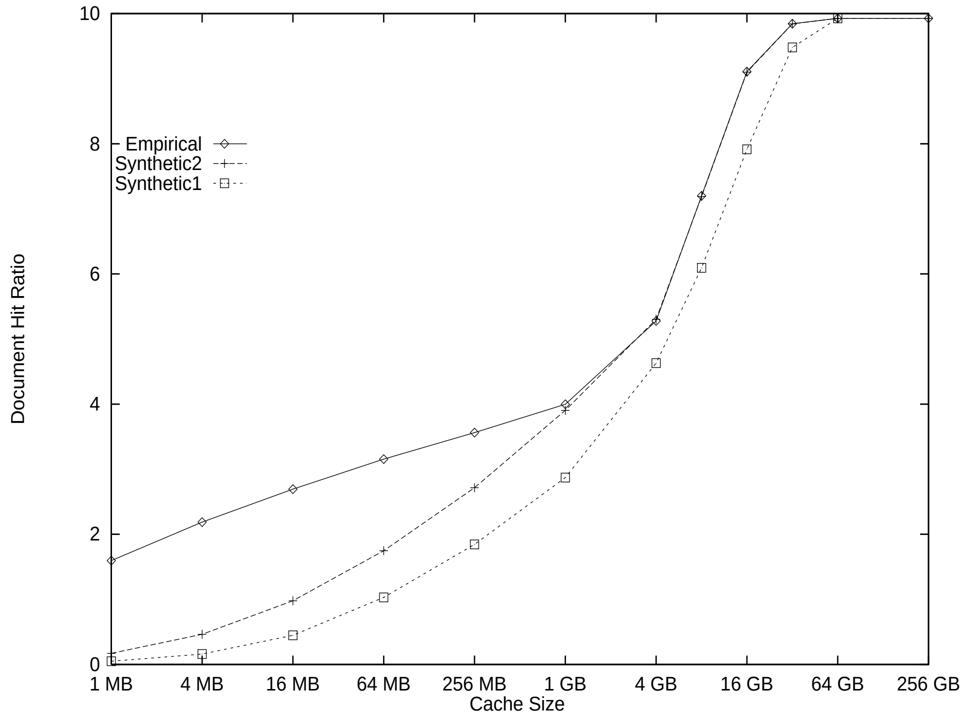
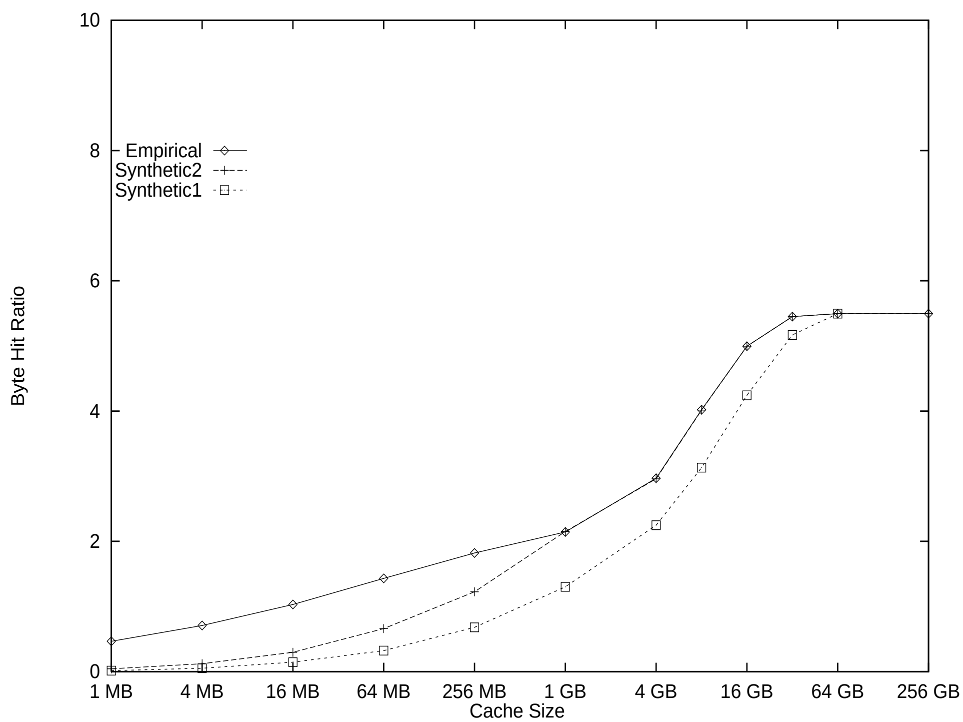


Figure 4.8: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)



(a)



(b)

Figure 4.9: Caching Performance Results for Empirical Trace versus Synthetic Trace for the CANARIE Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU

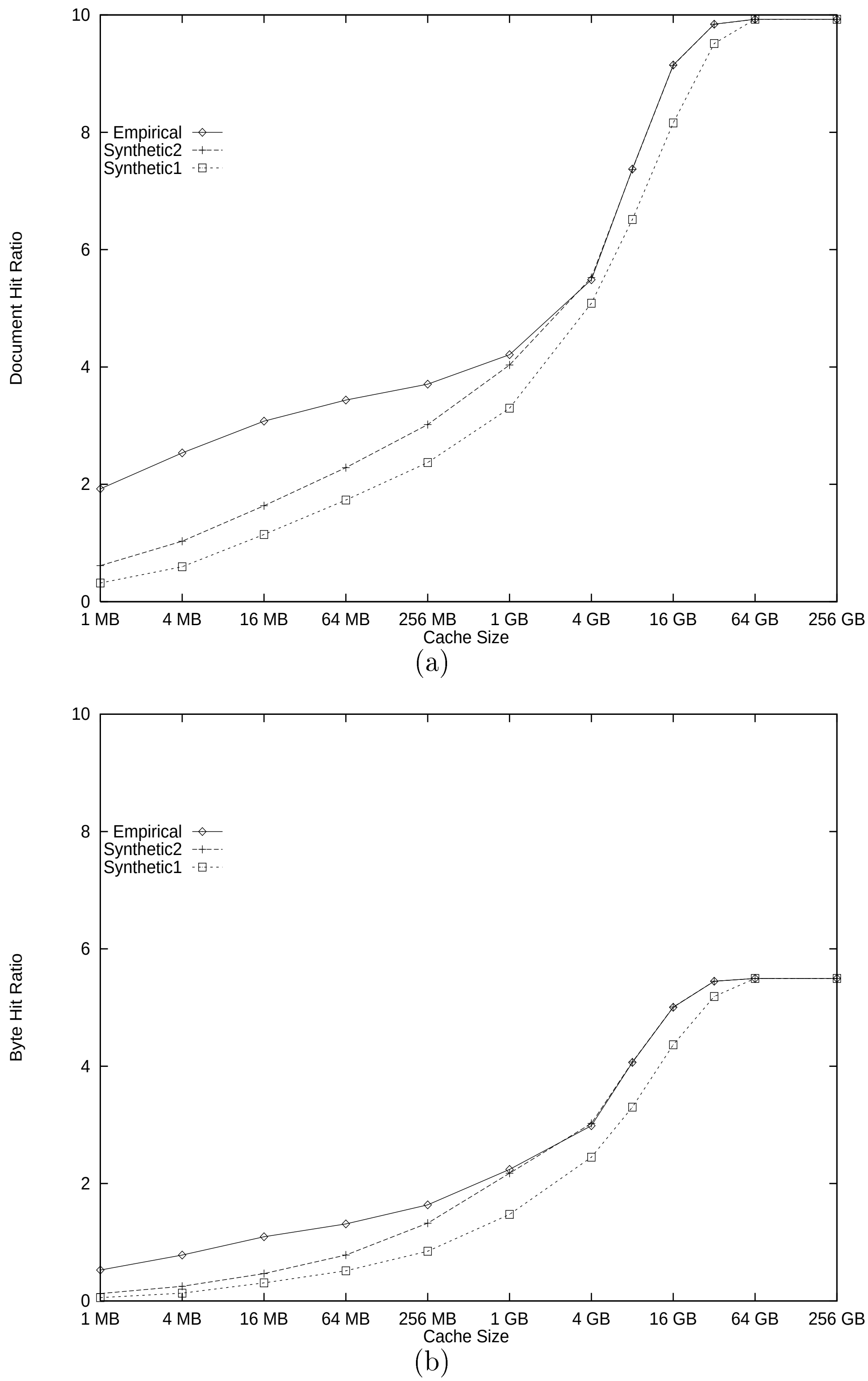
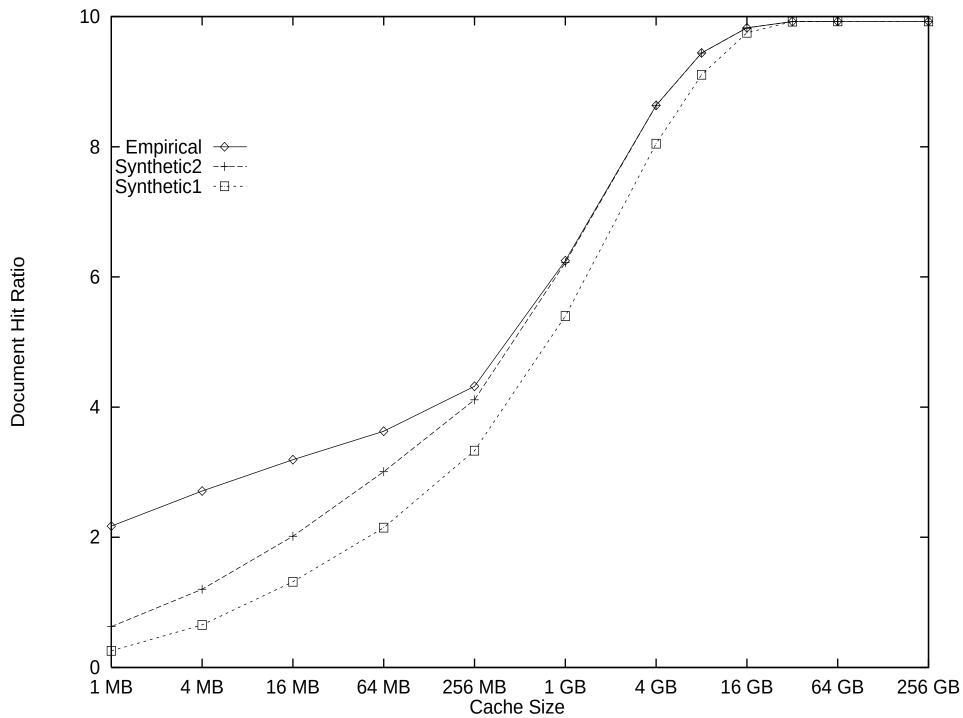
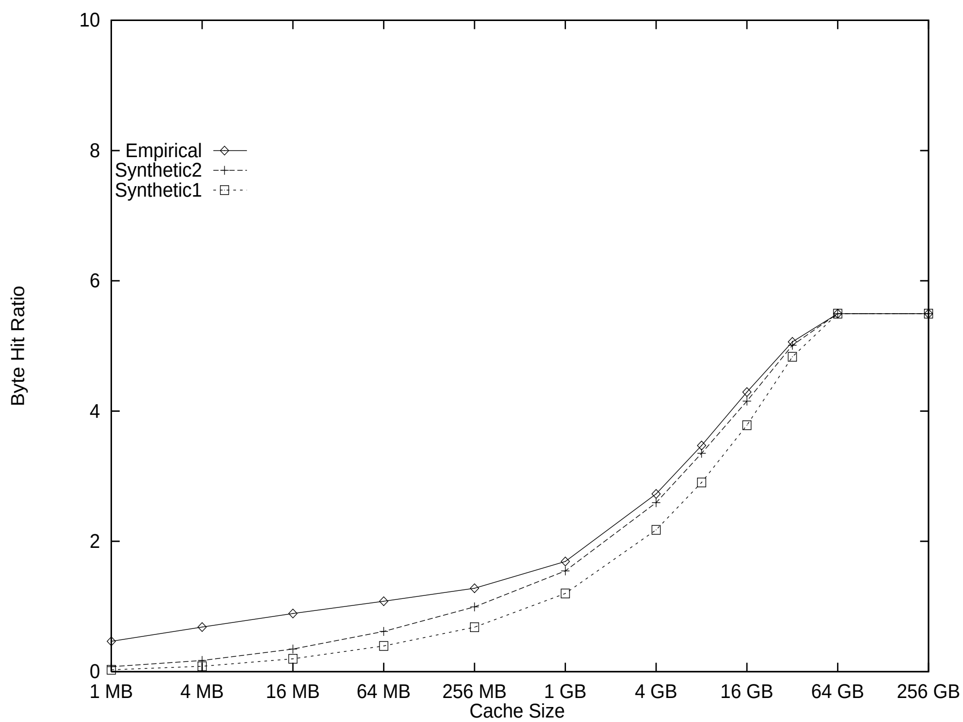


Figure 4.10: Caching Performance Results for Empirical Trace versus Synthetic Trace for the CANARIE Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging

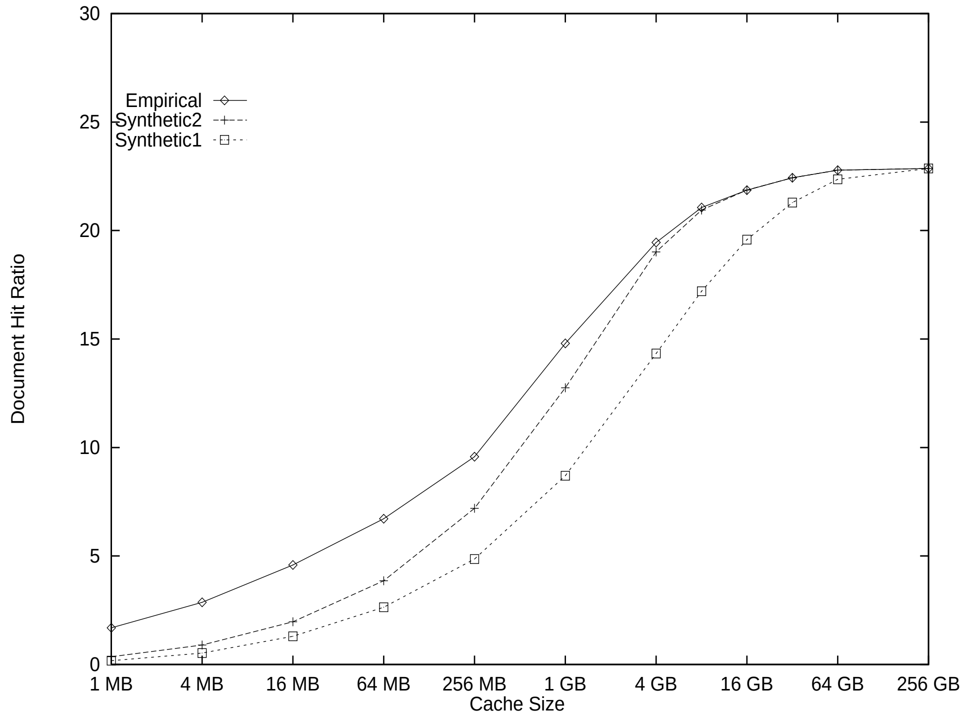


(a)

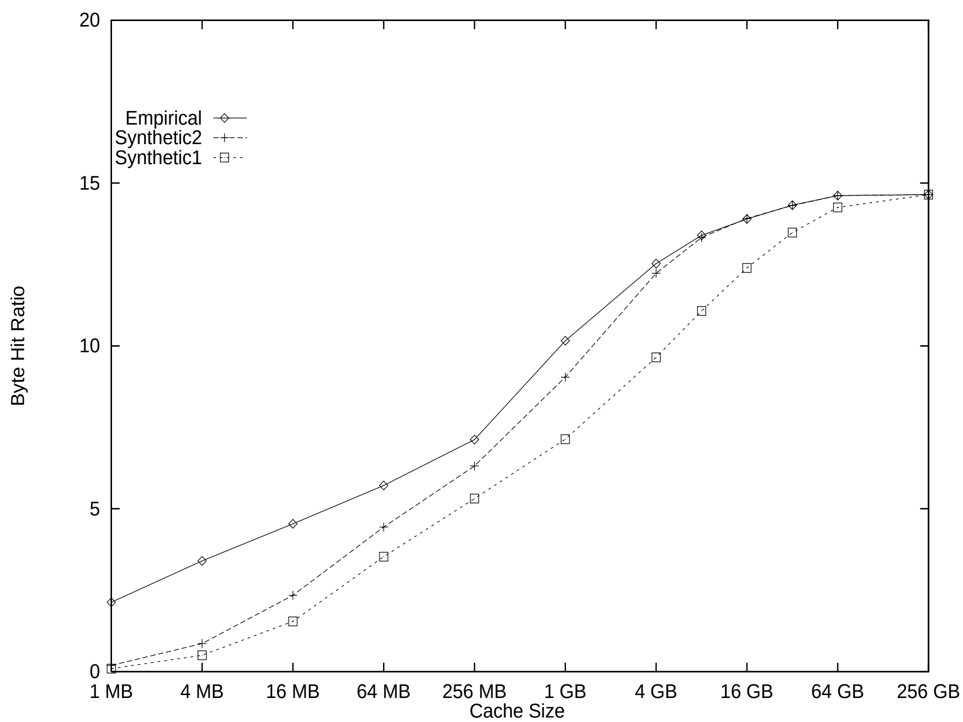


(b)

Figure 4.11: Caching Performance Results for Empirical Trace versus Synthetic Trace for the CANARIE Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)



(a)



(b)

Figure 4.12: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU

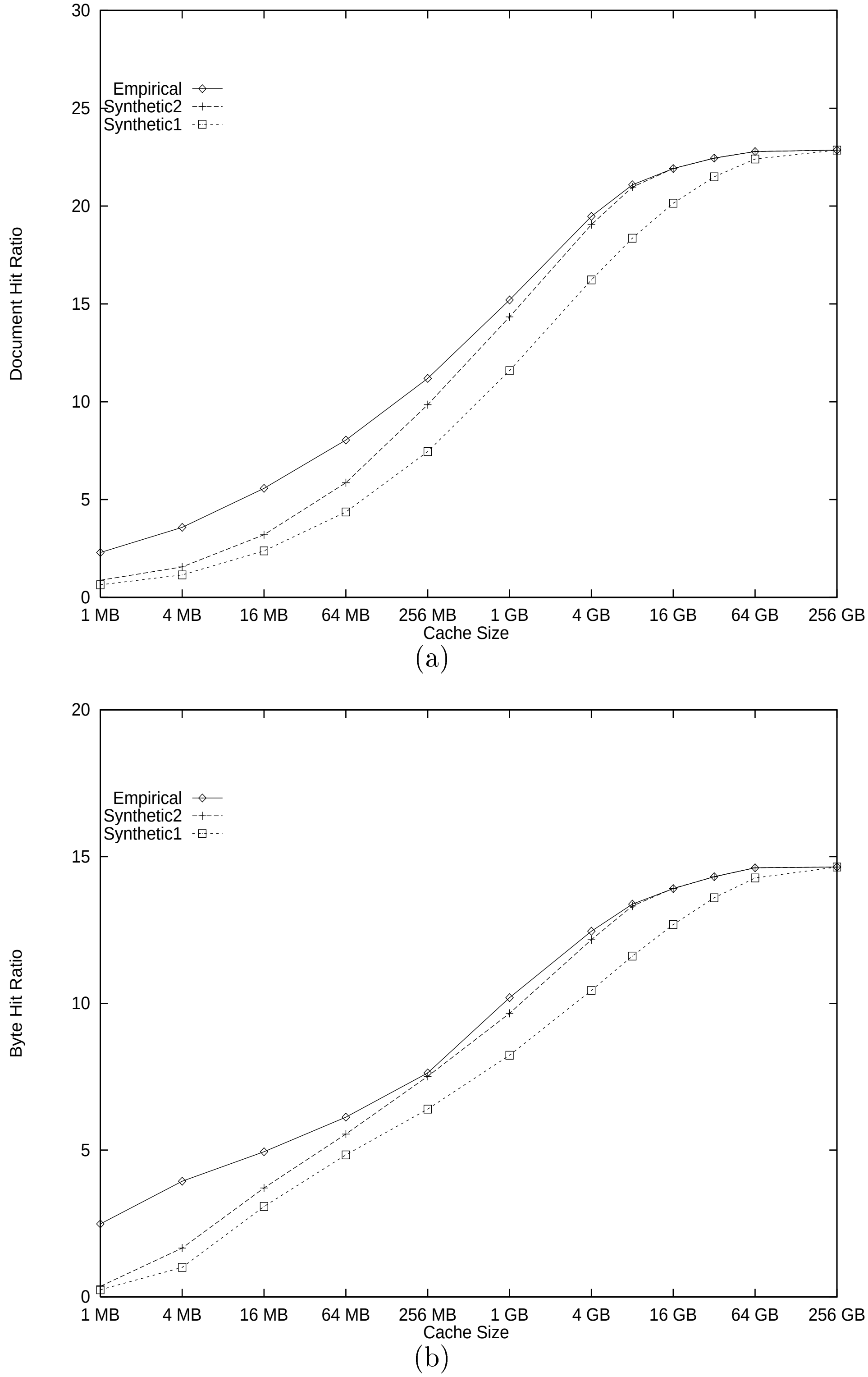
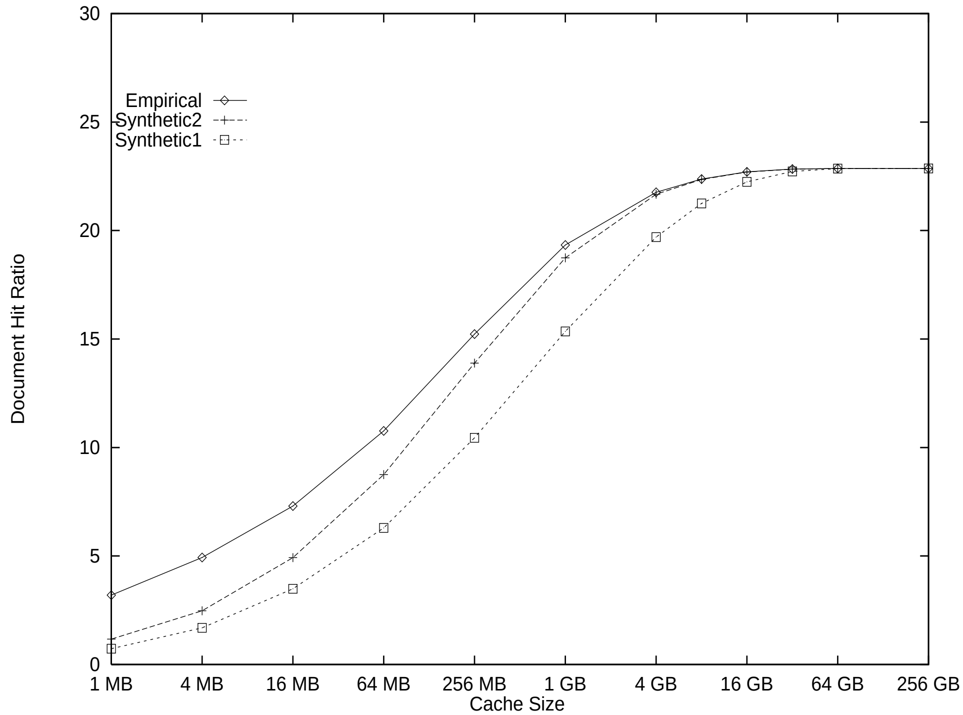
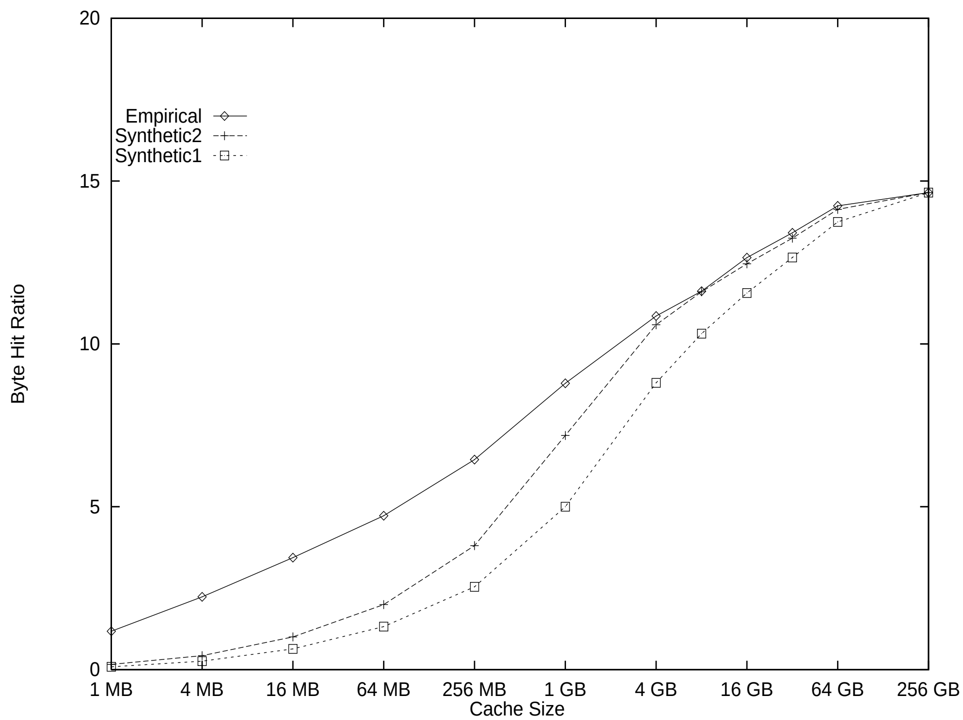


Figure 4.13: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging



(a)



(b)

Figure 4.14: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)

obtained by the empirical and `Synthetic1` traces for the USask data set under the three replacement policies at cache sizes of 256 MB and 4 GB. At a cache size of 256 MB, the document hit ratios obtained by GD-Size(1) for the `Synthetic1` trace is approximately 10 percentage points lower than that for the empirical trace, whereas at a cache size of 4 GB, the document hit ratio for the `Synthetic1` trace is only 4 percentage points lower than the empirical trace. On the other hand, the document hit ratios obtained by the LRU policy for the `Synthetic1` trace at cache sizes of 256 MB and 4 GB are 11 percentage points and 9 percentage points lower than those obtained for the empirical trace, respectively, while the LFU-Aging policy is least affected by the absence of temporal locality (see above for an illustration).

These results show that temporal locality is indeed important for analysing quantitative aspects of Web caching, and the impact of temporal locality varies with the replacement policies. It also establishes that a simple IRM-based model is not adequate for Web proxy workload generators designed to assess various qualitative and quantitative aspects of Web proxy cache performance. However, it is important to realise that Web caches are typically several gigabytes in size (e.g., the USask proxy uses a 5 GB disk cache [35]). In this context, a synthetic trace generated by applying IRM on a day-to-day basis can prove to be fruitful as the cache performance results achieved for such traces are close to those obtained for the corresponding empirical traces. The results also indicate that a good replacement policy for Web proxies should consider concentration, temporal locality and size of documents.

It is worthwhile to note that this study considered all static documents to be cacheable, which is far from being true (see Section 3.2.4). Therefore, the cache hit ratios reported here should be considered to be optimistic estimates. It should also be noted that much of the short-term temporal locality observed in the traces could be an artifact of the uncacheability of documents. As caching mechanisms improve, different temporal locality characteristics might be observed at the proxy caches, and the impact of short-term temporal locality might decrease.

4.4 Modelling Temporal Locality in Web Proxy Workloads

The results in the preceding section established the importance of temporal locality in Web proxy workloads. In this section, a model for incorporating both short-term and long-term temporal locality in synthetic workloads is considered. Section 4.4.1 describes the technique used for generating these synthetic traces and discusses their temporal locality characteristics. Section 4.4.2 presents cache performance results aimed at determining the applicability of the synthetic traces.

4.4.1 Finite Size LRU Stack Model

The simulation results in Section 4.3.2 show that if the "hot set" drift is modelled on a day-to-day basis, then the cache performance results from the synthetic traces are similar to those of the empirical traces for large caches (i.e., caches larger than the average distinct document set size). To introduce short-term locality in the synthetic trace, a finite size LRU stack is defined and configured with the (cumulative) probabilities of referencing the stack positions observed in the empirical trace.

When the synthetic trace generation begins, the LRU stack is empty and the references are made in accordance with the IRM model. Once the LRU stack is full, an uniform random number is generated to determine whether or not the next reference generated is for one of the documents in the LRU stack. If the random number generated is less than or equal to the cumulative probability of referencing documents from the LRU stack, then the next reference generated is for a document in the LRU stack. A search is made for a document from the top of the stack until the probability associated with a particular stack depth is greater than or equal to the random number generated; otherwise, an IRM model reference is generated. When the last reference to a particular document is generated (based on target document counts from the concentration analysis) and the document happens to be in the LRU stack, the document is removed from the LRU stack and all documents below the document being removed are pushed one position up in the LRU stack. After generating the reference stream for one day, the LRU stack is flushed and the process is repeated for the next day.

Two synthetic traces were generated for each empirical trace: `Locality1` with an LRU stack of size 100 documents; and `Locality2` with an LRU stack of size 1000 documents. These synthetic workloads preserve the concentration characteristics of the empirical traces and also preserve the long-term temporal locality (i.e., the synthetic traces have the same "hot set" drift characteristics as the empirical traces). The short-term temporal locality measure T (for the ten most popular documents) was calculated for each data set of the `Locality1` and the `Locality2` synthetic traces, and are shown in Tables 4.6 and 4.7, respectively. It is evident that all the documents exhibit short-term temporal locality. The T measure is higher near the top of the LRU stack and the T measure generally decreases as one moves down the stack. This is because an LRU stack is used to introduce closely spaced references in the synthetic trace and the probability of referencing documents at a particular stack position decreases in the empirical traces as deeper stack positions are considered. It is important to note that the short-term temporal locality characteristics in the synthetic traces are different from those exhibited by the empirical traces (e.g., compare Tables 3.11-3.13 with Table 4.6). The synthetic traces exhibit homogeneity of temporal locality, whereas the empirical traces show that some documents have more temporal locality compared to others. The impact of heterogeneity in temporal locality exhibited by documents will be quantified by the experiments in the following sections.

4.4.2 Performance of Cache Replacement Algorithms

In this section, the cache hit ratios obtained for the `Locality1` and `Locality2` synthetic traces are compared with those obtained for the corresponding empirical traces. Figures 4.15-4.23 present the results. The top graphs in each figure are for document hit ratios; the bottom graphs in each figure are for byte hit ratios. In each

Table 4.6: Short-Term Temporal Locality Measure for the `Locality1` Traces

USask										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	4.06	3.86	3.11	2.78	2.81	2.76	2.46	2.45	2.48	2.36
2	4.55	4.58	3.79	3.54	3.24	3.25	2.87	2.99	2.70	2.88
3	3.76	3.67	2.89	2.40	2.13	2.05	1.95	1.98	1.80	1.74
4	5.01	4.84	4.12	3.37	3.07	3.20	3.06	2.94	2.81	2.78
5	5.86	5.81	4.71	4.12	4.28	3.66	3.67	3.80	3.81	3.37
6	5.68	5.36	4.54	3.77	3.57	3.75	3.42	3.37	3.11	3.10
7	5.34	5.09	4.09	3.50	3.48	3.23	3.10	3.09	2.79	3.11
8	5.13	4.80	3.68	3.10	2.93	2.92	2.75	2.83	2.52	2.42
9	4.49	3.61	2.56	2.05	1.99	1.87	1.70	1.79	1.52	1.62
10	8.71	8.94	7.87	7.32	6.49	6.57	5.93	6.05	6.42	5.61
CANARIE										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	4.24	4.31	4.29	3.94	3.81	3.58	3.31	3.29	2.78	3.21
2	2.38	3.27	2.94	3.70	3.57	2.90	3.02	3.07	2.63	2.72
3	3.14	4.38	5.12	4.45	4.78	4.46	3.64	3.99	3.08	4.38
4	2.26	3.08	4.11	3.69	3.10	3.32	2.55	2.86	2.96	2.62
5	3.68	3.47	4.48	5.13	3.75	3.26	4.26	4.67	3.56	3.41
6	9.70	10.66	9.92	7.96	8.42	9.20	9.17	10.87	9.68	8.90
7	7.61	8.39	10.75	9.70	9.36	8.83	8.47	6.40	7.45	6.43
8	8.97	9.74	11.80	11.10	10.67	11.57	7.12	7.89	9.36	6.10
9	4.94	6.93	4.81	5.25	4.38	6.05	5.58	5.22	4.03	3.37
10	7.60	10.34	12.84	11.18	11.60	12.22	10.34	8.52	9.49	9.36
NLANR										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	4.83	4.86	4.44	4.18	4.06	4.34	4.08	4.28	4.01	4.06
2	3.55	3.90	3.51	3.29	3.22	3.36	2.96	3.25	3.06	2.88
3	2.03	2.17	1.91	1.82	1.75	1.70	1.39	1.49	1.65	1.55
4	1.96	2.04	1.79	1.46	1.32	1.49	1.28	1.25	1.25	1.30
5	2.41	2.37	2.15	1.92	1.87	1.94	1.66	1.75	1.60	1.64
6	8.53	8.48	8.28	7.55	7.43	8.07	7.23	7.49	7.24	7.09
7	2.92	3.40	2.97	2.70	2.44	2.68	2.34	2.36	2.27	2.40
8	2.86	3.04	2.41	2.36	2.32	2.24	2.31	2.28	2.26	2.07
9	11.11	11.84	11.52	10.47	11.80	10.65	10.07	10.07	9.67	10.25
10	17.69	17.03	16.47	15.51	14.12	13.22	11.92	12.62	12.34	10.83

Table 4.7: Short-Term Temporal Locality Measure for the `Locality2` Traces

Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
USask										
1	3.88	3.45	2.96	2.73	2.72	2.65	2.42	2.46	2.23	2.33
2	4.93	4.63	3.99	3.93	3.51	3.55	3.30	3.46	3.24	3.17
3	3.90	3.62	3.40	2.95	2.72	2.70	2.52	2.54	2.61	2.45
4	4.99	4.81	3.91	3.49	3.28	3.37	3.14	3.00	2.93	3.05
5	5.38	4.75	4.14	4.09	4.02	3.90	3.74	3.47	3.76	3.57
6	5.77	5.07	4.70	4.30	3.94	4.36	3.81	3.95	3.59	3.84
7	5.51	5.28	4.57	4.03	4.02	3.71	3.87	3.73	3.68	3.48
8	5.54	4.77	4.14	3.80	3.78	3.54	3.42	3.52	3.34	3.18
9	3.87	3.49	2.75	2.73	2.22	2.22	2.10	1.91	1.83	1.89
10	5.80	5.85	5.11	4.95	4.42	4.32	4.11	3.95	4.08	3.86
CANARIE										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	1.81	2.04	1.82	1.78	1.65	1.60	1.51	1.71	1.67	1.53
2	1.35	1.65	1.62	1.90	1.81	1.67	1.73	1.88	1.59	1.35
3	3.27	3.48	4.46	3.41	4.31	3.86	2.92	2.82	2.83	3.17
4	1.65	1.55	1.76	1.83	1.76	1.80	1.61	1.59	1.79	1.38
5	1.71	1.22	2.40	2.04	1.63	1.51	1.91	1.35	1.36	1.41
6	5.15	4.06	4.87	3.91	3.87	4.17	4.49	4.55	4.15	3.56
7	4.69	4.32	3.93	4.50	4.11	3.97	3.56	3.87	4.11	3.49
8	9.63	9.42	10.80	10.39	10.88	9.50	7.46	8.11	8.67	8.46
9	1.89	2.92	2.55	2.57	2.58	2.73	2.22	2.23	2.24	0.93
10	7.34	8.28	10.67	8.63	9.39	8.84	7.49	7.64	8.24	7.58
NLANR										
Document	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6	Bucket 7	Bucket 8	Bucket 9	Bucket 10
1	7.64	6.95	6.46	6.06	5.92	5.93	5.36	5.49	5.06	4.86
2	6.36	5.97	5.87	5.68	5.63	5.56	5.14	5.14	4.89	4.65
3	3.47	3.58	3.16	3.17	2.89	3.18	2.91	2.92	2.96	2.93
4	2.06	2.22	1.80	1.64	1.48	1.57	1.50	1.45	1.48	1.47
5	3.54	3.77	3.68	3.32	3.22	3.47	3.08	2.95	3.05	2.88
6	8.36	8.53	8.18	7.52	7.35	7.99	7.16	7.50	7.04	7.24
7	4.91	4.89	4.33	4.31	4.26	4.40	3.65	3.75	3.81	4.00
8	4.28	4.25	3.66	3.86	3.66	3.80	3.21	3.51	3.61	3.15
9	14.97	14.06	13.16	12.57	12.36	11.16	10.66	10.25	9.90	10.01
10	8.83	7.45	7.78	7.00	6.33	6.53	5.41	5.96	6.40	5.43

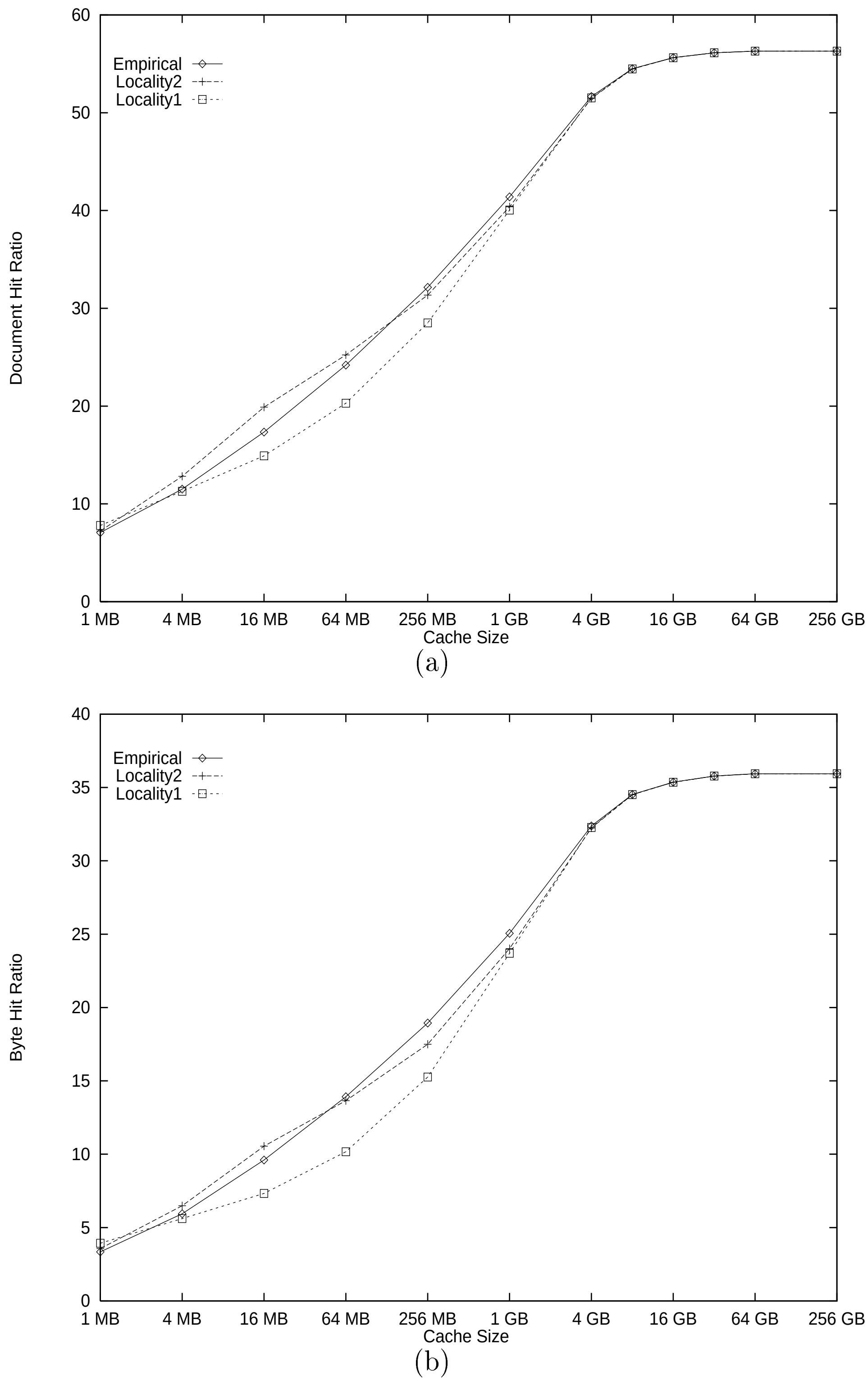
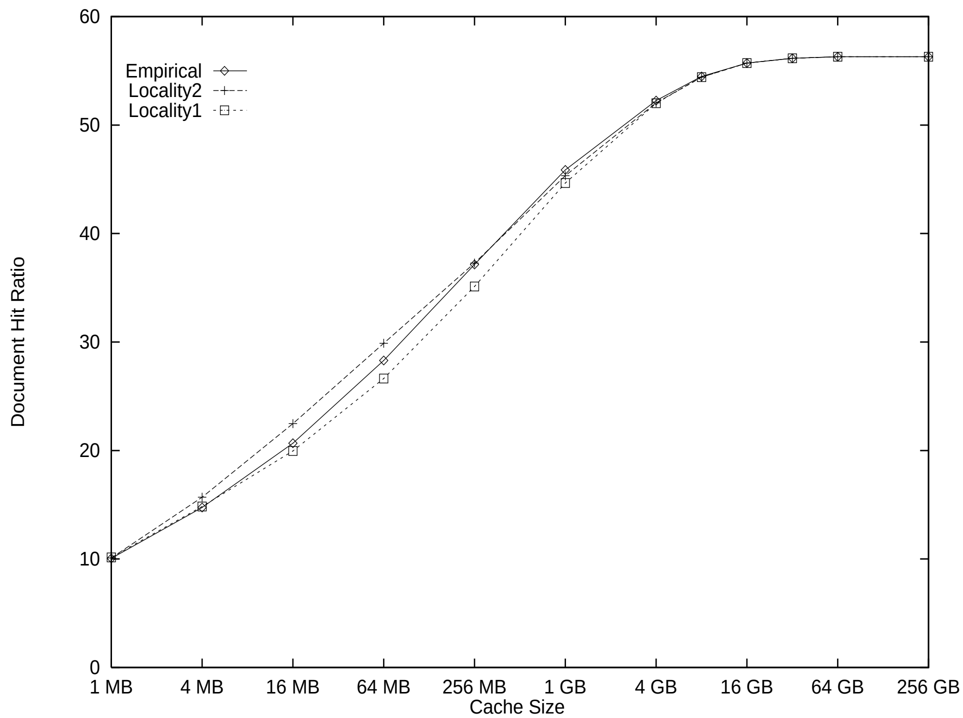
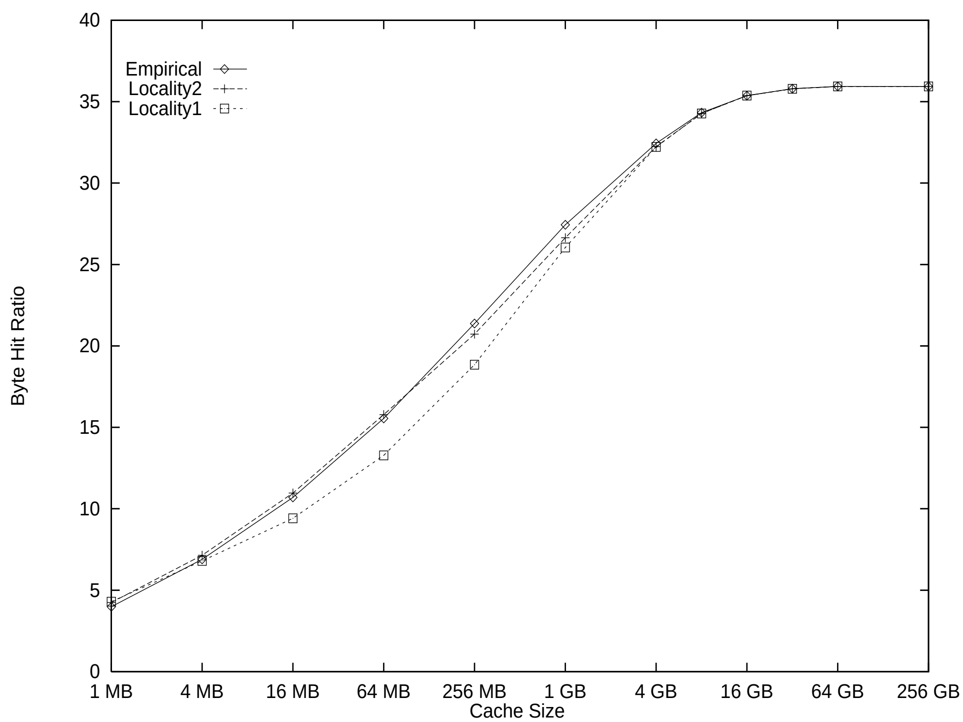


Figure 4.15: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU



(a)



(b)

Figure 4.16: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging

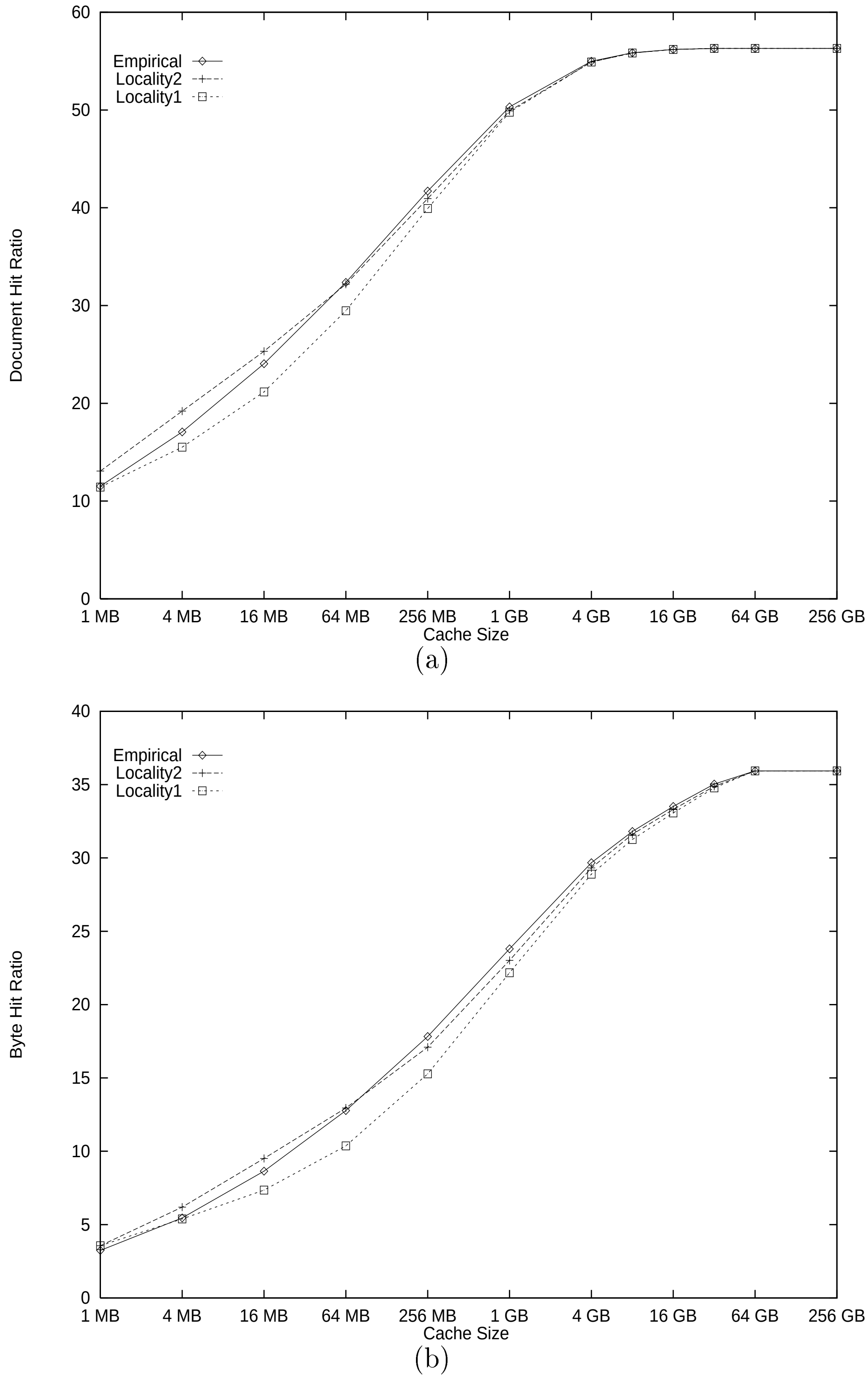
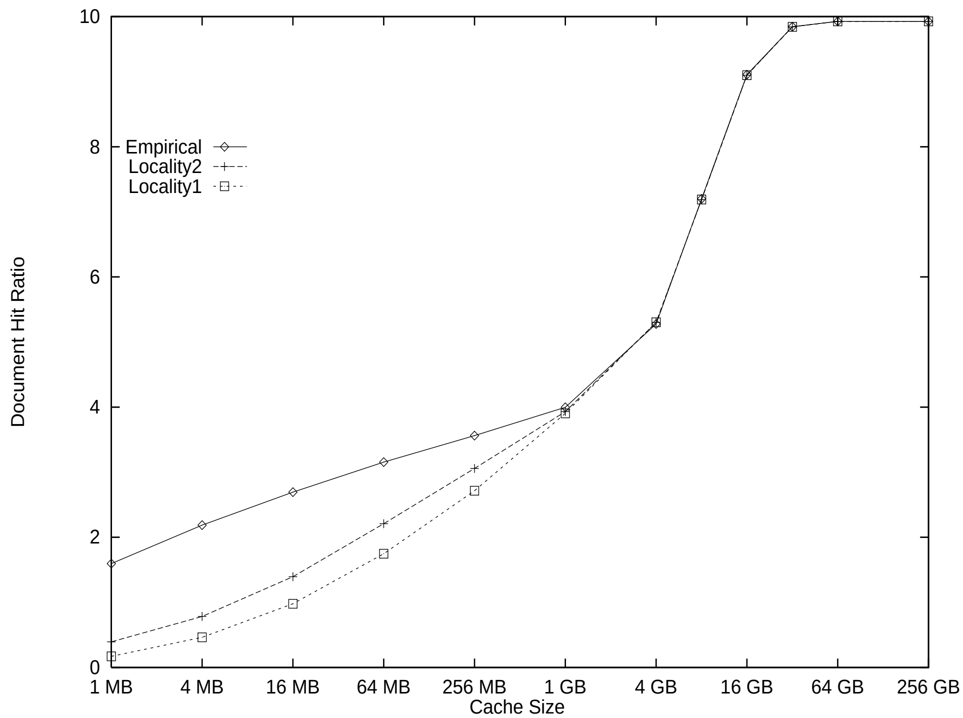
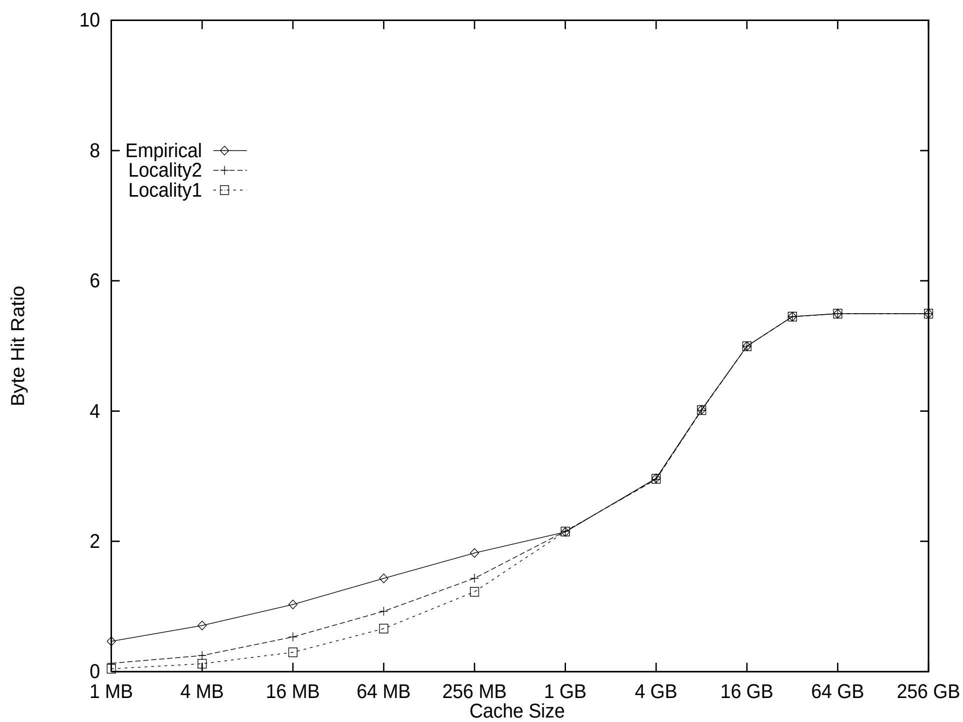


Figure 4.17: Caching Performance Results for Empirical Trace versus Synthetic Trace for the USask Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)

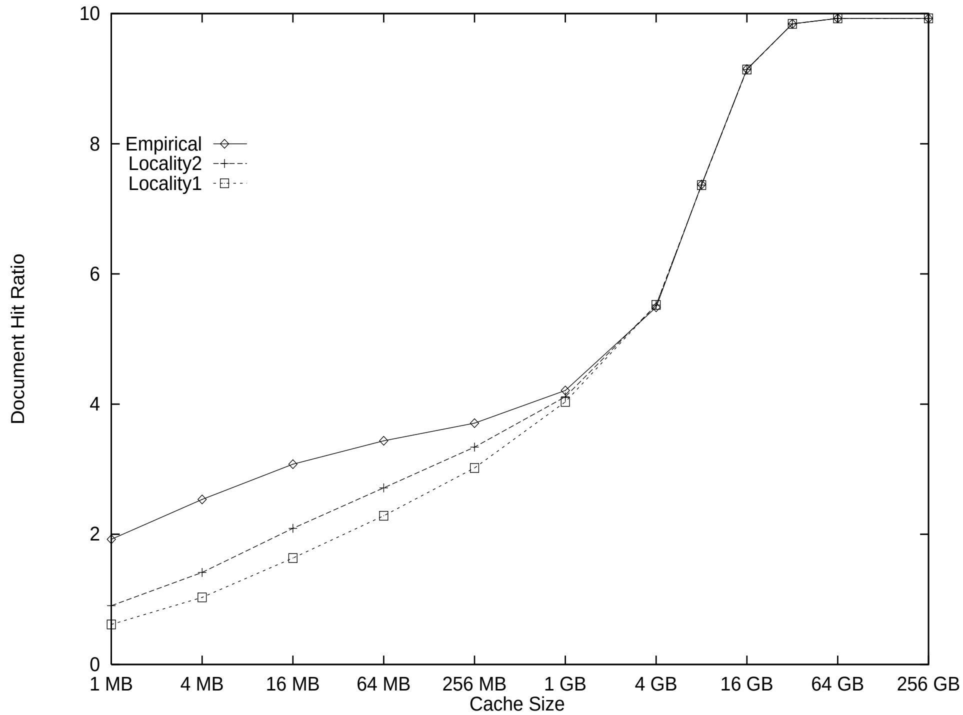


(a)

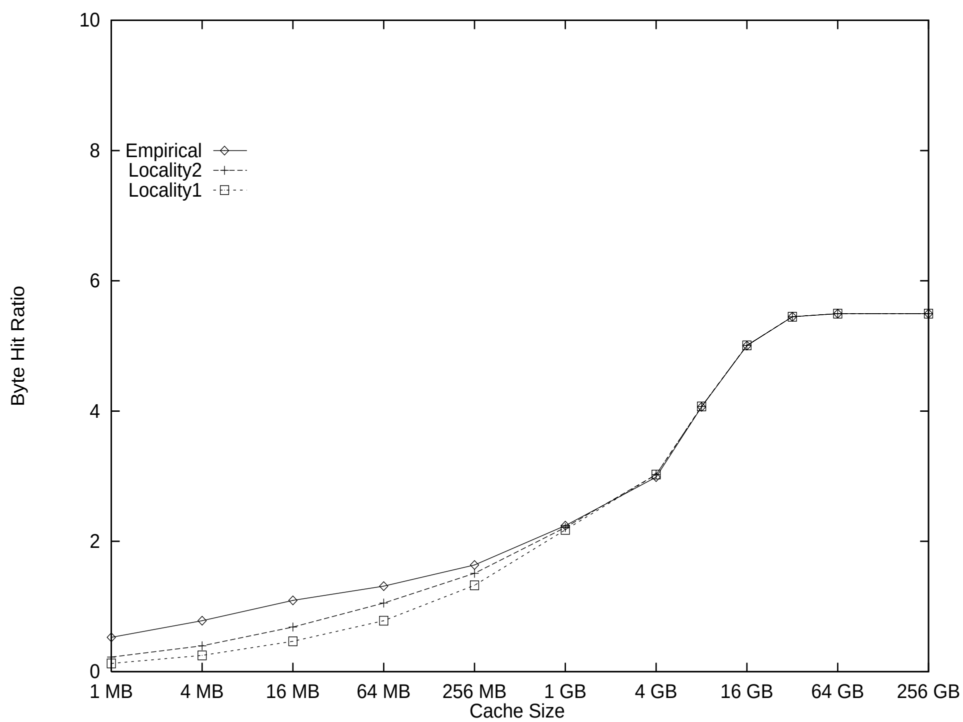


(b)

Figure 4.18: Caching Performance Results for Empirical Trace versus Synthetic Trace for the Canarie Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU



(a)



(b)

Figure 4.19: Caching Performance Results for Empirical Trace versus Synthetic Trace for the CANARIE Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging

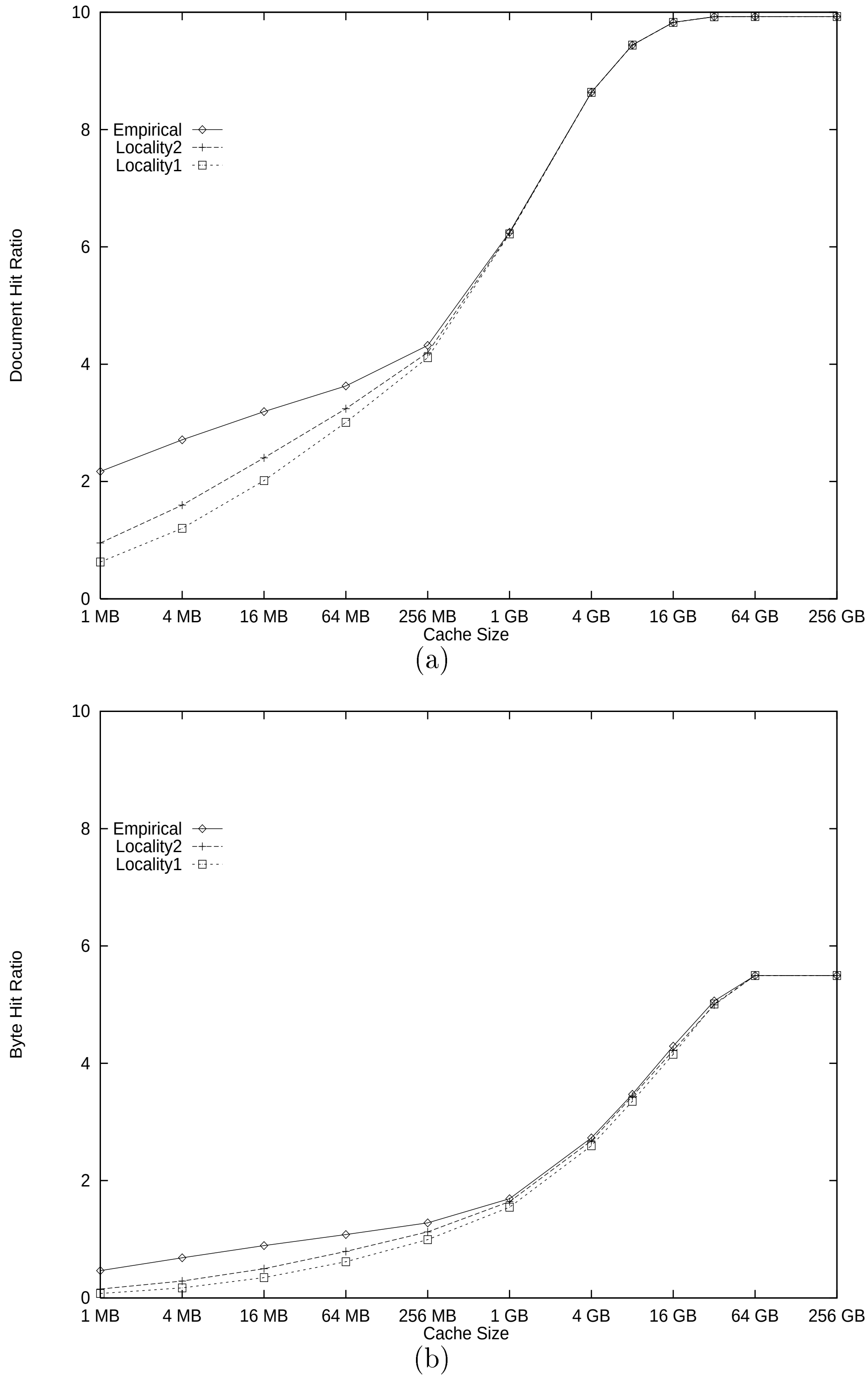


Figure 4.20: Caching Performance Results for Empirical Trace versus Synthetic Trace for the CANARIE Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)

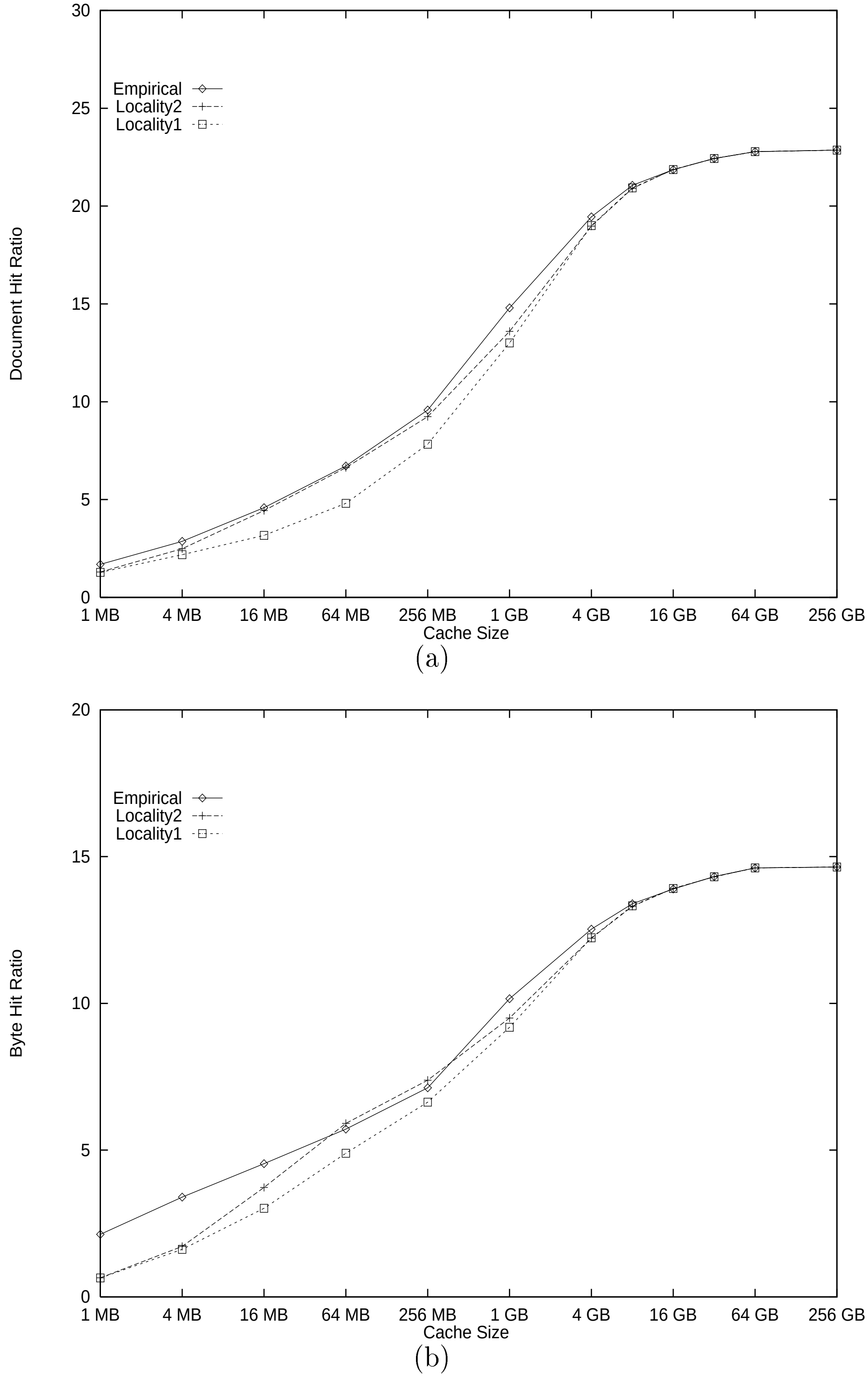
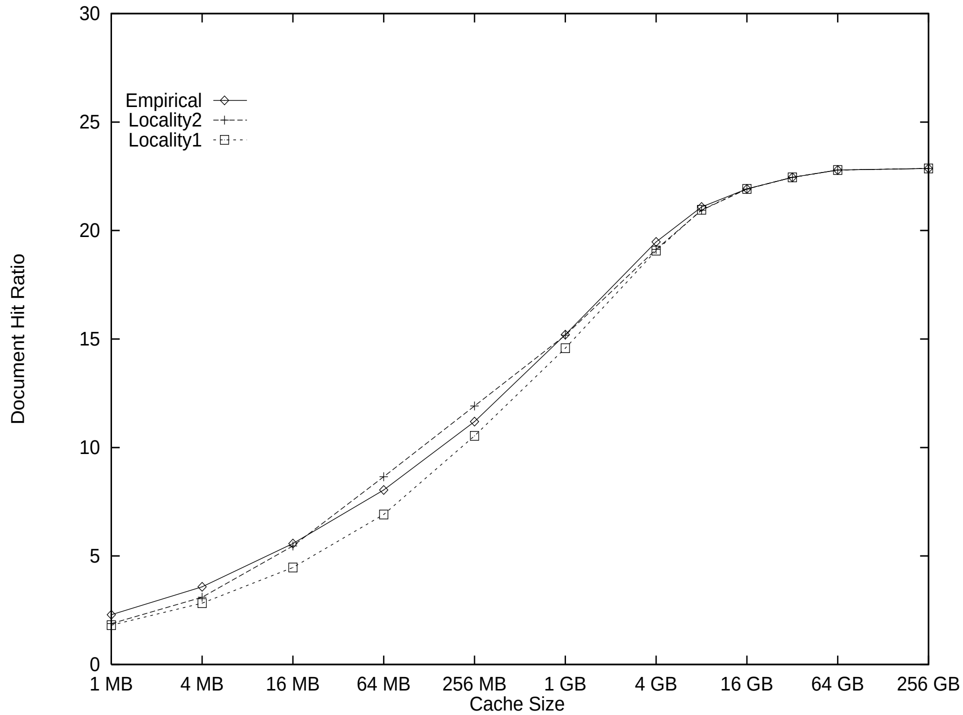
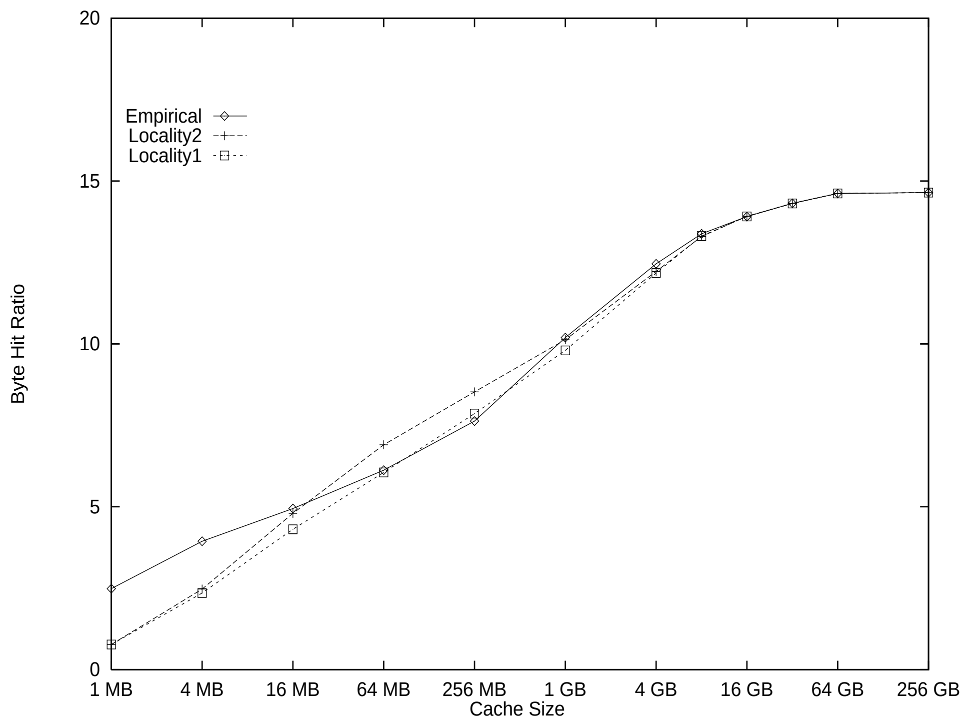


Figure 4.21: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for LRU; (b) Byte Hit Ratio for LRU

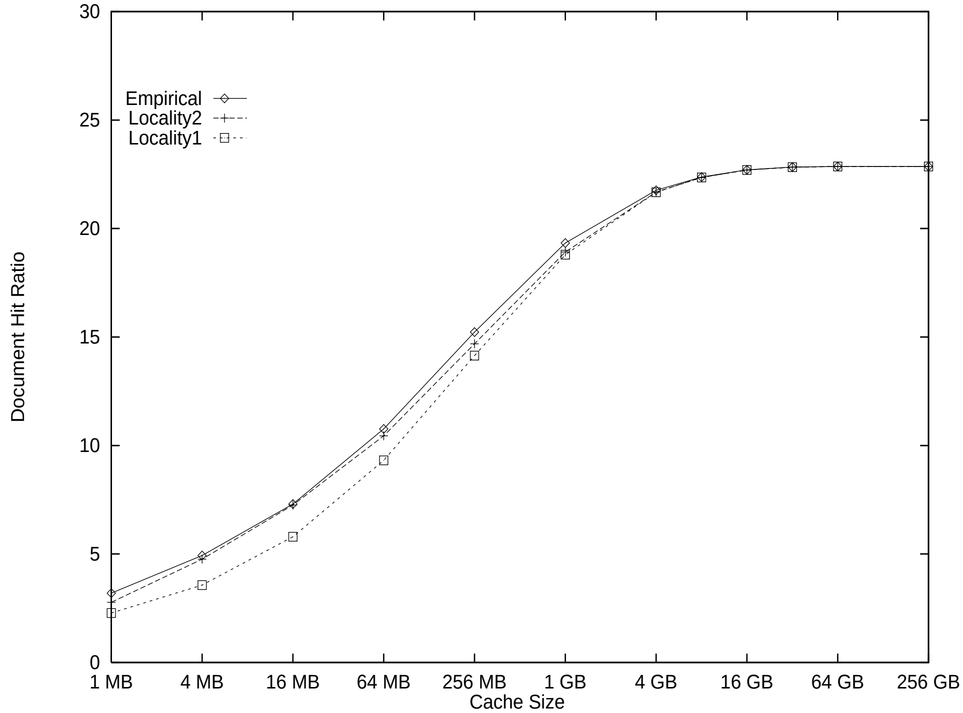


(a)

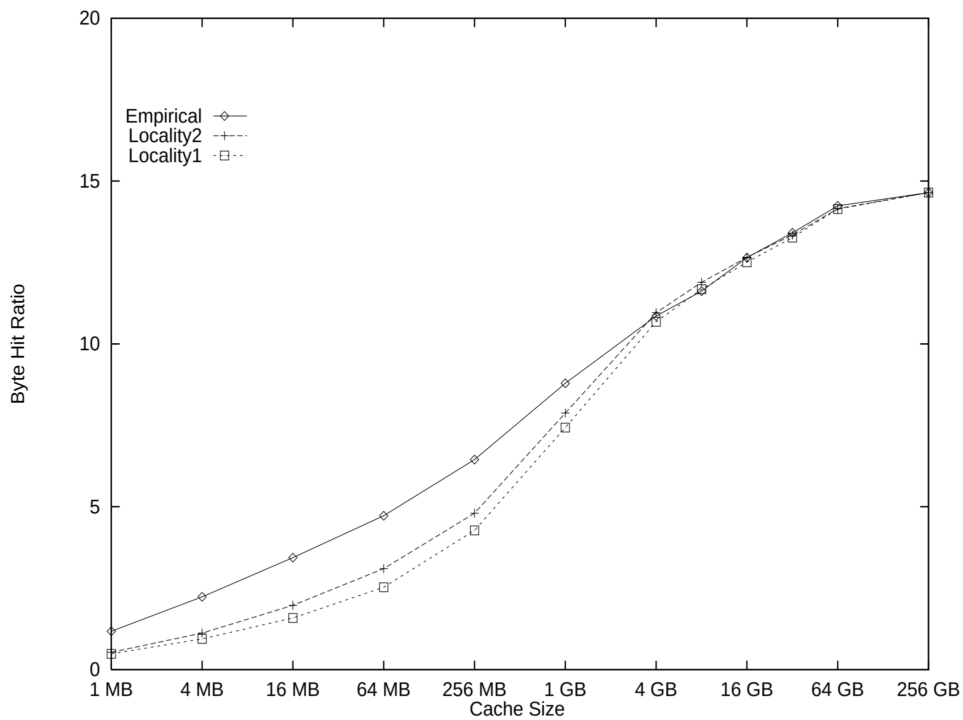


(b)

Figure 4.22: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for LFU-Aging; (b) Byte Hit Ratio for LFU-Aging



(a)



(b)

Figure 4.23: Caching Performance Results for Empirical Trace versus Synthetic Trace for the NLANR Data Set: (a) Document Hit Ratio for GD-Size(1); (b) Byte Hit Ratio for GD-Size(1)

graph, the y-axis is truncated to highlight the performance differences. As before, the caching performance results are presented for all the three replacement policies (LRU, LFU-Aging, and GD-Size(1)) for all the traces.

The results show that with use of the `Locality1` trace, cache hit ratios are underestimated for cache sizes less than the average total size of all unique documents accessed in a single day. It can also be seen that cache hit ratios obtained with the `Locality2` traces are very similar to those obtained by the empirical traces. However, the hit ratios obtained with the `Locality2` traces for small cache sizes can sometimes overestimate the cache hit ratios (see Figures 4.15, 4.16, and 4.23). The results also show that heterogeneity in short-term temporal locality exhibited by the popular documents (i.e., some documents exhibit temporal locality, whereas some documents do not exhibit temporal locality) does not have significant impact on the cache hit ratio results.

4.5 Summary

This chapter considered the impact of temporal locality on Web proxy cache performance. Three different cache replacement policies were used: LRU, LFU-Aging and GD-Size(1). First, a synthetic trace generator was developed that generated two types of synthetic trace: one that preserves the concentration characteristics of the original trace, but filters out any temporal locality present (`Synthetic1`); and a second type that preserves concentration as well as long-term temporal locality, but filters out any short-term temporal locality (`Synthetic2`). A Web proxy cache simulator was developed and cache performance analysis was performed using the empirical and the synthetic traces.

The results of the cache performance study show that GD-Size(1) provides higher document hit ratios, but does so by discriminating against large documents, and therefore achieves the lowest byte hit ratios among the three replacement algorithms considered. The LFU-Aging policy performed reasonably well in terms of document hit ratios because it kept frequently referenced (large and small) documents in its cache, and achieved the highest byte hit ratio among the three policies considered. The results also show that LRU performed worst in terms of document hit ratios and slightly better than GD-Size(1) in terms of byte hit ratios.

Comparison of the hit ratios for the empirical traces with those for the synthetic traces show that the synthetic traces based on the IRM model underestimate the hit ratios. However, for cache sizes larger than the average total size of all unique documents referenced in a single day, the cache hit ratios obtained with the `Synthetic2` traces are very similar to those obtained for empirical traces. These results indicate that short-term temporal locality matters only for small cache sizes.

To introduce short-term locality in the synthetic traces, a finite size LRU stack model is used. However, this model introduces temporal locality in all popular documents. This is not the case for the empirical traces, where some documents exhibit short-term locality while others do not exhibit short-term locality. However, the cache hit ratios obtained with this type of synthetic trace are very similar to those obtained with the empirical traces. This might indicate that heterogeneity in short-term temporal locality characteristics is unimportant for cache performance analysis.

Chapter 5

Summary, Conclusions, and Future Work

5.1 Thesis Summary

Chapter 1 presented the motivation behind the research. Two goals were identified: understanding the characteristics of the workloads presented to Web proxy servers and understanding how these characteristics change across different levels of a caching hierarchy; and determining how Web document references can be modelled.

Chapter 2 presented an overview of the World-Wide Web. The functioning of Web clients, proxies and servers was discussed. This was followed by a literature review of recent work on Web performance. Results of various workload characterisation studies [1, 4, 29] were outlined. Caching techniques employed in the Web were detailed, focusing on the differences between Web caching and traditional caching (e.g., in the domain of memory management, distributed file systems). Other issues considered were cache consistency, and techniques proposed for improving the performance of Web caching hierarchies.

Chapter 3 describes a workload characterisation study carried out for Web proxy servers. The study used access logs from three different Web proxies in a particular caching hierarchy. Necessary preprocessing was done on the access logs so as to extract useful information and keep the size of the access logs manageable. These reduced access logs formed the basis of the workload characterisation study. Several characteristics were studied, such as, document types and their sizes, proportion of documents accessed only once in the reduced access logs, distribution of transfer sizes, the popularity profile of documents, document modification ratios, distribution of document inter-reference distances, and temporal locality.

Chapter 4 described the methodology adopted for determining the impact of temporal locality on Web proxy workloads under different cache replacement policies. First, a synthetic trace generator was developed which produces two types of synthetic traces. One synthetic trace was constructed from the empirical trace by applying the Independent Reference Model (IRM) on the entire trace duration. This version of the synthetic trace filters out both the long-term and the short-term temporal locality present in the empirical trace. The other type of synthetic trace was generated by applying the IRM model on a per-day basis. This trace destroys the short-term temporal locality characteristics of the empirical trace but preserves the long-term temporal locality characteristics. A proxy cache simulator was developed and validated. Cache performance results from the synthetic and empirical traces showed that temporal locality matters in Web proxy workloads. Experiments also showed that short-term temporal locality matters only for small cache sizes and for most practical caches, a synthetic workload based on a per-day IRM model would provide reasonably accurate cache performance results. Chapter 4 concludes with a modelling approach that introduces short-term temporal locality into the synthetic traces.

5.2 Results and Contributions

The workload characterisation carried out in Chapter 3 contributed a number of observations. The main results are:

- A large number of requests (close to 90%) made to a Web proxy result in the client successfully obtaining the document.

- HTML and image files account for approximately 95% of the total requests seen at the proxies. Image type documents are the most referenced documents followed by documents of HTML type.

- The mean document transfer size is 7-15 KB, while the median transfer size is 2-3KB.

- One-timers account for a large number of references (17-45% of the total requests) in the Web proxies. The percentage of one-timers is higher at proxies than that reported for Web servers [4].

- The distribution of transfer sizes is heavy-tailed. The tail of the distribution is heavier at the higher-level proxies compared to that of lower-level proxies.

- The popularity of Web documents does not strictly follow the Zipf's law. The exponent in the Zipf relationship depends upon the level of the proxy in the caching hierarchy.

- Concentration of references is lower at Web proxies than that reported for Web servers [4]. The concentration of references is higher at the lower-level proxies than at the higher-level proxies.

- The distribution of inter-access times decays rapidly with time. A significant fraction of the re-references occurs within a short interval of time. These re-references occurred mainly because of document sharing, reloads, and immediate "expire" times for documents.

- Two new measures were proposed for measuring temporal locality. The drift measures showed that few documents have enduring popularity. The short-term temporal locality measure was specifically developed to distinguish "hot set" effects from temporal locality. The short-term temporal locality measure T was calculated for the most popular 25 documents in each data set. The results showed that some popular documents exhibit temporal locality.

The workload characterisation study across different levels of a caching hierarchy is important in light of the ever growing size of the Web. Two drift measures, and the short-term temporal locality measure, are an offshoot of the workload characterisation study. The drift measures are flexible and clearly illustrate how long-term temporal locality varies with time. On the other hand, the short-term temporal locality measure T correlates the probability of referencing a particular position of the LRU stack with the probability of a document being at a particular position of the stack without being referenced.

The simulation experiments reported in Chapter 4 showed that a simple IRM model is inadequate for modelling Web document references. The experiments also established that if long-term temporal locality is modelled, then for large caches the performance of empirical and synthetic traces is similar. The experiments also showed that the LFU-Aging replacement policy was influenced the least by the absence of temporal locality, whereas, LRU was affected the most by the presence or absence of temporal locality. Finally, a IRM model combined with a finite size LRU stack model was used to generate synthetic traces. Simulations showed that the cache performance with such synthetic traces was very similar to that with the empirical traces.

5.3 Directions for Future Work

There are several possible directions for future work. Some of these research directions follow directly from the work presented in this thesis, while other initiatives can complement the research results presented here.

In this thesis, various Web proxy workload characteristics have been analysed. Suggestions regarding ways of modelling these characteristics have been proposed. These results and observations can be used to design a flexible synthetic workload generator. Such workload generators can be used to learn more about the impact of different workload parameters on the performance of Web proxy caches.

The World-Wide Web is a continually evolving system. Forecasts indicate that by the year 2002, there will be about 320 million Web users, compared to approximately 100 million users in the year 1998 [45]. The number of multimedia services are also increasing, resulting in more audio and video traffic. Therefore, analysis of Web proxy traffic over time needs to be conducted to understand the changes occurring in the traffic characteristics.

In this study, access logs from different levels of one particular caching hierarchy was used. To comment on the general characteristics at different levels of a caching hierarchy, a study needs to look at more than one caching hierarchy, as was done for Web server [4], proxy [1], and client [29] workloads.

With the increasing popularity of Web caching hierarchies, it is necessary to evaluate the performance of the Web caches at the different levels of the hierarchy. It is known that the caches at higher-levels often become severely loaded. Such bottlenecks should be studied to arrive at better caching techniques in caching hierarchies.

The synthetic trace generation process outlined in Chapter 4 can be further refined. Recall that although the synthetic trace generated using the LRU Stack Model achieved hit ratios comparable to the empirical traces, its short-term temporal locality characteristics were very different compared to that of the empirical trace. In the empirical traces, it was observed that some popular documents exhibited short-term locality, while others did not, whereas the synthetic trace introduced short-term temporal locality in a homogeneous manner into all documents. A better approach of modelling short-term temporal locality is to introduce short-term locality only into those documents that exhibit temporal locality.

References

[1] G. Abdulla, E. Fox, M. Abrams and S. Williams, "WWW Proxy Traffic Characterization with Application to Caching", Technical Report TR-97-03, Computer Science Department, Virginia Tech., March 1997.
Available at URL: `http://www.cs.vt.edu/~chitra/work.html`

[2] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing Reference Locality in the WWW", *Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, Miami Beach, FL, pp. 92-103, December 1996.
Available at URL: `http://www.cs.bu.edu/~best/res/papers/pdis96.ps`

[3] Apache HTTP Server Project, Apache HTTP Server.
Available at URL: `http://www.apache.org/`

[4] M. Arlitt, *A Performance Study of Internet Web Servers*, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, 1996.
Available at URL: `http://www.cs.usask.ca/ftp/pub/discus/thesis_arlitt_bw.ps.Z`

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

[6] M. Arlitt, R. Friedrich and T. Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies", Technical Report HPL-98-97, Hewlett Packard Laboratories, May 1998.
Available at URL: `http://www.hpl.hp.com/techreports/98/HPL-98-97.html`

[7] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm, "World-Wide Web Caching- The Application Level View of the Internet", *IEEE Communications*, Vol. 35, No. 6, June 1997.

[8] M. Baker, J. Hartman, M. Kipfer, W. Shirriff and J. Ousterhout, "Measurements of a Distributed File System", *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pp. 198-211, October 1991.

[9] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", *Proceedings of the 1998 ACM*

SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, Madison, WI, pp. 151-160, July 1998.

[10] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications", *World Wide Web Journal (to appear)*, 1999.
Available at URL: `htt://cs-www.bu.edu/faculty/crovella/paper-archive/traces98.ps`

[11] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Nielson and A. Secret, "The World-Wide Web", *Communications of the ACM*, Vol. 38, No. 8, pp. 76-82, August 1994.

[12] T. Berners-Lee, L. Masinter and M. McCahil, "Uniform Resource Locators", RFC 1738, December 1994.
Available at URL: `ftp://ftp.internic.net/rfc/rfc1738.txt`

[13] A. Bestavros, R. Carter, M. Crovella, C. Cunha, A. Heddaya and S. Mirdad, "Application-Level Document Caching in the Internet", *Proceedings of the Second International Workshop on Services in Distributed and Network Environment (SDNE '95)*, Whistler, BC, June 1995.
Available at URL: `http://www.cs.bu.edu/~best/res/papers/sdne95.ps`

[14] D. Bickle, Personal Communication (e-mail), January 1999.

[15] C. Bowman, P. Danzig, U. Manber and M, Schwartz, "Scalable Internet Resource Discovery: Research Problems and Approaches", *Communications of the ACM*, Vol. 38, No. 8, pp. 98-107, August 1994.

[16] H. Braun and K. Claffy, "Web Traffic Characterization: An Assessment of the Impact of Caching Documents from NCSA's Web Server", *Computer Networks and ISDN Systems*, Vol. 28, Nos. 1 & 2, pp. 37-51, January 1995.

[17] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the Implications of Zipf's Law for Web Caching", *Proceedings of IEEE INFOCOM'99*, New York, March 1999.
Available at URL: `http://www.cs.wisc.edu/∼cao/papers/zipf-implications.html`

[18] R. Bunt and J. Murphy, "The Measurement of Locality and the Behaviour of Programs", *Computer Journal*, Vol. 27, No. 2, pp. 238-245, August 1984.

[19] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovinch, "Web Proxy Caching: The Devil is in the Details", *Performace Evaluation Review*, Vol. 26, No. 1, pp. 11-15, December 1998.

[20] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", *Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems*,

Monterey, CA, pp. 193-206, December 1997.
Available at URL: `http://www.cs.wisc.edu/~cao/papers/gd-size.html`

[21] P. Cao, L. Fan and Q. Jacobson, "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance", *Proceedings of the 1999 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Atlanta, GA, pp. 178-187, May 1999.

[22] P. Cao and C. Liu, "Maintaining Strong Cache Consistency in the World-Wide Web", *IEEE Transactions on Computers*, Vol. 47, No. 4, pp. 445-457, April 1998.

[23] CANARIE, CA*net II sanitized access logs.
Available at URL: `http://ardnoc41.canet2.net/cache/squid/rawlogs/`

[24] V. Cate, "Alex - A Global File System", *Proceedings of the 1992 USENIX File Systems Workshop*, Ann Arbor, MI, pp. 1-12, May 1992.

[25] CERN - European Laboratory for Particle Physics, "An Overview of the World-Wide Web".
Available at URL: `http://www.cern.ch/Public/ACHIEVEMENTS/WEB/Welcome.html`

[26] CERN - European Laboratory for Particle Physics, CERN httpd Web Server.
Available at URL: `http://www.w3.org/Daemon/`

[27] A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz and K. Worrell, "A Hierarchical Internet Object Cache", *Proceedings of the 1996 USENIX Technical Conference*, San Diego, CA, pp. 153-166, January 1996.
Available at URL: `http://excalibur.usc.edu/cache-html/cache.html`

[28] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 835-871, December 1997.

[29] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-Based Traces", Technical Report TR-95-010, Department of Computer Science, Boston University, April 1995.
Available at URL: `ftp://cs-ftp.bu.edu/techreports/95-010-www-client-traces.ps.Z`

[30] P. Danzig, M. Schwartz, and R. Hall, "A Case for Caching File Objects Inside Internetworks", *Proceedings of the ACM SIGCOMM '93 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, San Francisco, CA, pp. 239-248, September 1993.

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

[32] P. Denning, "Working Sets Past and Present", *IEEE Transactions on Software Engineering*, Vol. 6, No. 1, pp. 64-84, January 1980.

[33] B. Duska, D. Marwood and M. Feeley, "The Measured Access Characteristics of World Wide Web Client Proxy Caches", *Proceedings of the USENIX Symposium on Internet Technologies & Systems*, Monterey, CA, pp. 23-35, December 1997. Available at URL: `http://www.cs.ubc.ca/Spider/marwood/Projects/SPA`

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", HTTP Working Group, Internet Draft, November 1998.
`http://www.w3.org/Protocols/HTTP/1.1/draft-ietf-http-v11-spec-rev-06.txt`

[35] E. Fogel, Personal Communication (e-mail), January 1999.

[36] S. Glassman, "A Caching Relay for the World-Wide Web", *Computer Networks and IDSN Systems*, Vol. 27, No. 2, pp. 165-174, November 1994.

[37] J. Gwertzman and M. Seltzer, "World-Wide Web Cache Consistency", *Proceedings of the 1996 USENIX Technical Conference*, San Diego, CA,, pp. 141-151, January 1996.
Available at URL: `http://www.eecs.harvard.edu/~vino/web/usenix.196/`

[38] J. Gwertzman and M. Seltzer, "An Analysis of Geographic Push-Caching", *Proceedings of the Fifth IEEE Workshop on Hot Topics in Operating Systems*, 1995.
Available at URL: `http://www.eecs.harvard.edu/~vino/web/hotos.ps`

[39] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*, John Wiley & Sons, Inc., New York, NY, 1991.

[40] N. Johnson and S. Kotz (editors), *Encyclopedia of Statistical Sciences*, Vol. 6 & 9, John Wiley & Sons, Inc., New York, NY, 1988.

[41] M. Kendall, "Natural Law in the Social Sciences", *Journal of the Royal Statistical Society A*, Vol. 124, pp. 1-18, 1961.

[42] T. Kroenger, D. Long and J. Mogul, "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching", *Proceedings of the USENIX Symposium on Internet Technology & Systems*, Monterey, CA, December 1997.

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

[44] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0", RFC 1945, May 1996.
Available at URL: `ftp://ftp.internic.net/rfc/rfc1945.txt`

[45] L. Lange, "The Internet", *IEEE Spectrum*, Vol. 36, No. 1, pp. 35-40, January 1999.

[46] W. Leland, M. Taqqu, W. Willinger and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", *IEEE/ACM Transcations on Networking*, Vol. 2, No. 1, pp. 1-14, February 1994.

[47] A. Luotenen and K. Altis, "World-Wide Web Proxies", *Computer Networks and ISDN Systems*, Vol. 27, No. 2, pp. 147-154, November 1994.

[48] University of Kansas, Lynx Software.
Available at URL: `ftp://ftp2.cc.ukans.edu`

[49] C. Maltzahn, K. Richardson and D. Grunwald, "Performance Issues Of Enterprise Level Web Proxies", *Proceedings of the 1997 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems*, Seattle, WA, pp. 13-23, June 1997.

[50] B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freedman and Co., New York, 1983.

[51] E. Markatos, "Main Memory Caching of Web Documents", *Computer Networks and ISDN Systems*, Vol. 28, Nos. 7-11, pp. 893-905, May 1996.

[52] Microsoft Corporation, Microsoft Internet Explorer.
Available at URL: `http://www.microsoft.com/windows/ie`

[53] Microsoft Corporation, Microsoft Internet Information Server.
Available at URL: `http://www.microsoft.com/ntserver`

[54] J. 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: `ttp://www.research.digital.com/wrl/techreports/abstracts/95.5.html`

[55] J. Mogul, "The Case for Persistent-Connection HTTP", WRL Research Report 95/4, Digital Western Research Laboratory, Palo Alto, CA, May 1995.

[56] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential Benefits of Delta-Encoding and Data Compression for HTTP", *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Cannes, France, pp. 181-194, September 1997.

[57] National Center for Supercomputing Applications, Mosaic Software.
Available at URL: `http://www.ncsa.uiuc.edu/SDG/Software/Mosaic`

[58] D. Neal, "The HARVEST Object Cache in New Zealand", *Computer Networks and ISDN Systems*, Vol. 28, No. 7-11, pp. 1415-1430, May 1996.

[59] S. Nelson and C. Parks, "The Model Primary Content Type for Multipurpose Internet Mail Extensions", RFC 2077, January 1997.
Available at URL: `ftp://ftp.internic.net/rfc/rfc2077.txt`

[60] Netscape Communications Corporation, Netscape's Communicator Software.
Available at URL: `http://www.netscape.com/browsers`

[61] Netscape Communication Corporation, Netscape Application Server.
Available at URL: `http://www.netscape.com/servers`

[62] National Laboratory for Applied Network Research, Hardware for the Caches.
Available at URL: `http://www.ircache.net/Cache/hardware.html`

[63] National Laboratory for Applied Network Research, NLANR sanitized access logs.
Available at URL: `ftp://ircache.nlanr.net/Traces/`

[64] National Laboratory for Applied Network Research, NLANR Cache Status.
Available at URL: `http://www.ircache.net/Cache/status.html`

[65] N. Niclausse, Z. Liu and P. Nain, "A New Efficient Caching Policy for WWW", *Proceedings of the 1998 Internet Server Performance Workshop (WISP '98)*, Madison, WI, pp. 119-128, June 1998.
Available at URL: `http://www.cs.wisc.edu/~cao/WISP98/html-versions/Nicolas/final.html`

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

[67] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modelling", *Proceedings of ACM SIGCOMM '94*, London, England, pp. 257-268, August 1994.

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

[69] J. Peachey, R. Bunt and C. Colbourn, "Some Empirical Observations on Program Behaviour with Applications to Program Restructuring", *IEEE Transactions on Software Engineering*, Vol. 11, No. 2, pp. 188-193, February 1985.

[70] J. Postel, "Transmission Control Protocol", RFC 793, September 1981.
Available at URL: `ftp://ftp.internic.net/rfc/rfc793.txt`

[71] D. Povey and J. Harrison, "A Distributed Internet Cache", *Proceedings of the 20th Autralasian Computer Science Conference*, Sydney, Australia, pp. 175-184, February 1997.
Available at URL: `http://www.dstc.qut.edu.au/MSU/staff/povey/dcache.ps`

[72] L. Rizzo and L. Vicisano, "Replacement Policies for a Proxy Cache", Technical Report RN/98/13, Department of Computer Science, University College London, 1998.
Available at URL: `http://www.iet.unipi.it/~luigi/caching.ps.gz`

[73] C. Roadknight, I. Marshall and D. Vearer, "File Popularity Characterisation", *Proceedings of the 2nd Workshop on Internet Server Performance (WISP 99)*, Atlanta, Georgia, May 1999.
Available at URL: `http://www.cc.gatech.edu/fac/Ellen.Zegura/wisp99/papers/roadknight.ps`

[74] J. Robinson and M. Devarakonda, "Data Cache Management Using Frequency-Based Replacement", *Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Boulder, CO, pp. 134-142, May 1990.

[75] M. Satyanarayanan, "Scalable, Secure, and Highly Available Distributed File Access", *IEEE Computer*, Vol. 23, No. 5, pp. 9-21, May 1996.

[76] N. Smith, "The UK National Web Cache: A State of the Art Report", *Proceedings of the Fifth International World Wide Web Conference*, May 1996.

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

[78] Squid Internet Object Cache, Squid Proxy Software.
Available at: `http://squid.nlanr.net/`

[79] A. Tanenbaum, *Computer Networks*, Second Edition, Prentice Hall, Englewood, NJ, 1988.

[80] R. Tewari, M. Dahlin, H. Vin and J. Kay, "Design Considerations for Distributed Caching on the Internet", *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Austin, Texas, May 1999.
Available at URL: `http://www.cs.utexas.edu/users/dahlin/papers/icds99-WANCache.ps`

[81] J. Vass, J. Harwell, H. Bharadvaj, and A. Joshi, "The World Wide Web: Everything you (n)ever wanted to know about its servers", *IEEE Potentials*, Vol. 17, No. 4, pp. 33-34, October/November 1998.

[82] D. Wessels and K. Claffy, "Internet Cache Protocol (ICP), Version 2", July 1997.
Available at URL: `http://ds.internic.net/internet-drafts/draft-wessels-icp-v2-03.txt`

[83] D. Wessels, "Configuring Hierarchical Squid Caches", Squid Internet Object Cache, August 1997.
Available at URL: http://squid.nlanr.net/Squid/Hierarchy-Tutorial/tutorial.html.

[84] D. Wessels, "Information Resource Caching FAQ", Squid Internet Object Cache, August 1997.
Available at URL: http://www.ircache.net/Cache/FAQ/ircache-faq.html.

[85] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox, "Removal Policies in Network Caches for World-Wide Web Documents", *Proceedings of the 1996 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communnications*, Stanford, CA, pp. 293-305, August 1996.

[86] C. Williamson and R. Bunt, "Characterizing Short Term File Referencing Behaviour", *Proceedings of the Fifth Annual IEEE Phoenix Conference on Computers and Communications (IPCC '86)*, Phoenix, AZ, pp. 651-660, March 1986.

[87] D. Willick, D. Eager, and R. Bunt, "Disk Cache Replacement Policies for Network File Servers", *Proceedings of the 13th International Conference on Distributed Computing Systems (ICDCS)*, Pittsburgh, PA, pp. 2-11, May 1993.

[88] C. Wills and M. Mikhailov, "Examining the Cacheability of User-Requested Web Resources", *Proceedings of the 4th International Web Caching Workshop*, San Diego, CA, March/April 1999.
Available at URL: http://www.cs.wpi.edu/~mikhail

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

Appendix A

Glossary

1. `Cookie`: a small piece of information sent by a Web server to be stored on a Web browser, so that it can later be read back from the browser, the next time the user visits the Web site.

2. `Common Gateway Interface (CGI)`: a technology that allows a computer program to dynamically create Web documents in response to a client's request.

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

4. `Hypertext`: is used to organise documents on the Web and consists of texts and embedded links to other Web documents.

5. `HyperText Markup Language (HTML)`: the language used for writing Web documents.

6. `HyperText Transfer Protocol (HTTP)`: a simple application layer protocol that runs over Transmission Control Protocol/Internet Protocol, and provides reliable communication between Web clients and servers.

7. `Internet`: the collection of computers, networks, and routers that use the TCP/IP protocol suite and function as one large network.

8. `Internet Cache Protocol (ICP)`: a protocol used for efficient communication between Web caches.

9. `Independent Reference Model (IRM)`: a model that considers the elements of a reference stream to be an unending sequence of independent random variables with stationary probabilities.

10. `Least Recently Used (LRU)`: a replacement policy that removes the least recently referenced items from its cache.

11. `Least Frequently Used (LFU)`: a replacement policy that removes the least frequently referenced item from the cache.

12. `Transmission Control Protocol (TCP)`: a reliable transport layer protocol that transmits the application layer data stream in terms of small fragments, called packets.

13. `Universal Resource Locators (URL)`: a character string which easily allows the Web client to identify a particular Web document.

14. `World-Wide Web (WWW)`: a large distributed system based on the client-server architecture deployed on the Internet.

15. `Web client`: a program used to retrieve documents from Web servers.

16. `Web proxy`: a program which accepts document retrieval requests from a set of clients, forwards these requests to the appropriate Web servers (if required), and transmits the requested documents back to the clients. Proxies are used to allow restricted access to the Internet and for caching frequently requested Web documents. This term is often used to refer to the machine that runs the proxy server software.

17. `Web server`: a program that supplies the documents requested by clients by searching its repository of documents. This term is also used to refer to machine that runs the Web server software.