Caching Proxies: Limitations and Potentials

17 July 1995

Marc Abrams*, Charles R. Standridge**, Ghaleb Abdulla*, Stephen Williams*, Edward A. Fox*

*Computer Science Department, Virginia Tech, Blacksburg, VA 24061-0106, USA

**Department of Industrial Engineering, FAMU-FSU College of Engineering, Tallahassee, FL 32310, USA

{abrams,fox}@vt.edu, stand@evax12.eng.fsu.edu, {abdulla,williams}@csgrad.cs.vt.edu

---

Abstract

As the number of World-Wide Web users grows, so does the number of connections made to servers. This increases both network load and server load. Caching can reduce both loads by migrating copies of server files closer to the clients that use those files. Caching can either be done at a client or in the network (by a proxy server or gateway). We assess the potential of proxy servers to cache documents retrieved with the HTTP protocol. We monitored traffic corresponding to three types of educational workloads over a one semester period, and used this as input to a cache simulation. Our main findings are (1) that with our workloads a proxy has a 30-50% maximum possible hit rate no matter how it is designed; (2) that when the cache is full and a document is replaced, least recently used (LRU) is a poor policy, but simple variations can dramatically improve hit rate and reduce cache size; (3) that a proxy server really functions as a second level cache, and its hit rate may tend to decline with time after initial loading given a more or less constant set of users; and (4) that certain tuning configuration parameters for a cache may have little benefit.

Keywords: Proxy server, caching, workload characterization

---

1. Introduction

Without caching, the WWW would become a victim of its own success. As Web popularity grows, so does the number of clients accessing popular Web servers, and so does the network bandwidth required to connect clients to servers. Trying to scale network and server bandwidth to keep up with client demand is an expensive strategy.

An alternative is caching. Caching effectively migrates copies of popular documents from servers closer to clients. Web client users see shorter delays when requesting a URL. Network managers see less traffic. Web servers see lower request rates.

A cache may be used on a per-client basis or within networks used by the Web. We study the second alternative, known as a "proxy server" or "proxy gateway" with the ability to cache documents. We use the term "caching proxy" for short.

A caching proxy has a hard job. First, its arrival traffic is the union of the URL requests of many clients. For a caching proxy to have a cache hit, the same document must either be requested by the same user two or more times, or two different users must request the same document. Second, a caching proxy often functions as a second (or higher) level cache, getting only the misses left over from Web clients that use a per-client cache (e.g., Mosaic and Netscape). The left over misses usually do not contain the case of a document requested twice by the same user, so a caching proxy is left to cache documents requested by two or more users. This reduces the fraction of requests that the proxy can satisfy from its cache, known as the *hit rate*.

How effective could a caching proxy ever be? To answer this, we first examine how much inherent duplication there is in the URLs arriving at a caching proxy. We simulate a proxy server with an infinite disk area, so that the proxy contains forever every document ever accessed. This gives an upper bound on the hit rate that a real caching proxy could ever achieve. The input to the simulation is traces of all URL accesses during one semester of three different workloads from a university community. Overall we observe a 30%-50% hit rate. We also examine the maximum disk area required for there to be no document replacement. We then consider the case of finite disk areas, in which replacement must occur, and compare the hit rate and cache size resulting from three replacement policies: least recently used (LRU) and two variations of LRU. LRU is shown to have an inherent defect that becomes more pronounced as the frequency of replacements rises. Finally we use the best replacement policy and examine the effect on hit rate and cache size of restricting what document sizes to cache and whether to cache only certain document sizes, document types, or URL domains.

1.1 Caching in the World-Wide Web

Caching is used in two forms in the Web. The first is a *client cache*, which is built into a Web browser. A Web browser with caching stores not only the documents currently displayed in browser windows, but also documents requested in the past. There are two forms of client caches: persistent and non-persistent. A persistent client cache retains its documents between invocations of the Web browser; Netscape uses a persistent cache. A non-persistent client cache (used in Mosaic) deallocates any memory or disk used for caching when the user quits the browser.

The second form of caching, explored here, is in the network used by the Web (i.e., the caching proxy mentioned earlier). The cache is located on a machine on the path from multiple clients to multiple servers. Example of caching proxies are the CERN proxy server [LUOT94b], the DEC SRC gateway [GLAS94], the UNIX HENSA Archive [SMIT94], and in local Hyper-G servers [AFMS95]. Normally a caching proxy is not on a machine that runs a WWW client or an HTTP server. The caching proxy caches URLs generated by multiple clients. It is possible to use caching proxies hierarchically, so that caching proxies cache URLs from other caching proxies. In this case the caches can be identified as first level caches, second level caches, and so on. A hierarchical arrangement is only one possible configuration; Smith [SMIT94, Fig. 1] gives a non-hierarchical arrangement of large national and international caching proxies.

1.2 Why a Simulation Study?

One way to study cache performance is simply to configure a caching proxy, and to collect the desired statistics. In fact we did this with the workloads in this paper in another study [ABRA95]. However, this has some problems. First, the WWW at present has no mechanism to tell if a cached document is not the latest version from the original server. (See [GLAS94,SMIT94] for a discussion of the problem.) So studies that use real user workloads can never cache all documents to measure the maximum possible hit rate, because the users cannot live with outdated Web pages. Thus [ABRA95] and [GLAS94] only cache "non-local" documents (which are documents outside the Computer Science department in [ABRA95] and outside DEC SRC in [GLAS94]). Also [SMITH] requires the user to prefix URLs with an explicit caching proxy name to request a cache search. Our simulation study here "replays" the URL requests recorded from our workload, and hence we can cache all URLs. A second reason for simulation is to compare different policies for cache management. A policy might restrict caching to documents of certain sizes or media types. A policy is also required for replacing documents in a full cache on identical URL traces. Finally a simulation can be used in a classic experiment design with analysis of variance to identify what effect a number of factors (e.g., cache size, replacement policy, domain, and document size and types cached, workload) have on hit rate. Such an experiment and analysis is performed here.

1.3 Workload Used in This Study

A critical issue in any performance study is the workload selected for the study. For this paper, we recorded all HTTP URL requests made to proxy servers during one semester (Spring 1995) from three groups of users on the Virginia Tech campus. (Some days are missing from each workload due to problems in data collection, such as proxy server crashes.) The workload tracing procedure is discussed in the Appendix. The three workloads are:

"Undergrad (U)":
: workstations in an undergraduate computer science lab representing about 150 different user ids (however two of the ids were "guest" and "root", which means that more than 150 students used a Web browser), containing 79,718 accesses to the caching proxy;

"Graduate (G)":
: a popular host used by graduate computer science students, of whom at least 25 students used a Web browser, containing 55,186 accesses; and

"Classroom (C)":
: 26 workstations in a classroom on which students run a Web browser during sessions of a class on multimedia, containing 31,812 accesses.

Workloads U and G represent a mixture of Web usage for classes; for general or recreational Web browsing; and, in workload G, for research. One distinction between workloads U and G is that material for many courses that the undergrads take is provided through the Web; in fact four courses are "paperless," and in one even exams are taken through the Web. Workload C differs from the others in that the instructor often directs students to look at certain URLs or search for certain topics, and hence there should be a definite correlation between the URLs that different clients request. We previously used these traces in a study to characterize the workload (e.g., distribution of document sizes and types accessed, distribution of Web server sites accessed) and evaluate cache hit rates using the CERN proxy server [ABRA95].

The three workloads represent what university campus Web clients do. This is useful to complement other workloads studied in the literature (e.g., in Section 4). In our campus workload students are free to access any URL in the World-Wide Web for instructional, research, or recreational purposes, while a caching proxy in an industrial setting often restricts employee access to off-site material [GLAS94].

1.4 Caching Algorithms Studied

What document should be replaced when the cache is full and a request for a URL not in the cache arrives? This question is more complex than the equivalent question for virtual memory and computer cache memory, because the documents cached typically have *different* sizes. Should many small documents be removed for a large document? If cache size is limited, is it better to hold many small documents or a few large documents?

We compare three cache replacement policies in this paper, described below. Suppose the proxy receives a request for a URL that is not in the cache. Let S denote the size of the document corresponding to the URL.

LRU
: Classic least recently used [TANE92]: When the free space in the cache is smaller than S, repeat the following until the free cache space is at least S: replace the LRU document. (LRU may discard many small documents to make room for one large document.)

LRU-MIN
: A variant of LRU that tries to minimize the number of documents replaced. Let L and T denote, respectively, a list and an integer. (1) Set T to S. (2) Set L to all documents equal to or larger than T. (L may be empty.) (3) Remove the LRU of list L until the list is empty, or the free cache space is at least T. (4) If the free cache space is not at least S, set T to $T/2$ and goto step (2).

LRU-THOLD
: A variant of LRU that avoids the situation in which a document that is large compared to the cache size causes replacement of a large number of smaller documents. This policy is identical to LRU, except that no document larger than a threshold size is cached. (Even if the cache has room, a document whose size is larger than the threshold is never cached.)

2. Experiment Objectives and Design

2.1 Objectives

We want to assess:

1. What is the maximum cache hit rate that user access patterns allow, given an infinite cache size?
2. What is the cache size that various workloads require, if the cache is to have no replacement?
3. What is the mean *lifetime* of documents replaced in the cache (e.g., the time from when a document enters the cache until it is replaced, averaged over all replaced documents)?
4. How much effect do each of the following factors have on hit rate and cache size: disk area allocated for the cache, minimum document size cached, domains cached, document type cached, workload characteristics, and cache replacement policy.
5. What cache parameter settings are best?

2.2 Experiment Design

We perform four experiments, whose factors and levels are summarized in Table 1. We use the following notation to refer to the factor levels. The term "MaxNeeded" in the table refers to the disk area needed for no replacements to occur from Experiment 1 (listed later in Table 2). A "text" document, in factor "Document type cached," is one with either no file name extension or the extension .txt or .html. The workload mnemonic (i.e., U,G,C) is suffixed by /A or /S (e.g., U/A, C/S) to denote use of the all versus some of the eight day intervals into which the workload is partitioned in Section 3.1. Finally the workload mnemonic is suffixed by /MIN and /MAX (e.g., U/MIN) to denote the single eight day interval of the workload that produced, respectively, the minimum or maximum disk area needed for no replacements in Experiment 1.

The response variables referred to in Table 1 are the mean hit rate averaged over all cache accesses, the cache size needed for no replacements to occur, and the mean lifetime of replaced documents.

Table 1: *Factor-level combinations for all experiments.*

	Levels		
	Experiment 1	Experiment 2	Experiment 3
Factors:			
Disk area for cache	infinite	10, 50% of MaxNeeded	10, 50% 90% of MaxNeeded
Min. doc. size cached	0	0	0
Domain cached	all	all	all
Document type cached	all	all	all
Workload	U/A,G/A,C/A	U/S,C/S	U/S,C/S
Replacement policy	N/A	LRU-THOLD	LRU,LRU-MIN,LRU-THOLD
LRU-THOLD threshold	N/A	1,4,16,32K;1M	best in exp. 2
Output measures:	Hit rate Max cache size	Hit rate Max cache size Lifetime	Hit rate Max cache size Lifetime

	Experiment 4	
	Level -1	Level +1
Factors:		
Disk area for cache	50% MaxNeeded	50% MaxNeeded
Min. doc. size cached	1 kbyte	0 kbytes
Domain cached	off campus only	all
Document type cached	non-text only	all
Workload	U/S	C/S
Replacement policy	best in exp. 3	best in exp. 3
LRU-THOLD threshold	N/A	N/A
Output measures:	Hit rate Max cache size	

Each of the four experiments is described below.

Experiment 1: Maximum Possible Hit Rate for a Caching Proxy

The purpose of Experiment 1 is to identify the maximum possible hit rate for our three different workloads, assuming an infinite size cache. This represents the inherent "cachability" of client URL requests, regardless of the cache design. We used a one factor experiment design. The single factor varied is "Workload."

It will be seen in Section 3.1 that workloads U and G are somewhat similar; of these workload U exhibits the largest dynamic range of cache sizes. To reduce the number of trials required for Experiments 2 to 4, we only consider workloads U and C.

Experiment 2: Optimal Threshold for the LRU-THOLD Replacement Policy

In the next experiment, we try to answer which replacement policy (LRU, LRU-MIN, or LRU-THOLD) is best under which circumstances. But one of the three policies -- LRU-THOLD -- requires a parameter (i.e., the threshold). To do this, we must know what threshold value to use. This is the object of Experiment 2. We used a full factorial experiment with three factors (Disk area, Workload, and Threshold) and two replicas (with workloads U/MIN and U/MAX if factor "Workload" is U, or

C/MIN and C/MAX if factor Workload is C). This led to 40 simulation runs.

Experiment 3: Comparison of Cache Replacement Algorithms

The purpose of Experiment 3 is to compare four cache replacement algorithms. We used a three factor (Disk area, Workload, Replacement policy), full factorial experiment design, with two replications (just as in Experiment 2), requiring 16 simulation runs.

Experiment 4: Performance Impact of Caching

The purpose of Experiment 4 is to assess the effect of factors *other than* the replacement policy on hit rate and cache size. The factor varied are listed in Table 1. We used a (2**4)(2) experiment design, requiring 32 simulation runs. The experiment includes 2 replications as described in Experiment 2, and four factors with two levels (Disk area, Minimum document size cached, Domain cached, and Document type cached).

3. Experiment Results

3.1 Experiment 1

Due to a limitation in the language used for the simulation tool (SLAM II [PRIT87]), the simulation could not cache more than 8600 documents, which limited us to simulating eight days of traffic in the busiest case. Therefore we broke the observation period into eight day intervals, and within each eight day interval the maximum number of unique documents accessed is 8600.

Recall that the purpose of Experiment 1 is to determine the maximum possible hit rate, given an infinite disk area for caching. Figures 1 and 2 report, respectively, the hit rate and the cache size for no replacements for each interval and workload. It is evident from Figures 1 and 2 that workload C contains fewer days of trace data than the other workloads. This is because the class met four days a week, while the other workloads were collected seven days a week. Also workload C is not a multiple of eight days long, so when we divided it into eight day periods, we ignored the first few days to achieve a multiple of eight days. Finally Figures 1 and 2 show that we did not start collecting workload U until after the semester began.

Table 2 contains summary statistics of the data. The data in Table 2 suggest that if a separate caching proxy had been used for each of the three workloads, and the proxy was restarted with an empty cache every eight days, then over the course of the spring semester one would see the following: The cache hit rate for the two laboratory workloads -- U and G -- would be statistically indistinguishable, with a similar mean (30% and 29%, respectively), variance, and minimum and maximum hit rates. The ratio of the best to the worst hit rate observed in any eight day interval in the semester is roughly 2 to 1 for both workloads. Furthermore workload C has a 50% higher hit rate because the behavior of Web users are correlated in class, as mentioned in Section 1.3.

As for cache size required for no replacements, it is surprising that the the sample mean size was similar for all three workloads, ranging from 75 Mbytes to 86 Mbytes. However the standard deviation for workload G was considerably smaller than the other two workloads. The largest difference in cache size exhibited by a workload was the ratio of about 6 to 1 for the maximum to the minimum cache size required for no replacement for all eight day periods of workload U.

The results above correspond to a situation where a caching proxy is restarted with an empty initial cache every eight days. To explore the effect of interval length (eight days) on the results, we resimulated two of the workloads with longer intervals (12 days). Table 3 reports the statistics.

Recall that whenever a client uses a private cache, our logs will contain only those URLs that are a private cache miss. So how effective is a per-client cache in learning a user's behavior? Figure 1 provides some insight. In each of the three hit rate curves in Figure 1, most line segments have a negative slope; in other words, most of the time the hit rate is more likely on any day to go *down* rather than *up* compared to the previous day. While this might be a purely random phenomena, we speculate that there is an explanation. As stated earlier, a proxy cache is really a second level cache, if clients have their own cache. The proxy cache is only sent misses on the (first level) client caches. If the client cache is persistent between invocation of clients (i.e., the client cache uses disk), then the proxy cache hit rate should decline over time as the per-client cache fills (or, in some sense, adapts to user behavior). In Figure 1 the classroom curve has a smaller slope than the negative slope portions of the two other curves. This may be due to our resetting the client disk and memory cache size to zero before many classes. (We found that 10-20% of the workstation users turned the cache on between our resetting them to zero.) In contrast, no attempt was made to defeat client caching in workloads U and G. Finally the only of the three curves that contains more than one sharp increase in hit rate (i.e., the curve for workload U) does not start until about 1/3 the way into the spring semester; by this point the client caches had several weeks to fill, so that the data skips the period when the proxy cache hit rate declines rapidly due to the client cache filling.

Figure 1: *Cache hit rate as a function of measurement interval. The data collection period started on January 20, 1995, which is denoted by x-axis coordinate zero.*

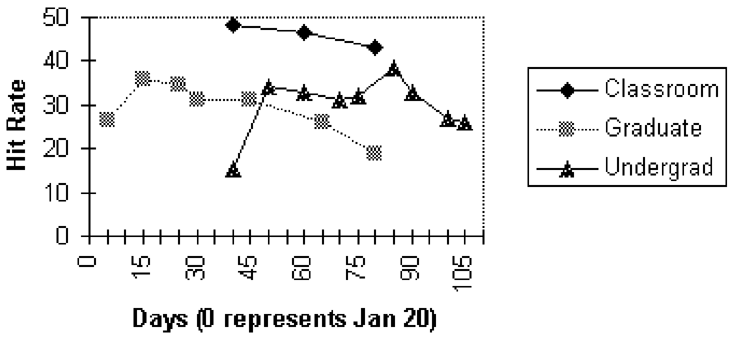


Figure 2: *Cache size required for no document replacements, as a function of measurement interval*

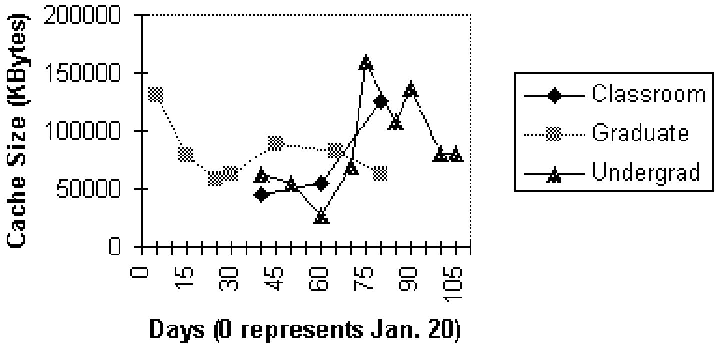


Table 2: *Statistics of hit rate and cache size for no replacement for Experiment 1 with an eight day interval*

	Undergrad		Graduate		Classroom	
	Hit Rate (%)	Cache Size (Mbytes)	Hit Rate (%)	Cache Size (Mbytes)	Hit Rate (%)	Cache Size (Mbytes)
Sample Mean	30.0	86.497	29.2	80.780	46.0	75.249
Std. Dev.	6.6	41.465	5.9	24.631	2.7	44.123
Minimum	15.4	27.525	18.8	58.680	43.1	44.902
Maximum	38.6	159.930	36.1	129.969	48.4	125.865

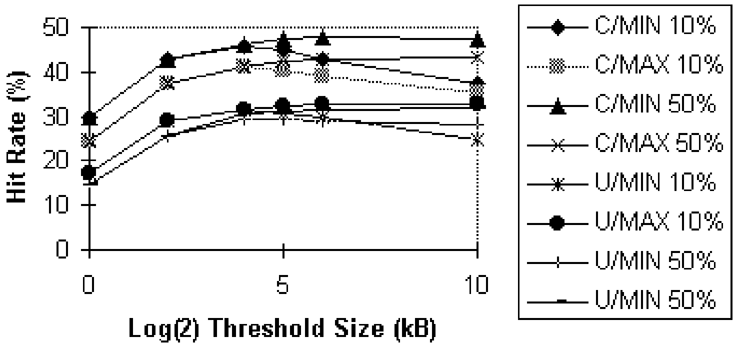
Table 3: *Statistics of hit rate and cache size for no replacement for Experiment 1 with a 12 day interval*

	Graduate		Classroom	
	Hit Rate (%)	Cache Size (Mbytes)	Hit Rate (%)	Cache Size (Mbytes)
Sample Mean	35.5	99.037	50.0	105.779
Std. Dev.	5.7	8.841	1.1	50.263
Minimum	27.4	91.906	49.2	70.238
Maximum	40.8	110.459	50.8	141.321

3.2 Experiment 2

Recall that the purpose of Experiment 2 is to determine the optimal value of the threshold to use for the LRU-THOLD document replacement policy. "Optimal" means the value that maximizes hit rate. Figure 3 contains the experiment results.

Figure 3: *Cache hit rate with LRU-THOLD replacement policy as a function of threshold values. Percentages (e.g., 10%) refer to sizes of disk area used for caching, as a fraction of MaxNeeded from Experiment 1.*



Analysis of variance (ANOVA) applied separately to data from each workload reveals that at alpha = 5% only the threshold level has a statistically significant effect on hit rate, explaining 90% and 77% of the variation in hit rates for workloads U and C, respectively.

Three of the four curves for each workload in Figure 3 reach a maximum value and then decline. The optimal threshold for all workloads is 2**4 Kbytes when the disk area is 10% of the cache size required for no replacements. When the disk size available for caching is 50% of the cache size required for no replacements, the optimal threshold is 2**6 Kbytes for workloads C/MIN and U/MIN, and 2**10 Kbytes for workload C/MAX and U/MAX.

3.3 Experiment 3

Recall that Experiment 3 compares document replacement policies LRU, LRU-MIN, and LRU-THOLD, to identify which maximizes hit rate and minimizes disk size required for the cache. For LRU-THOLD, we used the thresholds listed in Section 3.2. The experiment results are tabulated in Table 4.

In no case did LRU outperform the other replacement policies in Table 4. LRU-MIN achieved a higher hit rate than LRU-THOLD in five cases, and LRU-THOLD achieved a higher hit rate than LRU-MIN in four cases. Given that the optimal threshold for LRU-THOLD is a function of the workload and the ratio of the disk size available for the cache compared to the cache size needed for no replacement, LRU-MIN is the best policy: it requires no parameters and achieves the best performance most of the time. On the other hand, LRU-THOLD achieves dramatically smaller cache sizes with a small penalty in hit rate compared to LRU-MIN; thus LRU-THOLD is recommmended when disk size is limited compared to the cache size required for no replacement. Finally, the lifetimes reported in the table show that a document stays in the cache for greatly varying times -- as much as ten times longer for LRU-THOLD compared to LRU. The short lifetimes for LRU might explain why its hit rates are never higher than the other policies: documents are replaced too frequently.

Table 4: *Comparison of document replacement policies LRU, LRU-MIN, and LRU-THOLD with three different sizes of disk areas for caching (i.e., 10%, 50%, and 90% of MaxNeeded) using three performance measures. Zero entries for lifetime indicate that no document was ever replaced during simulation. Asterisks denote optimal values, unless all three policies yielded the same value.*

	Hit Rate (%)		Cache Size (Mbytes)		Lifetime (Hours)	
	U/MIN	U/MAX	U/MIN	U/MAX	U/MIN	U/MAX
10% MaxNeeded						
LRU	23.8	26.4	3.0	13.8*	1.3	15.0
LRU-MIN	31.0	28.1	3.0	13.8*	2.8	47.3
LRU-THOLD	31.5*	29.5*	3.0	16.0	7.2	151.6
50% MaxNeeded						
LRU	31.2	30.3	13.8	80.0	4.3	104.1
LRU-MIN	32.9*	31.0	14.0	76.5*	7.7	119.2
LRU-THOLD	32.9*	32.1*	10.0*	80.0	0.0	178.9
90% MaxNeeded						
LRU	32.9	32.1	24.8	144.0	8.7	170.3
LRU-MIN	33.0*	32.1	25.0	144.0	10.0	45.3
LRU-THOLD	32.9	32.1	10.0*	85.9*	0.0	0.0
	C/MIN	C/MAX	C/MIN	C/MAX	C/MIN	C/MAX
10% MaxNeeded						
LRU	35.4	30.6	5.0	13.0	80.9	32.2
LRU-MIN	46.3*	41.1	5.0	13.0	291.0	42.0
LRU-THOLD	46.0	41.2*	5.0	13.0	629.8	324.4
50% MaxNeeded						
LRU	46.0	38.7	22.9	61.4	498.6	76.3
LRU-MIN	48.1*	43.1*	23.0	62.8	647.1	225.7
LRU-THOLD	48.1*	42.7	14.8*	26.2*	0.0	0.0
90% MaxNeeded						
LRU	48.4*	42.6	40.9	114.0	838.2	278.6
LRU-MIN	48.4*	43.1*	41.0	111.5	840.1	297.9
LRU-THOLD	48.1	42.7	14.8*	26.2*	0.0	0.0

ANOVA on each workload's hit rate reveals the following. For workload U, the disk size and replacement policy were statistically significant (p=0.001 and 0.018), with the disk size explaining most of the variation in hit rate (58%), and the policy explaining 18%. The interactions of disk size and hit rate were not significant. Error (variances among the two eight day intervals simulated) accounted for 12% of the variation. Thus the disk size for caching has a much larger influence on hit rate than the replacement policy, but replacement policy is more important than variation due to differences in observation period. In contrast, in workload C no factor was statistically significant, with the variances among the two eight day intervals simulated explaining the most variation in hit rate (about 1/3).

3.4 Experiment 4

Recall that Experiment 4 investigates how various cache configuration parameters impact the hit rate and cache size. Table 5 lists the combination of factors that lead to the highest four hit rates. Interestingly, the workload did not matter in that the same factor-level combinations lead to the same position in the ranking table. The table also lists the resultant cache sizes, which varied no more than 10% from the maximum to the minimum observered. The conclusion from this data suggests that it is best to cache everything -- regardless of document type, document size, or domain. The penalty for not caching everything can be a dramatic drop in hit rate -- by at least 10% for workload C if we do anything else, and by almost half for workload U if we do not cache all domains and all document sizes.

Table 5: *Ranking of factor levels producing top four hit rates for each workload, with corresponding cache sizes*

Document type cached	Domain cached	Min. document size cached	Hit Rate U/S	Hit Rate C/S	Cache Size U/S	Cache Size C/S
+1	+1	+1	32.0%	45.0%	45.2Mb	42.9Mb
-1	+1	+1	30.0%	34.8%	40.0Mb	42.6Mb
+1	+1	-1	18.5%	34.2%	45.2Mb	43.0Mb
-1	+1	-1	16.8%	25.7%	40.0Mb	42.6Mb

Applying the analysis of variation technique to our data shows that all factors and factor interactions are statistically significant at alpha = 5%. Factors "Domain cached" and "Minimum document size cached" explain most of the variation in hit rate (72% and 18% for workload U and 75% and 9% for workload C, respectively). Factor "Document type cached" accounted for almost no variation in hit rate only in workload U, but 7% in workload C. All factor interactions and the error accounted for smaller percentages. Based on this, the decision of caching all documents, not just off campus documents, is the most important. In fact, the hit rate is 16.8% and 20.0% higher in workloads U and C, respectively, when on-campus documents are also cached compared to not caching them. This illustrates how critical it is to find a solution to the problem of identifying when cached documents are out of date with the HTTP proctocol to permit caching of all domains. On the other hand, caching text files provided little benefit to workload U (increasing hit rate only 1.3%), but was more critical in the workload C (increasing hit rate 6%).

4. Comparison to Other Cache Studies

We are aware of three studies in the literature that report cache performance: for the HENSA archive [SMIT94], for the DEC gateway [GLAS94], and for a proposed cache algorithm [PITK].

The cache size simulated in Experiment 1 is comparable to the data reported by Smith for the HENSA archive in March 1994 [SMIT94], which contained 7554 documents occupying 184 Mbytes of disk space (versus up to 8660 documents and a maximum of 159 Mbytes of disk space in Experiment 1). Given this, how do the hit rates compare? Smith reports that "on a typical working day we [the UNIX HENSA ARCHIVE] expect to serve at least 60% of requested documents out of the cache" [SMIT94]. Smith’s 60% hit rate is above the means of around 29-30% we found for workloads U and G and the 46% we found for workload C. However some differences exist. In the HENSA case, a user had to prefix a URL with the proxy name for caching to be activated. Therefore the HENSA server saw a subset of all user traffic. In contrast, our simulation study includes *all* URLs that clients send. Also, as noted in the Introduction, some clients in workloads U and G used a client cache, which reduces the hit rate in the caching proxy. Smith does not state if clients accessing the HENSA archive used caching; if they did not, this would be one reason that the HENSA hit rate is higher.

Glassman reports a 30%-50% hit rate for the DEC SRC gateway [GLAS94], which looks similar to the 29%-50% range in Tables 5 and 6. However the DEC gateway caches Gopher traffic, unlike our study, and the DEC gateway caches only non-local pages, whereas we cache all pages. Finally the number of users accessing the DEC gateway is much larger than in our study. Glassman asks in his paper, "Would a much larger or much smaller user population give different results?" As far as hit rate is concerned, the answer from our study is "no." Interestingly, the 75-86 Mbyte cache sizes we observed in eight day intervals are about the same as the cache size in the first eight day period of [GLAS94, Fig. 1]. (Apparently our caching of local files compensated for the fewer users in our workload.)

Recker and Pitkow [RECK94] also report hit rates. Unlike the workloads in this study or those of Smith and of Glassman, which represent all traffic originating from a set of clients, Recker/Pitkow’s workload is all traffic to a single HTTP server (i.e., www.gatech.edu), not to a proxy. Recker/Pitkow report a 67% hit rate for accesses from off-campus for a three month period. The server they studied had fewer documents (2000) than the study here or by Smith or by Glassman, but had a higher access rate than this study. We would expect the Recker/Pitkow hit rate to be higher than that observed here and by Smith and by Glassman because a URL names two fields (a server and a document), and accesses to a server cache are guaranteed to hit on one of the two fields (the server). In contrast accesses to a caching proxy has no such guarantee -- the accesses go to any Web server worldwide.

One other study infers caching proxy performance. Braun and Claffy [BRCL93] investigate the possibility of caching popular NCSA files at several sites around the country. Their study varies from this one in several ways. First, the recorded logs are only for two days of access. Second, the logging was done on a file server, not a proxy server. Third, the server in the study is unique, in that Mosaic clients use NCSA as the default home page. The study does show, however, that by caching a very small subset of client access, a substantial savings in network load can be achieved. The evaluation of NCSA file retrievals showed that the top 25 files requested represented only 0.065% of the different files accessed, but accounted for 59% of the number of requests and 45% of the bytes transferred.

5. Conclusions and Future Work

Our simulation study of three workloads (from an undergrad lab, graduate users, and a classroom with workstations for all students) reveals the following insights into caching proxy design:

1. We confirmed previous data in the literature that a caching proxy has an upper bound of 30-50% in its hit rate, given an infinite size cache.
2. With our workloads, a caching proxy's hit rate tends to *decline* with time. We hypothesize that the reason is that when Web browsers use their own caches, the proxy acts as a second level cache. The caching proxy hit rate declines as the Web browsers' caches fill over time. The effect is more pronounced if most Web browsers used have persistent caches, such as the Netscape browser provides.
3. When the cache is full and documents must be discarded to make room for a new document, LRU could never outperform the other policies that we studied. LRU ignores the sizes of documents. Thus to make room for, say, a large video for one user, it might replace many small documents resulting in misses for many users. A more effective policy is LRU-MIN, which applies LRU only to the largest documents, and then to groups of successively smaller documents. If the disk size becomes small compared to the cache size needed for no replacement to occur, then LRU-THOLD, which does not cache documents larger than a certain threshold, keeps the cache size moderate but yields performance comparable to LRU-MIN. The disadvantage of LRU-THOLD is that our study shows the optimal threshold to be primarily dependent on the workload and the disk size available for the cache. Finally one could use an adaptive policy: use LRU-MIN until the cache size approached 100% of the available disk size, and then switch to LRU-THOLD with a threshold that is gradually reduced until the cache size reaches a low water mark.
4. Our study reiterates the observation of Luotonen and Altis: "Caching is more effective on the proxy than on each client. This saves disk space since only a single copy is cached, and also allows for more efficient caching of documents that are often referenced by multiple clients as the cache manager can predict which documents are worth caching for a long time and which are not ... because it has many clients and therefore a larger sample size to base its statistics on" [LUOT94b]. This is particularly true for machines that are used in a laboratory, in which case a single disk shared among users may have many copies of the same documents! A more effective solution may be to use a *non-persistent* per-client cache when a caching proxy is available. (At the end of the semester in which we collected data, we deleted 0.5 Gbyte of per-client caches amassed by 150 user ids.)
5. Caching all documents -- not just ones in a particular domain, such as those outside one's orgainization -- dramatically increases hit rate. In our study, hit rate increases 16.8% to 20% when all document domains are cached.
6. A caching proxy is likely to have a variety of configuration parameters. We examined three: Should all document types, or just non-text be cached? Should all document sizes be cached, or only ones above some minimum size? Should all domains be cached, or just off campus? Our results are independent of workload. Naturally caching more files maximizes hit rate. However, caching just non-text files produces the smallest drop in hit rate, but it is small for one workload and large for another. Not caching small (less than 1 Kbyte) documents makes the next drop, and not caching documents that are text or small makes the next drop in hit rate. Interestingly, the cache size varies no more than 10% for all these alternatives, suggesting that such "tuning" of a caching proxy is ineffective.

One caveat is in order for our study: to some degree we overestimate hit rate because we can not distinguish which URLs represent CGI script generated documents, and hence should not be cached.

Our recommendation for someone configuring a cache for a situation similar in nature to our educational workloads (e.g., a department with workloads of up to 150 users) is that a modest disk size for caching per workload is required -- 160 Mbytes was adequate for all cases in our study. One should expect about 1/3 of the requests outside the classroom will be cache hits and about 1/2 of the requests in the classroom will be hits. The best strategy is to make the disk area for the cache sufficiently large to avoid replacement. When the cache becomes full and replacements must occur, use -- if available -- a replacement policy that tries to replace documents of comparable size to the incoming document.

Listed below are some open problems that need exploration:

- How effective are *multi-level* caching proxies?
- Caches require some mechanism of insuring that cached documents are up-to-date, such as issuing a CONDITIONAL GET from the proxy to the source HTTP server to verify up-to-date cached documents. But these methods increase proxy and network loads, and can increase the delay that a user experiences in retrieving a cached document. A study of the performance impact of various proposed cache consistency policies is needed.
- Luotonen and Altis [LUOT94b] suggest that the HTTP protocol be enhanced to allow multiple URL gets and responses over a single connection, rather than establishing one connection per GET. What impact would this have on the document transfer rate that an Web client user sees?

The full traces of the workloads used in this study; the tool used to analyze and transform the traces into simulation input files, called Chitra95 (described in [ABRA94]); and the Windows-based SLAM II [PRIT87] simulation used in this paper, called WebSim, are available from WWW location http://info.cs.vt.edu/~succeed.

Acknowledgements

We thank Leo Bicknell and Carl Harris for providing the workload U trace logs. This work was supported in part by the National Science Foundation through CISE Institutional Infrastucture (Education) grant CDA-9312611, CISE RIA grant NCR-9211342, and SUCCEED (Cooperative Agreement No. EID-9109853). SUCCEED is an NSF-funded coalition of eight schools and colleges working to enhance engineering education for the twenty-first century.

References

[ABRA95]
M. Abrams, S. Williams, G. Abdulla, S. Patel, R. Ribler, E. A. Fox, "Multimedia Traffic Analysis Using Chitra95," to appear in *ACM Multimedia 95*, San Francisco, Nov. 1995 (URL http://ei.cs.vt.edu/~succeed/95multimediaAWAFPR/95multimediaAWAFPR.html).

[AFMS95]
K.Andrews, F.Kappe, H.Maurer, K.Schmaranz, "On Second Generation Network Hypermedia Systems," *Proc. ED-MEDIA 95, World Conference on Educational Multimedia and Hypermedia,* Graz, Austria, June 17-21, 1995.

[BRCL94]
H. Braun and K. Claffy, "Web Traffic Characterization: An Assessment of the Impact of Caching Documents from NCSA's Web Server," *Proc. 2nd Int. WWW Conference*, Chicago, Oct. 1994 (URL http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/claffy/main.html).

[GLAS94]
S. Glassman, A Caching Relay for the World-Wide Web, In First International World-Wide Web Conference, pages 69-76, May 1994; also appeared in *Computer Networks and ISDN Systems 27*, No. 2, 1994 (URL http://www1.cern.ch/PapersWWW94/steveg.ps).

[LUOT94a]
A. Luotonen, *Configuration File of CERN httpd*, 1994 (URL http://www.w3.org/hypertext/WWW/Daemon/User/Config/Overview.html).

[LUOT94b]
A. Luotonen and K. Altis, "World-Wide Web Proxies", *1st Inter. Conf. on the WWW,* Geneva, May 1994; also appeared in *Computer Networks and ISDN Systems 27*, No. 2, 1994 (URL http://www1.cern.ch/PapersWWW94/luotonen.ps).

[PITK]
J. E. Pitkow and M. M. Recker, "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns," in *Proc. 2nd Int. WWW Conf.*, 1994 (URL ftp://ftp.gvu.gatech.edu/pub/gvu/tech-reports/94-39.ps.Z).

[PRIT87]
Pritsker, A. A. B., *Introduction to Simulation and SLAM II,* 3rd ed., Halsted, New York, 1987.

[RECK94]
M. M. Recker and J. Pitkow, *Predicting Document Access in Large, Multimedia Repositories*, Graphics, Visualization, and Usability Center Tech. Rep. VU-GIT-94-39, Georgia Tech, August, 1994 (URL ftp://ftp.gvu.gatech.edu/pub/gvu/tech-reports/94-35a.ps.Z and 94-35b.ps.Z).

[SMIT94]
N. Smith, *What Can Archives Offer the World-Wide Web,* Univ. of Kent at Canterbury, 22 March 1994.

[TANE92]
A. S. Tanenbaum, *Modern Operating Systems*, Prentice-Hall, 1992, pp. 111-112.

Appendix. Data Tracing Procedure

CERN HTTP servers were run as proxy servers during the spring semester of 1995 at Virginia Tech to collect data on client usage. This data was collected using two methods: using the proxy server log files and using *tcpdump* (http://www-nrg.ee.lbl.gov/ftp.html) to monitor the timings of network packets going into and out of the proxy server. (However the *tcpdump* data is not used in this paper, because we do not report on timings of document access.)

Our users run either Mosaic or Netscape. To log Mosaic client requests, we modified the "Mosaic" command on our systems to invisibly invoke our proxy server. To log Netscape client requests, we asked users in workload G to set their client preferences to use the proxy. In workload U, the hosts were all on an isolated network that was forced to use the proxy server as a gateway to the Internet, thus guaranteeing all client requests were logged. In the classroom the authors manually set the preferences of clients to use the proxy before class.

The CERN HTTP server permits logging of all server activity in either a default or verbose mode; we used the default mode. The log file generated contains one record for each request sent by client to the proxy. A record contains the client machine name, the date and time the proxy received the request, the command (e.g., GET), the source URL, the document size returned, and the command return code. We ran our proxy server with caching turned on, to collect actual data on cache effectiveness for another study ([ABRA94]), but the proxy still logged *all* requests generated by clients in two files (for documents returned by the proxy versus documents returned by the source HTTP server).

*Please send inquiries and commments to **abrams@vt.edu**.*