On Filter Effects in Web Caching Hierarchies

CAREY WILLIAMSON
University of Calgary

This paper studies the "filter effects" that occur in Web proxy caching hierarchies due to the presence of multiple levels of caches. That is, the presence of one level of cache changes the structural characteristics of the workload presented to the next level of cache, since only the requests that miss in one cache are forwarded to the next cache.

Trace-driven simulations, with empirical and synthetic traces, are used to demonstrate the presence and magnitude of the filter effects in a multi-level Web proxy caching hierarchy. Experiments focus on the effects of cache size, cache replacement policy, Zipf slope, and the depth of the Web proxy caching hierarchy.

Finally, the paper considers novel cache management techniques that can better exploit the changing workload characteristics across a multi-level Web proxy caching hierarchy. Trace-driven simulations are used to evaluate the performance of these approaches. The simulation results demonstrate that size-based partitioning and heterogeneous cache replacement policies each offer improvements in overall caching performance. The sensitivity of the results to the degree of workload overlap amongst child-level proxy caches is also studied.

Categories and Subject Descriptors: H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*World Wide Web (WWW)*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance Evaluation*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*; C.4 [**Performance of Systems**]: Measurement Techniques, Modeling Techniques, Performance Attributes

General Terms: Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Performance Evaluation, Simulation, Web Performance, Web Proxy Caching Hierarchies

1. INTRODUCTION

Caching proxies have gained widespread popularity on the Internet [Abdulla et al. 1997; Baentsch et al. 1997b; 1997a; Bestavros et al. 1995; Bolot and Hoschka 1996; Cohen et al. 1998; Zhang et al. 1997]. Proxies function as intermediaries between Web clients (browsers) and Web servers, accepting client requests and forwarding them to Web servers only as neccessary. When a requested document is returned by a Web server, the proxy server sends the document to the client and stores a copy of the document in its local cache. Depending on client request patterns, the proxy may be able to satisfy future client requests directly from the cache without

Author's address: Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada, T2N 1N4 Email: carey@cpsc.ucalgary.ca

contacting the Web servers.

In recent years, multi-level proxy cache configurations have received increasing research attention [Chankhunthod et al. 1996; Fan et al. 1998; Gwertzman and Seltzer 1995; Mahanti et al. 2000; Povey and Harrison 1997]. In a hierarchical configuration, proxies at or near the end-user constitute the lowest level of the hierarchy, often with sibling-sibling relationships with one another. The lowest level proxies may have a child-parent relationship to a higher level proxy, usually a (geographically) regional proxy. A regional proxy can in turn connect to a higher level proxy, such as a national proxy [Mahanti et al. 2000]. A request that cannot be satisfied by one proxy cache can be sent to a nearby sibling or to the parent using an Inter-Cache Protocol [Fan et al. 1998; Wessels and Claffy 1998]. Contacting the origin server to obtain the document serves as the last resort.

Interesting design issues arise with caching hierarchies, and performance trade-offs exist. For example, the potential advantages of reduced server load, reduced network traffic, and reduced end-user latency may be offset by inter-cache communication overhead, delays incurred at each level of the hierarchy, and performance bottlenecks at higher level proxies [Rodriguez et al. 1999].

Empirical measurements suggest that Web proxy caching hierarchies are not that effective [Mahanti and Williamson 1999; Mahanti et al. 2000; Rodriguez et al. 1999]. For example, the measurements reported in [Mahanti and Williamson 1999] for a three-level caching hierarchy indicate document hit ratios of 35-40% for a university-level Web proxy cache, hit ratios of 15-20% for a national-level Web proxy, and hit ratios of 5-10% for a root-level NLANR (National Laboratory for Applied Networking Research) cache. Thus a caching hierarchy can suffer "diminishing returns": the further up the hierarchy you go, the less likely you are to find the document of interest. In many cases, a request to the originating server is eventually needed to resolve the sequence of cache misses incurred.

The diminishing returns phenomenon makes sense intuitively, since the lower-level caches filter out many of the hits. As a result, the workload characteristics seen at higher-level caches become more "random" in nature. The only surprising aspect is that the diminishing returns occur *despite* the fact that higher-level caches are often larger (sometimes significantly larger) than the caches at the lower levels. These observations suggest that caching hierarchies are not that well-designed. Often, too much focus is placed on the performance of a proxy cache in isolation, rather than as part of an overall caching system [Weikle et al. 1998].

The purpose of this paper is to take an overall "system-level" view of Web caching hierarchies, and strive to improve their performance. First, the paper explores the behavioural characteristics of the filter effects in a multi-level Web proxy caching hierarchy, and their structural causes. Second, the paper proposes and evaluates several novel cache management techniques that can better cope with, or even exploit, the structural changes in Web workloads across the levels of a caching hierarchy.

This work uses trace-driven simulations, with empirical and synthetic traces, to study cache filter effects in a Web caching hierarchy. Empirical traces are taken from a university-level Web proxy cache in a Web caching hierarchy [Mahanti et al. 2000]. Synthetic traces are generated using a Web proxy workload generator called

ProWGen, developed in previous work [Busari and Williamson 2001a]. ProWGen traces capture the salient characteristics of Web proxy workloads (e.g., one-time referencing, Zipf-like document popularity, heavy-tailed file size distribution, temporal locality) that are most relevant to Web proxy cache performance [Busari and Williamson 2001a]. For simplicity, only a two-level caching hierarchy is considered, with three types of cache replacement policies: recency-based, frequency-based, and size-based. Synthetic workloads are used to investigate the performance of different combinations of replacement policies at different levels of the hierarchy, and then to investigate a size-based document partitioning approach.

The simulation results demonstrate the presence and magnitude of cache filter effects in a Web caching hierarchy, and their structural causes. The simulation results show that size-based partitioning and the use of different replacement policies at different levels of the hierarchy each improve performance. The sensitivity of the results to the degree of workload overlap amongst child-level proxy caches is also studied.

The remainder of this paper is organized as follows. Section 2 provides some background on Web proxy workloads and related work on Web caching hierarchies. Section 3 illustrates the presence of cache filter effects using trace-driven simulations. Section 4 studies ways to exploit cache filter effects to improve the overall performance of Web caching hierarchies. Finally, Section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

This section provides some background information on Web traffic workload characteristics, which are central to understanding the cache filter effects discussed in this paper. The section concludes with a brief synopsis of related research work on the design and performance of Web caching hierarchies.

2.1 Web Workload Characteristics

Several Web workload characterization studies have appeared in the literature. These empirical studies have focussed on Web client [Cunha et al. 1995], Web server [Arlitt and Jin 2000; Arlitt and Williamson 1997a], and Web proxy workload characteristics [Abdulla et al. 1997; Almeida et al. 1998; Duska et al. 1997; Mahanti et al. 2000].

From these empirical studies, several common workload characteristics emerge that are relevant to Web caching performance. These characteristics include a high degree of *one-time referencing*, a *Zipf-like document popularity distribution*, *heavy-tailed file and transfer size distributions*, and a *temporal locality property* in the document referencing behaviour. These characteristics are summarized as follows:

—One-Time Referencing

Studies of Web server and Web proxy workloads have shown that many documents requested from a server or a proxy are requested only once, regardless of the duration of the access log studied [Abdulla et al. 1997; Arlitt and Williamson 1997a; Mahanti et al. 2000]. These documents are referred to as "one-timers" in the literature. Clearly, there is no benefit to caching one-timer documents, since they are never accessed again. In fact, caching algorithms need to discriminate against such documents so that they do not clutter the cache and reduce its

effectiveness [Arlitt and Williamson 1997a].

One-timers are important because of their prevalence in Web workloads. Arlitt and Williamson [1997a] report that 15-40% of the unique files accessed from a Web server are accessed only once. The situation is even worse in Web proxy access logs, where one-timers can account for 50-70% of the documents [Abdulla et al. 1997; Mahanti et al. 2000]. Modeling the one-time referencing characteristic is thus important when generating workloads for evaluating different caching algorithms.

—Zipf-like File Popularity Distribution

One common characteristic in Web workloads is the highly uneven distribution of references to files [Arlitt et al. 1999; Mahanti 1999; Roadknight et al. 1999]. In many cases, Zipf's law has been applied to model file popularity [Almeida et al. 1996; Breslau et al. 1999; Cunha et al. 1995; Roadknight et al. 1999]. Zipf's law expresses a power-law relationship between the popularity P of an item (i.e., its frequency of reference) and its rank r (i.e., relative rank among the referenced items, based on frequency of reference). This relationship is of the form $P = c/r^{\beta}$, where c is a constant, and β is often close to 1. For example, Zipf's law arises in the frequency of occurrence of English words [Breslau et al. 1999]; when the number of occurrences is plotted versus the rank, the result is a power-law function with exponent close to 1.

In the Web context, a similar referencing behaviour is observed [Arlitt et al. 1999; Mahanti et al. 2000; Roadknight et al. 1999]. Some researchers have found that the value of the exponent β is close to 1 [Almeida et al. 1996; Cunha et al. 1995], precisely following Zipf's law. Others [Almeida et al. 1998; Breslau et al. 1999; Mahanti et al. 2000] have found that the value of β is less than 1, and that the distribution can be described only as "Zipf-like", with the value of β varying from trace to trace. This behaviour typically results in a straight line of (negative) slope β on a log-log plot of P versus r. The linear fit is usually good for the main body of the distribution, though it may deviate slightly at both the most popular end (due to "hot" documents and/or caching effects) and the least popular end (due to one-timers) [Mahanti et al. 2000].

—Heavy-Tailed Size Distributions

Workload characterization studies [Abdulla et al. 1997; Arlitt and Jin 2000; Arlitt and Williamson 1997a; Duska et al. 1997; Mahanti 1999] have shown that the file size distribution for Web transfers is heavy-tailed. A heavy-tailed distribution implies that relatively few large files account for a significant percentage of the data volume (in bytes) transferred to Web clients. This heavy-tail property contributes to the self-similarity observed in WWW traffic [Crovella and Bestavros 1997].

Clearly, the distribution of file sizes affects the design and performance of caching strategies. Caching only small files can reduce the number of requests to originating servers. This can result in a high document hit ratio, but a low byte hit ratio. On the other hand, caching large files can result in a higher byte hit ratio at the expense of document hit ratio (since many small documents may be forced out of the cache).

—Temporal Locality

Table I. Characteristics of Empirical and Synthetic Web Proxy Workloads

Item	Empirical	Synthetic
Total requests	5,000,000	4,965,779
Unique documents	1,739,119	1,700,000
Unique documents (% of requests)	34%	34%
One-timers	1,252,264	1,223,719
One-timers (% of unique documents)	72%	71%
Total Gbytes of unique documents	19	17
Smallest file size (bytes)	0	13
Largest file size (bytes)	53,857,877	42,975,450
Mean file size (bytes)	11,740	11,157
Median file size (bytes)	3,504	3,962
Zipf Slope	-0.808	-0.834
R^2	0.992	0.998
Tail index	-1.323	-1.326
R^2	0.980	0.998

Temporal locality refers to the tendency for Web documents referenced in the recent past to be referenced in the near future [Almeida et al. 1996; Jin and Bestavros 2000b]. Caching policies (e.g., Least-Recently-Used) often take advantage of this property when deciding what to cache and what to remove from the cache. Clearly, the presence (and strength) of the temporal locality property in the workload can have a dramatic effect on caching performance [Cherkasova and Ciardo 2000; Jin and Bestavros 2000a; Mahanti and Williamson 1999].

Among these workload characteristics, the Zipf-like document popularity distribution seems to have the most direct impact on Web proxy cache performance [Breslau et al. 1999; Busari 2000; Roadknight et al. 1999], and is thus the main focus of discussion throughout the paper.

2.2 An Empirical Trace Example

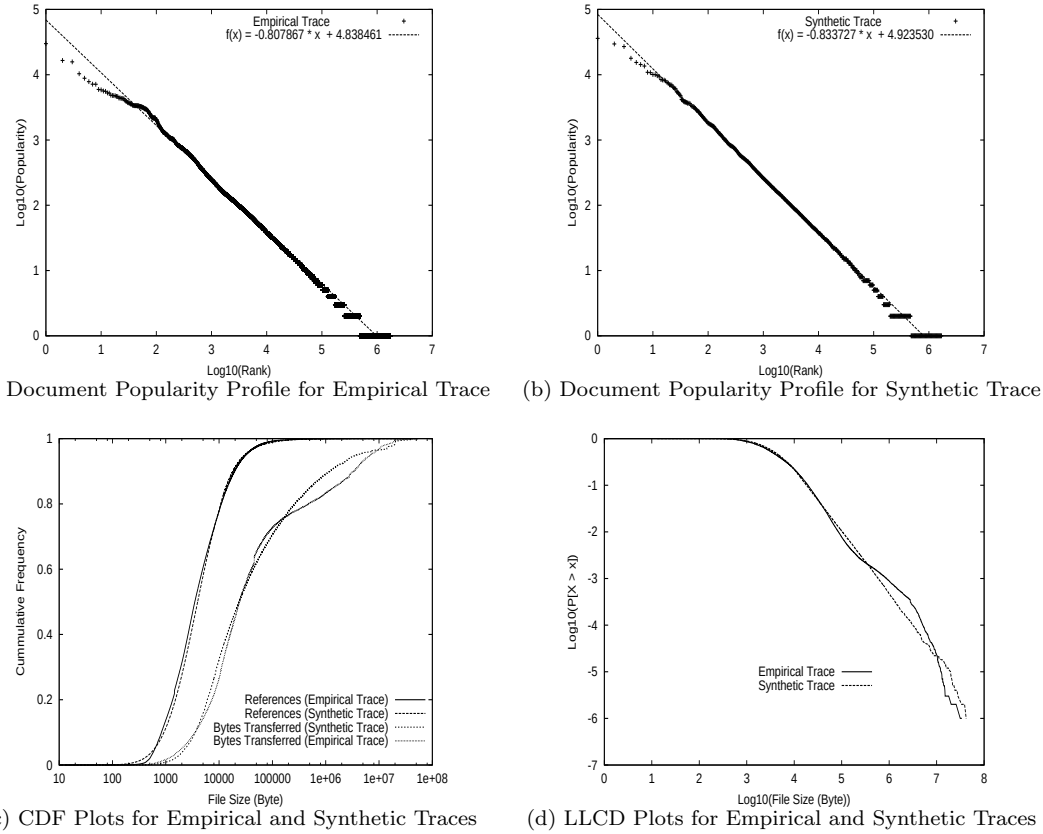
Table I provides a statistical summary of an empirical workload trace collected from a university-level Web proxy cache at the University of Saskatchewan in 1999. This empirical trace represents a typical two-week sample of a six-month trace analyzed in [Mahanti et al. 2000].

This empirical workload has structural properties consistent with those described in the previous subsection. There is a high percentage (72%) of one-timer documents. The Zipf-like document popularity distribution (see Figure 1(a)) has a slope of -0.808. The document size distribution (see Figure 1(c)) appears to be heavy-tailed (see Figure 1(d)), with a tail weight of 1.323. The temporal locality properties of this empirical trace have been studied in earlier work [Busari 2000; Busari and Williamson 2001a; Mahanti et al. 2000].

2.3 A Synthetic Trace Example

A Web proxy workload generation tool called ProWGen [Busari and Williamson 2001a] was used to synthesize an aggregate client workload, with workload parameters similar to the empirical trace. This workload is based on empirically observed workload characteristics at the lowest level of a Web proxy caching hierarchy [Busari 2000; Mahanti et al. 2000].

The statistical characteristics of the resulting workload produced are shown in



(a) Document Popularity Profile for Empirical Trace (b) Document Popularity Profile for Synthetic Trace

(c) CDF Plots for Empirical and Synthetic Traces (d) LLCD Plots for Empirical and Synthetic Traces

Fig. 1. Comparison of Characteristics of Empirical and Synthetic Web Proxy Workloads

the rightmost column of Table I. The statistical characteristics of the generated trace match closely with those of the empirical trace used.

A "graphical validation" of the synthetic workload is shown in Figure 1. Figures 1(a) and (b) show the file popularity results for the empirical trace and the synthetic trace, respectively. Figure 1(c) shows the cumulative distributions for file sizes and bytes transferred, while Figure 1(d) shows the tail behaviour using a log-log complementary distribution (LLCD) plot. In the latter two graphs, the results for the empirical trace are shown using solid lines, while the results for the synthetric trace are shown using dashed lines.

Figure 1 provides evidence that synthetic Web proxy workloads generated using ProWGen can match both the body and the tail of the empirical file size distribution, in addition to the Zipf-like referencing behaviour. The temporal locality properties of synthetic traces were analyzed in earlier work [Busari 2000; Busari and Williamson 2001a; Mahanti et al. 2000].

Synthetic traces are used selectively in the simulation experiments in this paper to demonstrate some of the performance characteristics and properties of Web proxy caching hierarchies.

2.4 Related Work

Several researchers have suggested ways to improve the performance of caching hierarchies. Tewari et al. [1999] suggested a distributed approach using metadata to track where copies of files are stored in the hierarchy. A similar technique is suggested by Povey et al. [1997], where only the lower level caches are responsible for storing documents, while upper level caches maintain information about the contents of lower level caches. The collaborative method proposed by Yu et al. [1998] employs a protocol that passes caching information down the proxy hierarchy for the lower level proxies to make better caching decisions. The Cache Array Routing Protocol (CARP) [Valloppillil and Ross 1998] is a form of distributed caching where multiple proxy servers are configured to appear as a single logical cache to the clients. In the "summary cache" scheme proposed by Fan et al. [1998], each proxy stores a summary of URLs of documents cached at every other proxy so that misses can be sent to a proxy with a copy of the requested document, or otherwise directly to the Web server.

Other authors have addressed the more general issue of hierarchical versus distributed caching approaches [Rodriguez et al. 1999; Wolman et al. 1999]. For example, Rodriguez et al. [1999] present mathematical analyses of hierarchical and distributed caching architectures, identifying performance tradeoffs for each approach, and proposing a hybrid scheme that combines the advantages of both. Wolman et al. [1999] discuss cooperative Web proxy caches, and analyze their scaling properties as a function of population size.

The approach in this paper is different, in at least two ways. First, this work assumes, as a starting point, the existence of a traditional Web proxy caching hierarchy, such as that in [Mahanti et al. 2000], and then strives to improve its performance. There is no attempt to re-design the overall caching infrastructure. Second, a primary focus in this work is on the explicit relationships between workload characteristics and Web proxy caching performance. The work is motivated by the observation that the workload characteristics differ across the levels of a caching hierarchy [Mahanti et al. 2000], due to filtering effects at lower-level caches [Che et al. 2001; Weikle et al. 1998]. This observation suggests the use of different (i.e., heterogeneous) caching policies at different levels of a caching hierarchy, or the use of "cache-aware" caching policies within the hierarchy. Relatively few papers [Breslau et al. 1999; Busari and Williamson 2001a; Doyle et al. 2001; Che et al. 2001] have taken this workload-based approach in the context of the Web, though similar problems have been addressed in the context of CPU cache hierarchies [Weikle et al. 1998], databases [Franklin et al. 1992], and client-server systems [Willick et al. 1993].

The initial focus of the paper is on understanding cache filter effects, and their structural causes. The latter part of the paper focuses on how to exploit knowledge of cache filter effects to improve the design and performance of Web proxy caching hierarchies.

3. UNDERSTANDING CACHE FILTER EFFECTS

This section provides graphical illustrations of cache filter effects, in an attempt to understand their structural causes and their impacts on Web proxy caching perfor-

mance. Trace-driven simulations are used, with empirical and synthetic traces, to illustrate the cache filter effects. A one-factor-at-a-time experimental design is used to identify the impact of selected factors, such as cache size and cache replacement policy, on the workload characteristics from one cache to the next. For simplicity, only a two-level caching hierarchy is considered in most of the experiments.

3.1 Number of Requests

The first experiment considers simulation run-length and warmup issues. This experiment examines the document referencing behaviour of the cache output stream (i.e., the stream of cache misses from the first-level cache) for input traces with 1,000,000 to 5,000,000 requests. The purpose of this experiment is to ensure that the traces used for subsequent experiments are of appropriate duration for analysis, and for general observations about the referencing behaviour. A cache size of 4 MB and a Least-Frequently-Used (LFU) replacement policy are used. Results for other policies are qualitatively similar.

Figure 3.1 shows the document popularity profile for the resulting workload destined for the second-level cache. The primary impact of the first-level cache is to truncate and flatten the top-left portion of the original Zipf-like document popularity distribution (see Figure 1(a)). In other words, most of the hits in the first-level cache are (as expected) for the highly popular documents that are re-referenced at short intervals. These requests are thus eliminated (filtered) from the request stream presented to the next-level cache. Similar observations are made by Doyle et al. [2001].

The impact of the first-level cache is quite consistent on each of the workload traces considered, producing a two-part plot that is no longer Zipf-like, but rather piecewise linear in shape. The leftmost portion of the plot is almost flat, reflecting fairly uniform popularity among the most popular documents seen by the second-level cache. The rightmost portion of the plot has a slope consistent with that of the original input trace (approximately -0.8, as shown in Figure 3.1). The boundary point between the two portions of the plot is dependent upon the document working set size of the trace relative to the cache size used (4 MB in this example).

In general, the document popularity profile shifts upward as longer traces are used, since longer traces often have more references to the popular documents. The proportion of one-timers in each of the input traces is approximately 73%.

Table II shows the document hit ratio (DHR) and byte hit ratio (BHR) results at the first-level cache, as a function of the trace length considered. The table also shows the number of requests passed on to the second-level cache, and some characteristics of the cache output stream. In addition to the change to the document popularity profile (reflected in the relatively poor R^2 values in Table II for the attempted least-squares linear fit of the Zipf slope), the presence of the first-level cache increases the percentage of unique documents and one-timers seen by the second-level cache (see Table I).

The results in Table II suggest that a trace length of 5,000,000 requests is adequate for assessing cache filter effects in simple Web proxy caching hierarchies.

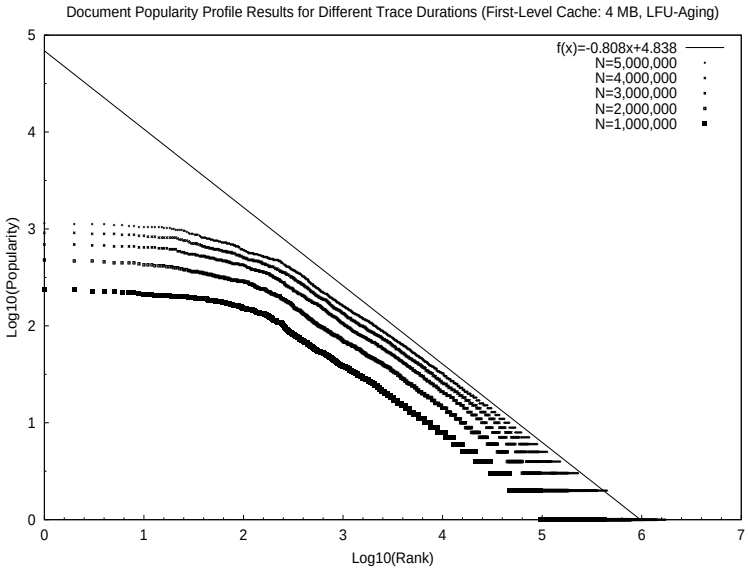


Table II. Simulation Results for Different Trace Durations (First-Level Cache: 4 MB, LFU-Aging)

Trace Duration	DHR	BHR	Requests	Unique	One-Timers	Zipf Slope	R^2
1,000,000	20.95%	16.07%	790,512	57%	79%	0.54	0.86
2,000,000	20.29%	14.91%	1,594,127	53%	79%	0.59	0.87
3,000,000	20.27%	15.08%	2,391,762	51%	79%	0.62	0.88
4,000,000	20.01%	14.65%	3,199,503	51%	79%	0.62	0.88
5,000,000	19.66%	14.33%	4,016,755	51%	79%	0.63	0.88

3.2 Cache Size

The purpose of the second experiment is to determine what effect the size of the first-level cache has on the document referencing characteristics of the cache output stream. The empirical input trace is used for input to the simulator. Results for an LFU replacement policy are shown, since the results for other policies are similar.

Figure 2 shows the resulting document popularity profile for the workload after the first-level cache. Each line in the graph shows the results for a different cache size, ranging from 1 MB to 1 GB. Recall that the empirical workload has about 19 GB of total unique content bytes.

The plots show that increasing the cache size makes the resulting document popularity distribution less and less Zipf-like. With a tiny 1 MB cache, the flattening behaviour in the top left of the plot is minimal, and the main body of the plot still follows the original Zipf slope of -0.8. As the cache size increases, the flattening behaviour due to cache filter effects becomes more and more pronounced. The rightmost part of the plot still provides a Zipf-like behaviour for all but the largest cache sizes considered in this experiment.

Table III summarizes the performance results for the first-level cache, and the resulting characteristics of the workload destined for the second-level cache. The Zipf-like property deteriorates greatly when the first-level cache is large (note the decreasing Zipf slopes and low R^2 values in Table III). The percentage of unique

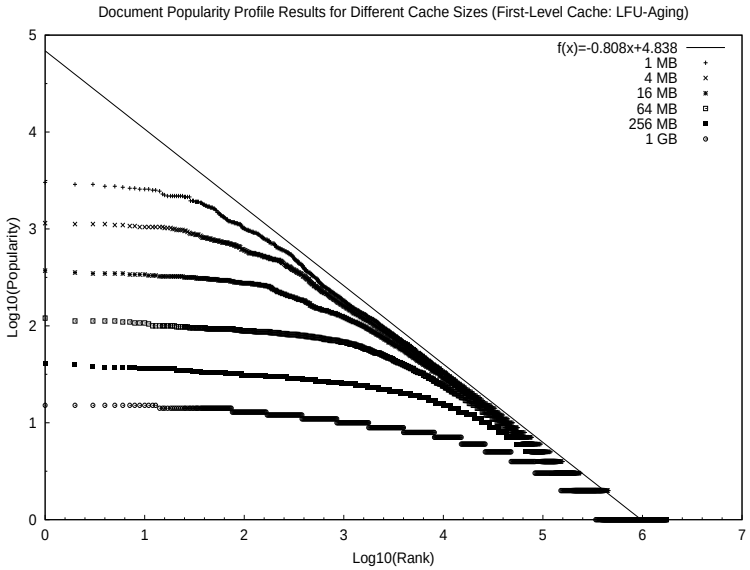


Fig. 2. Document Popularity Profile Results for Different Cache Sizes (LFU-Aging)

Table III. Simulation Results for Different Cache Sizes (First-Level Cache: LFU-Aging)

Cache Size (MB)	DHR	BHR	Requests	Unique	One-Timers	Zipf Slope	R^2
1	13.47%	8.57%	4,326,296	49%	79%	0.65	0.89
4	19.66%	14.33%	4,016,755	51%	79%	0.63	0.88
16	26.47%	20.18%	3,676,555	55%	80%	0.61	0.87
64	34.00%	26.03%	3,299,892	61%	82%	0.57	0.86
256	42.21%	32.06%	2,889,479	69%	83%	0.51	0.85
1,024	51.58%	39.43%	2,421,117	81%	87%	0.40	0.82
4,096	60.61%	47.51%	1,969,588	95%	95%	0.20	0.68
16,384	65.12%	52.13%	1,743,119	99%	99%	0.01	0.10

documents and the percentage of one-timers increase significantly for the second-level cache. Note that an infinite-size first-level cache would produce 100% one-timers at the second-level cache, since only "cold misses" would occur there. That is, all the documents are stored in the first-level cache, and with the exception of the first time reference to the document, all requests are served from this cache.

Subsequent simulations use a first-level cache size of 4-8 MB. This (small) size is large enough to have a pronounced filter effect on the workload emerging from the first-level cache, yet not so large as to represent an (uninteresting) infinite-size first-level cache.

3.3 Cache Replacement Policy

The third experiment studies the impact of the cache replacement policy on the document referencing behaviour of the cache output stream. The replacement policy for the cache determines which document(s) to remove from the cache when more space is needed to store a new incoming document. The empirical input trace is used for this experiment, with a cache size of 8 MB. This size is well below the

infinite cache size, as demonstrated by the previous cache size experiment.

Five different cache replacement policies are considered: RAND, FIFO, LRU, LFU, and GDS. The RAND policy chooses documents for replacement equiprobably at random. The FIFO policy removes documents on a strictly First-In-First-Out sequential basis. The RAND and FIFO policies are examples of simplistic "zero knowledge" policies. While rarely used in practice, these two policies provide a useful point of reference for the remaining three policies (LRU, LFU, and GDS), which are often considered in practical Web proxy cache implementations. LRU tries to keep recently active documents in the cache, by removing the document that is Least-Recently-Used. LFU tries to keep popular documents in the cache (with aging of old popular documents), by discarding the document that is Least-Frequently-Used. Greedy-Dual Size (GD-Size) [Cao and Irani 1997] tries to keep small documents in the cache, by associating a weight $H = 1/s$ with each document, where s is the size of the document in bytes.

This set of policies represents a broad range of candidate replacement policies (i.e., recency-based, frequency-based, and size-based). These policies are well-documented in the literature [Arlitt et al. 1999; Arlitt and Williamson 1997b; Cao and Irani 1997], and thus are not discussed at length here.

Figure 3 shows the document popularity profile results for the cache output stream from the first-level cache. Each line in the graph represents the results for one of the five replacement policies considered at the first-level cache. For most policies, a two-part piecewise-linear plot results. With all five replacement policies, the rightmost tail of the plot is Zipf-like with a slope of -0.8, as in the original workload trace.

Examining the plot shows that the LFU policy has the most pronounced impact on the document popularity profile, while the RAND and FIFO policies have the least impact. This result makes sense intuitively: since LFU is a frequency-based policy, the highly popular documents are favoured in the first-level cache, and the flattening effect in the document popularity profile for the second-level cache is more pronounced.

RAND and FIFO have less of a filtering effect on the workload, since they remove documents without any consideration for how frequently or how recently they have been referenced. Some filtering effect is still present for RAND and FIFO, however, since a highly popular document will re-enter the cache repeatedly. Upon each entry, there is an expected cache residency time prior to removal, during which re-references are possible. This residency time occurs since a document is unlikely to be chosen repeatedly as a candidate for removal (for the RAND policy), or must wait its turn for removal (for the FIFO policy).

The LRU policy produces workload characteristics between that of RAND and LFU. By considering recency of use, popular documents may receive many re-references in the first-level cache, thus filtering the workload to the next-level cache. The magnitude of this filtering effect depends of course on the relative size of the document working set compared to the cache size. Since the size of the first-level cache is relatively small (8 MB) in this example, the filtering effect is not that pronounced. Note that "recency of use" and (high) "frequency of use" are similar but not identical concepts, as illustrated in Figure 3.

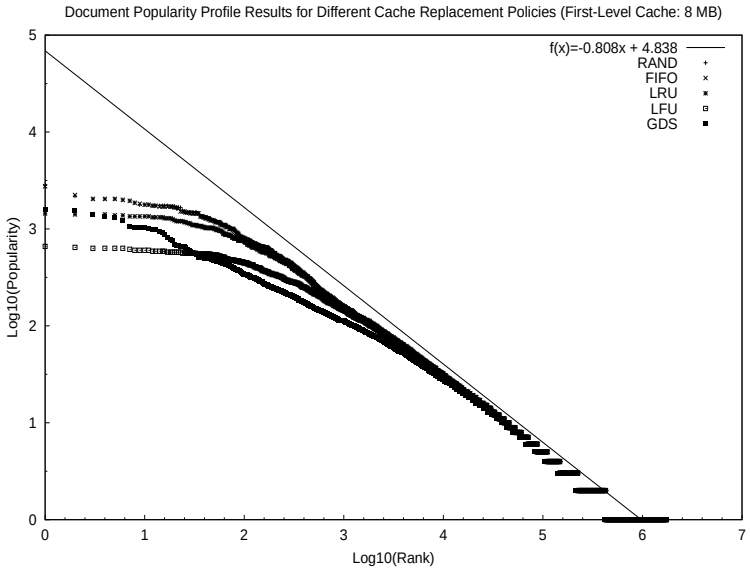


Fig. 3. Document Popularity Profile Results for Different Replacement Policies (8 MB)

Table IV. Simulation Results for Different Cache Replacement Policies (First-Level Cache: 8 MB)

Cache Policy	DHR	BHR	Requests	Unique	One-Timers	Zipf Slope	R^2
RAND	18.45%	13.96%	4,077,302	51%	80%	0.63	0.88
FIFO	19.15%	14.76%	4,042,424	51%	80%	0.62	0.88
LRU	20.61%	15.79%	3,969,433	52%	80%	0.62	0.88
LFU	22.99%	17.26%	3,850,620	53%	80%	0.62	0.88
GDS	27.10%	14.62%	3,644,856	56%	81%	0.60	0.87

The GDS policy has an interesting impact on the workload characteristics. The leftmost portion of the plot is not as flat as those for the other policies. The reason for this is that document size, rather than document popularity, is the central criterion for cache replacement decisions, and document size is (for the most part) independent of document popularity. The impact of the GDS policy is thus felt across a broader range of the document popularity profile, rather than concentrated on the highly popular documents. The transition point at which the plot becomes Zipf-like (with slope -0.8) is also further to the right, since the cache typically holds more documents (but not more bytes) than the other policies.

Table IV summarizes the performance results for the first-level cache, and the resulting characteristics of the workload destined for the second-level cache. The GDS policy provides the highest document hit ratio at the first-level cache, but not the highest byte hit ratio. For all cache replacement policies considered, the percentage of unique documents and the percentage of one-timers increase for the second-level cache, compared to the input workload for the first-level cache.

3.4 Zipf Slope

The fourth experiment studies the effect of the Zipf slope of the initial input trace on the document referencing behaviour of the cache output stream. For this ex-

Table V. Characteristics of Synthetic Traces with Different Zipf Slopes

Item	Trace A	Trace B	Trace C
Total requests	2,311,932	2,441,597	2,208,523
Unique documents	750,000	750,000	750,000
Unique documents (% of requests)	32%	31%	34%
One-timers	374,487	374,487	374,487
One-timers (% of unique documents)	50%	50%	50%
Total Gbytes of unique documents	9	9	9
Smallest file size (bytes)	25	25	25
Largest file size (bytes)	45,045,905	45,045,905	45,045,905
Mean file size (bytes)	12,824	12,814	12,815
Median file size (bytes)	3,826	3,825	3,824
Zipf Slope	-0.64	-0.78	-0.94
R^2	0.9993	0.9996	0.9996

Table VI. Simulation Results for Different Zipf Slopes (First-Level Cache: 8 MB, LFU-Aging)

Zipf Slope	DHR	BHR	Requests	Unique	One-Timers	Zipf Slope	R^2
0.64	12.69%	7.35%	2,018,593	37%	51%	0.72	0.94
0.78	27.30%	15.06%	1,174,958	42%	55%	0.65	0.93
0.94	40.37%	24.14%	1,316,911	56%	65%	0.48	0.85

periment only, synthetic input traces of approximately 2,500,000 requests are used, with approximate Zipf slopes of 0.60, 0.75, and 0.95. Table V summarizes the characteristics of these traces. These traces were generated using ProWGen, as described in Section 2.3. The cache size for these simulations is 8 MB, and the cache replacement policy is LFU.

Figure 4 shows that the filtering effects of a first-level cache occur regardless of the Zipf slope of the input trace. The two-part piecewise-linear document popularity profile is common for all three workloads, with the filtering effect most pronounced for the input trace with the steepest Zipf slope. In all three cases, the rightmost portion of the plot matches the Zipf slope of the input trace used.

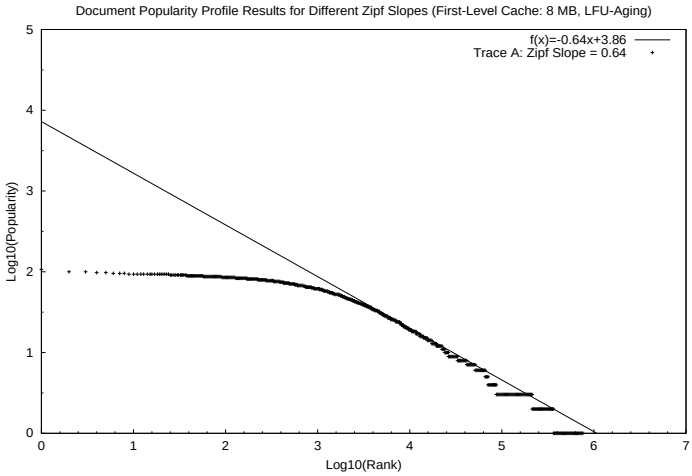
Table VI summarizes the workload characteristics and caching performance that result from the presence of the first-level cache. As expected, Web proxy caching performance is best when the Zipf slope is steep. In all cases, the filtering effects produce higher percentages of one-timers and unique documents at the second-level cache than in the original input trace.

3.5 Depth of Caching Hierarchy

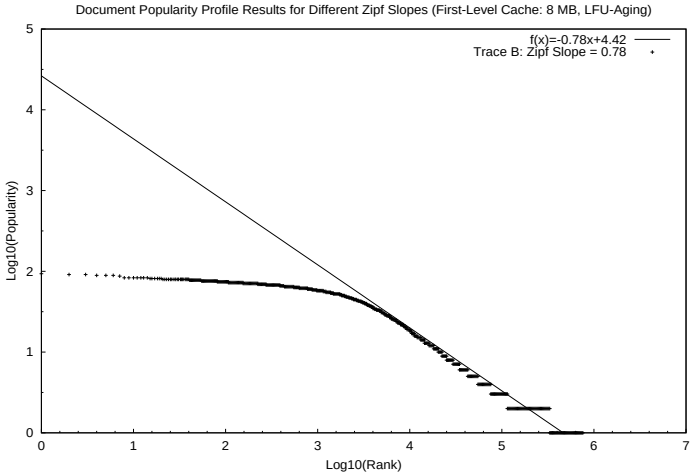
The final cache filtering experiment focuses on the *depth* of the caching hierarchy, by looking at the document referencing characteristics as the workload passes through three levels of caches. For this experiment, all caches use an LFU replacement policy and a cache size of 4 MB. A small cache size is used in order to see the impact of each successive level of cache.

Figure 5 shows that the first-level cache has the most pronounced filtering effect on the overall workload. Further levels of caching produce little change in the document popularity profile. This observation makes sense intuitively, since the higher level caches tend to have lower and lower hit ratios, and thus provide little change to the workload proceeding to the next higher level cache.

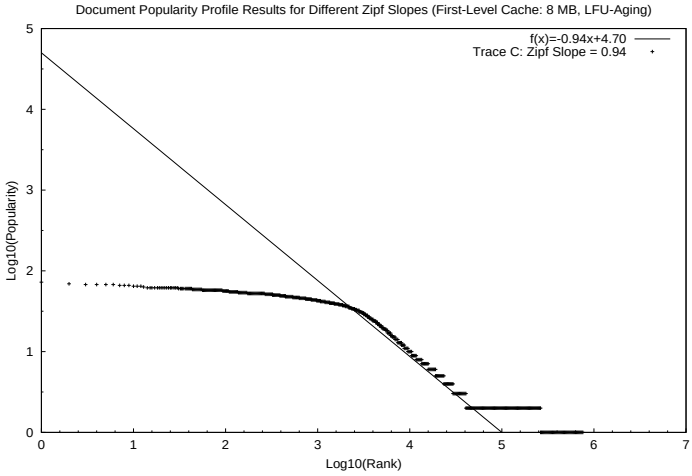
Table VII summarizes the details for cache performance and filtering effects for the three-level caching hierarchy experiment. The results in Table VII indicate that



(a) Zipf Slope = 0.64

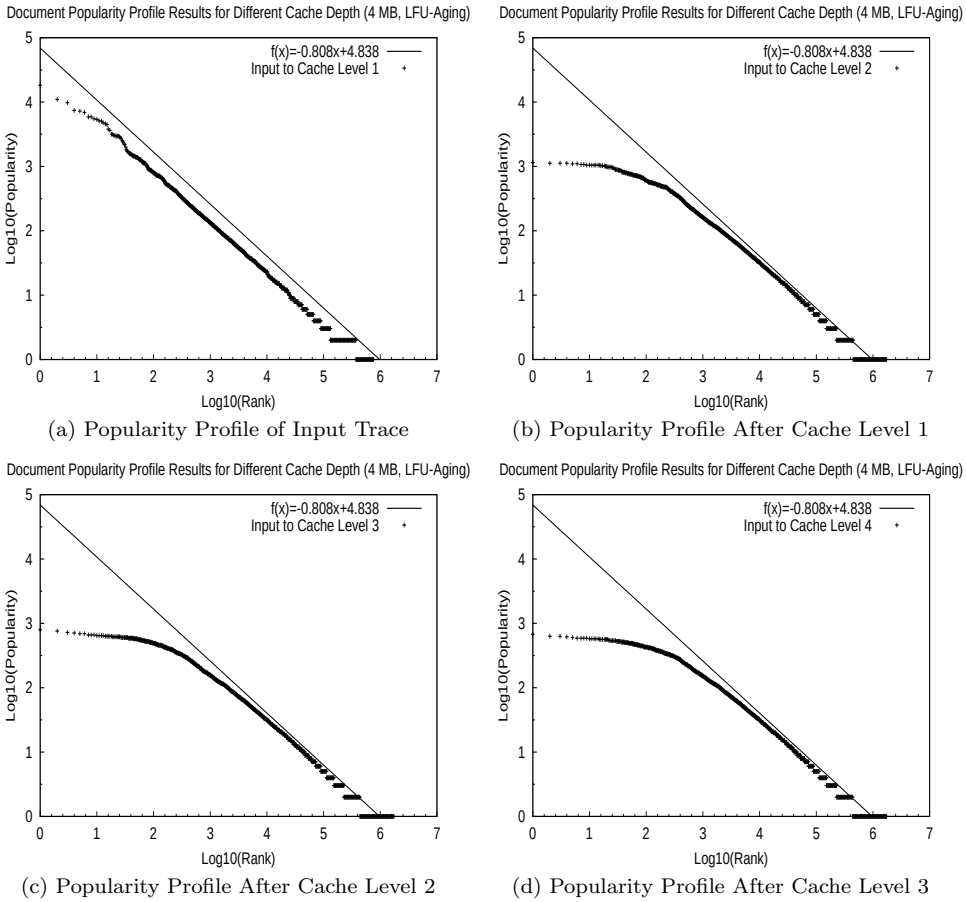


(b) Zipf Slope = 0.78



(c) Zipf Slope = 0.94

Fig. 4. Document Popularity Profile Results for Different Zipf Slopes (8 MB, LFU-Aging)



(a) Popularity Profile of Input Trace

(b) Popularity Profile After Cache Level 1

(c) Popularity Profile After Cache Level 2

(d) Popularity Profile After Cache Level 3

Fig. 5. Document Popularity Profile Results for Different Cache Depths (4 MB, LFU-Aging)

Table VII. Simulation Results for Different Cache Depths (First-Level Cache: 4 MB, LFU-Aging)

Cache Level	DHR	BHR	Requests	Unique	One-Timers	Zipf Slope	R^2
1	19.66%	14.33%	4,016,755	51%	79%	0.63	0.88
2	2.05%	1.50%	3,934,235	52%	79%	0.63	0.88
3	0.94%	0.66%	3,897,370	53%	79%	0.62	0.88

the higher level caches are not very effective: the document hit ratio and the byte hit ratio both drop significantly beyond the first-level cache.

This observation is consistent with findings reported in the literature [Abdulla et al. 1997; Busari and Williamson 2001b; Mahanti and Williamson 1999; Mahanti et al. 2000]. The poor performance of the cache explains the similarity between the document popularity profiles across the upper cache levels. If few requests for documents hit in the cache, most of the requests are passed through to the higher level, with the workload minimally changed.

3.6 Summary of Results

This section has explored the presence and magnitude of cache filter effects in simple Web proxy caching hierarchies. A summary of the key observations about cache filtering effects is as follows:

—Proxy caches filter out the most popular documents from a workload. The filtering effect typically changes the Zipf-like document popularity profile of the input stream into a two-part piecewise-linear document popularity profile for the output stream. The upper leftmost part of the plot is significantly flattened.

—The percentage of unique documents referenced and the percentage of one-timer documents tend to increase as the request stream passes through a cache.

—Among the caches in a caching hierarchy, the first-level cache has the most pronounced filtering effect on the workload.

4. EXPLOITING FILTER EFFECTS

The foregoing observations on cache filter effects paint a rather bleak picture for Web proxy caching hierarchies. These observations can be used to explain the ineffectiveness of Web caching hierarchies, and to argue for their demise in favour of distributed or cooperative caching approaches [Rodriguez et al. 1999; Povey and Harrison 1997; Tewari et al. 1999; Wolman et al. 1999; Yu and MacNair 1998]. Alternatively, these observations can be used to design *improved* Web proxy caching hierarchies that can take advantage of cache filtering properties in a multi-level caching system.

This section takes the latter approach, and considers approaches that might improve overall performance of a Web proxy caching hierarchy. The approaches considered are heterogenous replacement policies and size-based partitioning. Trace-driven simulations are used to evaluate the candidate approaches.

4.1 Experimental Methodology

The simulation experiments consider a two-level hierarchical Web proxy configuration as shown in Figure 6. In the simulation model, requests from the aggregate workload are forwarded to the lower level proxies, and misses are forwarded to the upper level proxy. Misses from the upper level proxy are forwarded to the (simulated) Web servers. There are no interactions directly between the two lower-level proxies, and there are no cache consistency mechanisms in the modeled hierarchy, since the workload is assumed to consist only of static documents (i.e., document size and content do not change with time).

The experimental methodology considers three main factors: *cache size*, *cache replacement policy*, and *cache management policy*. Cache sizes range from 1 MB to 32 GB in the experiments, with all three caches (parent and two children) identical in size. Three cache replacement policies are considered: Least-Recently-Used (LRU), Least-Frequently-Used-with-Aging (LFU-Aging), and Greedy-Dual-Size (GD-Size). The cache management policy determines how the different levels of the hierarchy operate. Two approaches, namely heterogeneous caching policies and size-based partitioning, are considered.

Two performance metrics are used to evaluate cache performance: *document hit ratio* and *byte hit ratio*. The document hit ratio is the number of requests satisfied

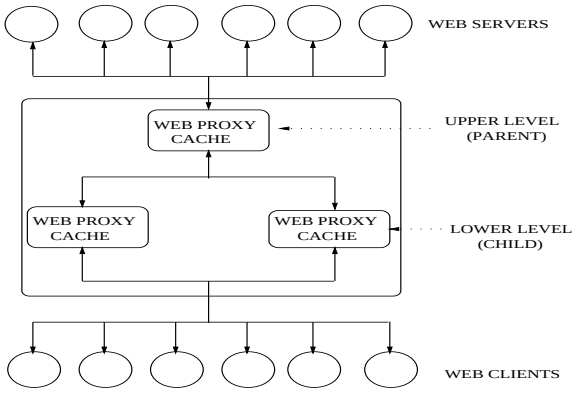


Fig. 6. Two-level Hierarchical Proxy Simulation Model

Table VIII. Characteristics of Synthetic Workload for Caching Simulations

Item	Value
Requests	9,749,703
Unique documents	3,000,000
Unique documents (% of requests)	31%
One-timers	2,099,045
One-timers (% of unique documents)	70%
Total bytes of unique documents (GB)	30
Total transferred content bytes (GB)	89
Smallest file size (bytes)	9
Largest file size (bytes)	51,600,679
Mean file size (bytes)	10,850
Median file size (bytes)	3,815
Correlation (file size and popularity)	-0.004982
Zipf Slope	-0.768985
R^2	0.9980
Pareto tail index	-1.300494
R^2	0.9998

by a particular proxy's cache divided by the total number of requests seen by the proxy. The byte hit ratio is the volume of data (in bytes) satisfied by the proxy's cache divided by the total volume of data requested from the proxy. Both metrics are required since Web documents can differ dramatically in size. In general, the higher the document hit ratio and byte hit ratio are, the better a replacement policy is. Furthermore, the closer the "hit" is to the client, the lower is the (expected) document retrieval latency.

4.2 Workload

The experiments in this section use a synthetically generated ProWGen workload. The synthetic trace has approximately 10 million requests. The details of the workload are summarized in Table VIII. Qualitatively, this workload is similar in structure to the empirical workload introduced in Section 2.2, though it has approximately twice as many requests.

The aggregate workload is then split across the two lower-level proxies to model three different scenarios: *complete overlap*, *partial overlap*, and *no overlap*. Each

scenario reflects a different degree of overlap (i.e., common URLs in the Web document request streams) in the workloads of the two lower level proxies.

The *complete overlap* scenario models the situation where the two lower-level proxies reside in "similar" organizations (i.e., the aggregate client sets behave similarly, in terms of the Web content requested). Thus the cache contents at the two lower level proxies are likely to be statistically similar, on average. This scenario is modeled by randomly dispatching each request in the workload to one of the lower-level proxies (equiprobably, at random).

The *no overlap* scenario models Web proxies with entirely different document request streams (i.e., completely different client behaviours). This scenario is modeled by assigning requests for odd-numbered documents to one lower-level proxy, and requests for even-numbered documents to the other lower-level proxy.

The *partial overlap* scenario represents an intermediate situation between the two extremes already discussed. In particular, this scenario has a 50% overlap in the workload of the two lower level proxies. This scenario is modeled by randomly choosing half of the documents to be shared (as in complete overlap), with the remaining documents split between the two child proxies on an odd-even basis.

4.3 Heterogeneous Replacement Policies

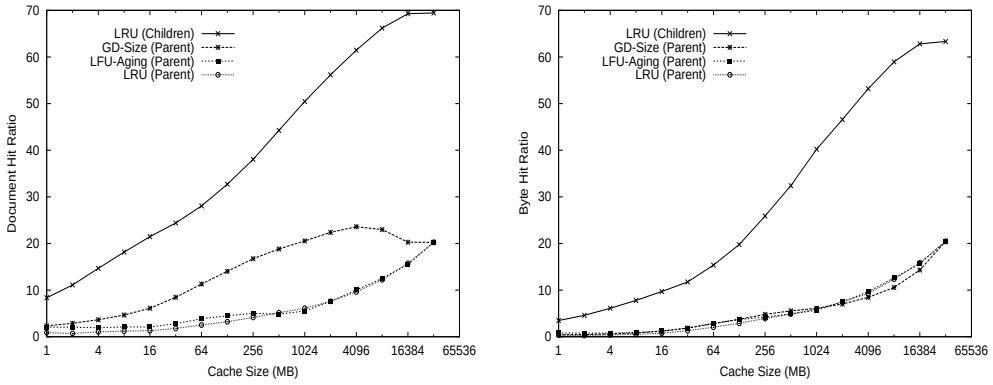
The experiments in this section are motivated by the simple observation that the workload characteristics are different at each level of the caching hierarchy. A natural follow-on from this observation is to try different caching techniques at each level of the hierarchy. Thus the experiments here consider different replacement policies (LRU, LFU-Aging, and GD-Size) at the child and parent caches in the caching hierarchy.

Simulation results for the *complete overlap* scenario are shown in Figure 7. In this figure, the leftmost column of graphs show document hit ratio results, while the rightmost column of graphs show the corresponding results for byte hit ratios. Figures 7(a) and (b) show the results for the LRU policy at the child level, while Figures 7(c) and (d) show the results for LFU-Aging at the child caches, and Figures 7(e) and (f) show the results for GD-Size at the child caches. On each graph, the uppermost line shows the results for the child[1] cache, and the remaining three lines show the results for the parent cache, for each replacement policy considered.

In general, the child proxy caches have much higher hit ratios than the parent proxy. This observation is not surprising, given that the parent proxy only sees the requests that miss at the lower level caches (i.e, the request stream is filtered by the lower level proxies) [Doyle et al. 2001; Weikle et al. 1998; Willick et al. 1993].

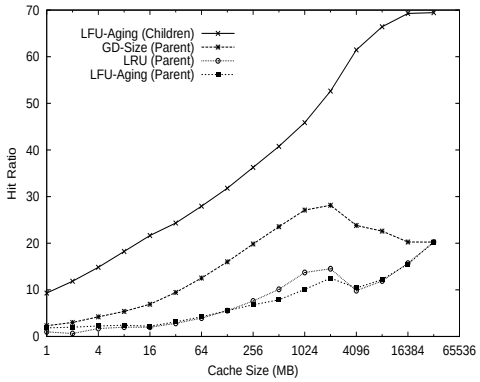
The graphs in Figure 7 illustrate some interesting differences in *marginal utility* (i.e., additional incremental value gained) when more cache space is added at either the child level or the parent level. This behaviour can be seen by noting the differences in the slopes of the hit ratio plots for the parent and child caches, as the cache sizes are increased (exponentially) from left to right. At some points, adding more child level cache is a big win; at other points the graph is fairly flat. Similar observations apply for the parent level cache.

[1]Since the hit ratios are similar for each child cache, only the results for one child cache are shown, for clarity.

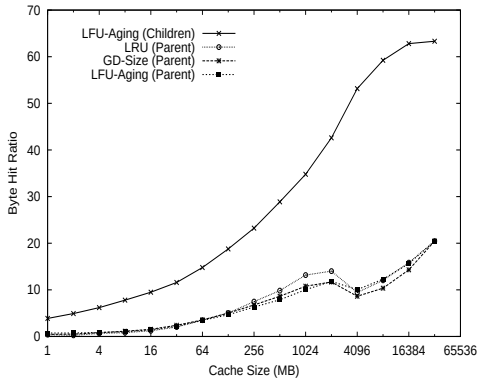


(a) Hit Ratio (Children: LRU)

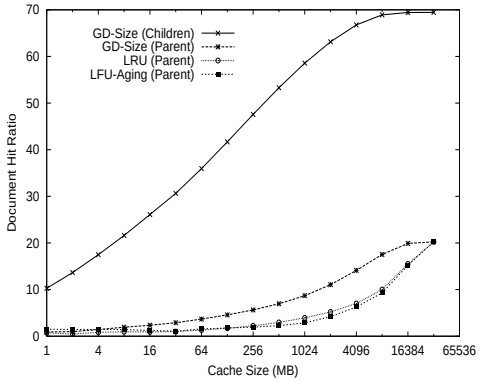
(b) Byte Hit Ratio (Children: LRU)



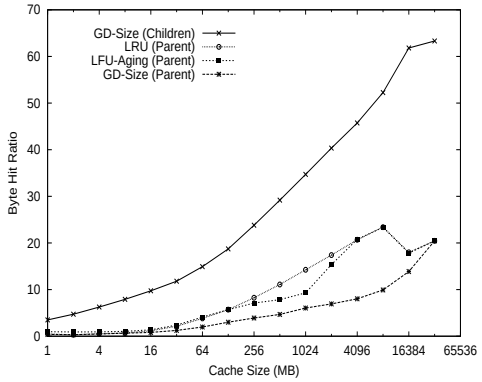
(c) Hit Ratio (Children: LFU-Aging)



(d) Byte Hit Ratio (Children: LFU-Aging)



(e) Hit Ratio (Children: GD-Size)



(f) Byte Hit Ratio (Children: GD-Size)

Fig. 7. Performance Results for Different Combinations of Replacement Policies in a Two-Level Hierarchical Proxy Configuration, with COMPLETE Overlap in the Workloads of the Lower Level Proxies.

In some cases, the hit ratio results for the parent level cache *drop* as the cache size is increased; this non-monotonic behaviour happens because the child cache in the simulation is also being increased in size, absorbing more hits, and reducing the number of requests to the parent level cache. In other words, the relative balance between "cold misses" (first request for a document) and "capacity misses" (subsequent request for a document that used to be in the cache, but has now been removed from the cache) changes as the cache sizes are scaled. This impact may be different at each level of the hierarchy.

The exact shape of these curves depends, of course, on the nature of the workload: the temporal locality property, the Zipf-like referencing behaviour, and the size (in bytes) of the "document working set", relative to the cache size used. These marginal utility trends also depend on the cache replacement policy used, since the replacement policy at one level changes the workload characteristics for the next level of cache.

Overall, Figure 7 shows that the GD-Size policy at the parent cache provides a significantly better document hit ratio than either LRU or LFU-Aging at the parent cache. This document hit ratio advantage is a factor of two or more for most of the cache sizes considered, when the child level cache uses LRU (Figure 7(a)) or LFU-Aging (Figure 7(c)). Furthermore, this advantage does *not* come at the expense of the byte hit ratio (see Figure 7(b) and (d)), as is often the case with GD-Size [Arlitt and Williamson 1997b]. The performance advantage of GD-Size at the upper level is less pronounced, but still present, when the child proxies use GD-Size (see Figure 7(e)). However, the document hit ratio advantage is compromised by a lower byte hit ratio (Figure 7(f)).

Determining which combination of policies is "better", from an end-user point of view (i.e., response time), is a challenging problem, since it depends on network capacity, network latencies, server load, version of HTTP, and TCP-level effects [Feldmann et al. 1999; Heideman et al. 1997]. Further discussion of this issue, and an approximation technique for cost-benefit analysis, appears in [Busari 2000].

In summary, the potential advantage of heterogeneous caching policies is most evident in Figure 7(a). For a given cache size, the effectiveness (hit ratio) of a second-level cache can be doubled or tripled by using a size-based replacement policy that is different from that used at the first-level cache (LRU, for example).

4.4 Sensitivity of Results to Workload Overlap

This section explores the sensitivity of the previous results for heterogeneous cache replacement policies to the degree of workload overlap between the two child-level proxies in Figure 6. The previous section assumed complete overlap in the workload: clients at either proxy are equally likely to access any given page in the Web document space. This section considers:

—a *partial overlap* scenario, in which 50% of the Web document space is common to all clients (accessed from any child proxy), and 50% is of regional interest only (accessed from only one child proxy)

—a *no overlap* scenario, in which the two child proxies handle requests for completely disjoint document sets

Results for the *partial overlap* workload scenario are shown in Figure 8. These results follow the same trend as in the complete overlap case, except for a slight improvement[2] in hit ratios for the child cache, and a noticeable drop in the hit ratios of the parent cache.

The explanation for this behaviour is the reduced overlap in the workloads of the two lower level proxies. That is, the 50% overlap in the workloads means that the left-most child proxy exclusively sees requests to 25% of the aggregate document set, the right-most child proxy exclusively sees requests to a different 25% of the aggregate document set, and the remaining 50% of the documents are (typically) seen by both proxies.

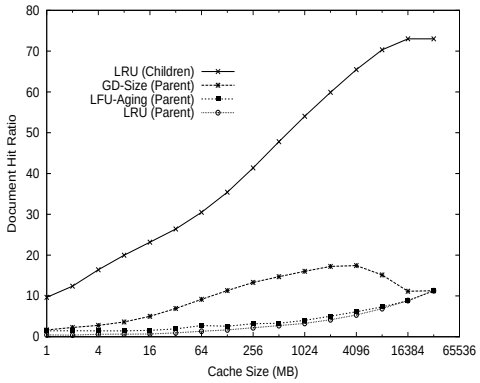
The partial overlap workload assumption has two important consequences. First, each child proxy sees only a subset (75%) of the total document space, which means fewer documents contending for cache space. Second, references to a particular document, as generated by ProWGen's document popularity and temporal locality models [Busari 2000; Busari and Williamson 2001a], may now be concentrated on a single child proxy, rather than randomly split across the two proxies. Again, this translates into better caching performance at the child proxies. On the other hand, the reduced overlap in the workload implies worse caching performance at the parent cache: the probability that a file requested by one child (and pulled into the parent cache from the origin server prior to delivering to the child) is requested later by the other child is reduced (by about half) compared to the complete overlap scenario. Furthermore, repeated hits at the parent cache for such a document can only occur if there are repeated capacity misses at the child level.

The results for the no overlap scenario (Figure 9) show the same trend: further improvement in the performance of the child caches, and a further decrease in the performance of the parent cache. This trend is consistent, regardless of the replacement policies used.

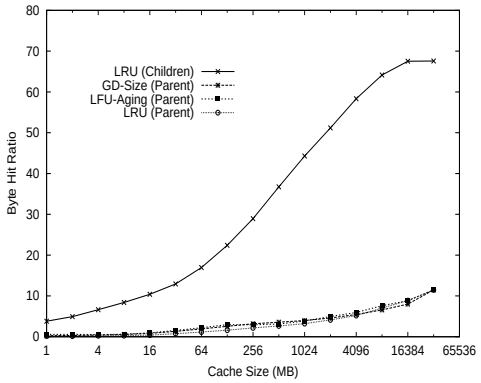
For the no overlap scenario, the only role for the parent cache is to serve capacity misses from the lower level caches. In such a case, the GD-Size policy at the parent cache has the best performance, since it typically stores the most documents. The results show that the parent cache has its highest hit ratio when the size of the child cache is about 1-2 GB: about 10% of the total size of the Web content accessed. As the child caches grow larger, fewer capacity misses occur, and the relative benefit of the parent cache diminishes.

In summary, the effectiveness of the parent cache in a caching hierarchy diminishes when there is little overlap in the Web workloads seen at the child-level proxies. For the scenarios and workloads studied here, the LRU or LFU-Aging policies at the lower level combined with GD-Size at the upper level always provide improvement in performance over an LRU-LRU combination. The GD-Size policy often doubles the document hit ratio at the parent cache, without any penalty in byte hit ratio. Using GD-Size at both levels improves the document hit ratio, but sacrifices the byte hit ratio.

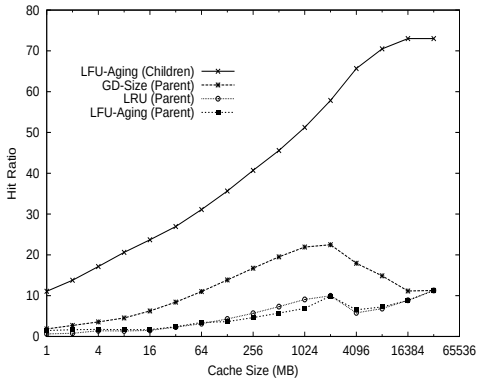
[2]Note the change in vertical scale from Figure 7 to Figure 8.



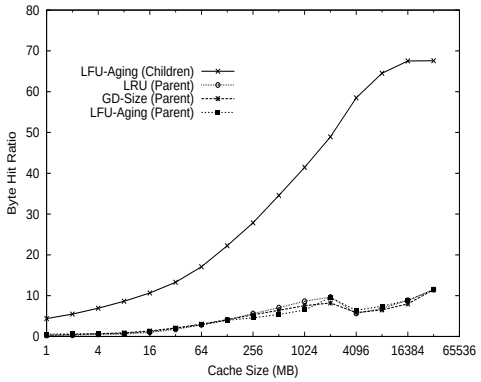
(a) Hit Ratio (Children: LRU)



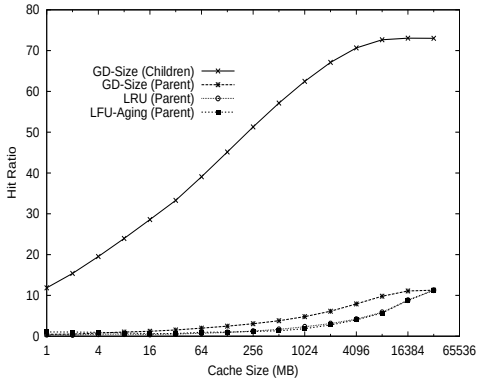
(b) Byte Hit Ratio (Children: LRU)



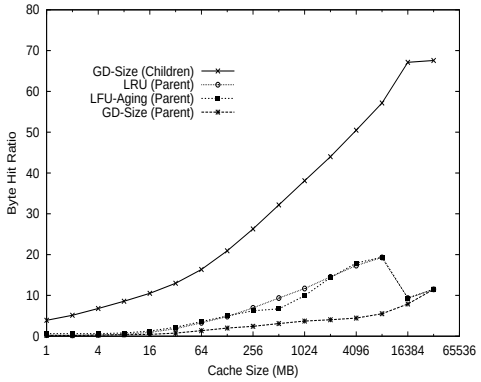
(c) Hit Ratio (Children: LFU-Aging)



(d) Byte Hit Ratio (Children: LFU-Aging)

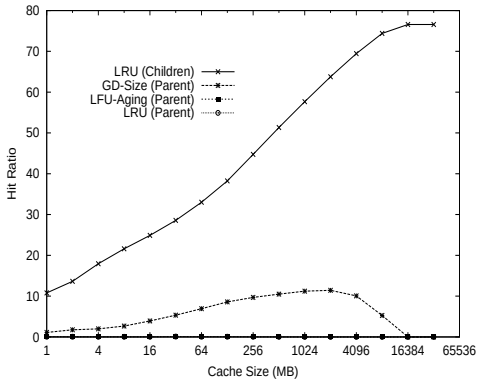


(e) Hit Ratio (Children: GD-Size)

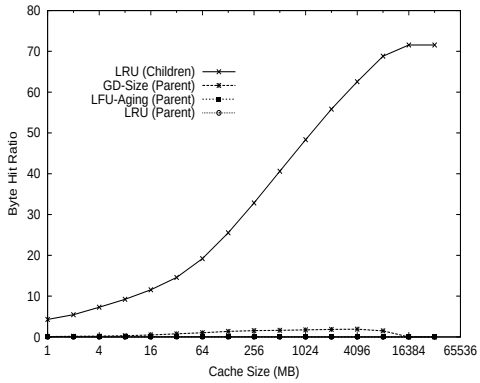


(f) Byte Hit Ratio (Children: GD-Size)

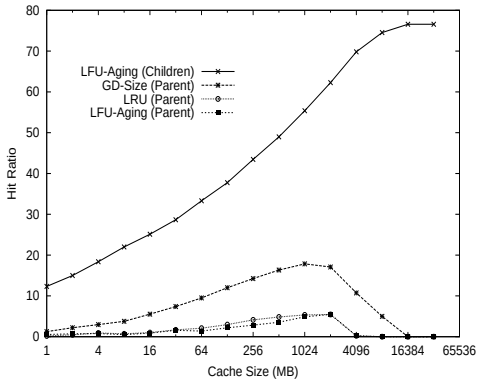
Fig. 8. Performance Results for Different Combinations of Replacement Policies in a Two-Level Hierarchical Proxy Configuration, with PARTIAL Overlap in the Workloads of the Lower Level Proxies.



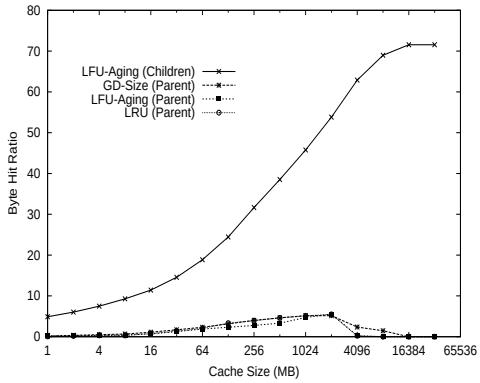
(a) Hit Ratio (Children: LRU)



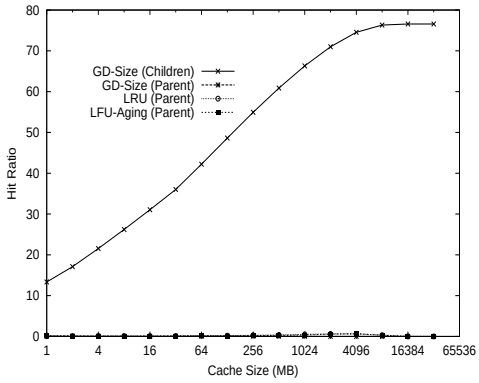
(b) Byte Hit Ratio (Children: LRU)



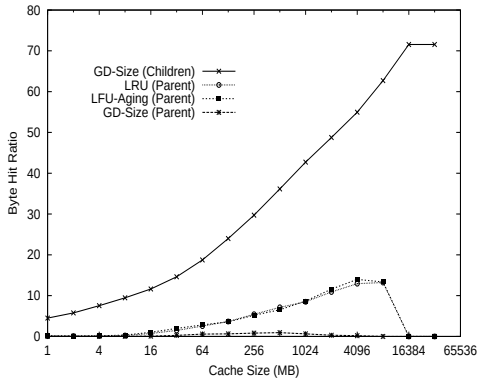
(c) Hit Ratio (Children: LFU-Aging)



(d) Byte Hit Ratio (Children: LFU-Aging)



(e) Hit Ratio (Children: GD-Size)



(f) Byte Hit Ratio (Children: GD-Size)

Fig. 9. Performance Results for Different Combinations of Replacement Policies in a Two-Level Hierarchical Proxy Configuration, with NO Overlap in the Workloads of the Lower Level Proxies.

4.5 Size-Based Partitioning

The next cache management strategy evaluated is a document partitioning approach called *size-based partitioning*. That is, based on a specified size threshold S, the lower level caches are allowed to store only files smaller than S, while the upper level cache is allowed to store only files of size S or larger. This simple policy provides a natural partitioning of the document space, using minimal information. Document size information is available to Web servers and proxies in the HTTP response header.

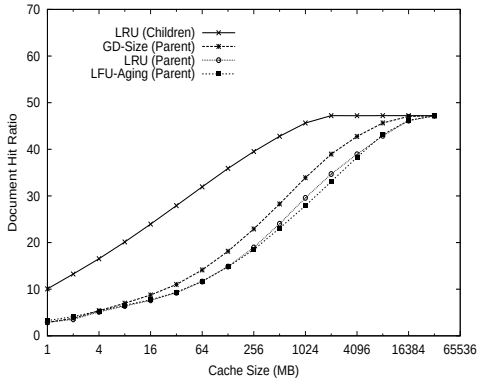
This partitioning is similar in nature to the "consistent hashing" approach used to partition the URL space in some distributed (flat) caching strategies, in that a specific document is only eligible to reside in a specific cache location (or cache level in this case). However, the partitioning is based on document size rather than URL, and there is no restriction on the number of first-level caches (for example) in which a copy of a given document could reside. The motivation for keeping small documents in the first-level cache is to minimize the round-trip latency for connection setup, which often dominates the retrieval time for small documents. Larger documents can be maintained at the second-level cache, since the larger connection setup latency can be amortized over the duration of a (longer) data transfer.

In its strictest form, size-based partitioning provides a way to flatten (i.e., defeat) the caching hierarchy. That is, the first-level caches handle small documents only, and the second-level cache handles large documents only. Variations of this approach could relax this principle. For example, one could use a document size threshold restriction on the first-level cache only, while allowing the second-level cache to cache any document, small or large. Alternatively, a hybrid scheme could devote part of the first-level cache (e.g., 90%) to caching small documents, and part to large documents, with these proportional settings reversed at the next-level cache. This approach would preserve the semantics of a caching hierarchy, and allow size-based partitioning and heterogeneous replacement policies to be combined within the same caching hierarchy. In this paper, only the simplest (strict) size-based partitioning approach is considered, to evaluate its tradeoffs.

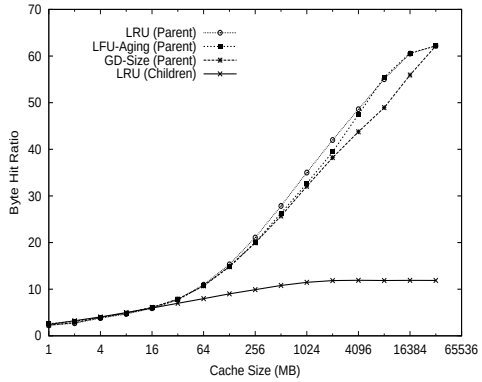
A key advantage of the size-based partitioning approach is that it confines the filtering effect of the first-level cache to only a portion of the incoming workload (e.g., small documents). Requests for large documents are passed through to the next-level cache unperturbed, preserving the Zipf-like document popularity profile in the request stream presented to that cache.

For completeness, the converse of this policy is also considered, namely large files at the lower level, and small files at the upper level. With either of these approaches, distinct documents are maintained at each level of the hierarchy. Some replication of documents in multiple caches at the child level of the hierarchy is still possible.

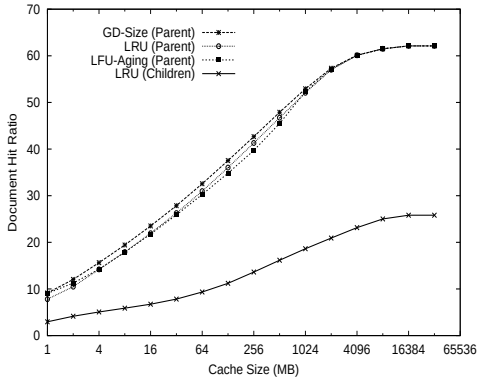
The first design issue for size-based partitioning is the choice of the threshold size S. To understand the impact of different threshold sizes, three values are considered: 5,000 bytes, 10,000 bytes, and 100,000 bytes. For each chosen threshold size, heterogeneous replacement policies are studied. For space reasons, only the results for the partial overlap workload scenario and the LRU replacement policy at



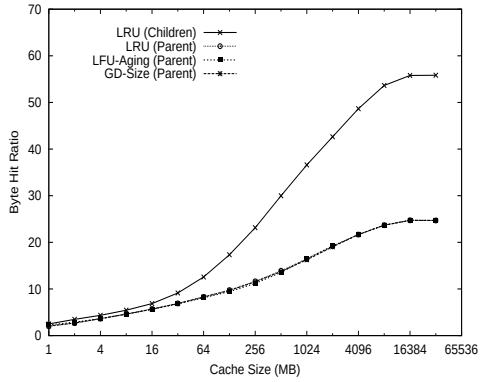
(a) Document Hit Ratio (Small files at the lower level)



(b) Byte Hit Ratio (Small files at the lower level)



(c) Document Hit Ratio (Large files at the lower level)



(d) Byte Hit Ratio (Large files at the lower level)

Fig. 10. Simulation Results for Size-Based Partitioning (Size Threshold = 5,000 Bytes)

the lower level caches are presented here. Complete results are available in [Busari 2000].

Figure 10 shows the results for a threshold size of 5,000 bytes. The top two graphs (Figures 10(a) and (b)) show the results when small files are kept at the lower level of the hierarchy, and large files at the upper level. The bottom two graphs (Figures 10(c) and (d)) are for the converse policy.

Figures 10(a) and (b) show that with size-based partitioning, the child caches have higher hit ratios than the parent cache (Figure 10(a)), but the parent cache achieves a much higher byte hit ratio (Figure 10(b)). The result for the parent cache is particularly interesting, in that it is able to achieve significant document hit ratios as well as byte hit ratios. This behaviour can be attributed to the high proportion of small files in the workload: about 60% of the requests are for files below this threshold size. The penalty for not keeping the large files (files $\geq$ 5,000 bytes) in the lower level cache is the lower byte hit ratios, as observed in Figure 10(b). The significant hit ratios and byte hit ratios achieved by the parent cache indicate that

many references occur to files stored in its cache, and these files are responsible for a significant fraction of the total volume of data transferred.

All three replacement policies considered at the parent cache are reasonably effective in Figure 10, since the incoming workload still has a Zipf-like document popularity profile, and the temporal locality property. That is, the filter effects of the first-level cache only apply to small documents. The original workload characteristics for requests to larger documents are preserved. Clearly, this workload is still amenable to caching at the parent level, since the document and byte hit ratios at the parent cache significantly exceed those of the heterogeneous caching approach for this workload (Figure 8).

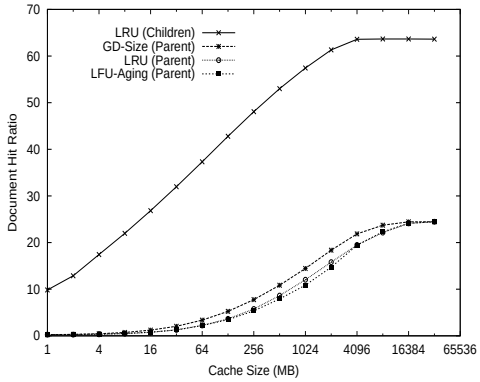
Figure 10(a) shows that the GD-Size policy still provides the best document hit ratio at the parent cache, among the policies considered. However, its performance advantage over LRU and LFU-Aging has diminished significantly from that in Figure 8(a). Furthermore, it has a slight disadvantage in terms of byte hit ratio (Figure 10(b)) at large cache sizes.

The flattening of the hit ratio plots for the child caches beyond a cache size of 2 GB indicates a form of "cache ineffectiveness" beyond this point. That is, while increasing the cache size beyond 2 GB can improve the performance of the parent cache, it has no further benefit for the child caches. The reason for this is that the child cache is already large enough to accommodate all requested files smaller than 5,000 bytes, without any replacements required for the workloads considered. The hit ratios thus stabilize beyond this "infinite" cache size.

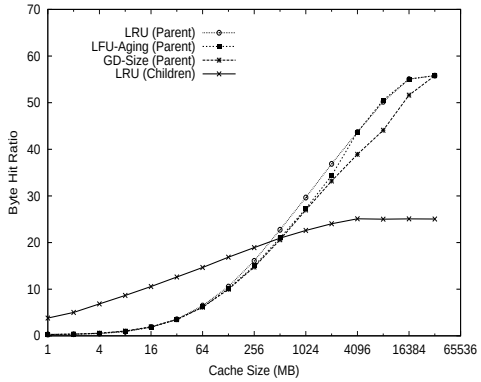
The performance results for the converse policy (i.e., keeping large files at the lower level and small files at the upper level) are shown in Figures 10(c) and (d). In these graphs, the parent cache shows consistently better document hit ratios than the child caches (Figure 10(c)). This is not surprising: the child proxies are not allowed to cache files smaller than 5,000 bytes, and these files account for a large fraction of the requests. While the observed document hit ratios at the child caches are low, the byte hit ratios (Figure 10(d)) are better than for the parent cache.

Figure 10(d) also shows that with the converse size-based partitioning approach, there are no noticable differences in byte hit ratio performance for the three different replacement policies considered at the parent cache (which is not allowed to cache large files). In other words, the performance impact of the size-based threshold scheme between levels of the proxy hierarchy is so dominant that the precise replacement policy used at the upper level is irrelevant. LRU, LFU-Aging, and GD-Size are equally effective (or ineffective) at the parent cache.

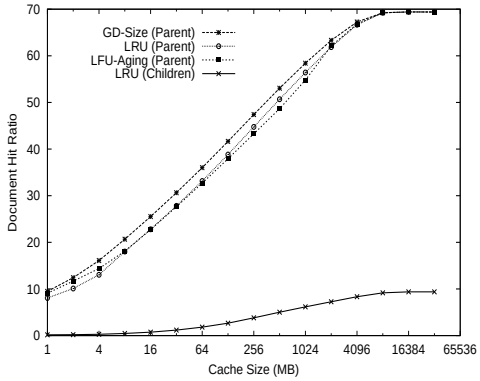
Figure 11 shows the results for size-based partitioning when the threshold size is 10,000 bytes. Compared to the results in Figure 10, Figures 11(a) and (b) show a significant improvement in performance for the lower level proxies, while the performance of the parent cache decreases. This trend occurs because the percentage of files that can be cached at the lower level of the hierarchy increases: approximately 80% of the requests are for files smaller than 10,000 bytes. The byte hit ratios for the child caches shift up noticeably. The drop in byte hit ratio for the parent cache is modest (compare Figure 10(b) and Figure 11(b)), compared to the drop in document hit ratio (compare Figure 10(a) and Figure 11(a)), because the large files cached at the upper level still contribute a significant fraction of the



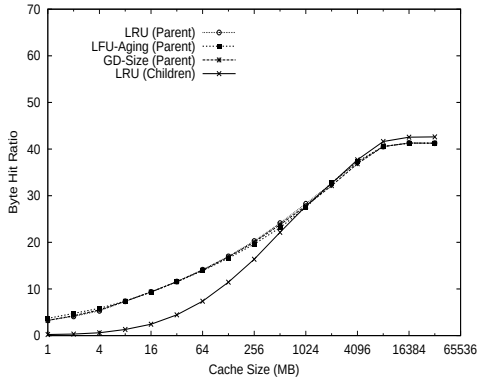
(a) Document Hit Ratio (Small files at the lower level)



(b) Byte Hit Ratio (Small files at the lower level)



(c) Document Hit Ratio (Large files at the lower level)



(d) Byte Hit Ratio (Large files at the lower level)

Fig. 11. Simulation Results for Size-Based Partitioning (Size Threshold = 10,000 Bytes)

total byte traffic volume.

The results for the reversed threshold policy (i.e., keeping large files at the lower level) are shown in Figures 11(c) and (d). Comparing these figures with Figures 10(c) and (d) shows that there is an increase in hit ratios for the parent cache, while the hit ratios for the child proxies decrease. By keeping only large files in the lower level proxies, the byte hit ratio is still high, but the document hit ratio is low because of fewer references to the large files. The GD-Size policy provides the best document hit ratio at the upper level, without compromising the byte hit ratio. Again, the byte hit ratio results for the parent level cache are largely independent of the replacement policy used.

Increasing the size threshold further ($S = 100,000$ bytes, not shown here) continues the same trends indicated previously [Busari 2000]. When small files are kept at the lower level, the document hit ratio at the parent drops drastically, though the byte hit ratio at the parent cache is still significant. Reversing the size threshold restriction improves hit ratios at the parent, but at the expense of the lower level

proxies.

4.6 Summary of Results

The foregoing experiments have illustrated the performance tradeoffs, in terms of document hit ratio and byte hit ratio, at both child-level and parent-level caches in a two-level Web proxy caching hierarchy. Several novel cache management strategies were explored, including heterogeneous cache replacement policies, and the partitioning of Web document content across the levels of the caching hierarchy based on document size.

The simulation results suggest performance advantages for the use of heterogenous replacement policies across the levels of a caching hierarchy. The results also suggest advantages for the use of size-based partitioning policies across the levels of a caching hierarchy. The latter policy is particularly attractive because of its (minimal) filtering effect on the workload characteristics as the request stream passes from one level of cache to the next. Combining heterogeneous replacement policies with the size-based partitioning approach makes little sense, since the heterogenous policies have their largest advantage on filtered request streams.

Determining which policy is "best", in terms of user-perceived response time, requires in-depth consideration of network capacity, network latencies, server load, HTTP, and TCP-level effects [Feldmann et al. 1999]. Such a rigorous evaluation is beyond the scope of the current paper. Further discussion of this issue, and an approximate cost-benefit analysis of selected caching strategies, appears in [Busari 2000].

5. CONCLUSIONS

This work used trace-driven simulations of Web proxy workloads to study cache filter effects in simple Web proxy caching hierarchies. First, the experiments demonstrated the impacts of cache size, cache replacement policy, Zipf slope, and hierarchy size on the document referencing characteristics at the next level of cache. Second, the paper considered two different approaches to cache management for a two-level Web proxy caching hierarchy. In particular, the experiments considered heterogeneous replacement policies within the hierarchy, and document partitioning across the levels of the hierarchy based on size.

The simulation results show that combining different replacement policies at different levels of the hierarchy can improve the performance of a caching hierarchy. The best performance was typically provided by the use of LRU or LFU-Aging at the lower level, combined with GD-Size at the upper level. The GD-Size policy is more effective at the parent cache because the cache filtering effects of the first-level cache result in a non-Zipf-like workload presented to the parent cache. Using GD-Size at both levels provides a better document hit ratio, but sacrifices the byte hit ratio. The results also show that the effectiveness of the parent cache depends a lot on the degree of overlap in the workloads of the child-level proxies.

For file partitioning, the simulation experiments show that size-based partitioning (with small files at the lower level of the hierarchy) can provide improved performance. However, the performance improvements are sensitive to the size threshold chosen.

Future work will involve studying network-level effects in Web proxy caching

hierarchies. Extending the ProWGen proxy workload generation tool to model document modifications, and the Web caching simulator to model cache consistency protocols, are also planned.

ACKNOWLEDGMENTS

The author is grateful to Mudashiru Busari and Alan Fedoruk for their substantial contributions to this work. Muda developed the ProWGen synthetic workload generator, and conducted the bulk of the simulation experiments on Web caching hierarchies as part of his M.Sc. thesis research at the University of Saskatchewan. Alan developed the scripts for the simulation and analysis of cache filtering effects, as part of his CMPT 855 course project at the University of Saskatchewan. This paper would not have been possible without their significant groundwork on this topic.

Financial support for this research was provided by TR*Labs* (Telecommunications Research Laboratories) in Saskatoon, NSERC research grant OGP0121969, and iCORE (Informatics Circle of Research Excellence).

REFERENCES

Abdulla, G., Fox, E., Abrams, M., and Williams, S. 1997. Www proxy traffic characterization with application to caching. Tech. rep., Virginia Tech. March.

Almeida, V., Bestavros, A., Crovella, M., and Oliveira, A. 1996. Characterizing reference locality in the www. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*. 92–103.

Almeida, V., Cesario, M., Fonseca, R., Meira Jr., W., and Murta, C. 1998. Analysing the behavior of a proxy server in light of regional and cultural issues. In *Proceedings of the 3rd International WWW Caching Workshop*.

Arlitt, M., Cherkasova, L., Dilley, J., Friedrich, R., and Jin, T. 1999. Evaluating content management techniques for web proxy caches. In *2nd Workshop on Internet Server Performance*. ACM.

Arlitt, M. and Jin, T. 2000. A workload characterization study of the 1998 world cup web site. *IEEE Network 14,* 3 (May/June), 30–37.

Arlitt, M. and Williamson, C. 1997a. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking 5,* 5 (October), 631–645.

Arlitt, M. and Williamson, C. 1997b. Trace-driven simulation of document caching strategies for internet web servers. *Simulation Journal 68,* 1 (January), 23–33.

Baentsch, M., Baum, L., Molter, G., Rothkugel, S., and Sturm, P. 1997a. Enhancing the web's infrastructure: From caching to replication. *IEEE Internet Computing 1,* 2 (March), 18–27.

Baentsch, M., Baum, L., Molter, G., Rothkugel, S., and Sturm, P. 1997b. World-wide web caching: The application level view of the internet. *IEEE Communications 35,* 6 (June), 170–178.

Bestavros, A., Carter, R., Crovella, M., Cunha, C., Heddaya, A., and Mirdad, S. 1995. Application-level document caching in the internet. In *Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments (SDNE'95)*. 166–173.

Bolot, J. and Hoschka, P. 1996. Performance engineering of www: Applications to dimensioning and cache design. In *Proceedings of 5th International World-Wide Web Conference*.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. 1999. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of IEEE INFOCOM*. 126–134.

Busari, M. 2000. Simulation evaluation of web caching hierarchies. M.S. thesis, University of Saskatchewan.

Busari, M. and Williamson, C. 2001a. On the sensitivity of web proxy cache performance to workload characteristics. In *Proceedings of IEEE INFOCOM*. 1225–1234.

Busari, M. and Williamson, C. 2001b. Simulation evaluation of a heterogeneous web proxy caching hierarchy. In *Proceedings of IEEE MASCOTS*. 379–388.

Cao, P. and Irani, S. 1997. Cost-aware www proxy caching algorithms. In *Proceedings of USENIX Symp. on Internet Technologies and Systems*. 193–206.

Chankhunthod, A., Danzig, P., Neerdaels, C., Schwartz, M., and Worrell, K. 1996. A hierarchical internet object cache. In *Proceedings of the 1996 USENIX Technical Conference*. 153–163.

Che, H., Wang, Z., and Tung, Y. 2001. Analysis and design of hierarchical web caching systems. In *Proceedings of IEEE INFOCOM*. 1416–1424.

Cherkasova, L. and Ciardo, G. 2000. Characterizing temporal locality and its impact on web server performance. In *Proceedings of ICCCN'2000*.

Cohen, E., Krishnamurthy, B., and Rexford, J. 1998. Improving end-to-end performance of the web using server volumes and proxy filters. In *Proceedings of ACM SIGCOMM*. 241–253.

Crovella, M. and Bestavros, A. 1997. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking 5,* 6 (December), 835–846.

Cunha, C., Bestavros, A., and Crovella, M. 1995. Characteristics of www client-based traces. Tech. rep., Boston University. July.

Doyle, R., Chase, J., Gadde, S., and Vahdat, A. 2001. The trickle-down effect: Web caching and server request distribution. In *Proceedings of the Web Caching and Content Delivery Workshop*.

Duska, B., Marwood, D., and Feeley, M. 1997. The measured access characteristics of world-wide web client proxy caches. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*. 23–35.

Fan, L., Cao, P., Almeida, J., and Broder, A. 1998. Summary cache: A scalable wide-area web cache sharing protocol. In *Proceedings of ACM SIGCOMM*. 254–265.

Feldmann, A., Caceres, R., Douglis, F., Glass, G., and Rabinovich, M. 1999. Performance of web proxy caching in heterogeneous bandwidth environments. In *Proceedings of IEEE INFOCOM*. 107–116.

Franklin, M., Carey, M., and Livny, M. 1992. Global memory management for client-server dbms architectures. In *Proceedings of the 19th International Conference on Very Large Databases (VLDB)*.

Gwertzman, J. and Seltzer, M. 1995. An analysis of geographical push-caching. In *Proceedings of 5th IEEE Workshop on Hot Topics in Operating Systems*. 51–55.

Heideman, J., Obraczka, K., and Touch, J. 1997. Modeling the performance of http over several transport protocols. *IEEE/ACM Transactions on Networking 5,* 5 (October), 616–630.

Jin, S. and Bestavros, A. 2000a. Greedy-dual* web caching algorithms: Exploiting the two sources of temporal locality in web request streams. In *Proceedings of the 5th Web Caching Workshop*.

Jin, S. and Bestavros, A. 2000b. Sources and characteristics of web temporal locality. In *Proceedings of IEEE MASCOTS*. 28–35.

Mahanti, A. 1999. Web proxy workload characterization and modeling. M.S. thesis, University of Saskatchewan.

Mahanti, A., Eager, D., and Williamson, C. 2000. Temporal locality and its impact on web proxy cache performance. *Performance Evaluation 42,* 2–3 (October), 187–203.

Mahanti, A. and Williamson, C. 1999. Web proxy workload characterization. Tech. rep., University of Saskatchewan. March. http://www.cs.usask.ca/faculty/carey/papers/workloadstudy.ps.

Mahanti, A., Williamson, C., and Eager, D. 2000. Traffic analysis of a web proxy caching hierarchy. *IEEE Network 14,* 3 (May/June), 16–23.

Povey, D. and Harrison, J. 1997. A distributed internet cache. In *Proceedings of the 20th Australian Computer Science Conference*.

Roadknight, C., Marshall, I., and Vearer, D. 1999. File popularity characterization. In *Proceedings of the 2nd Workshop on Internet Server Performance (WISP'99)*.

Rodriguez, P., Spanner, C., and Biersack, E. 1999. Web caching architectures: Hierarchical and distributed caching. In *Proceedings of the 4th Web Caching Workshop*. 37–48.

Tewari, R., Dahlin, M., Vin, H., and Kay, J. 1999. Beyond hierarchies: Design considerations for distributed caching on the internet. In *Proceedings of the 19th International Conference on Distributed Computing Systems*.

Valloppillil, V. and Ross, K. 1998. Cache array routing protocol v1.1. Tech. rep., Internet Draft. February.

Weikle, D., McKee, S., and Wulf, W. 1998. Caches as filters: A new approach to cache analysis. In *Proceedings of IEEE MASCOTS*. 2–12.

Wessels, D. and Claffy, K. 1998. Icp and the squid web cache. *IEEE Journal on Selected Areas in Communication 16,* 3 (April), 345–357.

Willick, D., Eager, D., and Bunt, R. 1993. Disk cache replacement policies for network fileservers. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*. 2–11.

Wolman, A., Voelker, G., Sharma, N., Cardwell, N., Karlin, A., and Levy, H. 1999. On the scale and performance of cooperative web proxy caching. In *Proceedings of ACM Symposium on Operating Systems Principles*. 16–31.

Yu, P. and MacNair, E. 1998. Performance study of a collaborative method for hierarchical caching in proxy servers. In *Proceedings of World-Wide Web Conference*. 215–224.

Zhang, L., Floyd, S., and Jacobson, V. 1997. Adaptive web caching. In *Proceedings of the NLANR Web Caching Workshop*.

Received August 2001; September 2001; accepted September 2001.