Exploring the Bounds of Web Latency Reduction from Caching and Prefetching

Thomas M. Kroeger[†]
Department of Computer Engineering
University of California, Santa Cruz

Darrell D. E. Long[‡]
Department of Computer Science
University of California, Santa Cruz

Jeffrey C. Mogul[§]
Digital Equipment Corporation
Western Research Laboratory

Abstract

Prefetching and caching are techniques commonly used in I/O systems to reduce latency. Many researchers have advocated the use of caching and prefetching to reduce latency in the Web. We derive several bounds on the performance improvements seen from these techniques, and then use traces of Web proxy activity taken at Digital Equipment Corporation to quantify these bounds.

We found that for these traces, local proxy caching could reduce latency by at best 26%, prefetching could reduce latency by at best 57%, and a combined caching and prefetching proxy could provide at best a 60% latency reduction. Furthermore, we found that how far in advance a prefetching algorithm was able to prefetch an object was a significant factor in its ability to reduce latency. We note that the latency reduction from caching is significantly limited by the rapid changes of objects in the Web. We conclude that for the workload studied caching offers moderate assistance in reducing latency. Prefetching can offer more than twice the improvement of caching but is still limited in its ability to reduce latency.

1 Introduction

The growth of the Web over the past few years has inspired researchers to investigate prefetching and caching as techniques to reduce latency [1, 12, 15]. While such techniques have seen significant success reducing latency in storage systems [7, 8, 9, 14] and in processor memory hierarchies [13], it remains to be seen how effective such techniques can be within the World Wide Web.

We classify caching and prefetching into four different methods and then derive bounds on these methods. Using traces taken over a three week period at Digital Equipment Corporation, we quantify these bounds.

We assume the use of a proxy server as an intermediary between the client (browser) and the web server. This proxy server accepts requests from clients and satisfies them using data that has been prefetched, cached or retrieved directly from an appropriate web server. This configuration is quite common in the Web today. Whether a proxy is used or not, this model serves to partition the latency of Web retrievals into two components: *external latency*, caused by network and web server latencies that are external to an organization, and *internal latency*, caused by networks and computers within the bounds of an organization. Figure 1 illustrates a common configuration.

Because the proxies are normally located on the organization's network, the communication between the client and proxy is normally a small portion of the overall latency. On the other side, the proxy-server communication normally accounts for a significant majority of the total event latency. The primary goal of proxy-based caching and prefetching is to reduce the amount of time the client waits for data by reducing or removing external latency. In our traces, external latency accounts for 77% of the latency seen in our entire trace set and 88% of the latency seen by subset of clients geographically close to the proxy.

With this potential for such a significant performance gain, the best improvement we saw from caching and prefetching reduced total latency by 60%. Additionally, we saw that the prefetching lead time, the amount of time between when prefetching begins and when the object is needed, significantly affects the amount of latency

[†] tmk@cse.ucsc.edu. Supported in part by Digital Equipment Corporation and the Office of Naval Research under Grant N00014–92–J–1807.

[‡] darrell@cse.ucsc.edu. Supported in part by the Office of Naval Research under Grant N00014–92–J–1807.

[§] mogul@pa.dec.com.

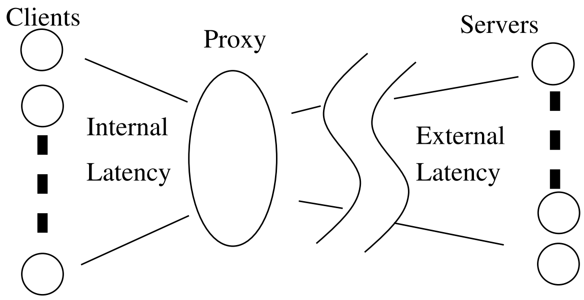


Figure 1: Typical proxy configuration

reduction. We found that when we limited our simulator to providing only a three-minute warning, the latency reduction dropped to 43%. Additionally, we observe that the latency reduction from caching was half of what it would have been for a data set with data objects that did not change. This observation agrees with several studies that show a high rate of change for objects in the web [11, 6, 3]. Comparing our results with the external latency we observe that, for the workload examined, Web latency consists of 23% internal latency, 20% external latency that cannot be cached or prefetched and 57% external latency that can be removed through prefetching and caching. The key point that we take from these results is that while caching and prefetching are helpful, under the current conditions there is a limit to their ability to reduce latency.

The rest of this article is organized as follows: section 2 categorizes caching and prefetching and presents four methods for using these techniques. We then present bounds for each method. In §3 we discuss the methodology used to quantify these bounds. In §4 we present the results of our simulations. Related work is then addressed in §5, and we present our conclusions in §6.

2 Bounding Caching and Prefetching

Our goal is to determine an upper bound on the effectiveness of proxy-based caching and prefetching for reducing latency in the Web. We classify caching and prefetching algorithms into different categories. We then construct models for bounding the performance of each category of algorithm. In order to ensure that our bounds are widely applicable, we do not present any specific algorithms and attempt to keep our analysis as general as possible.

2.1 Categories of Caching and Prefetching

We distinguish between caching algorithms based on whether a cache will remain *passive*, taking action only as a result of requests or if it is *active*, prefetching data in anticipation of future requests. We distinguish between prefetching algorithms based on where the information used to determine what to prefetch originates.

Traditionally, caching is thought of as a system that only reacts to requests. We define a *passive* cache as one that only loads a data object as a result of a client's request to access that object. In contrast, we use the term *active* cache to refer to caches that employ some mechanism to prefetch data. We note that passive caching systems also serve to reduce network traffic. However, for this work we focus on the use of passive caching for latency reduction.

We categorize prefetching into two categories, *local* and *server-hint*, based on where the information for determining which objects to prefetch is generated. In *local* prefetching, the agent doing the prefetching (*e.g.* a browser-client or a proxy) uses local information (*e.g.* reference patterns) to determine which objects to prefetch. Prefetching algorithms that do not make use of information from the server, whether employed at a client or at a proxy, would be considered local prefetching.

In *server-hint* based prefetching, the server is able to use its content specific knowledge of the objects requested, as well as the reference patterns from a far greater number of clients to determine which objects should be prefetched. The actual prefetching, however, must be done by an agent closer to the client. Therefore, the server provides hints that assist this agent (either a proxy or client) in prefetching. Implementation of this model is complicated by the requirement for modifications at both the client or proxy side as well as the server side.

Given these options for caching and prefetching, we examine four different methods: passive proxy caching with unlimited storage; an active cache with local prefetching and unlimited storage; server-hint based prefetching alone; and an active cache with server-hint based prefetching and unlimited storage.

2.2 Bounding Analysis of Prefetching and Caching

We set upper bounds for each model by basing our simulations on some best-case assumptions. We assume that each method works with full knowledge of future events. Then we place certain restrictions on each method.

For passive caching, we assume that a previously accessed object that has not been changed is still available from the cache.

For local prefetching, since an object must be seen at least once before it can be predicted for prefetching, we assume that only the first access to an object will not be prefetched, and that all subsequent accesses will be successfully prefetched. We do not assume the use of additional information, outside the traced reference stream, that would allow the prediction of a future reference to a URL that has never been seen in the trace.

For server-hint based prefetching, we assume that prefetching can only begin after the client's first contact with that server. Because we assume a system with full knowledge of future events, without this restriction each server would schedule the transfer of each object to complete just before the object was requested. Provided there is enough bandwidth, this model would eliminate all communication latency. In such a system servers would suddenly transfer objects to clients with which they had never before been in contact. To provide a more realistic and useful bound on the performance of server-hint based prefetching, we assume that in order for a server to provide prefetch hints for a client, it must have been accessed by that client. In this *first-contact* model, upon the first contact from a client, a proxy will simultaneously prefetch all of that client's future requests from that server. So, for example, if you contact *www.cnn.com* on Tuesday, this model would assume that all of your requests to *www.cnn.com* for Wednesday, Thursday, and Friday would have been prefetched on Tuesday.

Even this first-contact model may be somewhat unrealistic. We investigated the effects of placing limits on the prefetching lead time, and on the amount of data prefetched simultaneously. To limit the amount of data prefetched simultaneously we place a threshold on the bandwidth that can be used for prefetching. After the first contact, subsequent requests are scheduled for immediate prefetch until this bandwidth threshold is reached or exceeded. The next request is then only scheduled to begin once some current prefetches have completed and the bandwidth being used has gone below the threshold. We varied, bandwidth and lead time independently and in combination.

Finally, to bound the performance of active caching using server-hint based prefetching and unlimited storage, we test if an object could be found in a passive cache. If this is not the case, we test if this object could have been successfully prefetched under the first-contact model.

3 Quantifying Bounds

We used trace-based simulation to quantify these bounds. To obtain traces, we modified the firewall Web proxy used by Digital Equipment Corporation to record all HTTP requests from clients within Digital to servers on the Internet. The traces ran from from 29 August to 22 September 1996. Each event in this trace stream represents one Web request-response interaction for a total of 24,659,182 events from 17,354 clients connecting to 140,506 servers. This proxy provided no caching or prefetching; it simply served as a method for crossing the corporate firewall.

To provide separate samples for comparison, we extracted three mid-week segments from the traces, each covering a Monday through Friday. We labeled these samples *Week 1* through *Week 3* for each work week and *all* for the entire trace stream. We also examined a subset of the trace data consisting only of clients in Palo Alto, CA, where the proxy was located. These samples are labeled *PA 1* through *PA 3* and *PA all*. The workload from this subset would be more representative of a smaller, more localized organization.

Since our main concern is the latency seen when retrieving a Web page, our simulations focus on requests that use the HTTP protocol and the *GET* method. We ignored events that failed for any reason (*e.g.* the connection to the server failed during data transfer). For all of our analyses, we assumed that query events and events with *cgi-bin* in their URL cannot be either prefetched or cached. This convention is used by most proxy caches.

In our simulations, we use the observed total and external latencies to estimate total (t), external (e) and internal ($i = t - e$) event latencies that would be seen if this request were to occur again. If our model says that a request could have been successfully prefetched or cached, then we use the internal latency as our estimate for the modeled event's new duration ($n = i$). Otherwise, we use the previously observed total latency ($n = t$). Given these values, we then quantify the fraction of the latency reduced as $(t - n)/t$. This approximation ignores the possibility that the proxy was transferring information to the client during its communication with the server. However, since we are looking for a lower bound on the latency of this event, which is the same as an upper bound on the latency reduction, we can ignore this possibility.

Using these approximations and the bounding models described in §2, our simulation steps through each event in the trace stream and determines whether it could have been cached or prefetched. We compare the distribution of event latencies seen under these bounding models with those in the original trace (measured without caching or prefetching), in order to compute the average latency reduction, for the workload represented in these traces.

3.1 Sources of Inaccuracy

We note several possible sources of inaccuracy for the results presented here. Our assumption that URLs

Table 1: Passive caching with unlimited storage.

Measurement	Week 1	Week 2	Week 3	All	PA 1	PA 2	PA 3	PA All
Total latency t	4.6	4.7	4.3	4.1	2.8	2.4	2.7	2.4
External latency e	3.6	3.6	3.2	3.2	2.6	2.2	2.4	2.1
New latency n	3.5	3.6	3.4	3.0	2.5	2.2	2.4	2.0
Percentage external latency e/t	79%	77%	75%	77%	90%	90%	88%	88%
Total latency reduction	24%	22%	22%	26%	12%	11%	11%	15%
Hit ratio	48%	47%	48%	52%	19%	20%	20%	28%
Cache size (GB)	23	26	27	88	1.3	1.2	1.1	4.5

Latencies are averages in seconds. *PA 1–3* and *PA all* represent work week 1 through work week 3 and the entire trace stream for the Palo Alto subset.

Table 2: Bounds on latency reductions from local prefetching.

Measurement	Week 1	Week 2	Week 3	All	PA 1	PA 2	PA 3	PA All
Percentage external latency	79%	77%	75%	77%	90%	90%	88%	88%
Total latency reduction	41%	38%	36%	45%	26%	22%	24%	33%

with queries and *cgi-bin* cannot be prefetched or cached might cause our upper bound on latency reduction to be less than the best possible latency reduction. An algorithm that is able to cache or prefetch such items could see greater latency reductions then those presented here.

Our traces lack last-modified timestamps for about half of the entries because many servers do not provide this information. This left us to use changes in response size as the only indicator of stale data for these requests. The result is a simulation that in some cases would simulate a cache hit when one should not occur, causing us to overestimate the potential for latency reduction.

Our models assume that the latency for retrieving a successfully prefetched or cached item from the proxy is the time of the original event minus the time for proxy-server communications ($t - e$). This assumes that there is little or no overlap between the server-to-proxy data transfer and the proxy-to-client data transfer. When this assumption is wrong, as it is for the larger responses and more distant clients, our simulation will overestimate the possible latency reduction.

4 Results

The goal of our simulations was to use the workload traced to quantify the four bounding models presented in §2.2. We first present the latency reductions for passive caching and active caching with local prefetching. We then examine the *first-contact* model with and without unlimited storage caching. We address the variation in latency reduction by examining the distribution for latency reduction for each event and for the different types of objects that were requested.

4.1 Passive Proxy Caching Performance

First, we simulated a passive caching proxy with unlimited storage. Table 1 shows the results. The first three rows show the averages for original total latency, external latency and the simulated latency for a passive caching proxy. The next row shows what fraction of the original total latency was contributed by the external latency. This ratio serves as a limit: if we could remove all external latency, then our average event latency would be reduced by this percentage. The last three rows show the percent of latency reduced by the simulated caching proxy, the cache hit ratio and the size of the cache that would be required to hold all cached data.

From this table we can see that passive caching is only able to reduce latency from 22%–26% (15% for a smaller organization), a far cry from the 77% (or even 88%) of external latency seen in the traces. Also, while the cache hit ratio ranges from 47%–52%(19%–28%), the latency reduction is only half of that. This implies that the majority of the requests that saw a cache hit are for objects smaller than the average event, which confirms a similar observation made by Williams *et al.* [15]. That study showed a weak inverse relationship between the likely number of accesses per unchanged response and the response size.

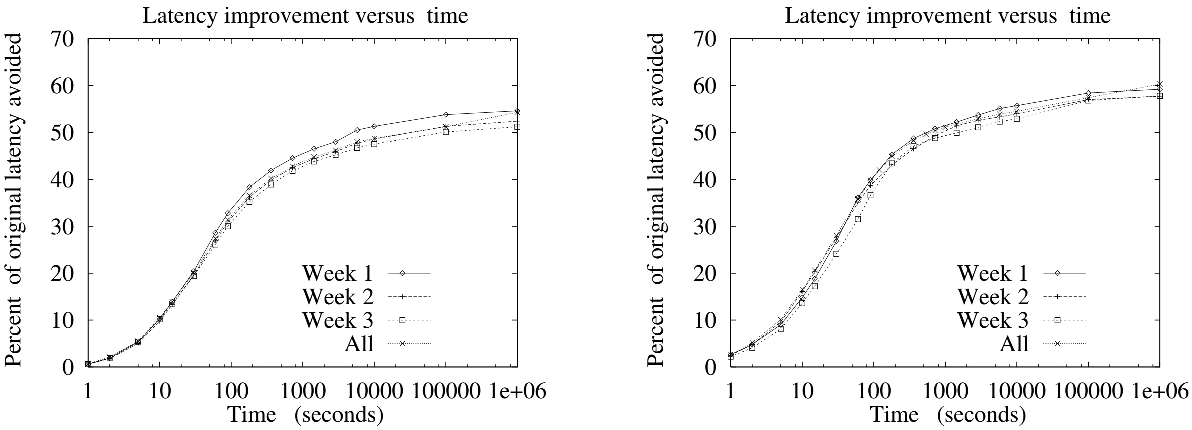
In Table 1, the latency reduction, hit ratio and cache size are larger for the entire-trace columns than for the single-week columns. This occurs because with the longer traces, there is a higher chance that given object will be referenced.

4.2 Local Prefetching

Next, we simulated prefetching based on locally-available information. Here, we assume that an object

Table 3: Results of server hint-based prefetching for an unlimited first-contact model.

Measurement	Week 1	Week 2	Week 3	All	PA 1	PA 2	PA 3	PA All
% external latency	79%	77%	75%	77%	90%	90%	88%	88%
Total latency reduction without caching	53%	51%	50%	53%	57%	56%	56%	58%
Total latency reduction with caching	58%	56%	54%	57%	59%	58%	57%	60%



(a) All clients in Digital

(b) Digital Palo Alto clients

Figure 2: Percentage of latency reduced versus how far in advance requests may be prefetched.

may be prefetched if and only if it has been seen before in the reference stream. This model provides a bound on active caching with local prefetching and unlimited storage. This model differs from the passive-caching model in that even if an object has changed (as indicated by a change in either the size or the last-modified timestamp), a subsequent access to that object can still be prefetched.

Table 2 shows that the latency reduction bound for local prefetching is almost double that for passive caching (see Table 1). The two results differ because the passive cache pays for a miss when an object changes; the high observed rate of change [3, 11] is what causes much of the poor performance of passive caching.

4.3 Bounds on Prefetching with Server Based Hints

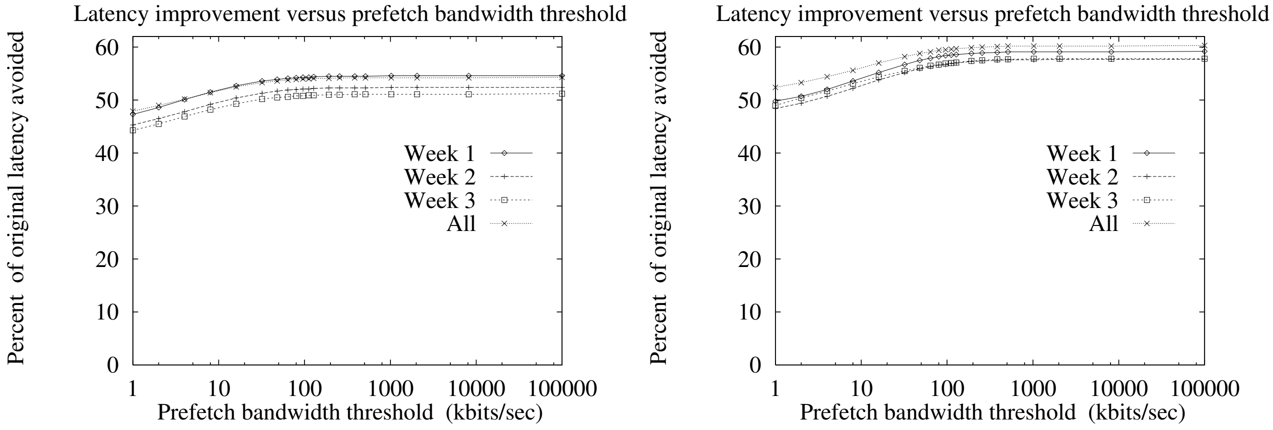
To simulate server-hint based prefetching, we assume that prefetching can only begin after the client has contacted the server for the first time. To simulate a combination of caching and prefetching, our simulator first checks if an object is in the cache. If not, then the simulator uses the server-hint based model to determine if this object could have been successfully prefetched. Table 3 shows that this first-contact model, where all future requests are simultaneously prefetched upon first contact, will reduce the latency by a little more than half. For a smaller, more centralized organization (as represented by the Palo Alto subsets), even though the external latency increases to 88% of the total latency, the improvement from server-hint based prefetching is only slightly better. In combination with an unlimited storage caching, server-hint based prefetching provides at best a 60% latency reduction.

4.3.1 Limiting the First-Contact Model

We modified our first-contact model to provide a more realistic bound by placing a limit on how far in advance of a request the data could be prefetched, also known as *prefetch lead time*. We then further modified our model to limit the amount of data prefetched simultaneously.

In order to examine the effects of limiting *prefetch lead time*, we modified our simulation to forget about contacts between client-server pairs that have been inactive for longer than a specified interval. This means that any subsequent request from the given client to the given server will be treated as a first contact. Figure 2 shows the results of varying prefetch lead times from one second to ∞ (represented in the figures as one million seconds). We note that for lead times below a few hundred seconds, the available reduction in latency is significantly less impressive.

To limit the amount of overlap between prefetch requests, we set a threshold on the bandwidth that can



(a) All clients in Digital

(b) Digital Palo Alto clients

Figure 3: Percentage of latency reduced versus bandwidth threshold.

used for prefetching. Figure 3 shows how this bandwidth limit affects the amount of latency reduction. At the left end of the graph, a small increase in the amount of bandwidth limiting prefetching significantly improves the reduction in latency. However, the total difference between unlimited simultaneous prefetching and sequential prefetching is no greater than 11%.

Figure 4 and Table 4 show the effects of varying both limits. The slope in Figure 4 along the axis for prefetch lead time shows that this parameter is the dominant factor. The relatively consistent slopes in the surface graphs imply the two parameters are relatively independent in their effects on latency. Therefore, the curves in Figures 2 and 3 are representative of the behavior of this model over variations in both parameters.

Table 5 shows the latency reduction seen by a server-hint based prefetching model with unlimited cache storage, for representative values of prefetch time and bandwidth available for prefetching. Comparing Table 5 with Table 4, we note that for a weak or moderately capable prefetching algorithm, the use of caching is especially important. For example, for a prefetching algorithm that can predict requests up to 3 minutes in advance, and has bandwidth threshold of 8 kbits/sec, caching will still offer an increase of approximately 11%. On the other hand, for an unbounded prefetching model, caching only offers an improvement of approximately 4%.

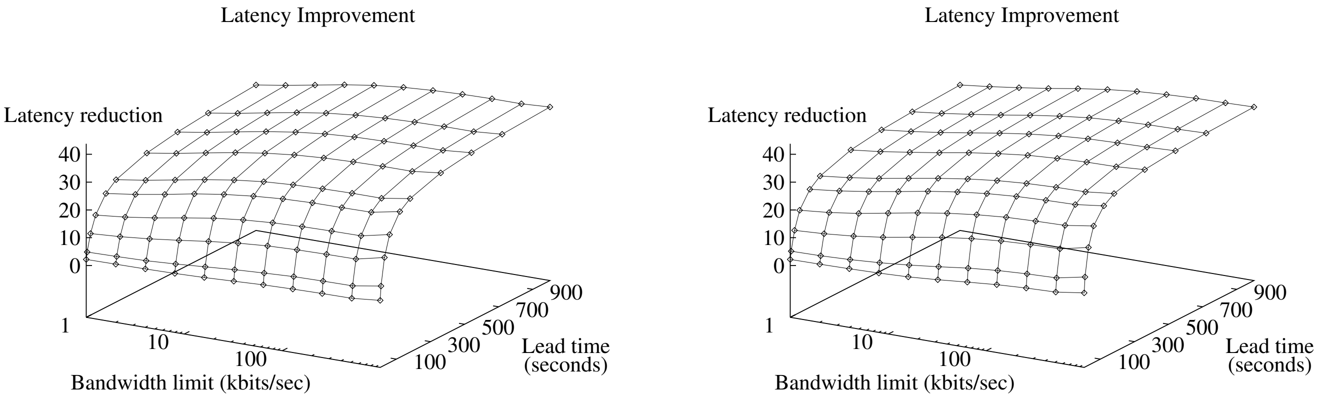
4.4 Reductions for Different Object Types

We examined how the type of object requested affects the latency reduced (object types were determined by an extension and if any, at the end of the URL path). Table 6 shows the results from caching and prefetching listed by object type. The category *NONE* represents URLs that had no extension, and the category *OTHER* represents objects with extensions other than those listed. The first two rows show the average for original total latency and the simulated latency. Rows 3 and 4 show the percentage of the total event stream that each type accounts for by event count and event duration, respectively. Rows 5 through 8 show the percentage of latency reduced by type for passive caching, local prefetching with unlimited cache storage, server-hint based prefetching without caching and server-hint based prefetching with unlimited storage caching.

This table shows that for the most part, the common types (*GIF*, *HTML*, *JPEG*, *CLASS*, *DATA*) all offer slightly above average latency reductions, while the less common or type-ambiguous requests offer significant less benefit. This result suggests that a cache which only kept objects of common types might make more effective use of its storage resources.

4.5 Variation in Latency Reduction

To examine the variation of latency reduction across separate events, Figure 5 shows histograms of the percent of latency reduced at each event for passive caching, local prefetching, server-hint based prefetching and server-hint based with unlimited storage caching. These graphs show a bi-modal distribution where an event either sees significant latency reductions (the peaks around 95% and 100%) or little to no reduction (the peak at 0%). What these distributions show is that under the models we have simulated one can expect a web request to either see minimal to no latency reduction or a significant



(a) All clients in Digital

(b) Digital Palo Alto clients

Figure 4: Reduction in latency for prefetching (bandwidth in kbits/sec, time in seconds). Note: the scale for the X axis is logarithmic.

Table 4: Selected latency reduction percentages for prefetching (bandwidth in kbits/sec, time in seconds).

b/w	Time	Week 1	Week 2	Week 3	All	PA 1	PA 2	PA 3	PA All
1	10	4.2%	3.9%	4.2%	4.3%	4.0%	5.1%	3.7%	5.6%
8	10	5.3%	4.9%	5.2%	5.3%	5.5%	6.6%	5.1%	7.0%
∞	10	9.8%	9.3%	9.6%	9.9%	13.8%	14.9%	12.9%	15.3%
1	60	16.1%	15.3%	14.9%	15.6%	18.7%	19.0%	16.7%	20.0%
8	60	20.8%	19.6%	19.1%	20.0%	24.7%	23.8%	21.4%	24.9%
∞	60	27.7%	26.1%	25.2%	26.5%	34.6%	33.2%	30.0%	33.9%
1	180	25.6%	24.0%	23.5%	24.4%	29.6%	27.7%	27.5%	29.5%
8	180	31.5%	29.4%	28.8%	29.8%	35.3%	33.0%	33.3%	34.7%
∞	180	37.4%	35.2%	34.2%	35.6%	43.6%	41.1%	41.7%	42.7%
1	∞	46.2%	44.3%	43.1%	46.7%	47.9%	46.3%	47.2%	50.1%
8	∞	50.4%	48.1%	47.0%	50.2%	51.7%	50.1%	51.2%	53.3%
∞	∞	53.4%	51.3%	49.9%	53.0%	57.1%	55.5%	55.8%	57.8%

reduction in the total latency.

5 Related Work

Many researchers have looked for ways to improve current caching techniques.

Padmanabhan and Mogul [12] described a server hint-based predictive model. In this model, the server observes the incoming reference stream to create a Markov model predicting the probability that a reference to some object A will be followed, within the next n requests, by a reference to some other object B (n is a parameter of the algorithm). On each reference, the server uses this model to generate a prediction of one or more subsequent references and sends this prediction as a hint to the client, including it in the response to the current reference. The client may then use this hint to prefetch an object if the object is not already in the client's cache. In their simulations, they estimated reductions of as much as 45%, but note that such techniques will also double the network traffic. Nevertheless, they show that a reasonable balance of latency reduction and network traffic increase can be found.

Bestavros *et al.* [1] have presented a model for the speculative dissemination of World Wide Web data. This work shows that reference patterns from a Web server can be used as an effective source of information to drive prefetching. They observed a latency reduction of as much as 50%, but only at the cost of a significant increase in the bandwidth used.

Williams *et al.* [15] presented a taxonomy of replacement policies for caching within the Web. Their experiment examined the hit rates for various workloads. The observed hit rates that range from 20% to as high as 98%, with the majority ranging around 50%. The

Table 5: Latency reduction percentages for both prefetching and caching (bandwidth in kbits/sec, time in seconds).

b/w	Time	Week 1	Week 2	Week 3	All	PA 1	PA 2	PA 3	PA All
1	10	26.2%	23.9%	23.5%	27.7%	15.0%	15.1%	13.6%	18.9%
8	10	26.8%	24.5%	24.0%	28.2%	16.2%	16.3%	14.7%	19.9%
∞	10	29.1%	26.8%	26.3%	30.2%	22.9%	23.0%	21.1%	26.0%
1	60	32.8%	30.6%	29.9%	33.6%	27.5%	27.4%	25.1%	30.6%
8	60	35.3%	33.1%	32.3%	35.8%	32.5%	31.4%	28.9%	34.4%
∞	60	38.9%	36.7%	35.7%	38.8%	40.8%	39.1%	36.0%	41.2%
1	180	39.3%	36.8%	35.9%	39.3%	36.8%	35.1%	35.0%	38.5%
8	180	42.4%	40.0%	38.8%	42.0%	41.4%	39.6%	39.7%	42.6%
∞	180	45.7%	43.4%	42.0%	44.9%	48.3%	46.2%	46.6%	48.6%
1	∞	53.5%	51.3%	50.0%	54.0%	50.2%	50.0%	49.9%	53.4%
8	∞	55.9%	53.6%	52.3%	55.9%	53.5%	53.3%	53.4%	56.1%
∞	∞	57.7%	55.5%	54.0%	57.4%	58.5%	58.1%	57.4%	59.9%

Table 6: Latency reduction for prefetching and caching by type of data requested (times in seconds).

Type	NONE	HTML	GIF	DATA	CLASS	JPEG	MPEG	OTHER
New latency	1.3	2.4	1.6	0.3	1.6	3.9	110.8	12.9
Original latency	2.5	4.3	3.3	1.5	3.8	7.2	151.7	17.0
Percentage by count	29.5%	12.1%	42.4%	0.2%	0.3%	11.1%	0.0%	5.4%
Percentage by time	17.8%	12.5%	34.3%	0.1%	0.3%	19.4%	0.4%	15.2%
Passive caching	26.7%	21.8%	34.5%	73.0%	40.4%	27.1%	4.5%	8.8%
Local prefetching	48.3%	43.7%	52.7%	79.9%	58.2%	45.8%	26.9%	24.3%
Server-hints without caching	49.3%	61.6%	62.5%	66.8%	64.4%	62.7%	27.5%	27.9%
Server-hints with caching	51.1%	61.5%	60.1%	79.9%	65.0%	58.4%	26.4%	27.6%

workload with the hit rate of 98% comes from a cache that is placed close to the server, rather than close to the clients, with the purpose of reducing internal bandwidth consumed by external HTTP requests. This workload addresses a different problem than that examined here. For their other workloads, our cache hit rates are reasonably comparable.

Several of the results found by other studies are close to or higher than the bounds resulting from our simulations. We note that these results are highly dependent on the workload, and on the environment modeled. One should take care in applying the results of any of these simulation studies, ours included, to a specific situation.

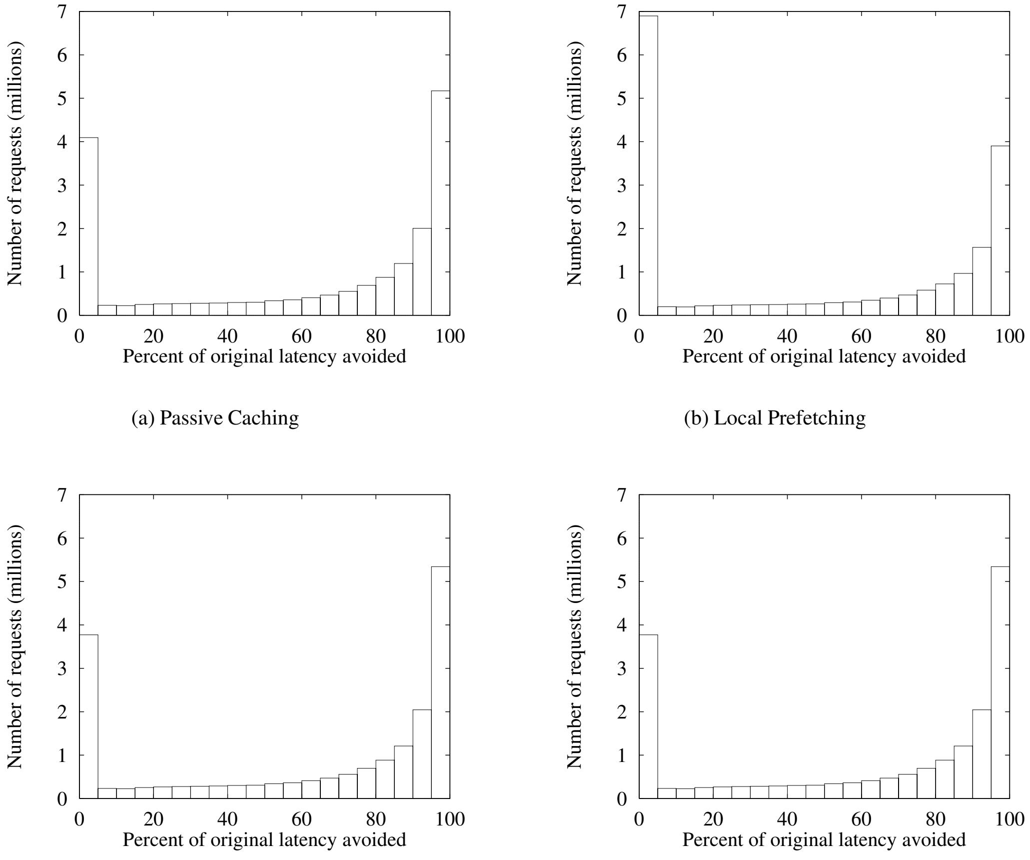
6 Summary and Conclusions

Using trace driven simulations we have explored the potential latency reductions from caching and prefetching. For the workload studied passive caching, with an unlimited cache storage, can reduce latency by approximately 26%. This disappointing result for passive caching is in part because data in the Web continually changes. On the other hand prefetching based on local information offers, saw a bound of approximately 41% reduction in latency. Adding server-hints increased this bound to approximately 57%.

We observed that prefetch lead time is an important factor in the performance of prefetching. Under more realistic constraints (prefetch bandwidth threshold of 8 kbits/sec, lead time of 3 minutes), the best we could hope for is a 35% reduction in latency. We also saw that the uncommon types of objects provided significantly less then average latency reduction.

Finally, we can make some observations based on comparing the results from these models with the total external latency. In the workload studied, 57% of the total latency is external latency that can be removed by caching or prefetching; 20% is external latency that cannot be removed, from events such as first contacts or queries and 23% is internal latency.

The results we present here serve to illustrate two conclusions. First, caching and prefetching can be effective in reducing latency in the Web. Second, the effectiveness of these techniques does have limits. The bounds produced by our simulations are for off-line algorithms that have full knowledge of the future. One would expect any on-line algorithm to be less effective. Again, we caution that these results are highly dependent on the workload and environment modeled. They should be applied with care. Nevertheless these results emphasize the need for improved prefetching techniques as well as additional techniques beyond caching and prefetching. These techniques might include wide-area replication [5], delta encoding [2, 11] and persistent TCP connections [10] which has been included in the HTTP/1.1 protocol [4].



(a) Passive Caching

(b) Local Prefetching

(c) Server-Hint Based without Caching

(d) Server-Hint Based with Caching

Figure 5: Histograms of the percentage of latency reduced at each event.

Acknowledgments

We are grateful to Digital Equipment Corporation for their support of this work and the use of their facilities, the Office of Naval Research for their support, Prof. Mary G. Baker for her kind guidance, Dr. Richard A. Golding for his insightful comments and enthusiasm, Prof. Pei Cao for her comments and input on this work, Carlos Maltzahn and Dr. Kathy Richardson for their work with the Squid proxy, Dr. Lawrence Brakmo for his input and comments on early portions of this work, Randal Burns, Tracey Sconyers and the other members of the concurrent systems group that provided comments on this work, and Suzan Fry.

Availability

Additional details on these traces, and the traces, are available from:

ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html

References

[1] A. Bestavros and C. Cunha, "A prefetching protocol using client speculation for the WWW," Tech. Rep. TR-95-011, Boston University, Department of Computer Science, Boston, MA 02215, Apr. 1995.

[2] R. C. Burns and D. D. E. Long, "Efficient distributed backup with delta c1ompression," in *Proceedings of the 1997 I/O in Parallel and Distributed Systems (IOPADS'97), San Jose, CA, USA*, Nov. 1997.

[3] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul, "Rate of change and other metrics: A live study of the world wide web," in *Proceedings of*

First USENIX Symposium on Internet Technologies and Systems, Dec. 1997.

[4] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, and T. Berners-Lee, “Hypertext transfer protocol – HTTP/1.1,” RFC 2068, HTTP Working Group, Jan. 1997.

[5] J. Gwertzman and M. Seltzer, “The case for geographical push caching,” in *Fifth Annual Workshop on Hot Operating Systems*, (Orcas Island, WA), pp. 51–55, IEEE Computer Society, May 1995.

[6] J. Gwertzman and M. Seltzer, “World wide web cache consistency,” in *Proceedings of the USENIX 1996 Annual Technical Conference*, (San Diego, CA), pp. 141–152, USENIX, Jan. 1996.

[7] T. Kimbrel, A. Tomkins, R. H. Patterson, B. Bershad, P. Cao, E. W. Felton, G. A. Gibson, A. Karlin, and K. Li, “A trace-driven comparison of algorithm for parallel prefetching and caching,” in *Proceedings of Second USENIX Symposium on Operating Systems Design and Implementation*, pp. 19–34, USENIX, October 1996.

[8] T. M. Kroeger and D. D. E. Long, “Predicting file-system actions from prior events,” in *Proceedings of the USENIX 1996 Annual Technical Conference*, USENIX, January 1996.

[9] H. Lei and D. Duchamp, “An analytical approach to file prefetching,” in *Proceedings of USENIX 1997 Annual Technical Conference*, USENIX, January 1997.

[10] J. C. Mogul, “The case for persistent-connection HTTP,” in *Proceedings of the 1995 SIGCOMM*, pp. 299–313, ACM, Sept. 1995.

[11] J. C. Mogul, F. Douglis, A. Feldmann, , and B. Krishnamurthy, “Potential benefits of delta encoding and data compression for HTTP,” in *Proceedings of the 1997 SIGCOMM*, (Cannes, France), pp. 181–194, ACM, Sept. 1997.

[12] V. N. Padmanabhan and J. C. Mogul, “Using predictive prefetching to improve world wide web latency,” *Computer Communications Review*, vol. 26, pp. 22–36, July 1996.

[13] A. J. Smith, “Cache memories,” *ACM Computing Surveys*, vol. 14, pp. 473–530, Sept. 1982.

[14] J. S. Vitter and P. Krishnan, “Optimal prefetching via data compression,” *Journal of the ACM*, vol. 43, pp. 771–793, September 1996.

[15] S. Williams, M. Abrams, C. R. Standridge, C. Abdulla, and E. A. Fox, “Removal policies in network caches for world-wide web documents,” in *Proceedings of the 1996 SIGCOMM*, pp. 293–305, ACM, July 1996.