17th ACM Symposium on Operating Systems Principles (SOSP '99)
Published as *Operating Systems Review* **34**(5):16–31, Dec. 1999

On the scale and performance of cooperative Web proxy caching

Alec Wolman, Geoffrey M. Voelker, Nitin Sharma,
Neal Cardwell, Anna Karlin, and Henry M. Levy

Department of Computer Science and Engineering
University of Washington
{*wolman,voelker,nitin,cardwell,karlin,levy*}*@cs.washington.edu*

Abstract

While algorithms for cooperative proxy caching have been widely studied, little is understood about cooperative-caching performance in the large-scale World Wide Web environment. This paper uses both trace-based analysis and analytic modelling to show the potential advantages and drawbacks of inter-proxy cooperation. With our traces, we evaluate quantitatively the performance-improvement potential of cooperation between 200 small-organization proxies within a university environment, and between two large-organization proxies handling 23,000 and 60,000 clients, respectively. With our model, we extend beyond these populations to project cooperative caching behavior in regions with millions of clients. Overall, we demonstrate that cooperative caching has performance benefits only within limited population bounds. We also use our model to examine the implications of future trends in Web-access behavior and traffic.

1 Introduction

Cooperative caching – the sharing and coordination of cache state among multiple communicating caches – has been shown to improve the performance of file and virtual-memory systems in a high-speed, local-area network environment [3, 17]. For example, when a file-page miss occurs, the local file cache may transfer the page from the file cache on another node. Cooperative caching works in this environment because network transfer time is much smaller than the disk access time required to service a miss.

Internet proxy caching has become a commonplace approach for improving the performance of Web browsers. Typically, the proxy sits in front of an entire company or organization. By caching requests for a group of users, a proxy can quickly return documents previously accessed by other clients. Ultimately, though, the hit rate of the proxy is a function of the size of the population it manages – a size often dictated by political, organizational, or geographic considerations. An obvious question, then, is whether multiple proxies should cooperate with each other in order to increase total client population, improve hit ratios, and reduce document-access latency. Whether such cooperative proxy caching is a useful architecture for improving performance depends on a number of factors. These include the sharing patterns of documents across organizations, the ratio of inter-proxy communication time to server fetch time, and the scale at which cooperation is undertaken.

Several cooperative-caching protocols have been proposed [9, 16, 28, 30, 33]; however, few studies have examined cooperative Web caching from a systemic viewpoint. As a result, we know neither the environments in which cooperative caching is useful (if any) nor its potential performance benefits. Answering such questions has been difficult in the past, because studying proxy cooperation requires *simultaneous* traces from multiple proxies.

In this paper, we take a two-pronged approach to exploring the limits and potentials of cooperative proxy caching. As the first approach, we collect and analyze traces from two environments: the University of Washington and the Microsoft Corporation. As a key component of our university trace, we identify each client in terms of its membership in one of about 200 university departments or programs. This gives us the equivalent of a simultaneous trace of 200 diverse, independent organizations, permitting us to analyze document sharing among those organizations and to measure the potential benefits of cooperation among organization-based proxies. We examine latency and bandwidth benefits of proxy caching for this data, as well. We then use the Microsoft trace of employee traffic to the Internet to explore the potential of cooperation between larger organizations. To do this, we analyze traces from Microsoft and the university that we collected over the same time period and processed with the same anonymization function. This permits a direct computation of the degree of document sharing, and hence the benefit of sharing, between two proxies each handling tens of thousands of clients.

As the second approach, we develop an analytic model of Web behavior that extends beyond the limits of our trace results. The model permits us to examine the impact of larger population sizes, to explore the tradeoffs among various cooperative-caching schemes, and to speculate on the performance implications of future trends.

Our results show the benefits of cooperative caching among collections of small organizations. However, we show that cooperative caching is unlikely to have significant benefits for larger organizations or populations. That is, with current sharing patterns, there is little point in designing highly scalable cooperative-caching schemes; all reasonable schemes will have similar performance in the low-end population range where cooperative caching works. Thus, the crucial problem that must be solved to improve Web performance is how to increase document cacheability.

The paper is organized as follows. The next section examines previous work and its relationship to our study. Section 3 describes our traces and presents and analyzes data from the traces. Section 4 develops an analytic model of steady-state Web proxy caching, and uses the model to study the performance of large-scale proxy caches. Section 5 then uses the model to compare cooperative caching schemes, and Section 6 summarizes and concludes.

2 Related work

Web tracing and caching are highly active research areas. Recent studies of Web traffic include analyses of Web access traces from the perspective of browsers [11, 26], proxies [2, 5, 6, 7, 10, 13, 14, 18, 20, 23, 30], and servers [1, 4, 29]. Earlier tracing studies were limited in request rate, number of requests, and diversity of population. The most recent tracing studies have been larger and more diverse. In addition to static analysis, some studies have also used trace-driven cache simulation to characterize the locality and sharing properties of very large traces [2, 6, 14, 18, 20, 23] and to study the effects of cookies, aborted connections, and persistent connections on the performance of proxy caching [6, 18].

Researchers have studied the temporal locality of Web proxy traces and examined how hit-ratio depends, asymptotically, on cache size and the number of requests. Several interesting findings have been identified. First, it has been well-documented that, for most traces, the relative frequency with which Web pages are requested follows a Zipf-like distribution, where the number of requests to the i^{th} most popular document is proportional to $1/i^{\alpha}$ for some constant α [2, 5, 8, 11, 19, 24]. Second, for infinite-sized caches, it has been shown empirically and analytically that the hit ratio for a Web proxy grows logarithmically with the client population of the proxy and the number of requests seen by the proxy [5, 8, 14, 20, 24].

There has also been extensive work on cooperative Web caching as a technique to reduce access latency and bandwidth consumption. Cooperative Web caching proposals include hierarchical schemes like Harvest and Squid [9, 32], hash-based schemes [21, 35], directory-based schemes [16, 27, 33] and multicast-based schemes [28, 34]. Although each of these research efforts included a performance evaluation of the protocols proposed and a discussion of algorithm scalability, only [22] presents empirical evaluations of cooperation for small populations, and none present empirical or analytical evaluations of the effectiveness of their schemes for the large client populations found in a wide-area setting.

Using client traces, Krishnan et al. studied the utility of cooperation among three Bell Labs proxies with a small user population [22]. They concluded that cooperative Web caching can be useful, but that a cache manager was necessary to dynamically determine when to cooperate, because of the extra server load imposed by cooperation.

This paper expands on these previous research efforts. We use trace-based analysis to quantify the potential advantages and drawbacks of inter-proxy cooperation for small- and medium-sized organizations. We use analytic modelling to examine cooperative-caching performance in wider-area environments. Our model is inspired by the model of Breslau et al. [5]. We have augmented their assumptions with a model of document rate of change, client request rate, and a small number of other parameters that allow us to study the steady-state behavior of cooperative-caching schemes as a function of the characteristics of the population they serve.

3 Web document sharing and proxy caching

This section poses and answers a number of questions about the potential of cooperative proxy caching. Before considering specific caching algorithms, we will explore the bounds of cooperative-caching performance. Key questions include:

1. What is the best performance one could achieve with "perfect" cooperative caching?

2. For what range of client populations can cooperative caching work effectively?

3. Does the way in which clients are assigned to caches matter?

4. What cache hit rates are necessary to achieve worthwhile decreases in document access latency?

To answer these questions quantitatively, we have collected and analyzed Web access data from two environments: (1) the University of Washington (UW), consisting of about 50,000 students, faculty, and staff, and (2) the Microsoft Corporation (MS), consisting of about 40,000 employees. Most significantly, the two traces were collected simultaneously and anonymized in the same way, allowing direct comparison of trace records, including URLs and server addresses. We use the UW trace as a way to analyze document sharing by 200 small, independent organizations within a diverse university; we use the UW and Microsoft traces together as a way to analyze document sharing across two large organizations. In both environments, we examine the potential benefits of proxy cache coordination from the perspective of the clients and the network; in this study, we do not investigate the effects of cooperative caching on server load in general or under hot spot conditions.

The following section presents results derived from these traces. In Section 4, we present an analytical model that goes beyond the limits of the trace to much larger populations.

Parameter	UW	Microsoft
HTTP Requests	82.8 million	107.7 million
HTTP Objects	18.4 million	15.3 million
Total Request Bytes	677 GB	(N/A)
Average Requests/Sec	137	199
Clients	22,984	60,233
Servers	244,211	360,586
Duration	7 days	6 days 6 hours

Table 1. Overall trace statistics.

3.1 Trace collection and characteristics

While several client traces exist in the public domain [11, 13, 14, 20, 26], most are several years old, and the information they contain is inadequate for our analysis. We therefore designed and implemented custom trace software and installed it on a computer connected to the outgoing switches through which our university's Internet traffic flows. Few proxies are deployed in our university, so we are able to see most of the Web client traffic generated from inside the university at the border. The traces are anonymized, but the anonymization preserves certain key aspects of the data that we require. In particular, we anonymize client IP addresses, but we first classify the client based on its "organizational" membership. This allows us to identify requests from different academic and administrative entities within the university, and classify each entity as a unique organization. We describe the use of this information later in this section.

We have been collecting university traces since October 1998; the trace used in this section was collected from May 7th through May 14th, 1999 and therefore shows very recent access characteristics. Table 1 presents the high-level details of this trace. As the table shows, we saw about 83 million requests by 23,000 clients to 244,000 servers over the seven-day period in the UW trace. Using this data, we can determine upper bounds on the performance of any cooperative caching algorithm. This tells us whether proxy cooperation is worthwhile even in the best case in our environment.

We also processed traces collected by the proxies handling all outgoing traffic from Microsoft Corporation. These traces were collected on the same days that we collected our UW trace. Our software anonymizes both traces using the same functions, so that URLs and server IP addresses in them can be directly compared. Table 1 shows that in the May 7 to May 14th, 1990 period, we saw about 108 million requests by 60,000 clients to 360,000 servers in the Microsoft trace.

3.2 Simulation methodology

The results presented in this section are based on Web cache simulations using our traces as input. This section discusses the methodology used for the experiments we performed.

We make assumptions in our simulator that a real cache would not make, and therefore do not model reality exactly. However, our goal is to investigate behavior, not to exactly reproduce hit rates, and we believe that our assumptions do not change our conclusions about cache behavior. The caches we simulate are infinite-sized and do not model expirations. As a result, they are somewhat optimistic. Real caches will incur misses due to capacity limitations that we do not model. However, capacity misses are rarely the bottleneck for Web caches. For example, only three percent of the requests to the Microsoft Web proxies from which we gathered our traces missed due to the finite capacity of the proxies (which have 9GB of RAM and 180GB of disk capacity). Real caches will also expire some objects that our simulations keep alive in the cache.

At the same time, our simulation experiments are conservative, because they include compulsory (cold start) misses. We minimize this effect by simulating traces over long periods of time. We also exclude the effect of compulsory misses using our steady-state model in Section 4.

We simulate two kinds of Web caches, a "practical" cache and an "ideal" cache. A practical cache closely models the cacheability of documents according to the algorithms in the Squid V2 implementation [32]. Our cacheability predicate accounts for HTTP 1.1 cache control headers, cookies, object names with suffixes naming dynamic objects, no-cache pragmas, uncacheable methods and response codes, and headers with Authorization and Vary fields. We reported the detailed breakdown of document cacheability in our traces in an earlier paper [37].

An ideal cache treats all documents as cacheable. It is well known that some Web objects, such as images used in advertisements, are marked uncacheable even though their contents do not change and could be cached. Future improvements to Web protocols and cache implementations can potentially be more aggressive and cache those objects that practical proxies cannot currently cache. Since we cannot anticipate all future improvements and implement them in our simulator, we instead use an ideal cache to report the upper bound that such improvements can hope to achieve on our workloads.

Many of the experiments examine cache performance as a function of client population. Because each UW modem is reused by many people, using a modem IP address to represent a client would be inaccurate. As a result, we exclude modem traffic in our analyses and focus on LAN users.

3.3 The impact of population size

In a cooperative-caching scheme, a proxy forwards a missing request to other proxies to determine if: (1) another proxy holds the requested document, and (2) that document can be returned faster than a request to the server. Whether such cooperation is worthwhile will depend on the number of proxies involved, their distances (inter-proxy communication latencies), their utilizations, the client populations served, and the complexity of the protocols used.

The result of cooperative proxy caching is simply to increase the effective client population. That is, at best, a collection of cooperating caches will achieve the hit rate of a single proxy acting over the *combined* population of all the proxies. In reality, the performance will be less than perfect, because proxies will not have perfect knowledge and will pay the overheads of inter-proxy communication latency. Examining a single, top-level proxy thus gives us an *upper*

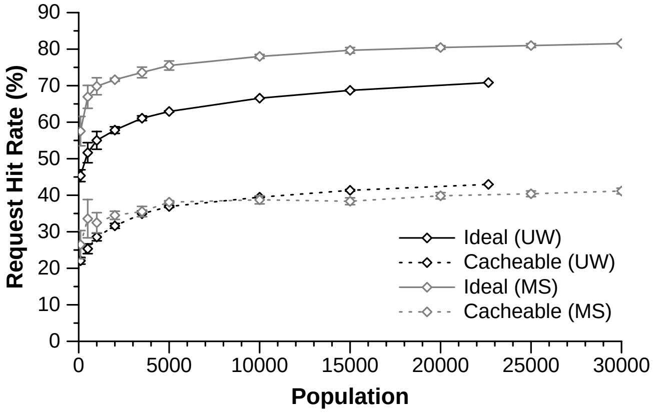

Figure 1. Proxy cache request hit rate as a function of client population.

bound on cooperative-caching performance.

Figure 1 graphs hit rate vs. client population size for our traces.[1] The dark black lines in this graph show the behavior of the university population. The dotted black line shows the "cacheable hit rate," which corresponds to the hit rate from a practical cache that considers the cacheability characteristics of the documents. Both the shapes of the curves and the cacheable hit rates are roughly consistent with previous proxy cache studies of client traces: [20, 23] both report hit rates above 50% (but disregard cookies), [14] reports hit rates of 40–45%, and [6] reports hit rates of 35% (but exaggerates the effects of cookies, which can often be cached in HTTP 1.1).

The solid black line in Figure 1 shows the "ideal hit rate" of an ideal cache for the UW trace, one that would be achievable *if* all shared documents were cacheable. In the future, improvements to Web protocols may move the cacheable line closer to the ideal line. On the other hand, future changes in document characteristics may move the cacheable line in either direction. We explore this issue further in Section 4.

The grey lines in Figure 1 show the behavior of the Microsoft population. It is interesting to note that the ideal curve asymptote is higher by about 13% than that of the university environment, an indication that document sharing within Microsoft is much higher than within the university. This suggests, not surprisingly, that the Microsoft population is much more homogeneous in its Web-access behavior than the university population. However, we also see that the Microsoft cacheable curve almost overlies the UW cacheable curve. This is a direct reflection of the different distributions of requests to cacheable documents in the traces; 60% of requests in the UW trace go to cacheable documents, but only 51% of requests in the Microsoft trace do so. As a result, even though there is more *sharing* among Microsoft users, document cacheability currently prevents a Microsoft proxy cache from achieving a hit rate that is any better than a UW proxy cache.

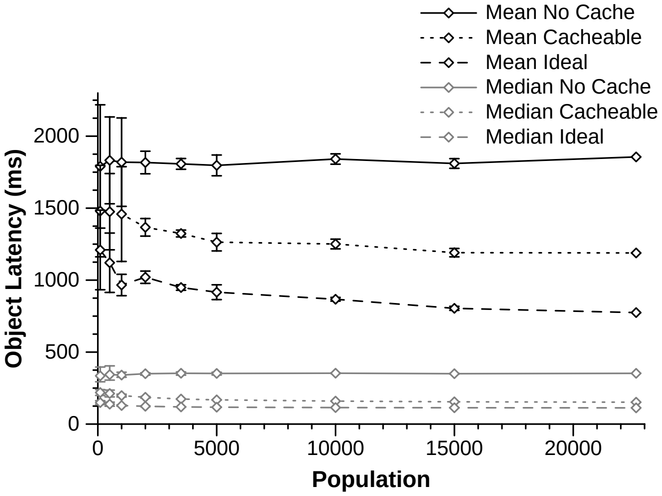

Figure 2. Mean and median request latency as a function of client population for the UW trace. The error bars on the median curves are the min and max medians across the trials.

From this simple figure, we can draw several key conclusions about cooperative caching. The graphs have a sharp knee at about 2500 clients. The steep increase in hit rate below that knee implies that a large potential benefit (hit rate increase) could exist from cooperative caching for multiple proxies with small client populations. For example, given 10 proxies, each handling a population of a few hundred clients, cooperative caching has the potential to significantly improve the hit ratio seen by the clients of those proxies. The improvement occurs by increasing the total population of each proxy from 200 to 2000 clients.

It is important to note that the total number of clients (below the knee) that can benefit from cooperative caching *could easily be handled by a single proxy cache* for our traces and user populations. Often this will not be possible, however, because decisions of proxy placement are based on political or geographical factors, such as company organization, location, and so on. While one organization may not trust another to proxy all of its requests, it may be willing to cooperate with other proxies for performance reasons.

Figure 1 also shows that hit rate increases very slowly with client population once past the knee of the curve. It is therefore not clear whether cooperative caching is beneficial in this region, for proxies whose populations are already above a few thousand clients. We will explore this question in more detail below and in Section 4.

3.4 Hit rate vs. latency and bandwidth

While many caching studies focus on hit rate, in the Web environment it is ultimately latency, not hit rate, that is crucial to clients. From the perspective of Internet service providers, hit rate translates into bandwidth savings over costly Internet links. These bandwidth savings can also reduce wide-area network congestion, potentially improving the performance of the Internet as a whole.

[1] In this graph and the others that follow, the clients for a given population size are randomly selected out of the pool of clients seen in our traces. Unless otherwise specified, each point on the graphs shown is the mean of four independent random trials, and error bars show the standard deviation across these trials.

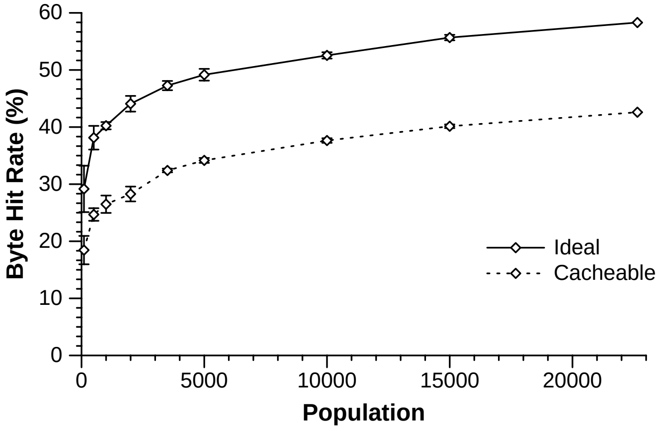

Figure 3. Proxy cache byte hit rate as a function of client population for the UW trace.

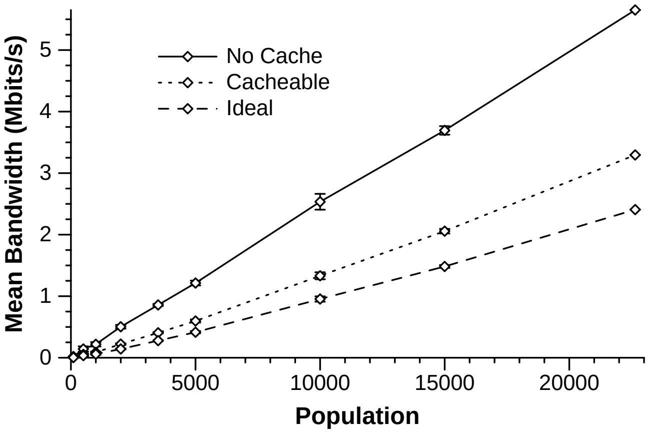

Figure 4. Bandwidth consumed as a function of client population for the UW trace.

Figure 2 shows document latency in our university trace as a function of the number of clients using a proxy cache. The top three lines, from top to bottom, show mean last-byte latency: (1) without a proxy cache (i.e., as extracted from the trace), (2) with a proxy cache respecting cacheability, and (3) with an ideal proxy cache. The bottom three lines show median last-byte latency under the same three conditions. Note that, unlike other curves, the error bars on the median lines correspond to the minimum and maximum median values among the trials at each population size. The three mean lines level out quickly, while the medians are essentially flat. This implies that caching will have little impact on mean and median latency beyond very small client populations. The mean trace curve is not constant, because each point represents requests from different client population samples, and mean latency will vary from one sample to another.

In Figure 2, the mean latency is much higher than the median due to many high-latency documents. Can cooperative caching reduce the percentage of these high-latency documents? Calculating the percentage of documents with a last-byte latency below two seconds, though, implies that this is not the case. When graphed as a function of population, this percentage is effectively a horizontal line for all three caching policies described above with very little difference among them. The ideal line is the highest (90%), and no cache is the lowest (82%). The insensitivity to population and closeness of these values demonstrate that neither cacheability nor increasing population will significantly reduce the number of high-latency documents. Our trace analysis indicates that these documents are slow due to document size, network latency (e.g., from congestion, low-bandwidth links, long distances), or both. Also, as described below, shared documents tend to be smaller than non-shared ones, biasing misses towards larger documents that consequently take longer to download.

A final dimension is bandwidth. Figure 3 shows the byte hit rate as a function of client population for the university trace. Once again, we see a knee in the curve at around 2500 clients. Comparing these results to the hit rates given in Figure 1, we can conclude that shared objects are smaller on average than other objects. Figure 4 shows the average bandwidth consumed as a function of client population for the three caching situations. We see that while caching reduces bandwidth consumption compared to no caching, there is no benefit to increased client population (i.e., there is no decrease in the slope of the bandwidth line).

3.5 Proxies and organizations

We produced Figures 1 and 3 by computing the hit rate of random subsets of clients at each population size; therefore, the figures assume that all clients are essentially identical in their access patterns. A crucial question is whether clients in a single organization, sharing a single proxy, have more in common with each other than with clients in different organizations sharing other proxies. If there is high locality within organizations, then populations smaller than the knee in Figure 1 could achieve the maximum hit rate. What benefit would clients in *real* organizations see if their proxies were to cooperate with other *real* organizational proxies?

To answer this question is difficult, because it requires a simultaneous trace of a large number of proxies in the Internet. Such traces have not existed in the past. We have tried to answer the question in our university environment, using UW as a small-scale model of the broader networked community. The University of Washington consists of a large collection of diverse organizations, e.g., museums of art and natural history, schools of nursing and dentistry, and departments such as music, Scandinavian languages, and computer science. Think of each such organization as an independent business entity, with its own interests and focus, which would typically have its own proxy sitting on its connection to the Internet.

In fact, somewhat fortuitously, few organizations on our campus currently employ proxies; this permits us to see most outgoing client requests and their responses. In our tracing software, before anonymizing each client's IP address, we first classify that client as belonging to one of about 200 independent university organizations. In this way, we preserve organizational membership information while protecting client identities. In effect, this gives us a simultaneous

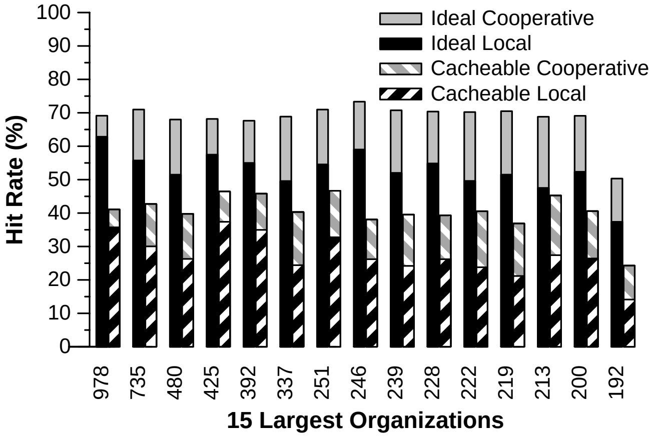

Figure 5. Breakdown of local and global proxy hit rates for the 15 largest UW organizations.

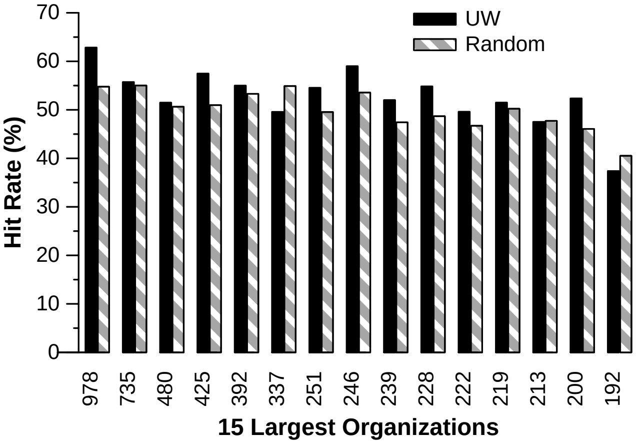

Figure 6. Comparison of the proxy hit rates for the 15 largest UW and randomly populated organizations.

trace of Web requests from 200 organizations.

Figure 5 shows the ideal (left-hand bar) and cacheable (right-hand bar) hit rates for the 15 largest UW organizations. The bars are labeled on the x-axis with the number of clients seen in the trace for each of the organizations shown: the smallest organization had 192 clients, while the largest had 978 clients. These bars thus represent 15 medium-sized companies or client communities.

The lower portion of each bar shows the hit rate that would be seen by a local proxy acting on behalf of that organization. The upper portion of the bar shows the *improvement* in hit rate that would be seen by that organization if *all* of the university's organizations used perfectly cooperating proxies. For the ideal bars, the average hit rate per group is 52%. The average cooperative-caching hit rate, i.e., the rate that would be seen by a single proxy over all organizations or by a perfect cooperative-caching scheme, is 69%. For the cacheable hit rate bars, the average hit rate per group is 29%, and the average cooperative hit rate is 38%. Therefore, if perfect cooperative caching were possible, it would achieve a noticeable improvement in hit rate for these proxies.

An interesting question raised above is whether clients in our UW population request documents randomly, or whether their access patterns are related in some way to those of other members of the same organization. To answer this question, we grouped clients at random into organizations with the same sizes as the 200 real organizations and compared local hit rates. Figure 6 shows a comparison of the hit rates for the 15 largest UW and randomly assigned organizations. The result indicates that there is a small (about 4%) average increase in hit rates for the real organizations compared to the randomly assigned ones. Therefore, there is some locality in organizational membership, but the impact is not significant in this case.

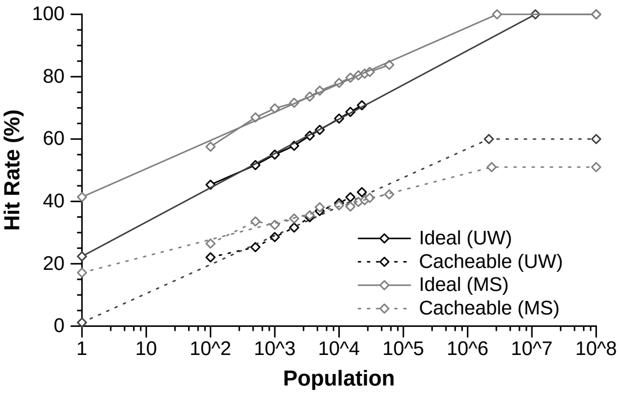

Figure 7. Proxy cache hit rate as a function of client population.

A related question is whether a better grouping exists of clients to proxies, for example, one based on each client's document interests. To examine this question, we conducted a clustering study. Using the trace data, we clustered clients based on their document access vectors, using a standard clustering algorithm (K-Means) that attempts to optimize intra-cluster sharing. The existence of such groupings within the university might imply the existence of an improved wide-area caching scheme based on clustering.

As with the university organizations, we compared the cluster-based client assignments to random assignments of clients into groups of the same size. The randomly assigned clusters have a consistently lower hit rate than the optimally clustered organizations. Somewhat surprisingly, the difference between the two assignments is just slightly more than the difference between the UW and random organizations (about 5%). Again, there is some affinity in client access patterns, but the impact on hit rate is not significant.

3.6 Impact of larger population size

We have seen that cooperative caching can increase hit rate, perhaps substantially, below the knee of the curve in Figure 1. What happens above the knee, i.e., can cooperative caching be effective in a wide-area network? Figure 7 shows the data from Figure 1 when it has been plotted on a log scale as a function of client population, fitted linearly using least squares, and then extrapolated past the client population we measured.

This graph suggests a number of interesting conclusions. First, the slopes of the UW lines are greater than the Microsoft lines, a relation we could only infer indirectly from Figure 1. Second, we notice that the slopes of the UW ideal and cacheable lines are similar. This indicates that there is little correlation between sharing and the cacheability of documents for the UW population. However, the slope of the Microsoft cacheable line is only 60% of the slope of the ideal line. This indicates that cacheable documents were shared to a lesser degree than uncacheable documents for the Microsoft population. Third, the cacheability curves are limited by the fraction of requests to cacheable documents, which is 60% in the UW trace and 51% in the Microsoft trace. In both of these cases, cooperative caching among populations larger than 2.4 million does not increase the hit rate to cacheable documents. Fourth, even if all documents were cacheable, the ideal hit rate reaches a maximum at a population of 11 million users for the UW trace and 2.9 million for the Microsoft trace.

The Microsoft client population is more homogeneous than the university population and therefore sees a higher degree of document sharing. Quantitatively, 83.8% of requests in the Microsoft trace are to previously requested documents; in the UW trace, 70.8% of requests are to previously requested documents. These statistics are the “ideal hit rates” that would be seen by a cache, if all documents were cacheable. But how much overlap is there to popular documents between the two populations? We looked at the most popular documents requested by each of the two populations, where we defined “most popular” to be those documents accessed more than 500 times. In the UW trace, there were 11,500 such documents, and in the Microsoft trace, there were 17,000 such documents. Looking at the 1000 most popular in each of the two populations, we see a 33% overlap; that is, of the 1000 most-popular documents accessed by Microsoft, 330 of them are also among the 1000 most popular accessed by UW. Therefore, many of the *same* documents are popular in both organizations.

In Section 4 we will present an analytical model to look in more detail at the behavior of cooperative caching in large-scale environments. Within the context of our own traces, though, we can perform an interesting experiment to create a larger population. Suppose that we implemented cooperative caching between the University of Washington and the Microsoft proxies. From the perspective of the UW proxy, this increases the size of the population it sees by a factor of 3.6; from the point of view of the Microsoft proxy, population increases by a factor of 1.4. What is the impact of combining the two populations through cooperative caching?

To estimate the benefit of cooperative caching between the two organizations, we did the following analysis. We ran the university trace through a simulated UW proxy; we then fed all misses to a second-level (cooperating) proxy preloaded with all of the objects seen in the Microsoft trace. Similarly, we ran the Microsoft trace simulating its proxy, with a second-level proxy preloaded with all objects seen by the UW trace. This gives us an idea of the maximum *incremental* hit-rate benefit each proxy would see if the two proxies would cooperate.

Figure 8 shows the results of this measurement. From the figure, we see that the UW proxy, whose effective population would increase 3.6-fold (from 23K to 83K clients), would see its ideal hit rate increase only 5.1% – from 70.1% to 75.9%; its cacheable hit rate would increase only 4.2% – from 42.7% to 46.9%. The Microsoft proxy, whose effective population would increase by a factor of 1.4 (from 60K to 83K clients), would see its ideal hit rate increase 2.7% – from 83.8% to 86.5%; its cacheable hit rate would increase only 2.1% – from 42.3% to 44.4%. To allow a more direct comparison of these results, we ran another experiment where the second-level cooperating proxy was preloaded by only a portion of the Microsoft population, thereby increasing the effective population size for the UW population by the same factor of 1.4. In this experiment, the ideal hit rate for the UW proxy increased by 1.6%, and the cacheable hit rate increased by 1.3%. When scaled by equal factors, it is interesting to note that Microsoft gains more benefit by cooperating with the UW population than the UW population gains by cooperating with Microsoft.

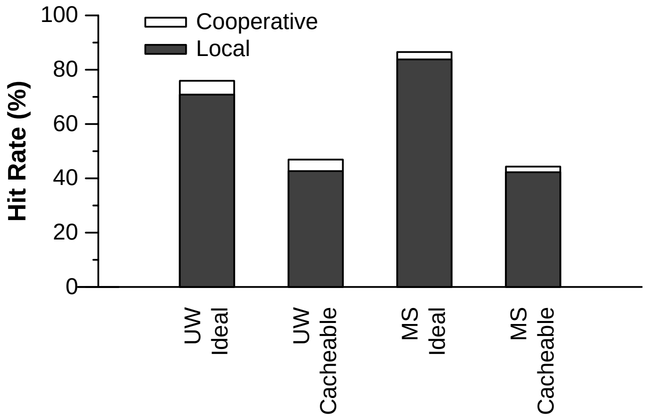

Figure 8. Hit rate benefit of cooperative caching between UW and Microsoft proxies.

These results are disappointing, but not surprising. The reason for these very small increases is that the *unpopular* documents are universally unpopular; therefore, it is unlikely that a miss in one of these large populations will find the document in the other population’s proxy. For the most-popular documents, cooperation does not help either, because only the first access (of the 500 plus accesses to a popular document) has the potential to benefit from another population’s proxy.

3.7 Summary

In this section, we analyzed traces from two environments to explore the bounds of cooperative-caching performance. From this data, we saw that:

1. The behavior of cooperative caching is characterized by two different regions of the hit rate vs. population curve. For smaller populations, hit rate increases rapidly with population; it is in this region that cooperative caching can be used effectively. However, these population sizes can be handled by a single proxy. Therefore, cooperative caching is only necessary to adapt to proxy assignments made for political or geographical reasons.

2. For larger populations (beyond the knee of the population vs. hit rate curve), cooperative caching is unlikely to provide significant benefit. We demonstrate this using our simultaneous traces of the Microsoft and university populations: a four-fold increase to a large university population via cooperative caching netted only a 2.7% increase in cacheable hit rate.

3. Although there exist organizations with significant locality, such as the Microsoft population, our clustering study shows that cooperative caching specialized to interest groups is unlikely to be effective.

4. While some of these general trends have been observed before, the inevitable conclusion for cooperative caching has not been explicitly noted.

Despite these negative conclusions, it is still possible that cooperative caching will perform better, either in steady-state or for different (possibly future) Web characteristics. In the domain where cooperative caching is effective, we also wish to understand which of the proposed cooperative-caching algorithms will be most beneficial. We explore these questions in the next two sections.

4 An analytic model of Web accesses

In Section 3 we considered the performance potential of cooperative caching using our trace data. In this section, we extend these results analytically to better understand the steady-state performance of cooperative caching for large client populations and to speculate on caching performance in light of future trends. We then examine the tradeoffs between various cooperative-caching schemes in Section 5.

4.1 Steady-state performance

The results in Section 3 bound the performance of caching schemes in the short term – over a period of one week. How would these results change if the cooperative caching systems were up and running for a month, a year, five years? In other words, how will cooperative caching schemes perform over the long run? In this section we examine the *steady-state* or *limiting* performance of caching schemes.

Our model of steady-state performance assumes that a cache can store all cacheable documents in the Web and that there are no capacity misses in the workload[2]. With this assumption, it is conceivable that in the long term the hit rate for cacheable documents would approach 100% since the relative importance of cold-cache effects, such as compulsory misses, would diminish. For example, this would happen if the Web were static, i.e., documents were not changing and new documents were not being generated. In this case, *all* documents would eventually be requested, and no further misses would be incurred thereafter. On the other hand, if new documents are constantly being created and old documents are changing, the hit rate for cacheable documents might remain low even over the long run.

The ultimate performance of a cache will therefore depend upon the rate at which documents change compared to the rate at which documents are requested. If the request rate dominates document rate of change, then the cache will still achieve near optimal hit rates. One request will miss in the cache whenever a document changes, but all subsequent requests to that document will be hits. However, if the request rate does *not* dominate document rate of change, then the cache will perform poorly. Repeated requests to a document will often find that the document has changed, so most of those requests will be misses. Since increasing the population served by a cache also increases the request rate, cooperative caching increases the likelihood that document request rate dominates document rate of change.

Similarly, the creation of new documents in the Web introduces cold misses into the workload when those documents are requested. As with document rate of change, if the Web grows slowly compared to the request rate, then caches will perform well: only a small fraction of requests will result in cold misses. However, if the Web grows significantly faster than the request rate, then the cache will be dominated by cold misses and will perform poorly.

We use our steady-state model to explore these effects in detail. We begin by introducing the model and then describe its parameterization in Section 4.3. Section 4.4 shows performance results of the model.

4.2 The model

Our model is inspired by that of Breslau et al. [5]. We make the following assumptions about clients and documents.

- There are N clients in the population. Clients are indistinguishable and act independently of one another.

- The total number of documents is n. For simplicity, we model documents as indivisible, rather than as compound, and assume that accesses to objects are independent.

- The fraction of all requests that are for the i-th most popular document, or the "popularity" of this document, is denoted by p_i. We assume that documents follow a Zipf-like distribution [5], i.e., that p_i is proportional to $1/i^\alpha$ for some constant α. The important characteristic of a Zipf-like distribution is that it is *heavy-tailed* – a significant fraction of the probability mass is concentrated in the tail, which in this case means that a significant fraction of requests go to the relatively unpopular documents. As α increases, the distribution becomes less heavy-tailed, and a larger fraction of the probability mass is concentrated on the most popular documents.

- The distribution of time between requests made by the client population is exponential with parameter λN, where λ is the average client request rate.

[2] We argue that the storage needed to do this is not intractable, and that capacity misses are therefore not an important aspect of the workload. Storage for the cacheable fraction of all documents currently estimated to be on the Web would require a large contemporary disk array, such as the 9TB EMC Symmetrix 5930 storage system [15].

- The distribution of time between changes to a document is exponential with parameter μ, independent of document size and latency, but not independent of popularity. We use two separate document change distributions, one for popular documents with mean μ_p and another for unpopular documents with mean μ_u. The number of popular documents is n_p. Document change can be used to model either expiration or actual change.

- The probability that a requested document is cacheable is p_c.

- The average document size is $E(S)$. Document size is independent of document popularity, latency, and rate of change.

- The last-byte latency to the server that houses that document has average value $E(L)$. Last-byte latency is independent of document popularity and document rate of change.

We justify the independence assumptions we make by the fact that all the correlation coefficients (between each pair of document size, document popularity, first-byte latency, and document cacheability) computed from the UW trace are close to 0. We have also found that cacheable objects are more likely to have low latencies, making the model conservative with respect to reality. The data from the UW trace strongly suggests that the rate-of-change distribution is heavy-tailed. However, to make our model tractable to solve analytically, we assume that the rates of change for popular and unpopular documents are distributed exponentially. Since the steady-state performance of a proxy cache improves as document inter-modification times increase, this approximation underestimates cache performance and again makes it conservative.

With these assumptions, we can compute a number of performance characteristics. In steady state, it is easily shown that for a single proxy cache serving a population of size N:

- The steady state hit rate is

$$H_N = p_c C_N,$$

where C_N is the probability that a request is a hit given that it is cacheable. The steady-state cacheable hit rate to cacheable documents C_N is

$$C_N = \sum_{1 \le i \le n} p_i \frac{\lambda N p_i}{\lambda N p_i + \mu}.$$

Taking p_i proportional to $1/i^\alpha$, a very close approximation to this sum is given by

$$\int_1^n \frac{1}{Cx^\alpha} \left(\frac{1}{1 + \frac{\mu x^\alpha C}{\lambda N}} \right) dx,$$

where

$$C = \int_{1 \le x \le n} \frac{1}{x^\alpha} dx.$$

The first integral can be evaluated exactly for $\alpha = 1$ and numerically for other values of α.

- The expected last-byte latency to serve a request is given by

$$lat_{req} = (1 - H_N)E(L) + H_N * lat_{hit},$$

where lat_{hit} is the latency for a cache hit.

- The average bandwidth savings per request due to proxy caching, measured in kilobytes *not* transferred, is denoted B_N and is given by

$$B_N = H_N E(S).$$

- The expected amount of storage required in a proxy cache for a population of this size is $nH_N E(S)$. This is actually optimistic, as it assumes that only objects that are cacheable and have not expired are cached.

The key differences between our model and that presented by Breslau et al. are that: (1) we consider the *steady-state* behavior of caching systems rather than caching behavior based on a finite request sequence, and (2) we incorporate document rate of change into the model rather than assuming that documents are static. Our goal in building the model also differs from the goals pursued by Breslau et al. They used their model to study proxy cache replacement algorithms; we use our model to understand the performance of large-scale, cooperative-caching schemes in terms of hit rate, latency, bandwidth savings, and storage consumed.

We note that a number of the assumptions made here do not match some empirical measurements, and therefore the results of our model cannot be compared directly to the trace results shown in Section 3. For example: (1) our empirical results are based on a one-week trace, while our model examines steady-state behavior, (2) the number of documents seen by the trace is significantly smaller than the number in the Web as a whole, and (3) the trace-based simulations did not expire documents from the cache and we cannot precisely model the rate-of-change distribution seen in the trace.

The goal of this section is *not* to exactly model empirical results. Rather, it is to examine at a high level the impact of changes to, or the sensitivity of, various workload parameters in light of future trends.

4.3 Model parameters

We parameterize the model using values computed from the UW trace. These values are summarized in Table 2.

We estimate the number of objects in the Web, n, using results from a study by Lawrence et al. [25]. Based upon February 1999 data, they estimate that the Web has 800 million compound Web documents. We used an estimate of 3.2 billion objects in the Web, since each compound document in the UW trace contained an average of four objects.

We assume that the time to serve documents from a proxy cache lat_{hit} is 10ms. Although this may be an optimistic value, particularly when caches are under heavy load and requests experience queueing delays, increasing lat_{hit}

Parameter	Value	Parameter	Value
α	0.8	p_c	0.6
n	3.2 billion	$E(S)$	7.7 KB
n_p	10.4 million	$E(L)$	1.9 seconds
λ	590 reqs/day	lat_{hit}	10 ms
Slow		*Fast*	
μ_p	1/14 days	μ_p	1/5 minutes
μ_u	1/186 days	μ_u	1/85 days
Mid Slow		*Mid Fast*	
μ_p	1/1 day	μ_p	1/1 hour
μ_u	1/186 days	μ_u	1/85 days

Table 2. Default model parameters from the UW trace.

Scenario	Popular		Unpopular	
	Mean	Median	Mean	Median
Normal	14	< 1	186	85
Always Change	3	< 1	129	23
Never Change	27	< 1	763	180
Cacheable	5	< 1	168	65
Uncacheable	< 1	< 1	22	< 1

Table 3. Document rate of change (in days between changes) for three different policies (Normal, Always Change, and Never Change), and then broken down by Cacheable and Uncacheable documents using the Always Change policy.

merely offsets the latency results by a similar amount until it reaches a second or more.

Table 3 summarizes the rate-of-change values observed in the UW trace for three policies. We found that document rate of change is correlated with document popularity, so the table contains results separated into popular and unpopular documents. This finding is consistent with previous rate of change studies. For example, Douglis et al. also found that the popular pages changed more often than the less popular pages [13]. We define popular documents as the most frequently requested documents that account for 40% of all requests. Due to the Zipf popularity distribution, the popular documents comprise only 0.3% of all documents. We account for this in the model by using two separate rate-of-change distributions, one for popular documents (mean μ_p) and another for unpopular documents (mean μ_u). In our model results, we use four parameterizations of these distributions. The "slow" parameterization uses the *mean* rates of change for popular and unpopular documents computed from the UW trace as the means μ_p and μ_u of the rate-of-change distributions in the model. The "fast" parameterization uses the *median* rates of change as computed from the UW trace as the means for the rate-of-change distributions in the model. And "mid slow" and "mid fast" represent intermediary values for the rate-of-change parameters.

Since our UW trace does not record the data transferred during a connection, we must rely on the "Last-Modified" HTTP header to detect document changes. However, this header is not always present in Web server responses. Other studies on rate of change, where the full document content was available, have determined that relying solely on HTTP header information has some pitfalls, the most common being when the headers signal a change when none has occurred [13, 36]. Wills et al. [36] also quantify how often documents change when the "Last-Modified" field is missing. The top three lines of Table 3 show results that differ only in how we treat requests to documents with incomplete header information (36% of the requests). For the "normal" results, we simply ignore those documents. For the "always change" results, we calculate an upper bound by assuming that those documents change between each access. For the "never change" results, we calculate a lower bound by assuming that no document missing this header field changes.

We also investigated whether the rate of change for cacheable documents was different than that of uncacheable documents. The "cacheable" and "uncacheable" lines in Table 3, generated using the "always change" policy, show that uncacheable documents change much more rapidly than cacheable documents. It is important to note that since our model is restricted to cacheable documents, the rate of change results for cacheable documents are more relevant as input parameters to the model.

4.4 Performance of large scale proxy caching

We begin by examining basic performance results from the model using parameters extracted from our trace data. We then consider the effects of possible future changes in the fundamental parameters of the Web. In particular, we examine the impact on performance of (a) the rate of change of Web documents μ, (b) the client request rate λ and population size N, (c) the Zipf parameter α of the popularity distribution, and (d) the rate of growth of the Web, measured in the number of accessible documents n.

4.4.1 Hit rate, latency, and bandwidth

Figure 9 shows the steady-state hit rate for cacheable documents C_N as a function of the population size N, graphed on a log scale. The figure shows four curves corresponding to four possible values for the rate of change parameters presented in Table 2. A key question to address is: given the client request rate, document rate of change, and document popularity distributions, how large a client population is needed to achieve a cache hit rate approaching p_c, the fraction of cacheable Web documents.

All these curves can be viewed as consisting of three regions: (1) an initial region in which hit rates grow slowly, (2) a large middle region in which hit rates grow linearly (as population grows exponentially), and (3) a final region in which hit rates grow slowly again, ultimately converging to 100%. In the initial region, the request rate is too low to dominate the rate of change for unpopular documents. As a result, the hit rate for unpopular documents remains close to zero and almost all of the hit rate improvement is accounted for by hits on popular documents. The transition to the middle region marks the beginning of a significant increase in the hit rate to unpopular documents. The heavy-tailed nature of the popularity distribution implies that an exponential increase in request rate is needed to obtain a linear increase in the fraction of requests to unpopular documents that are hits,

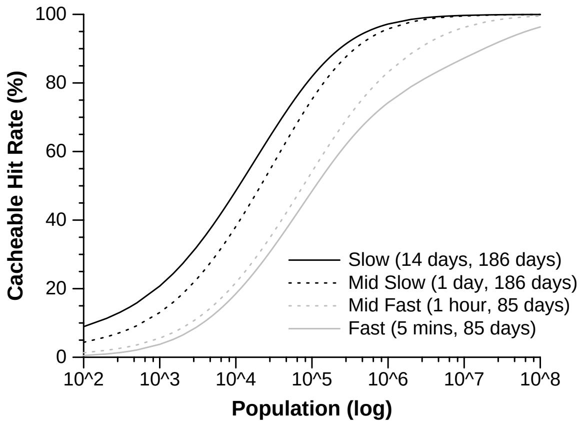

Figure 9. Cacheable request hit rate as a function of client population (log scale).

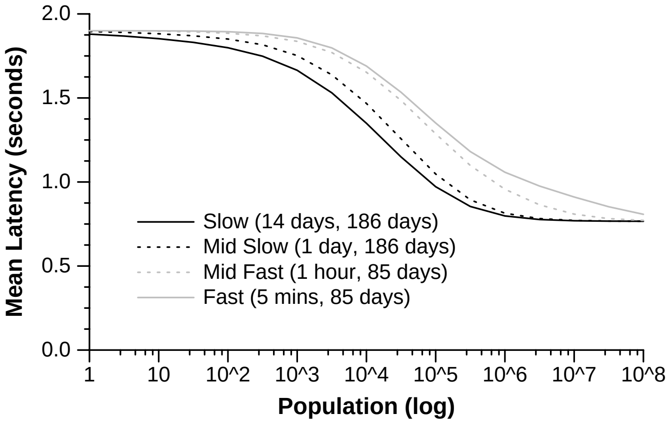

Figure 10. Mean request latency as a function of client population.

i.e., requests to documents that have been requested more recently than they have changed. The transition to the final region occurs when the hit rate for unpopular documents approaches 100%. Behavior in this region is once again accounted for by improvements in the hit rate for popular documents. Here, the request rates are high enough to dominate those inter-document modification times that are much smaller than the mean of the exponential distribution.

We see from Figure 9 that proxy cache hit rate is very sensitive to document rate-of-change parameters. The client population required to achieve 90% of the cacheable hit rate p_c is only 250,000 for the slowest parameters but nearly 20 million for the fastest parameters.

Finally, we would like to understand how hit rate translates into actual performance improvement. Hit rate fundamentally determines the improvement in object access latency and the reduction in network bandwidth consumed by transmitting Web objects. The effect on object access latency is shown in Figure 10. This figure graphs mean latency as a function of N, assuming a single cooperative cache for the entire population that serves cache hits with an average latency of 10ms. The curves asymptote at $(1 - p_c)E(L)$, the mean latency of uncacheable documents; recall that p_c=0.6 and $E(L)$=1.9 seconds, so the curves asymptote at 0.76 seconds. For the slower rates of change, most of the benefit is achieved at medium-sized populations: 95% of the maximum benefit is achieved at a population of 500,000. For the fast rates of change, 68% of the maximum benefit is achieved at a population of 500,000, and it decreases thereafter. Since the impact on latency and bandwidth is directly a function of hit rate, the curves for the effect on bandwidth are identical to those for latency. We therefore omit the graph of bandwidth as a function of population for brevity.

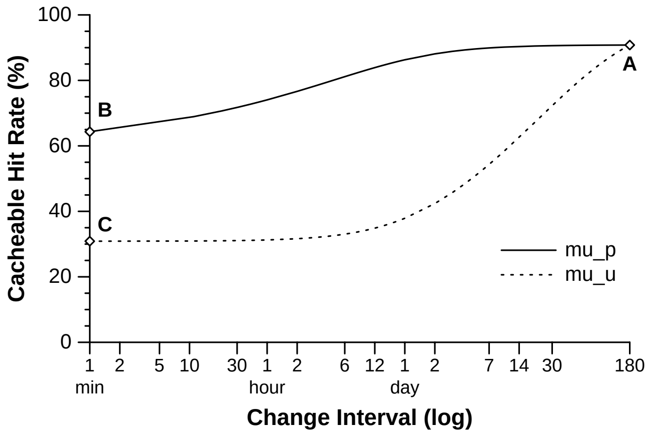

Figure 11. Sensitivity of hit rate to the rate of change of popular and unpopular documents, μ_p and μ_u. The hit rates were calculated for a population of 250,000.

4.4.2 *Document rate of change*

Two key issues regarding document rate of change concern: (1) how sensitive the hit rate is to document change rate, and (2) which parameter has the greatest impact on hit rate – the rate of change of popular documents μ_p or unpopular documents μ_u. Figure 11 shows the sensitivity of the proxy cache hit rate to the rate of change of popular and unpopular documents for a population of 500,000 clients. The x-axis is the mean interval between changes to a document (the inverse of μ) on a log scale, and the y-axis is the hit rate of cacheable documents. The top curve shows the effect on hit rate of varying the mean rate of change of popular documents μ_p from a very slow rate, viz., 1 change every 180 days (point A), to a very fast rate, viz., 1 change every minute (point B). For this curve, the rate of change of unpopular documents μ_u is held constant at the slow rate of 1 change every 180 days to minimize its impact. Similarly, the bottom curve shows the effect on hit rate of varying the mean rate of change of unpopular documents μ_u between the same extremes of slow (point A) and fast (point C) rates of change. For this curve, the rate of change of popular documents μ_p is held constant at the slow change rate.

From Figure 11 we see that the proxy cache hit rate is very sensitive to the change rates of popular and unpopular documents. For popular documents, hit rate varies moderately when documents change faster than once a day. When popular documents change slower than once a day, there is little impact on hit rate. In contrast, hit rate is sensitive to the rate of change of unpopular documents on an entirely different time scale. For unpopular documents, hit rate varies con-

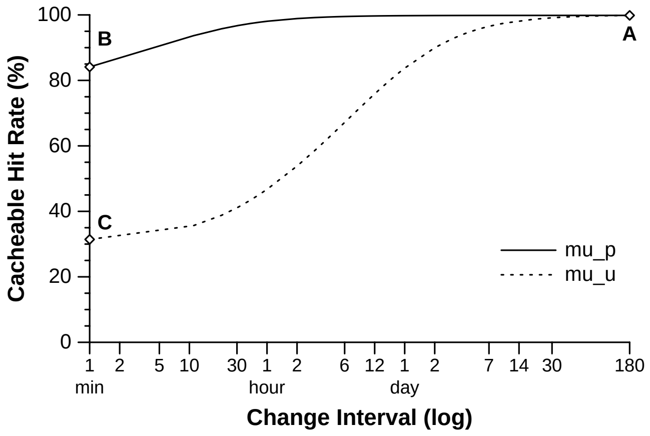

Figure 12. Sensitivity of hit rate to the rate of change of popular and unpopular documents, μ_p and μ_u. The hit rates were calculated for a population of 20 million.

siderably when documents change *slower* than once a day. Once unpopular documents change at least once a day, they are already changing faster than they are being requested. At this point, hit rate reaches a minimum and does not decrease, even at higher rates of change. Thus, for popular documents the issue is whether they change on the scale of minutes to hours; for unpopular documents it is whether they change on the scale of days to weeks to months.

We also see from Figure 11 that the rate of change of unpopular documents μ_u has a more significant impact on proxy cache hit rate than the rate of change of popular documents μ_p. This behavior is a result of the dynamics between document popularity and rates of change. Requests are heavily skewed to popular documents; therefore, even with high change rates, the request rate to popular documents dominates. While each document change makes the next request to that document miss in the cache, the popularity of the document is such that it will have many requests to the document before it changes again, and these requests hit in the cache. However, the request rate to unpopular documents is so low that the change rate dominates. Even if unpopular documents change at a moderate rate, requests to those documents always find out-of-date versions in the cache.

Figure 12 shows results similar to Figure 11, except for a much larger population of 20 million users. This population corresponds to a hit rate of 90% for the fast rates of change. Comparing the figures, we see a decrease in the time scales at which hit rate is sensitive to rates of change. For popular documents the time scale is now an hour or less, and for unpopular documents most of the variation in hit rate is between rates of change of 10 minutes to a month. At these very large population sizes, document request rates are significantly higher than with the smaller population in Figure 11. For both popular and unpopular documents, these high request rates dominate even higher rates of change.

4.4.3 Client request rate

The population needed to achieve a given hit rate varies inversely with the client request rate λ. When λ is very low, even large populations cannot dominate the object rate of change and therefore cannot keep the cache filled with up-to-date objects. On the other hand, when λ is extremely large, even small populations can maintain a filled, up-to-date cache. Trends indicate that λ is increasing significantly over time; the per client request rate in the UW trace is eight times that of the 1996 DEC trace [24]. Unfortunately, there is minimal current or historical data on document rate of change. The rates of change in the UW trace seem well within a factor of two of those presented in [13]. Based upon just these two points of reference, it does appear that λ is growing much faster than μ_u. As a result, the populations at which large-scale caching systems experience diminishing returns will decrease over time.

4.4.4 Document popularity and size of the Web

We also examined the sensitivity of the model to variations in the Zipf parameter α and the number of documents in the Web n. We do not have sufficient space to describe the results in detail, and so we briefly summarize them here. Increasing α skews the distribution even further towards popular documents. This greater skew towards popular documents significantly increases hit rates for slower rates of change, but only slightly increases hit rates for faster rates of change. Increasing the number of documents n simply shifts the curves for slow and fast rates of change to larger populations; it does not significantly change the shapes of the curves. This population shift is roughly in proportion to the increase in n: for example, for n=3.2 billion, the slow curve reaches a 90% hit rate at a population of 250,000; for n=32 billion, the slow curve reaches a 90% hit rate at a population of 25 million; and for n=320 billion, the slow curve reaches a 90% hit rate at a population of 250 million.

4.5 Summary

In this section, we developed an analytic model and used it to examine the steady-state performance of cooperative-caching schemes. Our model extends the results of our trace to a wider range of parameter values, including document popularity and document rate of change. We show again that relatively small populations achieve most of the performance benefits of cooperative caching.

5 Comparing cooperative caching schemes

The previous section explored the performance of large Web proxy-caching systems. This section examines the question of how these systems are designed and organized. Proxy caching starts at the level of individual organizations, small and large. To achieve the performance of caching systems with large populations, some form of cooperative caching among these organizational proxies will have to be used. In this section, we extend the calculations from the model to understand the differences in performance between various cooperative-caching systems as a function of scale. We compare three basic schemes: a hierarchical caching system inspired by Squid [32]; a flat hash-based caching system inspired by [21, 35]; and a directory-based scheme inspired by Summary Cache [16]. We evaluate these schemes at the

	Hierarchical	**Hash-based**	**Directory-based**
Request Arrival Rate	$\lambda N_i(1 - H_{N_{i+1})}$ to level i $(H_{N_{k+1}} = 0)$	$\lambda N/m$	$\lambda N/m$ $+\frac{\lambda N}{m}(1 - H_{\frac{N}{m}})H_{N(1-\frac{1}{m})}$ (2nd term – requests from other proxies)
Average Request Latency	$(1 - H_{N_1})E(L)$ $+\sum_{1 \leq i \leq k} L_i(H_{N_i} - H_{N_{i+1}})$	$(1 - H_N)E(L) + H_N L_c.$	$(1 - H_N)E(L)$ $+(1 - H_{\frac{N}{m}})H_{N(1-\frac{1}{m})}2L_c$ $+H_{\frac{N}{m}}L_l$
Storage Per Proxy	$\sum_{1 \leq i \leq k} nd^{i-1}H_{N_i}E(S)/m$	$nH_N E(S)/m$	$nH_{\frac{N}{m}}E(S)$ (lower bound)

Table 4. Cooperative caching performance parameters.

scale of a City, a State, and a large region (the West Coast of the U.S.). Our results will show why there is little motivation to scale to a region larger than a medium-sized city.

The *hierarchical caching system* assumes a hierarchy of k levels of caches, with a fanout of d at each level, where the bottom-level caches serve as proxy caches for populations of size N_k. A client's request is forwarded up the hierarchy until a cache hit occurs; if none occurs, the request is forwarded to the server. A copy of the requested object is then stored in all caches along the request path. In what follows, we will assume that the top-level cache serves an overall population of size $N_1 = N$, each second-level cache serves a disjoint subpopulation of size N_2, and on down to a set of k-th-level caches, each serving subpopulations of size N_k. (Concisely, $N_i = N/d^{i-1}$.) For simplicity, we assume fixed latencies between the caches in the hierarchy, where L_i is the latency between level $i + 1$ and level i caches.

The *hash-based caching system* assumes a total of m caches cumulatively serving a population of size N. (We will assume $m = d^k$ from the hierarchical scheme.) We assume a fixed average latency of L_c to transmit data from a random cache to a client in the population, and a hash function that randomly maps URLs to one of the m caches uniformly. Upon a request by a client, the client hashes the URL and forwards the request to the corresponding cache C. If the cache stores the document, it is forwarded to the client. Otherwise, the request is forwarded to the server, and a copy is returned to the client and to the cache C. The advantages of such a scheme are: (1) that load is balanced across the proxy caches, and (2) only one copy of each document is stored in the entire cooperative caching system. Such schemes have been proposed primarily for use in a local-area setting, since for large populations L_c may be sizeable.

Finally, in the *directory-based system*, we assume a total of m caches cumulatively serving a population of size N. In this system, the population is partitioned into subpopulations of size N/m, and each subpopulation is served by a single proxy cache. Each proxy cache maintains a directory that summarizes the set of documents stored at each of the other proxy caches in the system. When a client issues a request for a document, it is forwarded to its proxy. If the proxy has the object, it returns it directly to the client. Otherwise, the proxy cache checks its directory to see if another proxy cache in the system stores a copy of the document. If so, the request is sent to a random cache storing a copy, which then returns a copy of the document to the proxy cache and to the requesting client. We assume a latency of L_l to transmit data from a proxy cache to a client it serves, and a latency of L_c between proxy caches in the system.

To maintain the directories, each proxy cache periodically (every t time units) sends out an update about the contents of its cache. In particular, it multicasts the set of changes to its document set since the last request. We ignore the overhead of these messages in our subsequent analysis, as well as the extra misses caused by directory entries that become stale between updates.

These descriptions of cooperative caching systems are stated in general terms that emphasize the structure and operation of the systems, glossing over potential implementation details. For example, the description of the hierarchical system is in terms of multiple levels of caches. In practice, these caches may be special caches maintained at ISPs, or existing organizational caches that serve the role of first, second, and third level caches for different portions of the Web name space, as with [33]. The issue of whether it is better to use separate dedicated caches or to overload existing individual caches with multiple responsibilities is a detailed design issue that is beyond the scope of this paper.

Table 4 summarizes the performance of the three cooperative caching schemes, based on the model. Recall from Section 4 that $E(L)$ is the average latency to a server and $E(S)$ is the average document size. We compare the performance of these schemes at three different scales:

1. A medium-sized City (N=0.5 million users)
2. A small State (N=5 million users)
3. The west coast of the U.S. (N=50 million users)

From the UW trace, the average client issues just under 600 requests. Based on this, we assume that there are 50,000 clients behind each lowest-level proxy cache, which results in an average request rate of about 350 requests per second. This request rate is well within the load a single host can handle as a proxy cache (e.g., [12] reports 500 requests per second, and various single host proxies from the Web Caching Bake-off report throughputs ranging from 96–690 requests per second [31]). Based on the populations at different scales, this results in a total of $m = 10$ organizational proxy caches for the City, $m = 100$ caches for the State and $m = 1000$ caches for the west coast. For the hierarchical scheme, we assume that $d = 10$, giving us a two-level hierarchy for the city (a single top-level proxy cache on top of

	Hierarchical	Hashed	Directory
Arrival Rate	r_1 = 150M/day r_2 = 30M/day	30M/day	37M/day
Latency	0.86 secs	0.88 secs	0.89 secs
Storage	11 TB	1.5 TB	9.5 TB

Table 5. City cooperative caching performance.

	Hierarchical	Hashed	Directory
Arrival Rate	r_1 = 1.37B/day r_2 = 147M/day r_3 = 29.5M/day	29.5M/day	37.7M/day
Latency	0.79 secs	0.83 secs	0.85 secs
Storage	11 TB	150 G B	9.5 TB

Table 6. State cooperative caching performance.

the 10 organizational caches), a three-level hierarchy for the state, and a four-level hierarchy for the west coast.

We use the values in Table 2 from the UW trace to parameterize the model. For rates of change, we used the "mid slow" values, where μ_p is one change per day and μ_u is one change per 186 days. These rates of change most closely matched the performance observed in the UW and Microsoft traces. We further assume that the average last-byte latency to transfer a document: (1) from an organizational proxy cache to a client is 10ms, (2) between two random caches in the city is 50ms, (3) between two random caches in the state is 100ms, (4) and between two random caches on the west coast is 500ms. We derived these numbers by multiplying a basic latency by the number of round trips required to download an average document. For the basic latency, we used ping latencies from the University of Washington to popular Web sites whose distances correspond to the three levels of the cache hierarchy. Since the average document size in our trace is 7.7KB, we estimate that five round trips between the sender and the receiver of the document are required to complete the transfer (due to TCP/IP protocol overhead).

Because these latencies are based upon a simple model of the network (e.g., persistent connections might reduce the number of round trips if the connection has been ramped up to a high congestion window) and pings from a single network source, we also evaluated the sensitivity of our model results to these parameters. We did this evaluation indirectly by varying the average latency for downloading documents from servers, thereby changing the ratio of cache latency to server latency. In addition to the trace value of $E(L)$=1.9 seconds, we also evaluated the schemes using average documents latencies ranging from 250ms to 10 seconds. In each case, their relative performance was qualitatively similar when $E(L)$=1.9 seconds.

Tables 5, 6 and 7 present model results for the three cooperative-caching schemes using our parameterizations. From these tables we can draw a number of conclusions. First, we see that the bulk of the achievable benefit, in terms of latency savings, is already achieved at the scale of city-level cooperative caching. Indeed, the minimum possible average latency we could hope for is $(1 - p_c)E(L)$, which for our parameters is 0.76 seconds. All three schemes already achieve a value close to this in the city. Second, broadening the region to increase population also increases inter-proxy latencies. As a result, a flat cooperative-caching scheme is no longer effective. For the west coast, the average latency between proxy caches is sufficiently large that the performance of the flat hash-based and directory-based schemes is worse, in terms of document latency, than their performance at the level of the state, despite the ten-fold increase in population. Obviously, this problem could be solved by designing hierarchical variants of these two schemes.

	Hierarchical	Hashed	Directory
Arrival Rate	r_1 = 13B/day r_2 = 1.37B/day r_3 = 147M/day r_4 = 29.5M/day	29.5M/day	37.9M/day
Latency	0.78 secs	1.1 secs	1.13 secs
Storage	11 TB	15 GB	9.5 TB

Table 7. West Coast cooperative caching performance.

In terms of request rate, we see that for all schemes the lowest-level proxy caches have similar request rates (though the directory scheme has a slightly higher value), and are dominated by the requests from clients served directly by that proxy. However, even at the scale of city-wide cooperative caching, the top-level cache in the hierarchical scheme is a bottleneck. Since request rate will be directly correlated with queueing, the average latency that will be observed in the hierarchical scheme will be significantly higher than shown here, particularly as we scale up to the state or west coast level. Therefore, if scaling up to these levels is desirable (which is itself a questionable proposition at best), the load at the higher levels of the hierarchy must be distributed across multiple proxy caches. There are a number of fairly obvious and natural ways to do this. Finally, we see that the hash-based scheme has the advantages that each document is stored only in one proxy cache and the load is balanced across the caches.

In summary, all three schemes perform well in the region where cooperative caching is advantageous (e.g., at the level of a medium-sized city). Since documents are stored only in one cache, a hash-based scheme achieves the best storage efficiency. In a broader area (e.g., the size of the west coast), the increased latency of inter-proxy communication eclipses the very limited benefits of increased population.

6 Conclusions

This paper studied cooperative proxy caching in local- and wide-area environments. We used a combination of trace-based analysis and analytic modelling to evaluate cooperative caching, and proxy caching in general, at a wide range of population sizes, document characteristics, and access patterns. At a high level, our results show that:

1. In the absence of significant changes in client behavior, there is little point in continuing to expend effort on the design and evaluation of highly scalable, cooperative-caching schemes. The scale at which cooperative caching makes sense (viz., up to the level of

a medium-sized city) is sufficiently small that reasonable schemes will achieve most of the benefit.

2. The largest benefit for cooperative caching is achieved for relatively small populations. This is demonstrated by our analysis of cooperation among small organizations within the university environment. Our simultaneous traces of UW and Microsoft confirmed the marginal benefit of cooperative caching among organizations with populations of 20K clients or more. Scaling beyond such populations provides only minor improvement and therefore makes sense only in very high-bandwidth, low-latency environments.

3. Performance at the population level at which cooperative caching works effectively is basically limited by document *cacheability*. Therefore, increasing cacheability of documents is the main challenge for research aimed at improving Web cache behavior.

4. Cluster-based analysis of client access patterns indicates that cooperative-caching organizations based on mutual interest offer no obvious advantages over randomly assigned or organization-based groupings.

Fundamentally, the usefulness of cooperative Web proxy caching depends upon the scale at which it is being applied. From our trace data of users at the University of Washington and Microsoft Corporation, cooperative Web proxy caching is an effective architecture for small individual caches that together comprise user populations in the tens of thousands. At such small scales, any reasonable cooperative caching scheme will serve. But cooperative caching is not required for user populations of this size. If it is administratively and politically feasible, a single proxy cache can provide the same benefits with fewer resources and less overhead.

Whether or not they use cooperative caching locally, large organizations should use proxy caching for their user populations. A key issue is whether these large organizational caches benefit from cooperating. Experiments with our steady-state model indicate that cooperation among the organizational caches within a medium to large city will still provide benefit, although an incremental benefit, over cooperative caching at small scales. Assuming that bandwidth within a city is plentiful and latencies are small, the overhead of cooperative caching would be low and therefore worth the secondary benefits that such caching provides. In principle, the organizational caches within a city can use a hash-based scheme to maximize storage efficiency. In practice, however, given the cheap cost of disks, using a hash-based scheme to spread load is more important than storage efficiency. Extrapolating to yet larger scales, such as the state level and even the west coast of the U.S., our model results indicate that cooperative caching among cities would provide very limited additional benefit, particularly given the increased latencies among caches.

Finally, we note that our results on cooperative caching are based upon Web workload behavior currently observed. Fundamental shifts in Web workloads might change these results. For example, the workloads we have examined consist primarily of static documents. But we have also observed a growing presence in Web workloads of streaming multimedia traffic [37], and streaming multimedia objects have different characteristics than static objects. Their average size is orders of magnitude larger, so cooperative caching for storage efficiency becomes more appealing. Furthermore, last-byte latency is not a critical performance metric for streaming data. Instead, reducing jitter and making more effective use of the network become more important. Lastly, given the sizes of streaming objects, and the relatively long period of time over which they are transferred over the network, transport optimizations like multicast might prove more effective.

Acknowledgements

We would particularly like to thank those people who helped make our traces a reality. Paul Leach of Microsoft patiently and generously worked with us to produce the Microsoft data included in our study. At the University of Washington, Steve Corbato, Art Dong, Corey Satten, and the other members of the Computing and Communications organization at UW supported our effort. Finally, the SOSP referees provided thorough comments and suggestions. We are also very grateful to Jeff Mogul, our shepherd, for his guidance in improving the focus and clarity of our paper. This research was supported in part by DARPA Grant F30602-97-2-0226, National Science Foundation grant EIA-9870740, US-Israel Binational Science Foundation grant 96-00247, and Intel and Microsoft Graduate Research Fellowships.

References

[1] J. Almeida, V. Almeida, and D. Yates. Measuring the behavior of a World Wide Web server. Technical Report 96-025, Boston University, Oct. 1996.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. deOliveira. Characterizing reference locality in the WWW. Technical Report 96-011, Boston University, June 1996.

[3] T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Y. Wang. Serverless network file systems. *ACM Trans. on Computer Systems*, 14(1):41–79, February 1996.

[4] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *Proc. of the ACM SIGMETRICS '96 Conf.*, pages 126–137, May 1996.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proc. of IEEE INFOCOM '99*, pages 126–134, March 1999.

[6] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich. Web proxy caching: The devil is in the details. In *Workshop on Internet Server Performance*, pages 111–118, June 1998.

[7] P. Cao. Characterization of Web proxy traffic and Wisconsin proxy benchmark 2.0. http://www.cs.wisc.edu/∼cao/w3c-webchar-position, Nov. 1998.

[8] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proc. of the 1st USENIX Symp. on Internet Technologies and Systems*, pages 193–206, Dec. 1997.

[9] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A hierarchical Internet object cache. In *Proc. of the 1996 USENIX Technical Conf.*, pages 153–163, January 1996.

[10] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proc. of the ACM SIGMETRICS '96 Conf.*, pages 160–169, May 1996.

[11] C. R. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW client-based traces. Technical Report BU-CS-95-010, Boston University, July 1995.

[12] P. Danzig. NetCache architecture and deployment. In *Proc. of the 3rd Int. WWW Caching Workshop*, http://wwwcache.ja.net/events/workshop/01/NetCache-3_2.pdf, June 1998.

[13] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: a live study of the World Wide Web. In *Proc. of the 1st USENIX Symp. on Internet Technologies and Systems*, pages 147–158, Dec. 1997.

[14] B. Duska, D. Marwood, and M. J. Feeley. The measured access characteristics of World Wide Web client proxy caches. In *Proc. of the 1st USENIX Symp. on Internet Technologies and Systems*, pages 23–36, Dec. 1997.

[15] EMC Corporation, http://www.emc.com/products/ enterprise_storage_systems/systems.htm. *Symmetrix 3000 and 5000 Enterprise Storage Systems Product Description Guide*, 1999.

[16] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *Proc. of ACM SIGCOMM '98*, August 1998.

[17] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing global memory management in a workstation cluster. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, pages 201–212, Dec. 1995.

[18] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of web proxy caching in heterogeneous bandwidth environments. In *Proc. of IEEE INFOCOM '99*, March 1999.

[19] S. Glassman. A caching relay for the World Wide Web. In *Proc. First Int. World Wide Web Conf.*, pages 60–76, May 1994.

[20] S. D. Gribble and E. A. Brewer. System design issues for Internet middleware services: Deductions from a large client trace. In *Proc. of the 1st USENIX Symp. on Internet Technologies and Systems*, pages 207–218, Dec. 1997.

[21] D. Karger, T. Leighton, D. Lewin, and A. Sherman. Web caching with consistent hashing. In *Proc. of the 8th Int. World Wide Web Conf.*, May 1999.

[22] P. Krishnan and B. Sugla. Utility of co-operating Web proxy caches. In *Proc. Seventh Int. World Wide Web Conf.*, April 1998.

[23] T. M. Kroeger, D. D. E. Long, and J. C. Mogul. Exploring the bounds of Web latency reduction from caching and prefetching. In *Proc. of the 1st USENIX Symp. on Internet Technologies and Systems*, pages 13–22, Dec. 1997.

[24] T. M. Kroeger, J. C. Mogul, and C. Maltzahn. Digital's Web proxy traces. ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html, August 1996.

[25] S. R. Lawrence and C. L. Giles. Accessibility of information on the Web. *Nature*, 400(6740):107–109, July 1999.

[26] B. A. Mah. An empirical model of HTTP network traffic. In *Proc. of IEEE INFOCOM '97*, pages 592–600, April 1997.

[27] J.-M. Menaud, V. Issarny, and M. Banatre. A new protocol for efficient transversal Web caching. In *Proc. of the 12th Int. Symp. on Distributed Computing*, pages 288–302, September 1998.

[28] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive Web Caching: Towards a New Global Caching Architecture. *Computer Networks and ISDN Systems*, 30(22–23):2169–2177, Nov. 1998.

[29] J. C. Mogul. Network behavior of a busy web server and its clients. Technical Report 95/5, DEC Western Research Laboratory, Oct. 1995.

[30] M. Rabinovich, J. Chase, and S. Gadde. Not all hits are created equal: Cooperative proxy caching over a wide area network. In *Proc. of the 3rd Int. WWW Caching Workshop*, June 1998.

[31] A. Rousskov, D. Wessels, and G. Chisholm. The first ircache web cache bake-off. Technical report, National Laboratory for Applied Network Research, April 1999.

[32] Squid internet object cache, http://squid.nlanr.net.

[33] R. Tewari, M. Dahlin, H. Vin, and J. Kay. Design considerations for distributed caching on the Internet. In *The 19th IEEE Int. Conf. on Distributed Computing Systems*, May 1999.

[34] J. Touch. The LSAM proxy cache - a multicast distributed virtual cache. In *Proc. of the 3rd Int. WWW Caching Workshop*, June 1998.

[35] V. Valloppillil and K. W. Ross. Cache array routing protocol v1.0. ftp://ftp.isi.edu/internet-drafts/draft-vinod-carp-v1-03.txt, Feb. 1998.

[36] C. E. Wills and M. Mikhailov. Towards a better understanding of Web resources and server responses for improved caching. In *Proc. of the Eighth Int. World Wide Web Conf.*, pages 153–165, May 1999.

[37] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy. Organization-based analysis of Web-object sharing and caching. In *Proc. of the 2nd USENIX Symp. on Internet Technologies and Systems*, Oct. 1999.