IEEE Data Engineering Bulletin, 19(3):3-11, September 1996.

Server-initiated Document Dissemination for the WWW

Azer Bestavros (best@cs.bu.edu)
Carlos Cunha (carro@cs.bu.edu)

Computer Science Department
Boston University, MA 02215

Abstract

In this paper we overview the merits of a data dissemination mechanism that allows information to propagate from its producers to servers that are closer to its consumers. This dissemination reduces network traffic and balances load amongst servers by exploiting geographic and temporal locality of reference properties exhibited in client access patterns. The level of dissemination depends on the relative popularity of documents, and on the expected reduction in traffic that results from such dissemination. We present results of log analysis and trace-driven simulations that quantify the performance gains achievable through the use of such a protocol.

1 Introduction

The effectiveness of client-based caching for very large distributed information systems (like the WWW) is limited. This was established in a comprehensive study of client-based caching for the Web, which was conducted by our Oceans group (`http://www.cs.bu.edu/groups/oceans/`). In that study [4], the effectiveness of session caching, machine caching, and proxy caching were established using a unique set of 5,700 client traces (almost 600,000 URL requests), which were obtained by intrumenting Mosaic as detailed in [6]. This study concluded that proxy caching is ultimately limited by the low level of sharing of remote documents amongst clients of the same site. This finding agrees with Glassman's predictions [9] and was further confirmed for general proxy caching by Abrams *et al* [1].

In order to explain the limited effectiveness of WWW client-based caching, we need to consider the locality of reference properties that contribute to an enhanced cache performance [2]. Access patterns in a distributed information system (such as the WWW) exhibit three locality of reference properties: temporal, geographical, and spatial. Temporal locality of reference implies that recently accessed objects are likely to be accessed again in the future. Geographical locality of reference implies that an object accessed by a client is likely to be accessed again in the future by "nearby" clients. This property is similar to the processor locality of reference exhibited in parallel applications [8]. Spatial locality of reference implies that an object "neighboring" a recently accessed object is likely to be accessed in the future.

If client-based caching is done on a *per-session* basis (*i.e.*, the cache is cleared at the start of each client session), then the only locality of reference property that could be exploited is the temporal locality of reference. The results of our client-based caching study [4] suggest that for a single client, the temporal locality of reference is quite limited, especially for remote documents. In particular, we found that even with a cache of infinite size, the average byte hit rate is limited to 36% and could be as low as 6% for some client traces. This poor performance could be attributed to the "surfing" behavior of clients, which implies that recently-examined documents are *rarely* revisted in the future.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

	www.cs.bu.edu		www.stones.edu
	Data Set 1	Data Set 2	Data Set 3
Period	56 days	182 days	110 days
URL requests	172,635	585,739	4,068,432
Bytes transferred	1,447 MB	6,544 MB	112,015 MB
Average daily transfer	26 MB	36 MB	1,018 MB
Files on system	2,018	2,679	N/A
Files accessed (remotely)	974 (656)	1,628 (1,032)	N/A (1,151)
Size of (accessed) file system	50 MB (37 MB)	62 MB (42 MB)	N/A (402 MB)
Unique clients (10+ requests)	8,123	8,474	60,461

Table 1: Summary statistics for log data used in this paper

If client-based caching is done on a *per-site* basis (*i.e.*, a common "proxy" cache is shared among all clients), then one would expect an improved cache performance as a result of the geographical locality of reference property. However, the results of our client-based caching study [4] suggest that for remote documents the amount of sharing between clients is limited. In particular, we found that even with an infinite proxy cache size, the average byte hit rate improves from 36% to 50%. Figure 1, which is reproduced from the data in [4] illustrates this point by plotting the rate at which the proxy cache must be inflated—a measure termed the *Cache Expansion Index* (CEI)—to maintain a constant *byte hit rate*. Figure 1 indicates that the CEI is proportional to the number of clients (N) sharing the proxy cache—it shows that for a large number of clients concurrently using the proxy cache, the CEI is linearly related to N and does not seem to level off.

It must be noted that Figure 1 does not imply that the relationship between CEI and N will continue to be linear for even larger values of N; it only implies that for the levels of concurrency likely to be aggregated at a single site through a proxy cache, the relationship *is* linear. To get a higher level of concurrency, we need to think about an economy of scale far beyond what a proxy cache at (say) an organization can provide.

The above discussion shows that temporal and geographical locality of reference properties for WWW access patterns are not strong enough to result in an effective caching strategy at a single client or site. However, what is yet to be answered is whether temporal and geographical locality of reference properties at a much larger scale (*e.g.*, thousands of clients) could be exploited. In the remainder of this paper, we provide an affirmative answer to this question.

2 The Premise of Dissemination

In order to be able to quantify the available locality of reference properties that could be exploited on the WWW, we collected extensive server traces from the HTTP server of the Computer Science Department at Boston University `http://www.cs.bu.edu` and from the HTTP server of the Rolling Stones web site `http://www.stones.com/`. These traces are summarized in table 1. Unless otherwise stated, our model validation and trace simulations were all driven by this data.

The highly uneven popularity of various Web documents has been documented in [6]. This study confirmed the applicability of Zipf's law [13, discussed in [12]] to Web documents. Zipf's law was originally applied to the relationship between a word's popularity in terms of rank and its frequency of use. It states that if one ranks the popularity of words used in a given text[1] (denoted by ρ) by their frequency of use (denoted by P) then $P \sim 1/\rho$. Note that this distribution is a parameterless hyperbolic distribution. *i.e.,* ρ is raised to exactly -1, so that the n^{th} most popular document is exactly twice as likely to be accessed as the $2n^{\text{th}}$ most popular document. Our data shows that Zipf's law applies quite strongly to *Web documents serviced by Web servers.* This is demonstrated in Figure 2 for all 2,018 documents accessed in the BU trace (data set 1). The figure shows a log-log plot of the total number of references (Y axis) to each document as a function of the document's rank in overall popularity (X axis). The tightness of the fit to a straight line is strong ($R^2 = 0.99$), as is the slope of the line: -0.95 (shown in the figure). Thus the exponent relating popularity to rank for Web documents is very

[1] Zipf's law has subsequently been applied to other examples of popularity in the social sciences.

nearly -1, as predicted by Zipf's law. Out of some 2000+ files available through the WWW server, the most popular 256KB block of documents (that is 0.5% of all available documents) accounted for 69% of all requests. Only 10% of all blocks accounted for 91% of all requests!

The above observation leads to the following question: How much bandwidth could be saved if requests for *popular* documents from outside the LAN are handled at an earlier stage (*e.g.*, using a proxy at the "edge" of the organization)? Figure 3 shows the percentage of the server load (measured in total bytes serviced) that would be saved if various block sizes of decreasing popularity are serviced by some other server.

A closer look at the logs of `http://www.cs.bu.edu`, reveals that there are three distinct classes of documents. Figure 4 shows the ratio of remote-to-local (and local-to-remote) accesses for each one of the 974 documents accessed at least once during the analysis period. Out of these 974 documents: 99 documents had a remote-to-local access ratio larger than 85%—we call these *remotely popular documents*; 510 documents had a remote-to-local access ratio smaller than 15%—we call these *locally popular documents*; 365 documents had a remote-to-local access ratio between 85% and 15%— we call the remaining 365 documents *globally popular documents*.

The classification of documents into globally, remotely, and locally popular (*i.e.*, based on temporal/geographical locality of reference) could be easily done by servers in order to decide which documents to disseminate and where such documents should be dessiminated.

An important question that may affect the performance of our dissemination protocol is related to the rate at which popular documents are updated. Notice that the more frequently these documents are updated, the more frequently the home server is to refresh the replicas at the proxies. In a related study, we monitored the *date of last update* of remotely, locally, and globally popular documents for a 6-month (26 weeks) period that started with the 56-day period of table 1, and continued for another 126 days). We observed that both remotely popular and globally popular documents were updated very infrequently (less than 0.5% update probability per document per day), whereas locally popular documents were updated more frequenlty (about 2% update probability per document per day).[2] This result is important because it suggests that those documents most likely to be disseminated are statistically the ones less likely to change. The significance of these findings was further investigated by Gwertzman and Seltzer in the context of WWW cache consistency [11]. It is important to note that our measurements and observations apply to information systems of *archival* nature (*e.g.* institutional Web pages). For information systems of *temporal* nature (*e.g.* stock market tickers, news feeds, *etc.*) it is likely that the most popular documents are those that change most frequently.

3 Dissemination Model and Experiments

In our model, information is disseminated from *home servers* (producers) to *service proxies* (agents) closer to *clients* (consumers). We assume the existence of a many-to-many mapping between home servers and service proxies. A service proxy along with the set of home servers it represents form a *cluster*. We model the WWW (Internet) as a hierarchy of such clusters.[3]

Our notion of a service proxy is similar to that of a client proxy, except that the service proxy acts on behalf of a cluster of servers rather than a cluster of clients. In practice, we envision service proxies to be information "outlets" that are available throughout the Internet, and whose bandwidth could be (say) *"rented"*. Alternately, service proxies could be public engines, part of a national computer information infrastructure, similar to the NSF backbone. For the remainder of this paper we use *proxy* to refer to a service proxy.

Our model does not limit the number of proxies that could be used to "front-end" a particular server. Each server in the system may belong to a number of clusters, and thus may have a number of proxies acting on its behalf, thus disseminating its documents along multiple routes (or towards various subnetworks). A server is allowed to use (through bidding for example) a subset of these *proxies* to disseminate its data to clients.

In order to evaluate our dissemination protocol, we had to come up with a realistic clustering of the Internet to reflect the dissemination model introduced above and detailed in [3]. This clustering could be based on several criteria. For example, it could be based on geographical information (*e.g.*, distance in miles between clients as suggested in [10]). Alternately, it could be based on institutional/network boundaries (*e.g.*, assign one service proxy per institution or network). Our choice was to base the clustering on the structure of the routes between a

[2] Multiple updates to a document within one day were counted as *one* update.

[3] A detailed treatment of this model, including an optimal resource allocation strategy for proxies to ration their resources amongst competing home servers belonging to the same cluster can be found in [3].

server and its clients. Such a clustering (based on the distance between the server and client measured in *actual route hops*) allows us to quantify the savings achievable through dissemination because it makes it possible to measure the bandwidth saved in *Bytes* $\times$ *Hops* (BHops) units. We define a *BHop* to be equivalent to a single byte transfered over one hop. A reduction in BHops implies either a reduction in the total number of bytes transfered or a reduction in the transfer distance (in hops), or a reduction in both.

Using the `record route` option of TCP/IP, it is possible to build a complete tree originating at the server with clients at the leaves. We built such a tree for all *traceable* clients of `http://www.cs.bu.edu`. By a traceable client, we mean a client for which `traceroute()` succeeded in identifying a route that remained consistent over several traceroute experiments. Almost 90% of all bytes transferred from `http://www.cs.bu.edu` were transferred to traceable clients. For data set 2 of table 1, the constructed tree consisted of 18,000+ nodes.

Figure 5 shows a histogram of the number of clients (leaf nodes) versus the distance in hops of these clients from the `http://www.cs.bu.edu` server. This histogram shows three distinct client populations. The first population is within a distance of 2-3 hops and represents on-campus clients at Boston University. The second population is within a distance of 4-9 hops and represents clients on the New England Academic and Research network (NEARnet). The third is within a distance of more than 9 hops and represent WAN clients. For servers where most of the demand is generated by WAN clients, Figure 5 suggests that popular documents could be disseminated as much as 8-9 hops away from the server. Such a dissemination would result in large savings. Specifically, replicating the most popular 25 MB from `http://www.stones.com` on proxies 8-9 hops closer to clients would yield whoping network bandwidth savings of more than 8 GByte$\times$Hops per day.

We performed simulations of our dissemination strategy based on the structure[4] of the server-to-client routing tree discussed above. Our simulations were driven by the 26-week traces obtained from `http://www.cs.bu.edu` (data set 2 in table 1).

In our simulations, replicas of the most popular files on `http://www.cs.bu.edu` (the root of that tree) were disseminated every week down to proxies (internal nodes of the tree) *closer* to the clients. The location of such proxies depends on the demand during the previous month (4 weeks) from the various parts of the tree and aims to balance the load amongst those proxies.

The general dissemination model presented in [3] assumes that proxies are organized in a hierarchy. Our implementation of this idea was simpler, whereby the hierarchy was flattened by having the home server (the root of the tree) compute using the access patterns of all its clients the ultimate placement for the replicas. In other words, our simulations did not consider multi-level proxying (*i.e.*, the possibility of proxying a proxy) as would be possible under the general dissemination model.

In our study we assumed that any internal node is available as a service proxy. In a real system, this assumption may not be valid since internal nodes are routers, unliquely to be available as service proxies. In practice, we envision that the set of proxies available *"for rent"* by a particular server could be matched up to the optimum place obtained using our protocol.

In our simulations, we require clients to always request service from the home server. If the requested documents are available at proxies closer to the client, then the home server forwards the request to the proxy that is closest to the client. This forwarding mechanism is supported by the HTTP protocol and requires much less overhead than the more general hierarchical name resolution strategy described in [7].

In our simulations, we assume that the cost of propagating updates to proxies could be ignored. To establish the validity of this assumption, consider the ratio of the number of times a *popular* document is changed per unit time to the number of times that document is accessed per unit time. As noted earlier in this paper, popular documents are updated least. This means that the ratio of updates versus access will be very small (we measured this ratio at less than 0.001 for popular documents). Thus, the inaccuracy of our simulations (by not taking into account the cost of propagating updates) is very insignificant.

Figure 6 shows the percentage reduction in bandwidth (measured in *bytes* $\times$ *hops*) that is achievable by disseminating a percentage of the most "popular" data available on the server. The horizontal axis shows the number of proxies to which this data was disseminated. Two curves are shown. The first assumes that the most popular 10% of the data ($\approx$ 5 MB) is to be disseminated, whereas the second assumes that the most popular 4% ($\approx$ 2 MB) of the data is to be disseminated. Obviously, a larger number of proxies results in a deeper dissemination, and thus larger bandwidth savings.

[4] Analysis of `http://www.cs.bu.edu` logs suggests that the shape of the tree (especially internal nodes) and the distribution of load is quite static over time.

One observation from figure 6 is that most of the bandwidth saved through the use of dissemination is due to the dissemination of a very small amount of data. For example, increasing the level of dissemination by 150% (from 2MB per service proxy to 5MB per service proxy) results in a meager 6% additional bandwidth savings. Another observation from figure 6 is that most of the bandwidth saved results from using a small number of service proxies. For example, tripling the number of service proxies from 20 to 60 results in only 21% additional bandwidth savings. Finally, figure 6 shows that the payoff per replica decreases as the number of replicas increases.

The results illustrated in figure 6 are based on a dissemination strategy that is done only once (on day one), based on analysis of the complete logs for all 26 weeks used to drive the simulations. Such an approach would be reasonable if the popularity profile is static. A more practical (and accurate) approach would be to periodically compute the popularity profile based on past access patterns. Figure 7 shows the performance results achievable using such an approach. In particular, it shows the reduction in bandwidth for four experiments, in which the dissemination was performed once a week using the logs of the last n weeks to compute the popularity profile, for n=1, 2, 3, and 5, respectively. For all of these experiments, the 10% most popular data (computed over the n-week period) was disseminated. Figure 7 shows that the longer the history used to compute the popularity profile, the better the performance, especially when the number of proxies is quite large.

An interesting observation from figure 7 regards the non-monotonicity of the performance gains with respect to the level of dissemination. In particular, while the general trend in figure 7 is for the performance to improve as the number of proxies is increased, there are many instances in which an increase in the number of proxies results in a minor performance degradation, especially when the history used to compute the popularity profile is short (*e.g.*, one week). This anomaly can be explained by noting that deploying a larger number of proxies results in a deeper dissemination and that the further a proxy is from the home server, the more lilkely it is for the load generated at that proxy to fluctuate. In other words, dissemination based on a smaller ratio of clients-to-proxies (which is the case when deep dissemination is attempted) is less effective. Recall that, at the limit when documents are propagated all the way to client proxies (*i.e.* closer to the leaves of the tree), dissemination reduces to client caching which has been shown earlier (in Section 1) to be of limited effectiveness due to the weak sharing amongst clients.

Figure 8 documents this observation by showing the load-variation index (on the Y axis) for the top 17 levels of the dissemination tree (on the X axis). This index is computed by normalizing the *change* in the percentage of load at a proxy from one week to the next using the Z-score technique.

In our simulation, the same data is disseminated to all proxies. Better results (and less succeptibility to variation in geographical locality of reference) are attainable if the dissemination strategy takes more advantage of the geographic locality of reference (by disseminating different data to different proxies based on the access patterns of clients served by each proxy).

4 Summary and Future Research

Current service protocols in large-scale distributed information systems are client-based; they do not make use of the knowledge amassed at servers concerning client access patterns. In a series of studies, we have demonstrated that such knowledge could be used effectively to perform dissemination. Using extensive trace simulations we have quantified the potential gains of our protocol. We showed that dissemination is most effective in reducing network traffic (by as much as 40-50%) and in balancing the load amongst servers. These gains are *above and beyond* what is achievable through client-based caching.

The basic premise of our work is that servers are in a unique position to decide *which* documents are worth replicating through dissemination, *how many* replicas should be disseminated, and *where* to place these replicas. This, however, does not imply that clients and their proxies do not (or should not) play a role in improving the performance of information retrieval and in alleviating communication bottlenecks. For example, our work is not a substitute for client-based caching; rather, it is a complement. The primary purpose of client-based caching is to reduce the latency of information retrieval, whereas the primary purpose of information dissemination is to reduce traffic and balance the load among servers. Of course, these purposes are not completely independent—a protocol whose primary purpose is to reduce traffic and balance load is likely to improve the latency of information retrieval. Nevertheless, the most reduction in latency is likely to be the result of a protocol whose primary purpose is just to do that (client-based caching in this case).

Another example of how our work complements other client-based protocols is related to the notion of dynamic server selection, introduced by Crovella and Carter in [5]. The primary purpose of a client-based dynamic server selection protocol is to determine "on the spot" which one of several servers to use in order to retreive a document replicated on all of these servers. This selection is based on dynamic measurements that attempt to locate the server with the least congested route. Such a technique could complement dissemination by allowing servers to communicate with clients the location of all (or some) replicas that are deemed "close" to the client, while leaving to the client-based server selection protocol the final decision as to which one of these replicas to retrieve based on the dynamic conditions of the network.

Our experiments demonstrate the potential savings achievable through dissemination. More experiments are needed to generalize these findings by considering traces from a larger set of servers. Also, more experiments are needed to study the effects of dissemination from competing service proxies. Finally, it should be noted that our study has concentrated primarily on Web document retrieval. As the Web evolves in terms of the type of information and the manner in which this information is presented[5] more elaborate dissemination protocols may have to be developed.

Acknowledgments:
I would like to thank all members of the *Oceans* group `http://www.cs.bu.edu/groups/oceans` for the many discussions and feedback on this work. This work has been partially supported by NSF (grant CCR-9308344).

References

[1] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. In *Proceedings of the Fourth Interntional Conference on the WWW*, Boston, MA, December 1995.

[2] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing Reference Locality in the WWW. In *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[3] Azer Bestavros. Demand-based document dissemination to reduce traffic and balance load in distributed information systems. In *Proceedings of SPDP'95: The 7^{th} IEEE Symposium on Parallel and Distributed Processing*, San Anotonio, Texas, October 1995.

[4] Azer Bestavros, Robert Carter, Mark Crovella, Carlos Cunha, Abdelsalam Heddaya, and Sulaiman Mirdad. Application level document caching in the internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.

[5] Mark Crovella and Robert Carter. Dynamic server selection in the internet. In *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95)*, August 1995.

[6] Carlos Cunha, Azer Bestavros, and Mark Crovella. Characteristics of www client-based traces. Technical Report TR-95-010, Boston University, CS Dept, Boston, MA 02215, April 1995.

[7] Peter Danzig, Richard Hall, and Michael Schwartz. A case for cashing file objects inside internetworks. Technical Report CU-CS-642-93, University of Colorado at Boulder, Boulder, Colorado 80309-430, March 1993.

[8] S.J. Eggers and R.H. Katz. A characterisation of sharing in parallel programs and its application to coherence protocol evaluation. In *Proceedings of the 15th International Symposium on Computer Architecture*, pages 373–383, May 1988.

[9] Steven Glassman. A caching relay for the world wide web. In *Proceedings of the First Interntional Conference on the WWW*, 1994.

[10] James Gwertzman and Margo Seltzer. The case for geographical push caching. In *Proceedings of HotOS'95: The Fifth IEEE Workshop on Hot Topics in Operating Systems*, Washington, May 1995.

[11] James Gwertzman and Margo Seltzer. World wide web cache consistency. In *Proceedings of the 1996 USENIX Technical Conference*, San Diego, CA, January 1996.

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

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

[5]For example, the increasing percentage of Common Gate Interface (CGI) queries, Java applets and scripts.

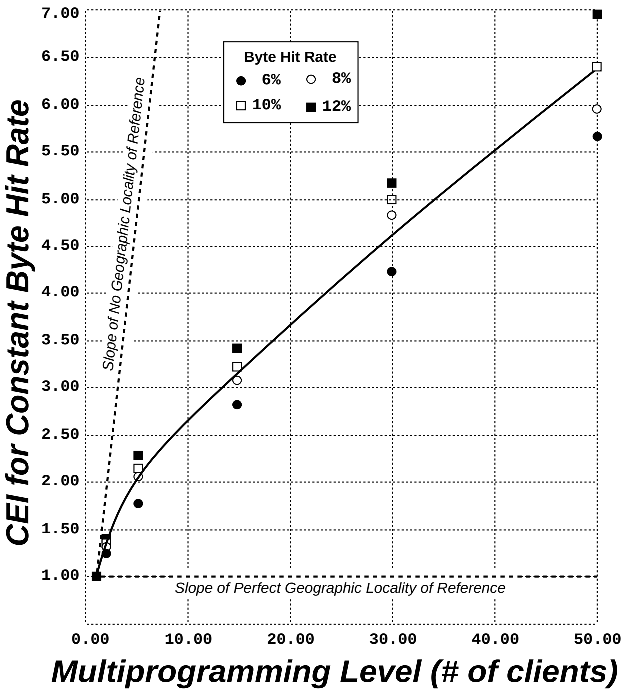

Figure 1: Cache size *vs* MPL for a constant hit rate

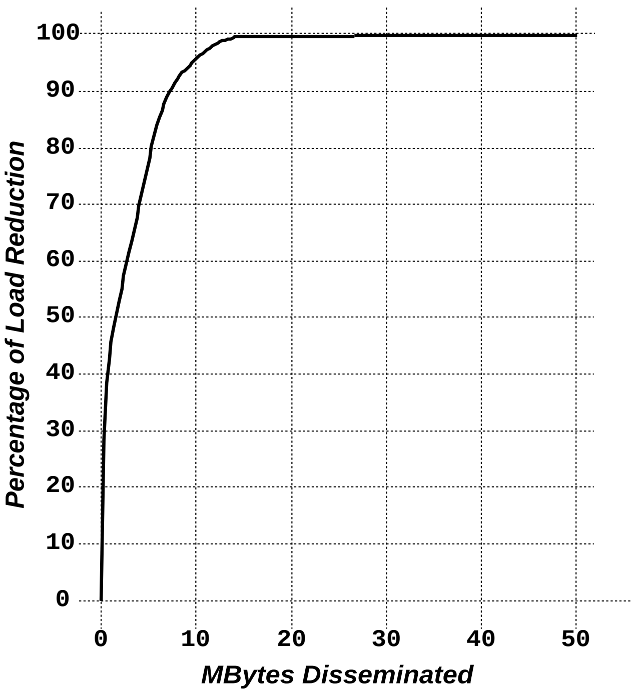

Figure 3: Load reduction due to popularity profile

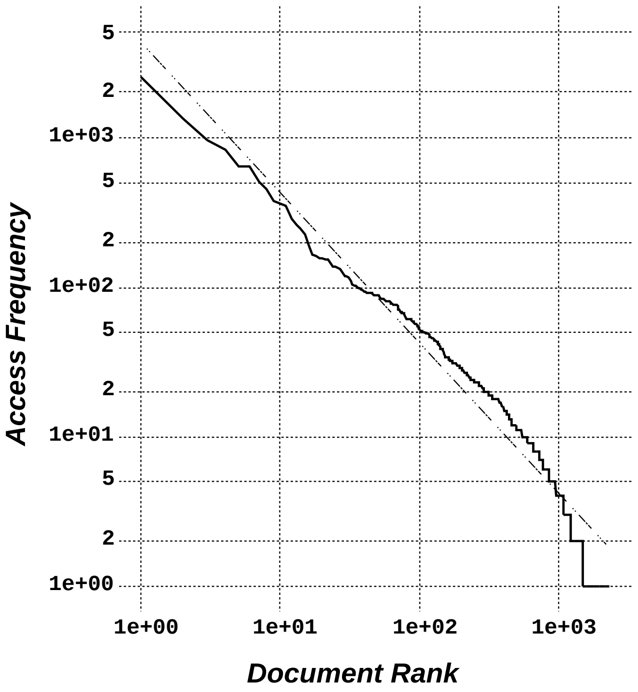

Figure 2: Popularity of documents versus rank

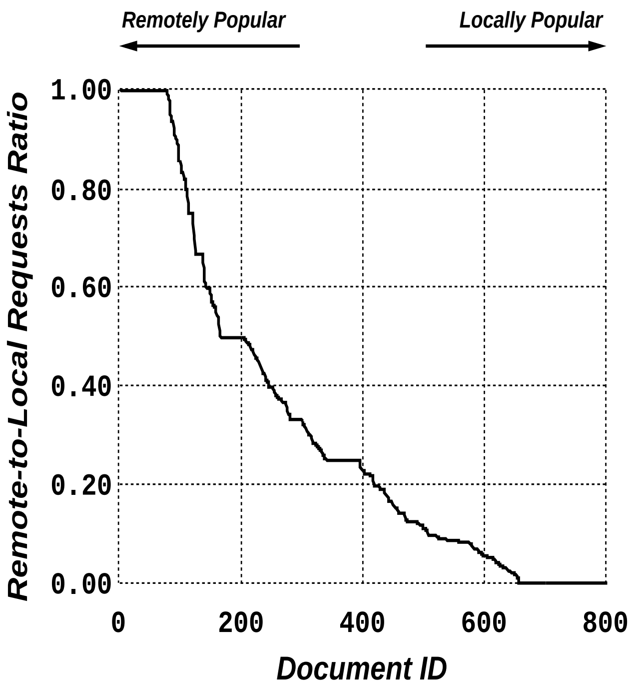

Figure 4: Local vs remote popularity of documents

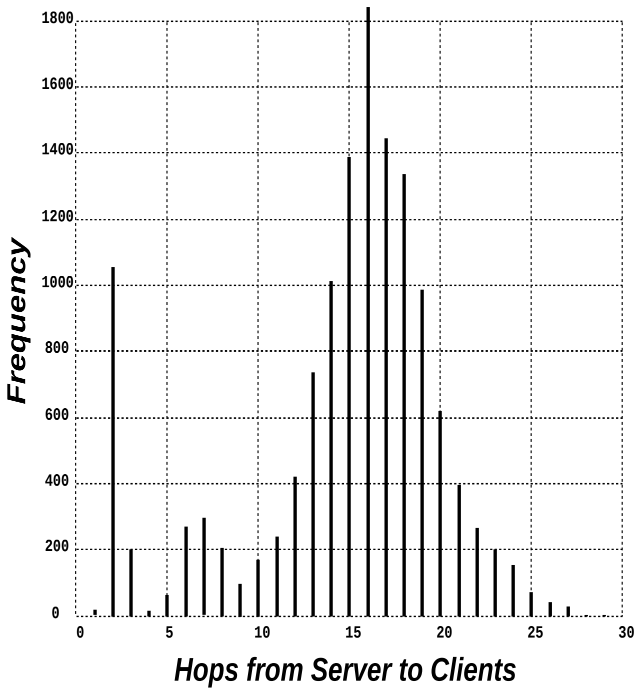

Figure 5: How far away are clients?

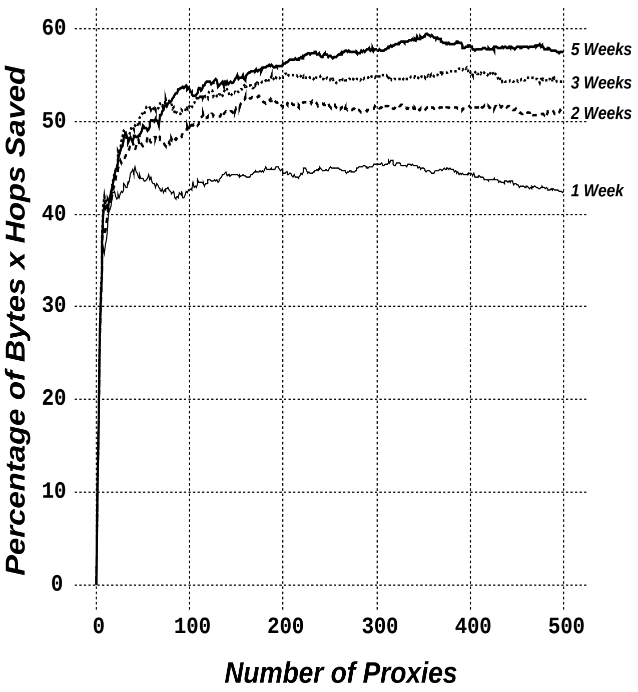

Figure 7: Performance succeptibility to history length

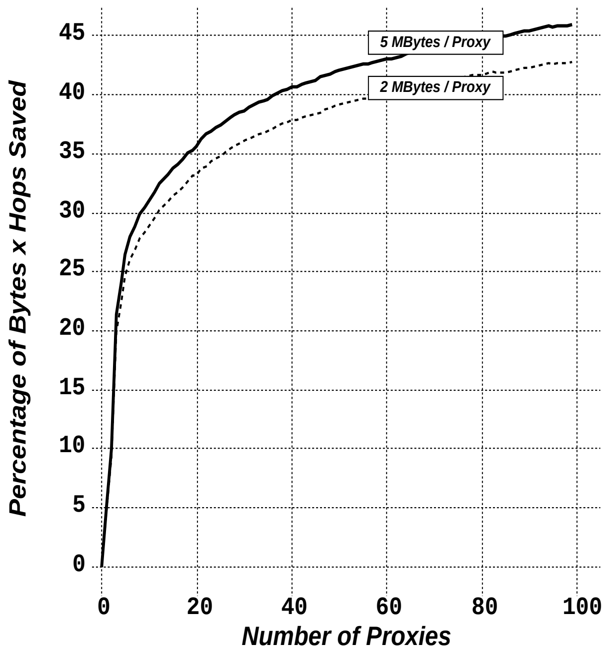

Figure 6: Bandwidth saved through dissemination

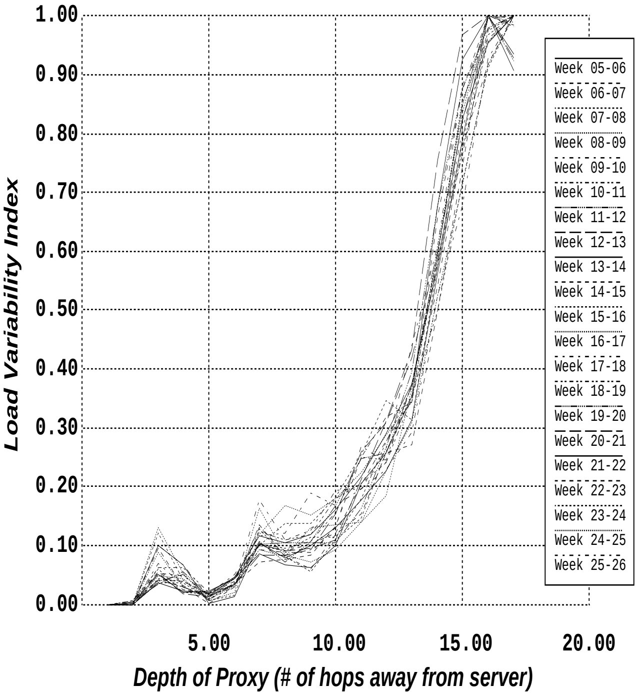

Figure 8: Load variability versus depth