Implications of Proxy Caching for Provisioning Networks and Servers*

Mohammad Raunak, **Prashant Shenoy**, **Pawan Goyal**‡**Krithi Ramamritham**†and **Purushottam Kulkarni**
Department of Computer Science, University of Massachusetts, Amherst, MA 01003
{raunak,shenoy,krithi,purukulk}@cs.umass.edu
‡Ensim Corporation, 1366 Borregas Avenue Sunnyvale, CA 94089
goyal@ensim.com

Abstract

In this paper, we examine the potential benefits of web proxy caches in improving the effective capacity of servers and networks. Since networks and servers are typically provisioned based on a high percentile of the load, we focus on the effects of proxy caching on the tail of the load distribution. We find that, unlike their substantial impact on the average load, proxies have a diminished impact on the tail of the load distribution. The exact reduction in the tail and the corresponding capacity savings depend on the nature of the workload and the percentile of the load distribution chosen for provisioning networks and servers—the higher the percentile, the smaller the savings. For workloads considered in this study, compared to over a 50% reduction in the average load, the savings in network and server capacity was only 20-35% for the 99^{th} percentile of the load distribution. We also find that while proxies can be somewhat useful in smoothing out some of the burstiness in web workloads; the resulting workload continues, however, to exhibit substantial burstiness and a heavy-tailed nature. We identify one-time requests for large objects to be the limiting factor that diminishes the impact of proxies on the tail of load distribution. We conclude that, while proxies are immensely useful to users due to the reduction in the average response time, they are less effective in improving the capacities of networks and servers.

Keywords: client-server systems, Internet, communication systems performance, web proxy caching

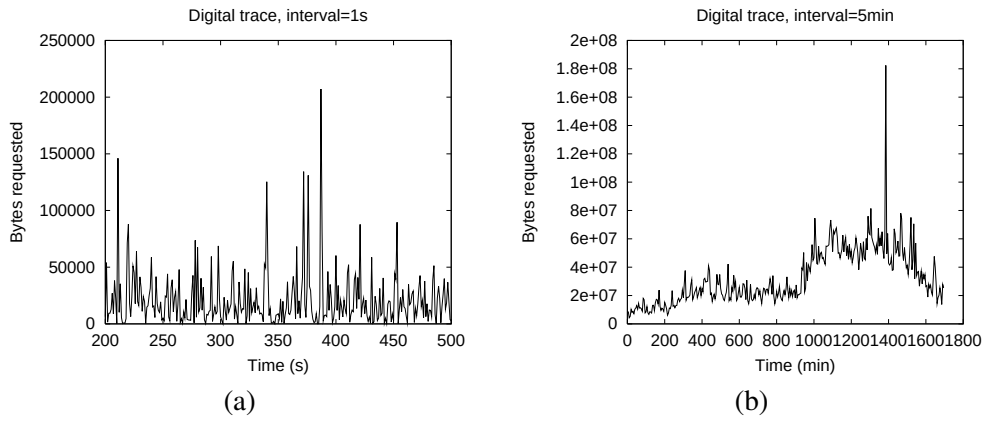
1 Introduction

1.1 Motivation

The past decade has seen a dramatic increase in the amount of web traffic in the Internet; from an insignificant fraction in 1993, web traffic has grown to become the largest component of the total traffic on the Internet today. Recent studies have revealed that web accesses tend to be non-uniform in nature, resulting in frequent server and network overload, and thereby significantly increasing the latency for information access. Proxy caches provide a way to alleviate this drawback. In a proxy-based architecture, clients send web requests to proxies; proxies respond to these requests using locally cached data or by fetching the requested object from the server. By caching frequently accessed objects and serving requests for these objects from the cache, proxies can yield a number of performance benefits: (i) deploying a proxy in a local area network or near a slow inter-continental link helps reduce client response times, (ii) deploying a proxy near an access link (i.e., the link that connects the organization to the

*This research was supported in part by National Science Foundation grants ANI-9977635, CCR-9984030, CCR-0098060, EIA-0080119, IRI-9619588, IBM, Intel, EMC, Sprint, and the University of Massachusetts.

†Also affiliated with the Dept. of Computer Science and Engg, Indian Institute of Technology, Powai, Bombay.



(a) (b)

Figure 1: The bursty nature of web workloads. The figures demonstrate the the bursty nature of web workloads over time scales of one second and five minutes.

rest of the Internet) helps organizations and network service providers reduce their network bandwidth usage, (iii) deploying a proxy close to web servers (referred to as server-side caching) helps reduce load on these servers, and (iv) deploying proxies in the network core helps reduce aggregate network load on backbone links.

Thus, by absorbing a portion of the workload, proxy caches can improve client response times as well as increase the capacities of servers and networks, thereby enabling them to service a potentially larger clientele. The latter hypothesis has been the basis for the large-scale deployment of proxies by organizations and network service providers in their computing infrastructure [7]. Surprisingly, however, there has been no study that quantifies these benefits. Most performance studies to date have demonstrated the efficacy of proxies in reducing the *average* response time, network bandwidth usage and server load [6, 11, 17]. In contrast, servers and networks are typically provisioned based on a high percentile of the load (rather than the average load) [13]. For instance, a server may be provisioned such that the 95^{th} percentile of the response time does not exceed 100ms, or a network link may be provisioned such that the 99^{th} percentile of the bandwidth requirements does not exceed the link capacity. Conventional wisdom has implicitly assumed that a certain reduction in the average load due to caching yields a corresponding reduction in a high percentile of the load. For instance, white papers on commercial proxy servers frequently argue that a byte hit ratio of 40% yields a corresponding 40% savings in network capacity. In contrast, recent studies that have characterized web workloads [1, 2, 8] suggest that a high percentile of the load (i.e., the tail of the load distribution) may behave differently as compared to the average load due to the following reasons:

- *Heavy-tailed nature:* Web workloads tend to be heavy-tailed in nature. It has been shown that sizes of web requests as well as the resulting network bandwidth usage have a Pareto distribution with long tails [1, 8]. Whereas proxies have been shown to be effective in reducing the mean of this distribution, their effectiveness in reducing the tail of the distribution has not been quantified.

- *Impact of burstiness on locality:* Web workloads exhibit burstiness at multiple time-scales (see Figures 1(a) and (b)) [8, 12]. Periods of intense bursts in such workloads govern the tail of the load distribution. If the locality exhibited by the workload reduces during intense bursts, then the tail of the distribution will be

relatively unaffected even in the presence of a proxy cache (since the reduced locality will result in a large number of cache misses). On the other hand, if the locality increases during intense bursts, then proxies may be able to reduce the tail of the distribution. Whereas past studies have investigated the locality exhibited by web workloads over relatively long intervals [2, 10], the precise dependence of locality on burstiness at the fast time scale has not been investigated.

- *Dynamic Objects:* Dynamically generated objects (e.g., cgi-bin) are less amenable to caching since they are regenerated on every user request. Studies reveal that the fraction of dynamic and uncacheable objects in web workloads can be as high as 30% and is increasing with time [11]. An increasing proportion of such objects in web workloads will further reduce the overall effectiveness of proxy caches (in particular, their effectiveness in reducing the tail of the load distribution).

Due to the above reasons, it is not a priori evident whether proxies yield capacity savings while provisioning servers and networks and if so, to what degree. A systematic study that quantifies the effects of proxy caches on the provisioning of servers and networks and examines the factors underlying these effects is the objective of this paper.

1.2 Research Contributions

In this paper, we examine the impact of proxy caching on the provisioning of networks and servers by answering the following questions:

- Do proxy caches help in reducing the tail of the network and server loads imposed by web requests? If so, what are the resulting savings in (a) network capacity and (b) server capacity due to caching?
- Does deploying a proxy cache in the network help smooth out the burstiness in web traffic? That is, does a proxy cache (which acts as a buffer in the network) make the resulting traffic less bursty?
- What are the factors that contribute to the tail of the load distribution? In particular, how does the locality exhibited by the workload during intense bursts affect the tail of the distribution?

Observe that the first two questions examine *what* impact a proxy has on network and servers, whereas the third question examines *why* proxies have this impact.

To answer these questions, we consider trace workloads from real proxies and servers and use simulations to evaluate the impact of proxies on these workloads. Our experimental evaluation yields three key results:

- We find that proxies have a diminished impact on the tail of the load distribution (as compared to their impact on the average load). The exact reduction in the tail of the load distribution and the corresponding capacity savings depend on the percentile of the distribution chosen for provisioning networks and servers; the higher the percentile, the smaller the capacity savings.
  - Network capacity can be provisioned based either on the *bandwidth usage* or on the *client response times*. We find that proxies are somewhat effective in reducing the tail of the network bandwidth usage, but have a smaller impact on the tail of the client response time distribution. For workloads considered in this

study, the savings in network capacity for the 99^{th} percentile was only 30-35% for networks provisioned on the basis of bandwidth usage and 15-20% for networks provisioned on the basis of response times (as compared to a 50-60% reduction in average response time and bandwidth usage).

  - Server capacity is usually provisioned on the basis of the *server response time*. We find that server-side caches are less effective in reducing the tail of the response time distribution as compared to the reduction in the average response time. Specifically, the reduction in the 99^{th} percentile of server response time due to caching and the resulting savings in server capacity is 15-20% (as compared to a 30-50% reduction in average response time). However, not all web server workloads exhibit this behavior—server-side caches can be very effective for workloads that access a small number of static web pages, resulting in very high hit rates and substantial savings.

- Proxies that are deployed in the network do indeed smooth out some of the burstiness in the load. However, the resulting traffic continues to be quite bursty and heavy-tailed.

- We find that intense bursts are caused by an increase in the request rate as well as the size of requested objects. Since the locality exhibited by large objects is often poor, requests for larger objects diminish the impact of caching on the tail of the load distribution.

The rest of this paper is structured as follows. Section 2 explains our experimental methodology. We present the results of our experimental evaluation in Section 3. In Section 4, we discuss various tradeoffs and factors that may influence our results. Section 5 discusses related work, and finally, Section 6 presents some concluding remarks.

2 Experimental Methodology

Since the objective of our research is to investigate the implications of proxy caching on provisioning networks and servers, we consider two different simulation environments—one to study the effects of a proxy placed near a network access link and the other to study the effects of a server-side proxy cache attached to a web server. Figure 2 depicts these scenarios. In the rest of this section, we first describe our simulation environments in detail (Section 2.1) and then discuss the details of the trace workloads and performance metrics used in our experimental evaluation (Sections 2.2 and 2.3, respectively).

2.1 Simulation Environment

2.1.1 Simulation Environment for a Proxy near an Access Link

Consider a proxy deployed by an organization near its access link. As shown in Figure 2(a), clients within the organization send web (HTTP) requests to the proxy. The proxy services these requests using locally cached data (in the event of a cache hit), or by fetching the requested object from the server (in the event of a cache miss). The proxy is assumed to employ a disk-based cache to store frequently accessed objects. Objects in the cache are managed by the proxy using the LRU cache replacement policy. Furthermore, the proxy is assumed to employ a cache consistency mechanism to ensure that the cached data is always up-to-date with that stored on servers. Our simulations assume a

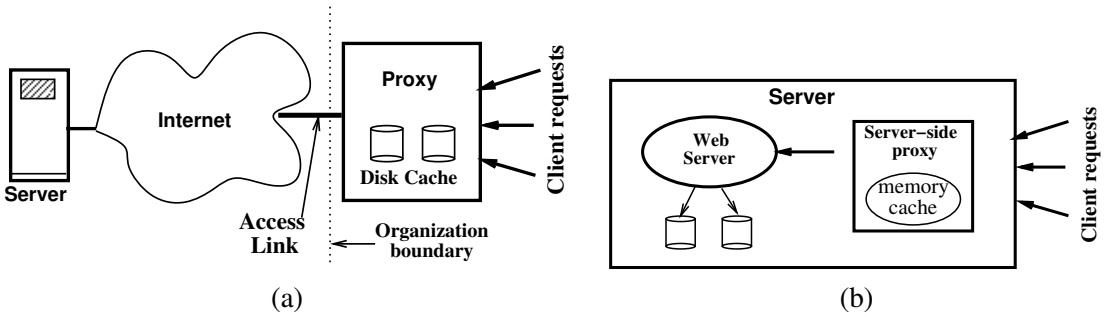


Figure 2: Simulation environments for (a) a proxy near an access link and (b) a server-side proxy.

strong consistency mechanism—one where cached data is always invalidated upon modification and the proxy never provides stale data to clients [4, 21]. In practice, however, many proxies employ cache consistency mechanisms that can occasionally provide stale data; however, we choose to simplify our study by assuming that stale data is never present in the cache (i.e., strong consistency). The possible impact of other cache consistency mechanisms and cache replacement algorithms on our results is discussed in Section 4.

In such a scenario, cache hits result in retrievals from disk and are modeled using an empirically derived disk model [5]; the model has been validated against a real disk in [20]. We choose the Seagate Barracuda 4LP disk [19] to parameterize the disk model. A (small) fixed overhead is added to the retrieval times predicted by the model to account for OS and proxy processing overheads.

Cache misses, on the other hand, result in data retrievals over the network. To model network retrieval times, we assume that the access link has a certain capacity b_{link}. The rest of path from the proxy to the server is modeled as a single bit pipe of capacity b_{int}. This is because we assume the bandwidth received by each connection is limited by the presence of a bottleneck link somewhere on the proxy-server path; the bandwidth received by the connection on this link is modeled using b_{int}. Note that different connections might encounter different such bottleneck links; we simply our model by assuming the bandwidth received on any such link is b_{int}. In such a scenario, if there are n ongoing connections on the access link, the bandwidth received by a new connection is $b = \min(\frac{b_{link}}{n+1}, b_{int})$. Given the round trip time (RTT) and the bandwidth b received by a new connection, the simulator can then compute the network latency—the time required to download an object of size s over the network.[1] Our network latency calculations take into account effects such as TCP slow-start at the beginning of a connection, but ignore packet losses and additional slow-start periods these would entail [11]. These approximations suffice for our purpose since we are interested in studying the bandwidth requirements of web workloads, rather than the effects of various Internet dynamics on these workloads (recent studies on the performance of web proxies have also made similar assumptions [11]).

To ensure that our results do not depend on the idiosyncrasies of a particular workload environment, we consider two different scenarios—a network service provider environment consisting of mostly dial-up (modem) clients, and a commercial organization consisting of mostly LAN-based clients. As explained in Section 2.2, we choose trace workloads corresponding to these two scenarios for our experimental evaluation. Our simulations assume $b_{link} = 45$

[1] Note that b may change over the lifetime of a connection if new requests arrive.

Mbps (a T3 link), $b_{int} = 256$ Kbps, a proxy-server round trip time of 250 ms and a packet size of 1500 bytes. For LAN-based environments, we choose a client-proxy bandwidth of 2 Mbps and a round trip time of 2.5 ms; for dial-up (modem) environments we choose a client-proxy bandwidth of 28.8 Kbps and a round trip time of 100 ms. While we report experimental results for these parameter values, we conducted additional experiments to measure the sensitivity of results to the choice of parameters on the client-proxy path . We found that our results were not very sensitive to the particular choice of parameters on the client-proxy path (since most of our metrics focus on the proxy-server path—see Section 2.3).

2.1.2 Simulation Environment for a Server-side Proxy

Consider a web server that employs an in-memory proxy cache as shown in Figure 2(b). All web requests to the server are intercepted by the proxy; cache hits are serviced using locally cached data, while cache misses are forwarded to the server for further processing. Cached objects are assumed to be managed using the LRU cache replacement policy. We assume that cached objects are always consistent with those at the server (such strong consistency guarantees are easy to provide for server-side proxy caches, since these proxies are under the control of the server).

Cache hits result in retrievals from an in-memory cache and are assumed to be infinitely fast (since typical memory speeds are several orders of magnitude faster than disk speeds, this is a reasonable assumption for our simulations). Cache misses, on the other hand, require the server to retrieve the requested object from disk. To model such retrievals, we assume that web objects (files) are stored on disk in terms of 8KB disk blocks (as is typical in most UNIX file systems). Blocks of a file are assumed to be randomly placed on disk. Given a request for a web object, our simulator maps it to the corresponding file on disk and then issues read requests for all disk blocks of that file. The underlying disk is assumed to employ the SCAN disk scheduling algorithm to retrieve disk blocks; disk retrieval times are computed using an empirically derived disk model [5]. As with the simulation environment for the proxy near an access link, we choose the Seagate Barracuda 4LP disk [19] to parameterize the disk model. Also, as before, we add a small, fixed overhead to each request (both hits and misses) to account for proxy and server processing overheads. Finally, in the context of server-side proxies, since we are only interested in response time at the server, our simulations do not account for the network latency to transmit the requested object to the client.

2.2 Workload Characteristics

To generate the workload for our experiments, we employ traces from real proxies and servers. We employ six different traces for our study—three proxy and three server traces. Together these traces represent environments ranging from commercial organizations to educations institutions, and from LAN-based clients to dial-up (modem) clients. The characteristics of these traces are shown in Table 1.

We use three proxy traces to evaluate the impact of caching on network bandwidth usage. The Digital trace represents a proxy servicing LAN-based clients in a commercial organization (Digital). The Berkeley trace represents the workload at a proxy servicing dial-up clients in an educational institution (Berkeley's HomeIP dial-up service), whereas the NLANR trace represents requests served at a proxy from NLANR's nationwide proxy hierarchy. As

Table 1: Characteristics of trace workloads.

Trace	Type	Duration	#Requests	#Unique Objects	#Unique Clients	Bit rate (KB/s)	Request rate (req/s)	Avg Req Size (KB)
Digital	Proxy	2 days 4 hrs	1141412	385607	5488	120.82	11.2040	9.68
Berkeley	Proxy	1 day 19 min	965172	391461	3845	52.7	6.2422	7.34
NLANR	Proxy	1 day	1064036	550088	161	76.74	12.32	6.231
NASA	Server	31 days	1415318	13436	74149	9.78	0.5284	18.43
ClarkNet	Server	7 days	1500129	32159	73466	23.45	2.4807	9.45
WorldCup	Server	8 hrs 42 min	6755067	8687	60223	1038.78	223.49	4.648

shown in Table 1, all traces span one or more days and consist of a million requests each; these requests were generated by a population of several thousand clients. Each record in the trace represents a client request and contains information such as the arrival time of a request, the requested URL, the client requesting the object, the object size, the last modified time of the object, etc. We use the last modified times for maintaining consistency of cached objects—an object is invalidated from the cache upon modification.

Turning our attention to dynamic objects, our examination of the Digital and Berkeley traces indicated that only a small fraction of the requests ($< 7\%$) accessed objects that were either dynamically generated or marked uncacheable (these traces were gathered in 1995 and 1996, respectively, and consist of mostly static web pages). The NLANR trace is more recent (April 2001) and consist of 12% uncaheable requests. Studies of other recent trace workloads [11] have also shown a larger fraction of uncacheable objects (around 30%) and this fraction is reported to be increasing with time. Consequently, the results reported in this paper are optimistic—a larger fraction of dynamically generated objects will further reduce the efficacy of proxy caching.

We use three server traces to evaluate the impact of caching on server response times. The first trace represents a web server belonging to a network service provider (ClarkNet), the second a server at a government institution (NASA), while the third trace is from a server for a sports event (World Cup Soccer 1998). The traces were gathered over a period of 7 days, 31 days and 8.7 hours, respectively; the number of requests ranges from 1.4 to 6.75 million requests (see Table 1). The traces provide information similar to that provided by proxy traces. Again we use the last modified times of an object to ensure that the object is invalidated from the cache upon modification.

2.3 Performance Metrics

We use two different metrics to study the impact of deploying proxy caches near access links—*network bandwidth usage* and *client response time*. The network bandwidth usage on the access link is defined to be the amount of data retrieved over the link per second. The client response time of a request is defined to be the total time to service a client request (it includes the response time of the proxy for cache hits and the network latency incurred to download an object for cache misses).

To study the impact of caching by proxies on the load experienced by a server, we use the *server response time* as our metric. The server response time is defined to be the time required by the server to retrieve the requested object from disk (it does not include the network latency incurred to transmit the object from the server to the client).

For each of these metrics, we first compute a distribution of the metric using simulations. We then compute the mean as well as various percentiles of these distributions and study their variation for different cache sizes.

3 Experimental Evaluation

The objective of this section is to answer the three questions posed in Section 1.2. We consider each question in turn. Sections 3.1 and 3.2 examine the impact of caching on the tail of the network and server loads, respectively. Section 3.3.1 examines whether caching can help reduce burstiness in the load, while Section 3.3.2 investigates the factors governing the tail of the load distribution.

3.1 Implications of Proxy Caching for Network Provisioning

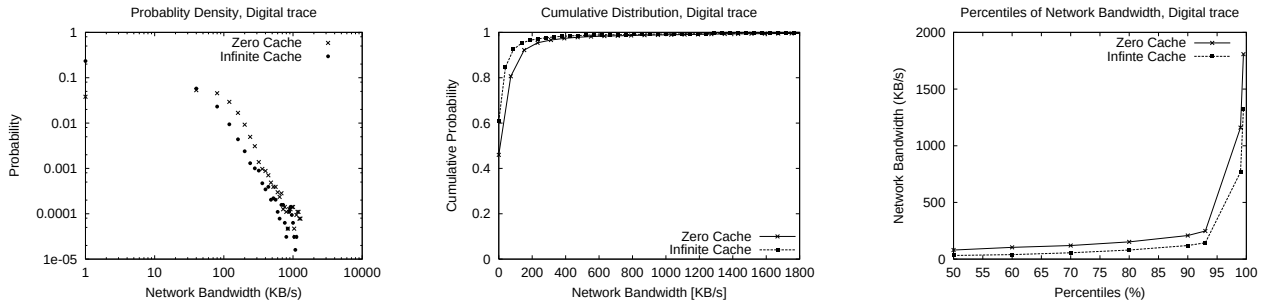
To determine the impact of proxy caching on the load on an access link, we varied the cache size from zero to infinity and computed the distributions of the network bandwidth usage and the client response time.

3.1.1 Effect on Network Bandwidth Usage

Figures 3(a) and (b) plots the probability density function (on log scale) and the cumulative distribution function of the network bandwidth usage. The figure shows that the distribution of network bandwidth usage has a long tail in the absence of a proxy cache. More importantly, the distribution of network bandwidth usage continues to exhibits a long tail even with the presence of an infinite cache. This suggests a diminished impact of proxy caches in reducing the tail of the distribution. (Section 3.3 provides a more precise characterization of the tail of the distribution and discusses the reasons for this behavior).

To determine the impact of proxy caching on the tail of the distribution, we computed the average bandwidth usage and the 99^{th} percentile of the bandwidth usage for different cache sizes. For the Digital and the Berkeley traces, we find that (i) the average bandwidth usage reduces by half as we increase the cache size from zero to infinity, and (ii) the 99^{th} percentile of the bandwidth usage also reduces but to a smaller extent. Figure 4 quantifies this reduction in the network bandwidth usage. As shown in the figure, the reduction in the 99^{th} percentile of the bandwidth usage is 30-35% (as compared to a nearly 50% reduction in average bandwidth usage). Moreover, as shown in Figure 4(b)) the reduction in bandwidth usage decreases with increasing percentiles of the bandwidth distribution (from around 30-40% for the 95^{th} percentile to 25% for the 99.5^{th} percentile).

The behavior of the NLANR trace is somewhat different. Like the Digital and the Berkeley trace, the NLANR trace shows smaller bandwidth savings for higher percentiles. However, the savings do not monotonically decrease with increasing percentiles. As shown in Figure 5(a), the savings reduce from 35% to 20% as the percentile increases from 96 to 99.8. In contrast, the reduction in the 98^{th} percentile of the bandwidth usage is higher (around 58%). To better understand the reason for this behavior, we examined requests for large objects ($>$ 1 MB in size). It has been observed that large objects are more likely to be one-time requests (i.e., requested only once and never again) than small objects. Consequently, requests for large objects are more likely to be compulsory misses, diminishing the impact of caching on the tail of the distribution. In contrast, as shown in Figure 5(b), the NLANR trace has several 1.3MB objects that are very popular, resulting in a large number of cache hits. This is the likely cause of increased



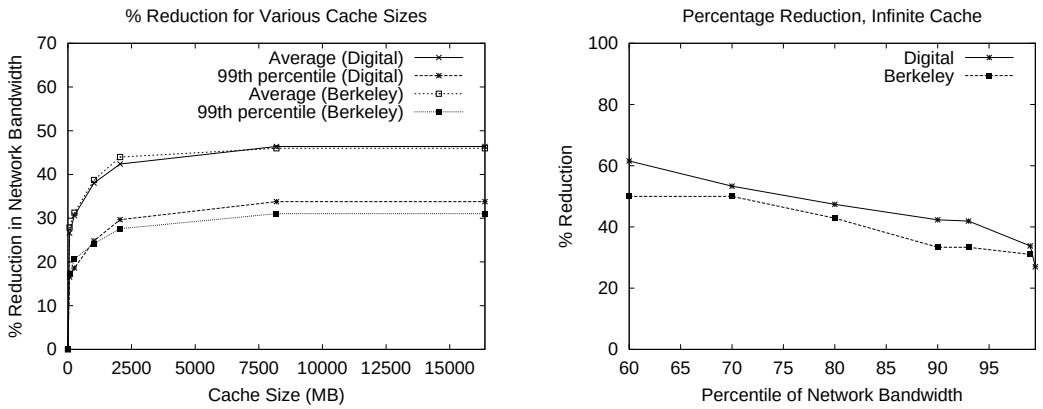
(a) Probability density function (b) Cumulative Distribution function (c) Percentiles of bandwidth usage

Figure 3: Distribution of network bandwidth usage in the presence and absence of caching

savings at the 98^{th} percentile. The fraction of cache misses increases for objects larger than 1.3MB, resulting in smaller savings for higher percentiles. In the Digital trace, on the other hand, requests for large objects consistently result in more cache misses than hits (see Figure 5(c)), resulting in monotonically decreasing savings for higher percentiles.

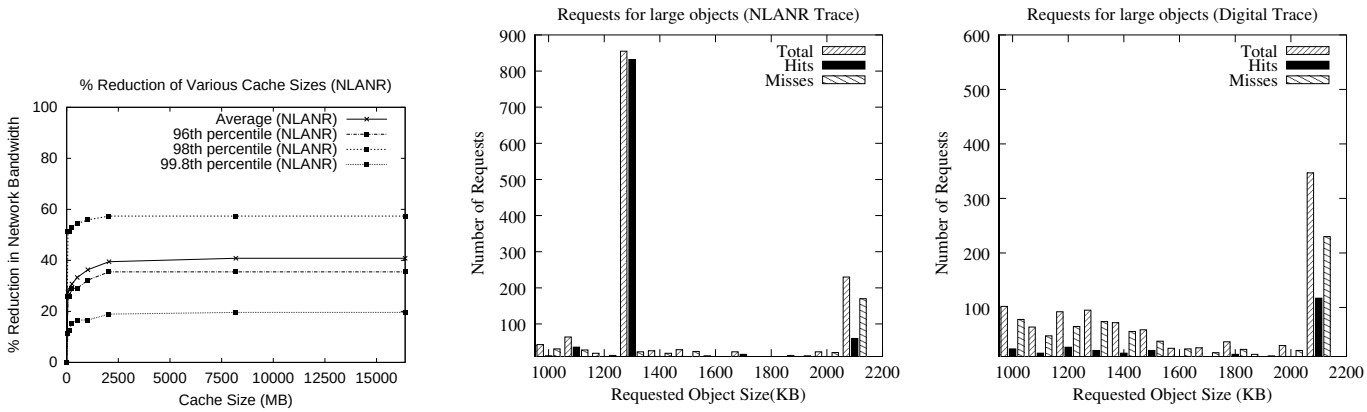
Together Figures 3 and 4 lead us to the following conclusions.

Result 1 *Proxy caches can help reduce the tail of the network bandwidth usage and thereby increase overall network capacity (or equivalently, yield capacity savings for a given workload). However, the magnitude of the capacity savings that can be achieved is smaller than the reduction in the average bandwidth usage. Moreover, these savings depend on the percentile of the bandwidth distribution chosen for provisioning the network—the higher the percentile, the smaller the savings. The savings also depend on the number of one-time requests for large objects—the larger that number, the smaller the savings.*



(a) Reduction for different cache sizes (b) Reduction for different percentiles

Figure 4: Reduction in network bandwidth usage due to caching



(a) Reduction for different cache sizes

(b) Number of requests for large objects (NLANR)

(c) Number of requests for large objects (Digital)

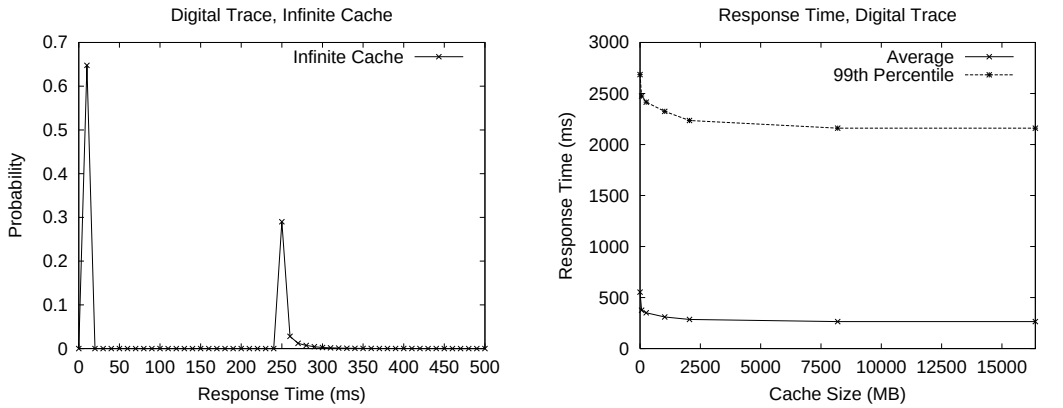
Figure 5: Reduction in network bandwidth usage and the nature of requests for large objects.

3.1.2 Effect on Client Response Times

Since a network may be provisioned on the basis of the response time (rather than bandwidth usage), in what follows, we examine the impact of proxy caching on the client response times. To do so, as before, we varied the cache size from zero to infinity and computed the distribution of the client response time for different cache sizes. Figure 6(a) depicts the distribution of the response time obtained for an infinite cache. The figure shows that the response time distribution is bimodal and has a long tail. Since the response time in the event of a cache hit is significantly smaller than that for a miss, the two peaks in this bimodal distribution represents hits and misses, respectively. Figure 6(b) depicts the variation in average response time and the 99^{th} percentile of the response time for different cache sizes. The figure shows that the average response time decreases due to an improvement in cache hit ratio for larger caches. The 99^{th} percentile of the response time also decreases but to a smaller extent. To understand the reasons for this behavior, we observe that, due to their large network retrieval times, cache misses dominate the set of requests that govern a high percentile of the response time. Consequently, we investigate the impact of cache misses on the network latency (which constitutes the response time of miss requests).

Figure 7(a) plots the probability density function of the network latency; as expected, the distribution consists of a single peak (which corresponds to the peak due to cache misses in the bimodal response time distribution). Figure 7(b) plots the variation in the 99^{th} percentile of the network latency for different cache sizes. Interestingly, the figure shows that the 99^{th} percentile of the latency *increases* with increasing cache sizes. To understand the reasons for this increase, observe that network latency depends on two factors: (i) the size of the requested object, (ii) the bandwidth available to download the object. We consider the effect of each factor on the tail of the network latency.

- *Size of the requested object:* Studies have shown that small objects are more likely to be reaccessed as compared to large objects [2]. Moreover the average size of a requested object in our workloads is small ($<$ 10 KB). It follows that a cache hit is more likely to access a small object over a large object. Increasing the cache size increases the number of cache hits, causing the cache to absorb a larger fraction of requests for small objects. Put another way, as the cache size increases, an increasing fraction of the cache misses are for large



(a) Probability density function (b) Response times for different cache sizes

Figure 6: Impact of caching on response times. Figure (a) plots the probability density function of the response time while figure (b) shows the variation in the average and 99^{th} percentile of the response time with increasing cache sizes.

objects. This causes the tail of the request size distribution for cache misses to increase with the cache size, and hence, the tail of the network latency shows a corresponding increase.

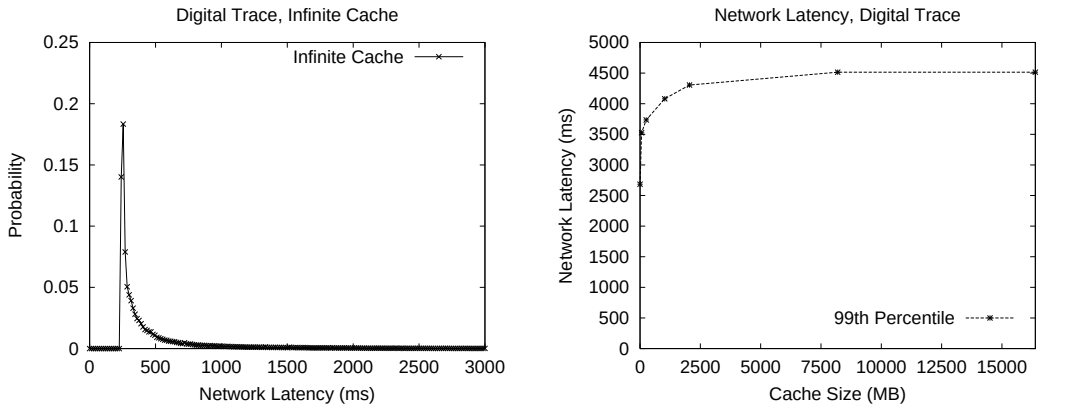
- *Bandwidth available for a web request:* By absorbing a fraction of the workload, a proxy cache reduces the load on the access link and increases the average bandwidth available for each network retrieval. An increase in available bandwidth results in faster downloads and a reduction in network latency. However, the bandwidth requirements of our trace workloads are significantly smaller than the capacity of the 45 Mbps T3 link assumed in our simulations. This causes the link to be lightly loaded and hence, the improvement in bandwidth availability due to caching is negligible.

Since the decrease in network latency due to improved bandwidth availability is more than offset by increase in latency due to large objects, the 99^{th} of the network latency shows an overall increase.[2] Due to their inability to reduce the tail of the network latency, proxy caches yield only a small decrease in the tail of the response time distribution (despite the increase in the tail of the latency, the tail of the response time shows a net decrease due to the improvement in hit ratios for larger caches). Figure 8(a) quantifies the reduction in the average response time and the 99^{th} percentile of the response time due to caching. The figure shows that the 99^{th} percentile of the response shows a 15-20% reduction due to caching (in contrast to over a 50% reduction in the average response time). Moreover, as shown in Figure 8(b), the percentage reduction in response times falls with increasing percentiles of the response time distribution (from around 40% for the 95^{th} percentile to only 15% for the 99.5^{th} percentile).

Thus we have the following result:

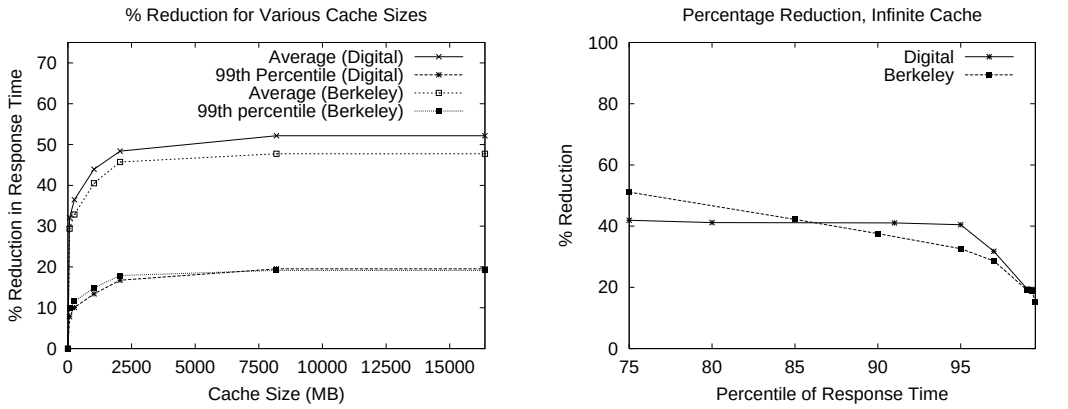
Result 2 *For the workloads considered in this study, we find that networks provisioned on the basis of the response time (rather than the bandwidth usage) see a small (15-20%) improvement in capacity due to proxy caching (as*

[2]We verified this hypothesis by simulating a low bandwidth link of capacity 128KB/s. In this case, the increase in available bandwidth per request is more significant than the increase in request sizes, causing the tail of the latency to show an overall *decrease*.



(a) Probability density function (b) Network latency for different cache sizes

Figure 7: Impact of caching on the network latency incurred by cache misses



(a) Reduction for difference cache sizes (b) Reduction for different percentiles

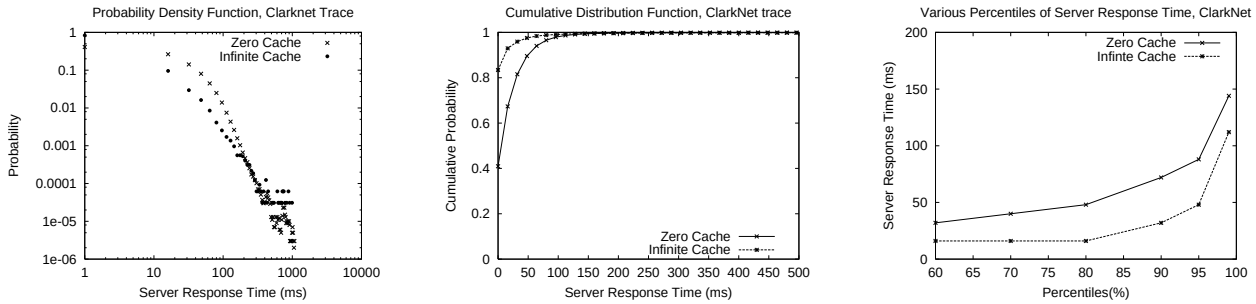
Figure 8: Reduction in response time due to caching

compared to over a 50% improvement in the average response time). This is because the tail of the network latency increases with increasing cache sizes and counters some of the improvements in response times due to increasing cache hit ratios.

3.2 Implications of Proxy Caching for Server Provisioning

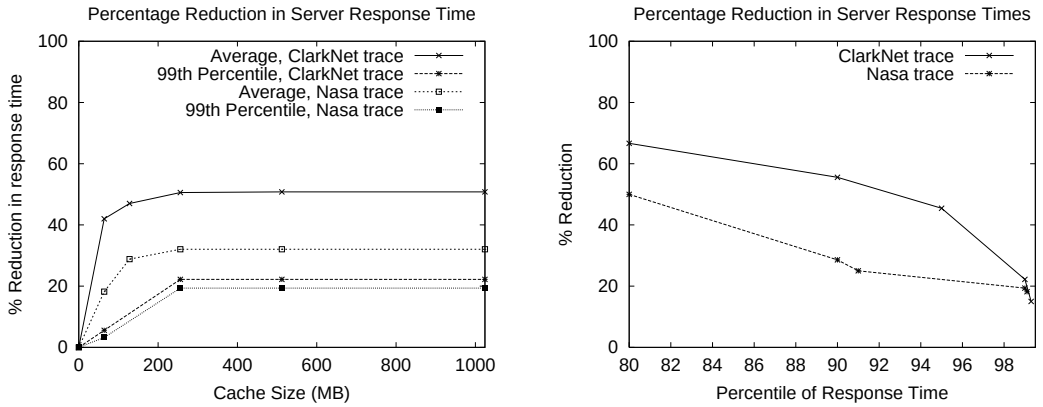
To study the impact of a server-side proxy cache on the server load, we varied the cache size and computed the distribution of the server response time for various cache sizes.

Figure 9 depicts the distribution of the server response time in the presence and absence of a proxy cache. The figure shows that the response time distribution has a long tail even in the presence of an infinite cache (again indicating a reduced impact of caching on the tail of the response time distribution). We computed variation in the average response time and the 99^{th} percentile of the response time for different cache sizes. For the ClarkNet and NASA traces, we found that both the average and the 99^{th} percentile of the response time decrease with increasing



(a) Probability density function (b) Cumulative Distribution function (c) Percentiles of response time

Figure 9: Distribution of server response times in the presence and absence of caching.



(a) Reduction for different cache sizes (b) Reduction for different percentiles

Figure 10: Reduction in Server Response Times due to Caching

cache sizes, but the percentage reduction in the 99^{th} percentile is smaller than that in the average response time. Figure 10 quantifies this reduction. It shows that, as in the case of network bandwidth, the average response time reduces by 35-50% as we increase the cache size from zero to infinity, but the reduction in the 99^{th} percentile of the response time is only 20%. Moreover, this reduction decreases with increasing percentiles of the response time distribution; the percentage reduction varies from 25-40% for the 95^{th} percentile to 15% for the 99.5^{th} percentile (see Figure 10(b)). (Observe that Figures 9 and 10 are the *server response time* counterparts of Figures 3 and 4 for network bandwidth usage.)

The characteristics of the WorldCup trace are very different from the other two traces we studied. As shown in Table 1, the trace consists of a large number of requests to a very small set of web pages. Consequently, the resulting cache hit ratio is very high (99% or higher) even for caches as small as 4MB in size (see Table 2). Due to these high hit rates, there is a substantial reduction in the average and high percentiles of the response time (see Figure 10). Consequently, server-side caches can indeed be very effective for workloads that access a small set of mostly static web pages.

Thus, we have the following result:

Table 2: Cache hit rates for World Cup Trace

Cache Size (MB)	Hit ratio	Byte hit ratio
0	0	0
4	0.9958	0.9885
16	0.9968	0.9908
Infinite	0.9981	0.9953

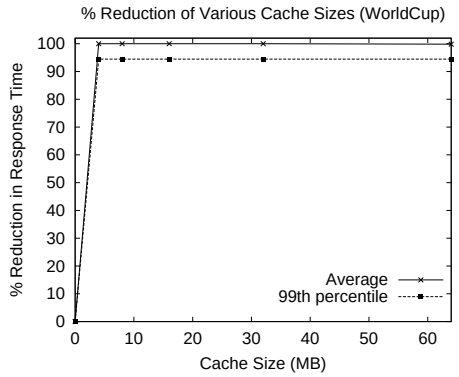


Figure 11: Reduction in Server Response Time for the World Cup trace

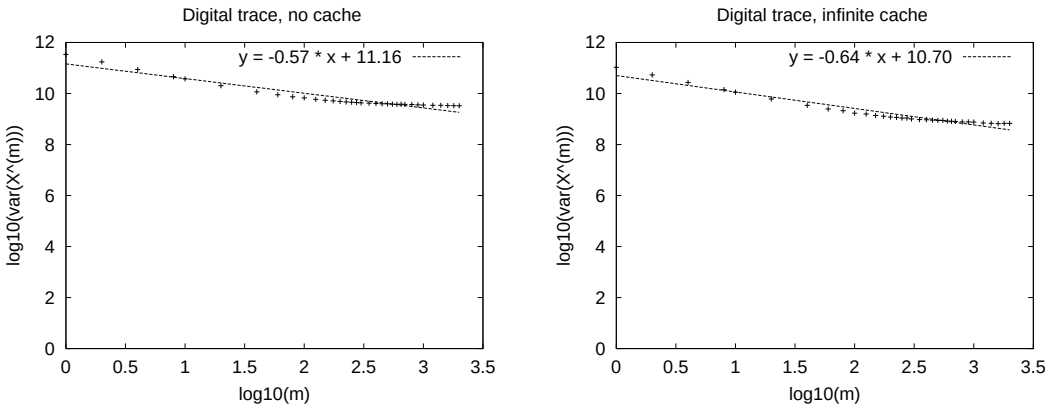
Result 3 *Server-side proxy caches can be somewhat effective for provisioning servers. The increase in capacity of a server (or the savings in capacity for a particular workload) depends on the percentile of the response time distribution chosen for provisioning the server. For our trace workloads, these gains varied from 25-40% for the* 95^{th} *percentile to only 15% for the* 99.5^{th} *percentile. The savings in capacity may be further limited by constraints on the size of an in-memory cache (our results provide upper bounds on the response time reduction by assuming an infinite cache). Finally, server-side caches can be very effective when the workload accesses a small set of mostly static web objects.*

3.3 Impact of Caching on Burstiness and Locality

Our experiments thus far have examined the impact of proxy caches on the capacity of servers and networks. In this section, we examine why proxies have such an effect on the web workloads. Specifically, we examine (a) whether proxies can help smooth out the burstiness in web workloads, and (b) the relationship between locality and the burstiness of web workloads.

3.3.1 Effect of Caching on Burstiness

Past studies have shown that web workloads exhibit burstiness at multiple time scales [8, 12]; the workloads employed in our study are no different (see Figures 1(b) and (c)). In this section, we determine whether a proxy cache can help smooth out the burstiness in web workloads, or whether the resulting workload becomes more bursty. A



(a) No cache (b) Infinite cache

Figure 12: Determination of the Hurst parameter using variance-time plots

Table 3: Effect of caching on burstiness

	Digital		Berkeley	
	No cache	Infinite cache	No cache	Infinite cache
Mean bit rate (KB/s)	120.82	75.97	52.7	35.2
Hurst parameter and 95% CI	0.71 ± 0.06	0.68 ± 0.05	0.75 ± 0.06	0.68 ± 0.05
Pareto and 95% CI	1.32 ± 0.009	1.33 ± 0.013	1.5 ± 0.03	1.62 ± 0.04

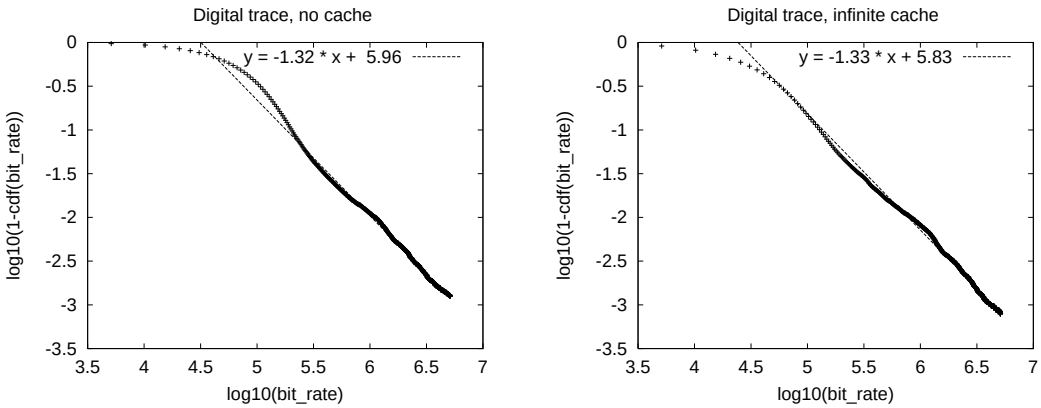
better understanding of these issues will help network designers determine if proxies should be deployed to smooth out the aggregate web traffic in a network and the implications of doing so (a more bursty workload yields statistical multiplexing gains, but also increases the probability of transient congestion in the network).

We use two different metrics to quantify the burstiness of a workload: (i) the *Hurst Parameter* H, which is a measure of the self-similarity (i.e., burstiness) exhibited by a workload [8]; and (ii) the heavy-tailed nature of the workload as defined by the *Pareto* distribution: $P(X > x) \approx x^{-\alpha}$, as $x \to \infty$, $0 < \alpha < 2$.

To determine the impact of caching on burstiness, we compare the original workload to that generated in the presence of an infinite cache. To do so, we first compute the bit rate of the workload over one second intervals (assuming no proxy is present). Next, we filter out cache hits from the workload assuming the presence of an infinite cache and repeat this process. We then compute the Hurst parameter H and the parameter α of the Pareto distribution for these two workloads as follows.

To compute the Hurst parameter, we assume that the bit rate of the workload over one second intervals represents a time series $X = (X_t; t = 1, 2, 3, ...)$. We then compute the m-aggregated series $X^{(m)} = (X_k^{(m)}; k = 1, 2, 3...)$ by summing the original series X over non-overlapping blocks of size m. The variance of $X^{(m)}$ is then plotted against m on a log-log plot, yielding the so-called *variance-time* plot [8, 12, 16]. A straight line with slope $-\beta$ is indicative of self-similarity and the Hurst parameter H is given by

$$H = 1 - \frac{\beta}{2}; \quad 0 < \beta < 2$$



(a) No Cache (b) Infinite Cache

Figure 13: Determination of the α parameter of the Pareto distribution using LLCD plots.

The workload is said to self-similar if $0.5 < H < 1$, with larger values of H indicating an increase in burstiness. (see [8, 12, 16] for a more detailed treatment of self-similarity). Figure 12 illustrates this process by: (i) depicting the variance-time plot for the Digital trace, and (ii) computing a linear fit through these data points using a simple linear regression model (least-squares method). The figure yields $\beta = 0.57$ and $\beta = 0.64$ for cache sizes of zero and infinity, respectively, and corresponding Hurst parameters of $H = 0.71$ and $H = 0.68$, respectively. Table 3 summarizes the Hurst parameter and the 95% confidence intervals obtained for our workloads. The table shows that H reduces in the presence of a proxy cache for both the Digital and Berkeley workloads, suggesting that a proxy can potentially reduce the burstiness in web workloads.[3] However, the Hurst parameter of the resulting workload continues to be in the range $0.5 < H < 1$, indicating that the workload continues to exhibit significant burstiness.

Next, we determine if the asymptotic shape of the bit rate distribution has a heavy-tailed nature (i.e., if it conforms to the Pareto distribution). To do so, we use log-log complementary distribution (LLCD) plots [8]. These are plots of the complementary cumulative distribution $\overline{F}(x) = 1 - F(x) = P(X > x)$ on the log-log axes. If the LLCD plot is approximately linear over a significant range in the tail, then the distribution is heavy tailed, and the slope $-\alpha$ of the linear portion yields the corresponding Pareto distribution. A decrease in α indicates that an increasing portion of the mass is present in the tail, making the distribution more heavy-tailed (see [8] for a more detailed description). Figure 13 shows the LLCD plots for the Digital trace as well as a linear fit through the tail of the distribution using the least squares method. The slopes of the linear fit as well as the 95% confidence intervals are summarized in Table 3. Together, Figure 13 and Table 3 show that: (i) the original workload and that in the presence of an proxy cache are heavy-tailed and conform to the Pareto distribution, (ii) α increases in the presence of a cache, indicating that a proxy is somewhat successful in reducing the tail of the distribution (however, the increase in α is small ($< 8\%$) and the resulting workload continues to be heavy-tailed).

Thus, we have the following result:

[3] Since the 95% confidence intervals of the two workloads overlap, we can not draw any definite conclusions about a reduction in burstiness due to caching.

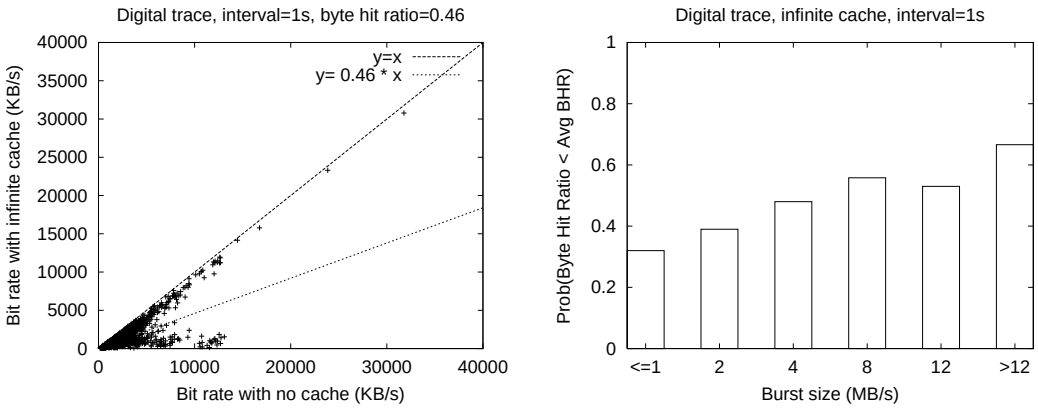


Figure 14: Effect of burstiness on the locality.

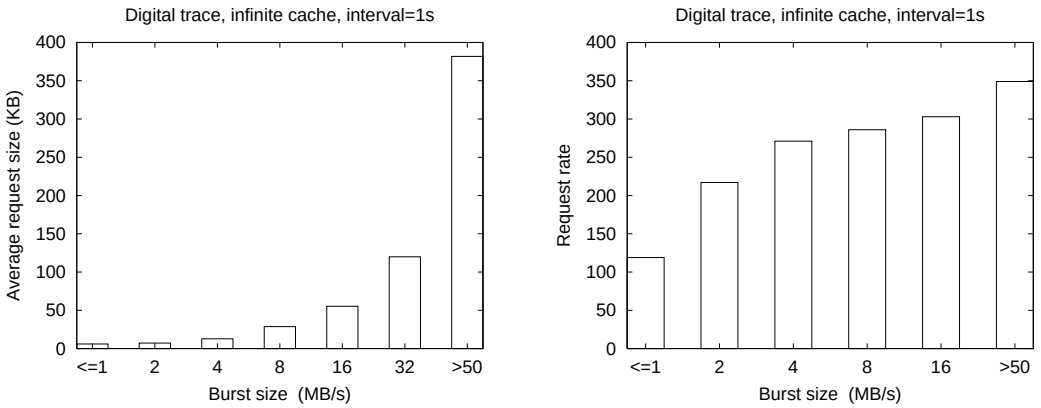
Result 4 *Deploying proxies in the network not only reduces the average volume of traffic due to web requests (as shown in Section 3.1), but also produces a reduction, albeit small, in the burstiness in the workload. The resulting workload continues, however, to exhibit significant burstiness and a heavy-tailed nature.*

3.3.2 Effect of Burstiness on Locality

To understand the effect of burstiness on locality, let B_0^t and B_∞^t denote the bit rate of the trace in the interval $[t, t+1)$ when no proxy is present and in the presence of a proxy with an infinite cache, respectively. Let γ denote the average byte hit ratio of the workload over the entire trace. Observe that, if $B_\infty^t < \gamma \cdot B_0^t$ then the workload exhibits better than average locality within that interval. On the other hand, if $B_\infty^t > \gamma \cdot B_0^t$ then the trace exhibits worse than average locality in that interval. Figure 14(a) plots (B_0^t, B_∞^t) pairs for each one second interval in the Digital trace. The figure shows that, for high bit rates, it is more likely that the trace exhibits worse than average locality. In fact, for some intense bursts, the figure shows that $B_\infty^t \approx B_0^t$, indicating that most requests in that interval were cache misses.

To understand the reasons for such poor locality, we computed the fraction of the intervals in which the trace exhibited worse than average locality for various burst sizes. Figure 14(b) plots these values. As shown in the figure, the probability that the locality exhibited by the trace is worse than average increases with increasing burst sizes (i.e., bit rates). To further understand this behavior, we computed the average requests sizes and the number of requests/s for different burst sizes. Figures 15(a) and (b) plot these values. The figure shows that both the request rate and the average size of a requested object increases with burst size. Since the increase in request size is significantly larger than that in request rate, we conjecture that requests for large objects with poor locality (i.e., the so-called "one-timers" [3]) contribute to the reduction in overall locality during intense bursts.[4] We are currently conducting a separate study using recent traces to examine the validity of this hypothesis, and to gain further insights into the characteristics of large objects.

[4]A prior study has found a weak correlation between the object size and its access frequency—larger objects are less likely to be requested again as compared to smaller objects [2]. Our study seems to indicate that even a weak correlation can have significant performance implications, especially on the tail of the distribution.



(a) Average request size per burst (b) Request rate

Figure 15: Variation in the average requests size and the request rate with increasing burst size.

Thus, we have the following result:

Result 5 *Intense bursts result in an increase in the request rate as well as a sharp increase in the average request size. Since large objects are more likely to exhibit poor locality as compared to small objects, an increase in the request size causes the overall locality of the workload to reduce during intense bursts (and consequently, reduces the impact of caching on the tail of the load distribution).*

4 Additional Considerations and Directions for Future Work

In this section, we consider the possible effects of the assumptions made in our simulation study on our results and discuss possible directions for future work.

- *Effect of hierarchies/cooperating caches:* Our study assumed a single proxy cache that is deployed near an access link or a web server. Many large organizations make use of a set of cooperating caches that are often organized in a hierarchical manner. Cooperating caches can improve hit ratios by sharing cached data across a larger set of clients. However, *for an identical clientele*, a set of cooperating caches will be unable to achieve a larger degree of sharing than the infinitely large proxy cache assumed in our study. Consequently, even in the presence of multiple cooperating caches, the network bandwidth usage on the access link will be similar to that for a single infinite proxy cache. The response times, however, will be different since cache hits in former could be either local hits or remote hits. Moreover, a set of cooperating caches will always outperform a set of non-cooperating proxies, each of which services only a subset of the total clientele.

- *Effect of cache replacement algorithms:* Our study assumed that proxies manage their caches using the LRU cache replacement policy (since LRU is the algorithm of choice in most proxies). Recent studies indicate that cache replacement policies that take into account locality, size, and access costs of web objects yield better hit ratios than LRU. Whether these improvements in hit ratios have a significant impact on the tail of the load distribution is a question that we plan to address as part of future work. We conjecture that the impact on the

tail is likely to be marginal, at best, since even an improvement in hit ratios is unlikely to have an impact on the poor locality exhibited by large objects (the locality of an object is a characteristic of the clients access patterns and not the cache replacement algorithm).

- *Effect of cache consistency mechanisms:* For simplicity, our study assumed a strong consistency mechanism that ensured the stale data is never present in the cache. In practice, many proxies provide weaker consistency guarantees and can occasionally supply stale data to clients. By delaying the fetching of updates from the server and servicing interim requests using stale data, weak consistency mechanisms can yield higher hit ratios and result in lower bandwidth requirements. In spite of these differences, we conjecture that, like strong consistency, weak consistency mechanisms will have a negligible impact on the tail of the load distribution. This is because the poor locality of large objects that governs the tail of the load distribution is unaffected by the cache consistency mechanism.

- *Effect of uncacheable objects:* The fraction of uncacheable objects in recent web workloads is reported to be 30% and is gorwing [11]. We believe that an increase in the fraction of dynamic and uncacheable objects is likely to exacerbate the heavy-tailed behavior of web loads and further reduce the effectiveness of proxy caching.

- *Effect of intelligent prefetching:* It is well known that prefetching can be an effective technique for reducing client response times and improving hit ratios [15]. Consequently, one possible avenue for future research is to design intelligent prefetching schemes that counter the heavy-tailed nature of web workloads by prefetching large objects. However, the gains due to such a prefetching policy may be offset by the bandwidth costs of prefetching large objects that exhibit poor locality. Hence, it is not a priori evident if such policies can be effective in reducing the tail of the load distribution.

5 Related Work

Due to the growing popularity of the world wide web, research on optimizing web performance has received increasing attention. Several research groups have focused on characterizing web workloads so as to gain a better understanding of these workloads [1, 3, 2, 8, 9, 10]. These studies have shown that web workloads exhibit significant short-term burstiness (self-similarity) and are heavy-tailed. The performance benefits of web proxies in reducing the average response time, network bandwidth usage and server load has been shown in [11, 7, 17, 18]. Most of these studies focus on the average load, or on issues such as the impact of low bandwidth connections and uncacheable objects (in contrast to our study which investigates the tail of the load distribution). Finally, other issues such as strong and weak consistency mechanisms [4, 21], cache replacement algorithms [14], hierarchical proxy caching [22] and prefetching techniques [15] have also received significant research attention; these efforts are complementary to our current study.

6 Concluding Remarks

In this paper, we examined the potential benefits of web proxy caches in improving the effective capacity of servers and networks. Since servers and networks are provisioned on the basis of a high percentile of the load, we focused on the effects of proxy caching on the tail of the load distribution. We found that, unlike their substantial impact on the average load, proxies have a diminished impact on the tail of the load distribution. For workloads considered in this study, compared to over a 50% reduction in the average load, the savings in network and server capacity was only 20-35% for the 99^{th} percentile of the load distribution. The exact reduction in the tail and the corresponding capacity savings depend on the percentile of the load distribution chosen for provisioning networks and servers—the higher the percentile, the smaller the savings. Further, the savings depend on the number of large one-time requests in the workload—the larger that number, smaller the savings. We also found that not all server workloads exhibit this behavior—server-side caches can be very effective for workloads that access a small number of static web pages, resulting in very high hit rates and substantial savings. Our study showed that proxies can be somewhat useful in smoothing out some of the burstiness in web workloads, the resulting workload continues to exhibit substantial burstiness and a heavy-tailed nature. We identified large objects with poor locality (one-time requests) to be the limiting factor that diminishes the impact of proxies on the tail of load distribution. We conclude that, while there are good reasons to deploy proxies to enhance user performance (due to the effectiveness in reduces the average response times), they are less effective in improving the capacities of networks and servers. This somewhat negative result calls for a rethinking of the cost-benefit tradeoffs of deploying proxies for capacity improvements in servers and networks and also for more research into techniques such as intelligent prefetching that can possibly counter these effects.

References

[1] M. Arlitt, R. Friedrich, and T. Jin. Workload Characterization of a Web Proxy in a Cable Modem Environment. Technical Report HPL-1999-48, Hwelett-Packard Laroratories, Palo Alto, CA, 1999.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of Infocom'99, New York, NY*, March 1999.

[3] M. Busari and C. Williamson. On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics. In *Proceedings of IEEE Infocom'01, Anchorage, Alaska*, April 2001.

[4] P. Cao and C. Liu. Maintaining Strong Cache Consistency in the World-Wide Web. In *Proceedings of the Seventeenth International Conference on Distributed Computing Systems*, May 1997.

[5] P. Chen and D. Patterson. Maximizing Performance in a Striped Disk Array. In *Proceedings of ACM SIGARCH Conference on Computer Architecture, Seattle, WA*, pages 322–331, May 1990.

[6] E. Cohen, B. Krishnamurthy, and J. Rexford. Improving end-to-end performance of the Web using server volumes and proxy filters. In *Proceedings ACM SIGCOMM'98, Vancouver, BC*, September 1998.

[7] Inktomi Corp. The Economics of Large-Scale Caching. www.inktomi.com/Tech/EconOfLargeScaleCache.html, 1997.

[8] M R. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.

[9] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of Change and Other Metrics: A Live Study of the World Wide Web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 147–158, December 1997.

[10] B. Duska, D. Marwood, and M. Feeley. The Measured Access Characteristics of World-Wide-Web Client Proxy Caches. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.

[11] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of Web Proxy Caching in Heterogeneous Environments. In *Proceedings of the IEEE Infocom'99, New York, NY*, March 1999.

[12] S. D. Gribble, G. Manku, D. Roselli, E. Brewer, T. Gibson, and E. Miller. Self-Similarity in File Systems. In *Proceedings of ACM SIGMETRICS '98, Madison, WI*, June 1998.

[13] M. Harchol-Balter. The Effect of Heavy-Tailed Job Size Distributions on Computer System Design. In *Proceedings of the Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics, Washington DC.*, June 1999.

[14] B. Krishnamurthy and C. Wills. Proxy Cache Coherency and Replacement—Towards a More Complete Picture. In *Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS)*, June 1999.

[15] T. Kroeger, D. Long, and J. Mogul. Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.

[16] W E. Leland, M. Taqqu, W. Willinger, and D V. Wilson. On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[17] C. Maltzahn, K. Richardson, and D. Grunwald. Performance Issues of Enterprise Level Web Proxies. In *Proceedings of the SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, June 1997.

[18] A. Rousskov and V. Soloviev. On Performance of Caching Proxies. In *Proceedings of ACM SIGMETRICS Conference, Madison, WI*, pages 272–273, June 1998.

[19] Seagate Technology, Inc. *ST-11200N SCSI-2 Fast (Barracuda 4) Specification*, August 1994.

[20] J. Thisquen. Seek Time Measurements. Technical report, Amdahl Peripheral Products Division, May 1988.

[21] J. Yin, L. Alvisi, M. Dahlin, and C. Lin. Volume Leases for Consistency in Large-Scale Systems. *IEEE Transactions on Knowledge and Data Engineering*, January 1999.

[22] H. Yu, L. Breslau, and S. Shenker. A Scalable Web Cache Consistency Architecture. In *Proceedings of the ACM SIGCOMM'99, Boston, MA*, September 1999.

Mohammad Raunak received his BS from North South University in Bangladesh and his M.S in Computer Science from the University of Massachusetts Amherst in 2000. He is currently on leave from the Ph.D program at the University of Massachusetts and is a Lecturer in Computer Science at the North South University in Bangladesh. His research interests are caching in the Internet, cache consistency, intelligent content delivery and wireless communications.

Prashant Shenoy received his B. Tech in Computer Science and Engineering from IIT Bombay, India in 1993, and his M.S. and Ph.D. in Computer Sciences from the University of Texas at Austin in 1994 and 1998, respectively. He is currently an Assistant Professor in the Department of Computer Science at the University of Massachusetts Amherst. His research interests are multimedia file systems, operating systems, computer networks and distributed systems. Over the past few years, he has been the recipient of the National Science Foundation CAREER award, the IBM Faculty Development Award, the Lilly Foundation Teaching Fellowship, and the UT Computer Science Best Dissertation Award. He is a member of the ACM and the IEEE.

Pawan Goyal received his B. Tech in Computer Science and Engineering from IIT Kanpur, India in 1992, and his M.S. and Ph.D. in Computer Sciences from the University of Texas at Austin in 1993 and 1997, respectively. He is currently a Member of Technical Staff at Ensim Corporation. Prior to joining Ensim, he was a Senior Member of Technical Staff at AT&T Research Labs. His research interests are computer networking, operating systems and multimedia systems.

Krithi Ramamritham received the Ph.D. in Computer Science from the University of Utah and then joined the University of Massachusetts. He is currently on leave from the University of Massachusetts and is the Verifone Chair Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Bombay. His interests span the areas of real-time systems, transaction processing in database systems, real-time databases systems and the web. He is a Fellow of the IEEE and a Fellow of the ACM.

Purushottam Kulkarni received his B.E. in Computer Science from University of Pune, India in 1997, and his M.S. in Computer Science from the University of Minnesota, Duluth in 2000. Before joining University of Minnesota he worked as a Software Engineer at Tata Technologies India Limited, India. He is currently a graduate student in the Department of Computer Science at the University of Massachusetts, Amherst. His research interests include scalable data dissemination over the web, computation dissemination, performance evaluation and simulation models for web caching environments.