Implications of Proxy Caching for Provisioning Networks and Servers*

Mohammad S. Raunak, **Prashant Shenoy**, **Pawan Goyal**‡and **Krithi Ramamritham**†

Department of Computer Science,
University of Massachusetts,
Amherst, MA 01003
{raunak,shenoy,krithi}@cs.umass.edu

‡Ensim Corporation,
1215 Terra Bella Ave,
Mountain View, CA 94043
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 percentile of the load distribution chosen for provisioning networks and servers—the higher the percentile, the smaller the savings. In particular, compared to over a 50% reduction in the average load, the savings in network and server capacity is 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 the poor locality of 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 load, they are less effective in improving the capacities of networks and servers.*

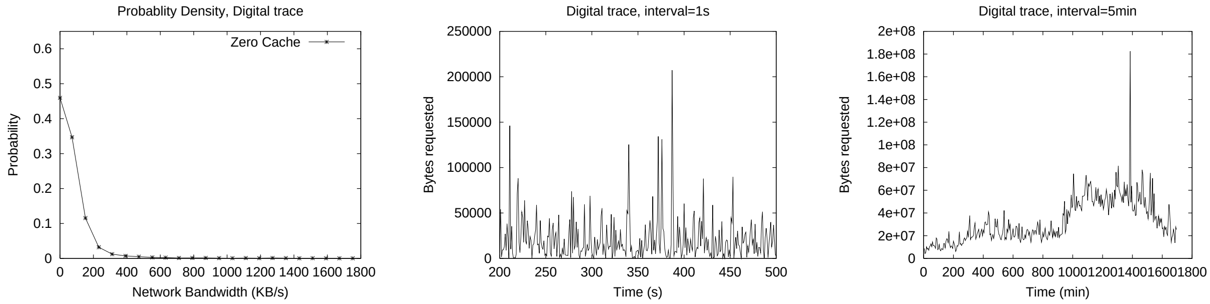
1 Introduction

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 [29]. 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 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.

*This research was supported in part by Intel Corporation, National Science Foundation grant no. ANI 9977635, and the University of Massachusetts.

†Currently on leave at the Dept. of Computer Science and Engg, Indian Institute of Technology, Powai, Bombay.



(a) (b) (c)

Figure 1: Heavy-tailed and bursty nature of web workloads. Figure (a) shows that the distribution of network bandwidth usage is heavy tailed. Figures (b) and (c) demonstrate the bursty nature of web workloads over time scales of one second and five minutes.

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 [9]. 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 [1, 8, 15, 23]. In contrast, servers and networks are typically provisioned based on a high percentile of the load (rather than the average load) [18]. 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. By relying on existing performance studies, these organizations and network service providers have implicitly assumed that *a reduction in the average load results in a corresponding reduction in a high percentile of the load.* In fact, white papers on commercially available proxies explicitly make such assumptions while arguing for the benefits of proxies in increasing the capacities of servers and networks.[1] In contrast, recent studies that have characterized web workloads [1, 2, 4, 11] 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 request sizes as well as the resulting network bandwidth usage have a Pareto distribution with long tails (see Figure 1(a)) [1, 11]. 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(b) and (c)) [11, 16]. 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

[1] For instance, a white paper on the cost benefits of large-scale proxy caching argues that an average byte hit ratio of 40% yields a corresponding 40% savings in network capacity [9]. Since commercial organizations rarely provide details of their experimental evaluations for proprietary reasons, we were unable to determine the basis for these conclusions.

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 [4, 13], 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 not amenable to caching since they are regenerated on every user request. Studies reveal that the fraction of dynamic and uncacheable objects in web workloads is around 30% and increasing with time [15]. 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 proxy caches 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 heavy-tailed nature 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. The savings in network capacity for the 99^{th} percentile is 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).

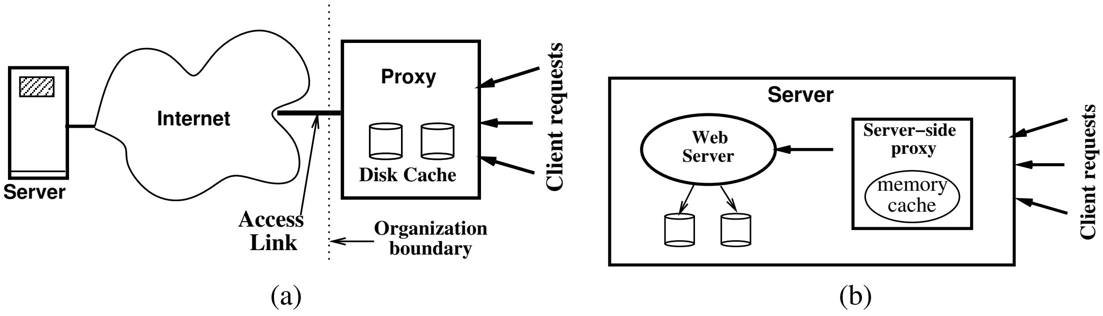


Figure 2: Simulation environments for (a) a proxy near an access link and (b) a server-side proxy.

  - Server capacity is usually provisioned on the basis of the *server response time*. Again, we find that proxies 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 proxy caching and the resulting savings in server capacity is 15-20% (as compared to a 30-50% reduction in average response time).

- 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 the locality exhibited by web workloads is likely to reduce during intense bursts. This is because bursts are caused by concurrent requests for large objects, many of which exhibit poor locality and hence do not benefit from caching. Thus, poor locality exhibited by large objects is a key factor that diminishes the impact of proxy caching on the tails 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 proxy 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 strong consistency mechanism—one where cached data is always invalidated upon modification and the proxy never provides stale data to clients [6, 32]. In practice, however, many proxies employ cache consistency mechanisms that can occasionally provide stale data [17]; 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 [7]; the model has been validated against a real disk in [30]. We choose the Seagate Barracuda 4LP disk [27] 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 the Internet is modeled as a single bit pipe of capacity b_{int}. The rationale for doing so is the inherent presence of a single congested link on the path from the proxy to the server that usually governs the bandwidth received by a connection; we model the presence of such a link by assuming that the bandwidth received by the connection on that link is b_{int}. In such a scenario, if there are n active 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.[2] 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 [15]. 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 [15]).

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 $b_{link} = 45$ 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.

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

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

Table 1: Characteristics of trace workloads.

Trace	Type	Duration (s)	#Requests	#Unique Objects	#Unique Clients	Bit rate (KB/s)	Request rate (req/s)	Avg Req Size (KB)
Digital	Proxy	101871.8	1141412	385607	5488	120.82	11.2040	9.68
Berkeley	Proxy	154618.1	965172	391461	3845	52.7	6.2422	7.34
NASA	server	2678370	1415318	13436	74149	9.78	0.5284	18.43
ClarkNet	server	604717	1500129	32159	73466	23.45	2.4807	9.45

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 [19], 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.[3] 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 [7]. As with the simulation environment for the proxy near an access link, we choose the Seagate Barracuda 4LP disk [27] 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 four different traces for our study—two proxy and two 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 two 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), whereas the Berkeley trace represents the workload at a proxy servicing dial-up clients in an educational institution (Berkeley's HomeIP dial-up service). As shown in Table 1, both traces span multiple days and consist of approximately a million requests each; these requests were generated by a population of several thousand clients.[4] Each record in the trace represents a

[3]Modern file systems use optimizations such as clustering to achieve near-contiguous allocation of large files [28]. Our simulator supports such optimizations; in such a scenario, each cluster consists of several contiguous disk blocks and is randomly placed on disk.

[4]The original traces are larger and span multiple weeks; due to memory constraints on our simulation machines, we restricted our simulations to the first few days of the original trace.

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 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). More recent trace workloads [15] have a larger fraction of such objects (around 30%) and this fraction is reported to be increasing (unfortunately these traces are not publicly available). 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 two server traces to evaluate the impact of caching on server response times. One represents a web server belonging to a network service provider (ClarkNet) and another that at a government institution (NASA). The traces were gathered over a period of 7 and 28 days, respectively, and consist of more than a million requests each (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 unit time. 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 three questions:

1. Do proxy caches help in reducing the tail of (a) the network load and (b) the server load imposed by web requests? That is, do proxies yield savings in network and server capacity?

2. Does deploying a proxy cache help smooth out the burstiness due to web traffic and have a stabilizing effect on the network? That is, does the resulting load become less bursty?

3. What are the factors that contribute to the heavy-tailed nature of the load distribution? In particular, what is the effect of burstiness on locality and how does it impact the tail of the load distribution?

In what follows, we consider each of these questions in turn. Sections 3.1 and 3.2 address the first question, focusing on network provisioning and server provisioning, respectively. Sections 3.3.1 and 3.3.2 address the second and third questions, respectively.

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 and the cumulative distribution function of the network bandwidth usage. The figure shows that the network bandwidth usage has a heavy-tailed nature in the absence of a proxy cache.[5] More importantly, the network bandwidth usage exhibits a heavy tailed nature even in 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 heavy-tailed nature 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 (see Figure 4). The figure shows 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 5 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 5(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).

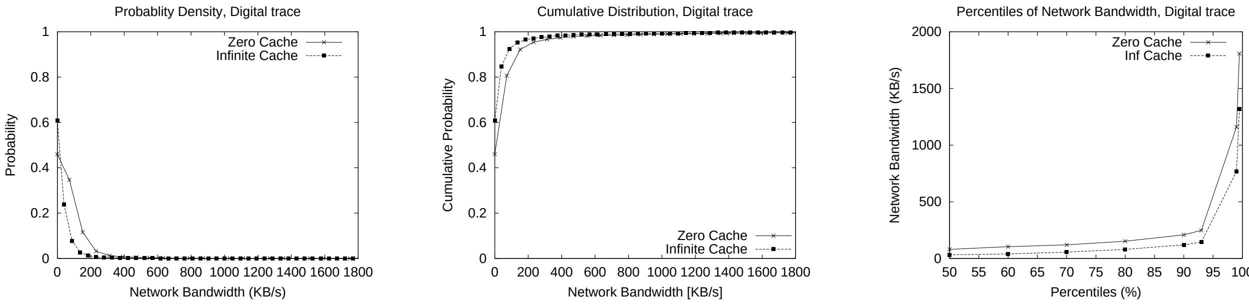
Together Figures 3, 4 and 5 lead us to the following conclusions.

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.*

3.1.2 Effect on Client Response Times

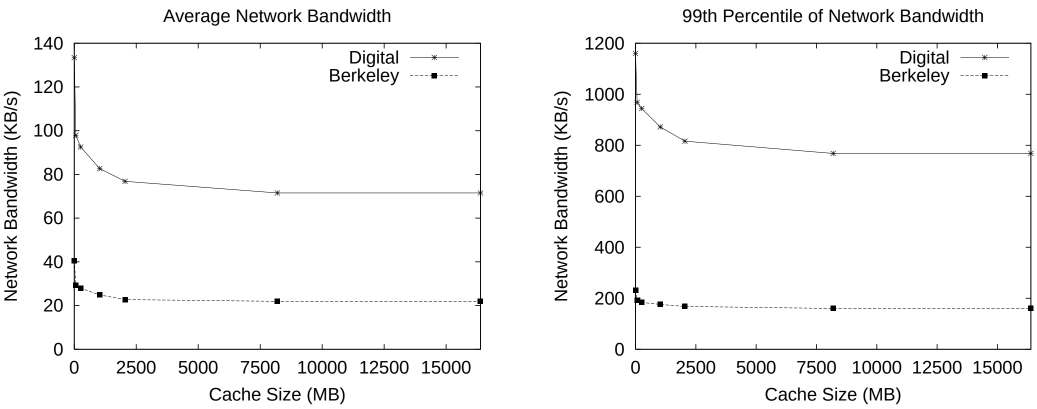
Since a network may be provisioned on the basis of the response time (rather than bandwidth usage), in what follows, we examine the impact of proxy caching on the client response times. To do so, as before, we varied the cache size from zero to infinity and computed the distribution of the client response 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. 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. Since the cache hit ratio increases with increasing cache sizes, the average response time shows a corresponding reduction. The 99^{th}

[5] The figure shows only the initial portion of the tail; the actual tail is significantly longer. Note also that, due to its bursty nature, the workload has several silence periods where no requests arrive—these intervals correspond to zero bandwidth usage in the figure.



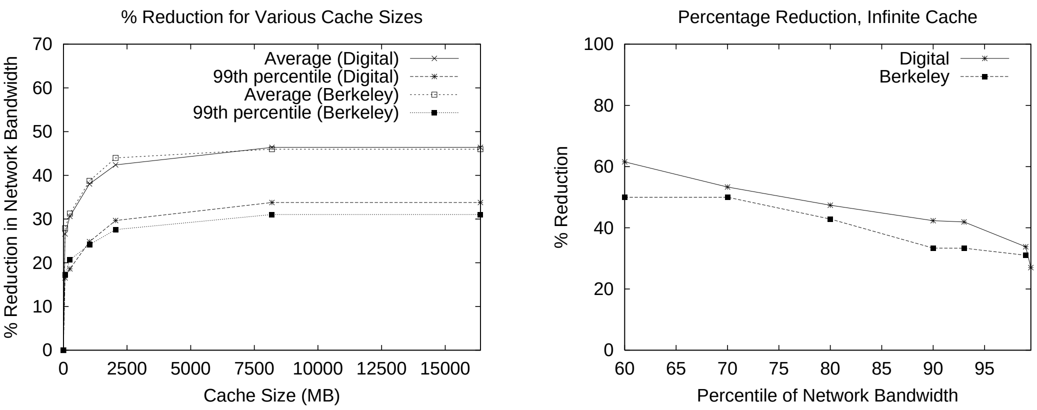
(a) Probability density function (b) Cumulative Distribution function (c) Percentiles of bandwidth usage

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



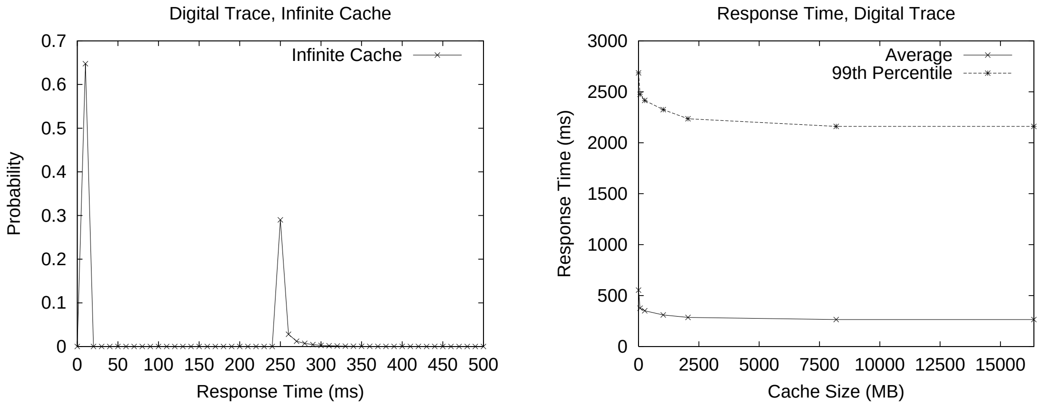
(a) Average network bandwidth (b) 99^{th} percentile of the bandwidth

Figure 4: Effect of cache size on network bandwidth usage



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

Figure 5: Reduction in network bandwidth usage due to caching



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

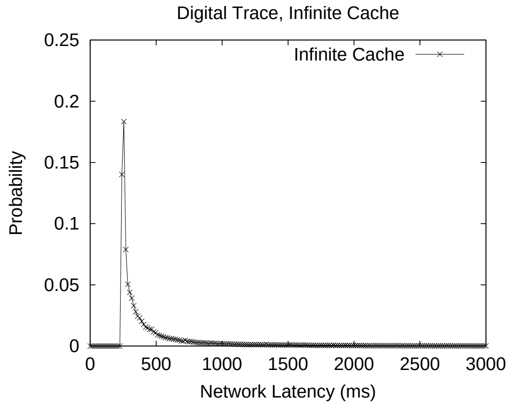
Figure 6: Impact of caching on response times. Figure (a) plots the probability density function of the response time while figure (b) shows the variation in the average and 99^{th} percentile of the response time with increasing cache sizes.

percentile of the response time also reduces 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).

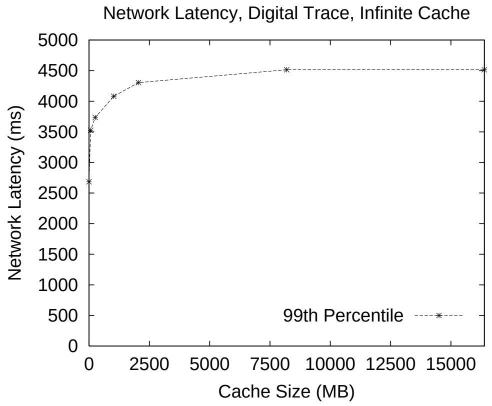
Figure 7(a) plots the probability density function of the network latency; as expected, the distribution consists of a single peak (which corresponds to the peak due to cache misses in the bimodal response time distribution). Figure 7(b) plots the variation in the 99^{th} percentile of the network latency 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:* The tail of the network latency distribution is governed by downloads of large objects. As will be shown in 3.3.1, many large objects exhibit poor locality (i.e., are accessed only once and never again). Increasing the cache size causes an increasing fraction of small objects (which exhibit better locality) to be serviced as cache hits. The resulting network workload consists of an increasing fraction of large objects, causing the 99^{th} percentile of request size distribution to increase with cache size. 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. The resulting decrease in latency is more than offset by

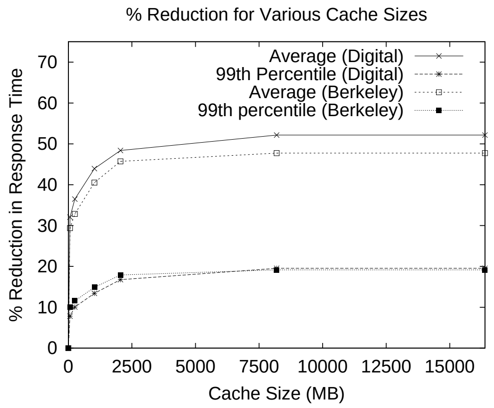


(a) Probability density function

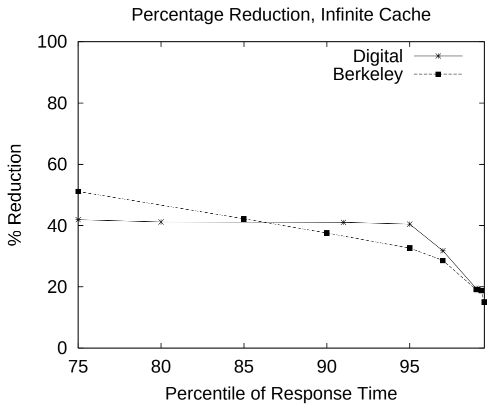


(b) Network latency for different cache sizes

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

the increase in latency due to larger object sizes (as explained above), causing the tail of the latency to show an overall increase.[6]

Due to the above reasons, proxies are unable to reduce the tail of the network latency; hence, the reduction in the tail of the client response times is small. (Although the tail of the network latency increases with cache sizes, the tail of the response time distribution shows a net decrease due to the improvement in hit ratios.) 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 *Networks that are 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 compared to over a 50% improvement in the average response time). This is because the tail of the response time distribution is governed by large objects that exhibit poor locality and proxies are ineffective at improving response times for such objects (since they always result in cache misses).*

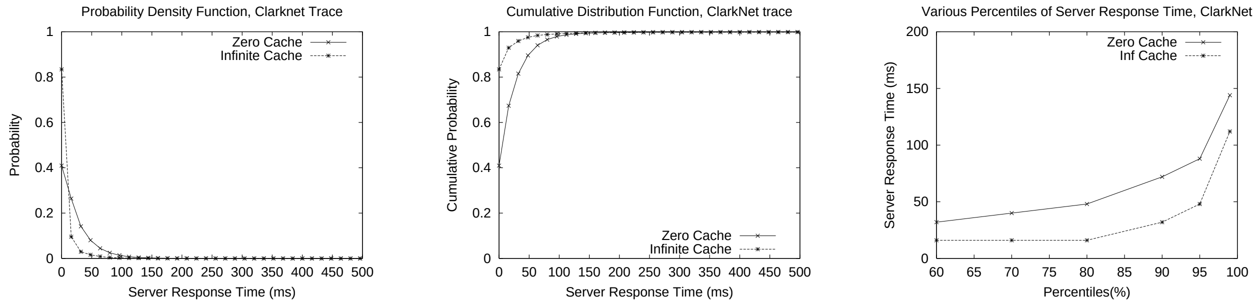
3.2 Implications of Proxy Caching for Server Provisioning

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 heavy-tailed nature even in the presence of an infinite cache (again indicating a reduced impact of caching on the tail of the response time distribution). Figure 10 plots the variation in the average response time and the 99^{th} percentile of the response time for different cache sizes. As shown in the figure, both the average and the 99^{th} percentile of the response time decrease with increasing cache sizes, but the percentage reduction in the 99^{th} percentile is smaller than that in the average response time. Figure 11 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 11(b)). (Observe that Figures 9 through 11 are the *server response time* counterparts of Figures 3 through 5 for network bandwidth usage.)

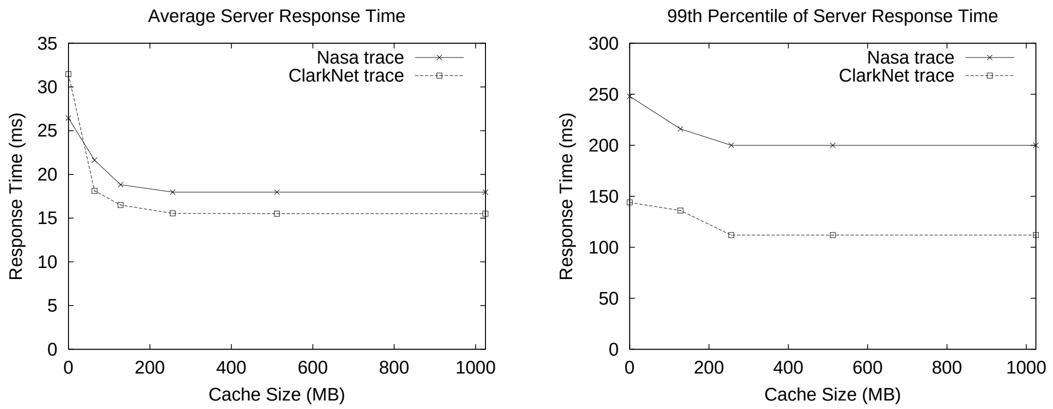
Thus, we have the following result:

[6]To verify this hypothesis, we simulated an access link with a capacity of only 128 KB/s, causing the link to be heavily loaded. The increase in bandwidth available per connection due to caching is more pronounced for such saturated links. Consequently, the decrease in latency due to the improved bandwidth availability was larger than the increase in latency due to larger object sizes, causing the tail of the latency to show an overall *decrease*.



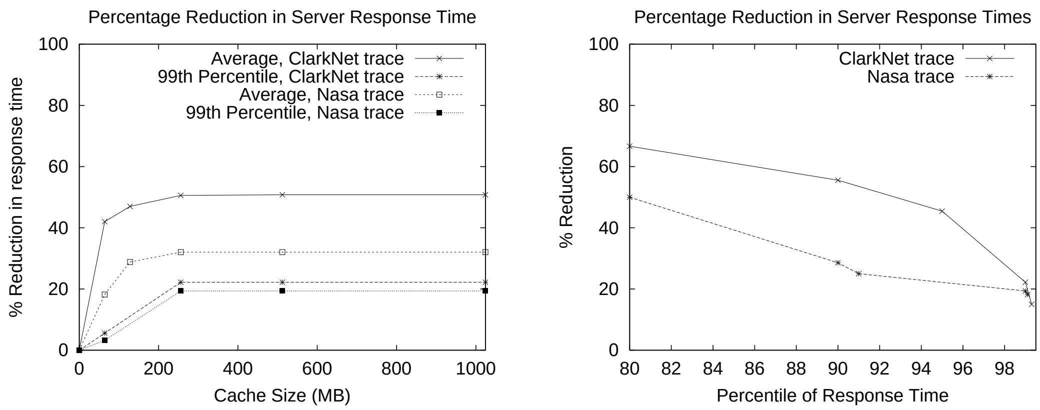
(a) Probability density function (b) Cumulative Distribution function (c) Percentiles of response time

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



(a) Average response time (b) 99^{th} percentile of the response time

Figure 10: Effect of cache size on server response times.



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

Figure 11: Reduction in Server Response Times due to Caching

Table 2: 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

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).*

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.

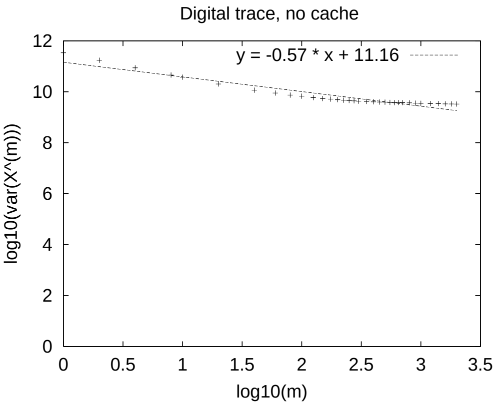
3.3.1 Effect of Caching on Burstiness

Past studies have shown that web workloads exhibit burstiness at multiple time scales [11, 16]; 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 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[7] 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).

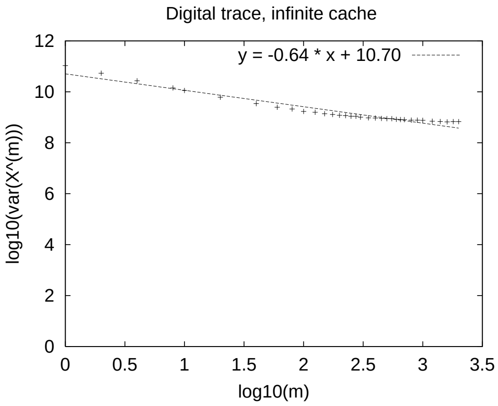
We use two different metrics to quantify the burstiness of a workload: (i) the *Hurst Parameter* H, which is a measure of the self-similarity (i.e., burstiness) exhibited by a workload [11]; and (ii) the heavy-tailed nature of the workload as defined by the *Pareto* distribution: $P(X > x) \approx x^{-\alpha}$, 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

[7]Doing so requires Layer 4 switching algorithms in the routers to intercept web requests and forward them to a proxy. We do not consider the costs of Layer 4 switching algorithms in this paper, and focus only on the performance benefits of deploying proxy caches near network routers.



(a) No cache



(b) Infinite cache

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

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.

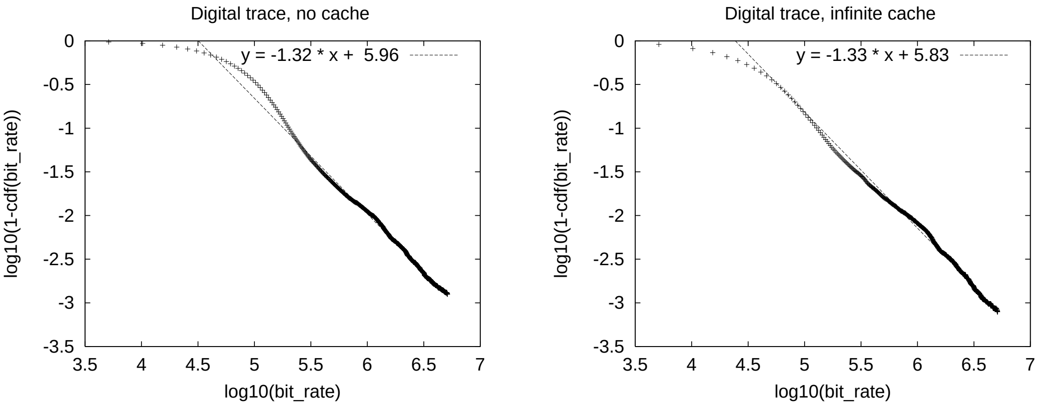
To compute the Hurst parameter, we assume that the bit rate of the workload over one second intervals represents a time series $X = (X_t; t = 1, 2, 3, ...)$. We then compute the m-aggregated series $X^{(m)} = (X_k^{(m)}; k = 1, 2, 3...)$ by summing the original series X over non-overlapping blocks of size m. The variance of $X^{(m)}$ is then plotted against m on a log-log plot, yielding the so-called *variance-time* plot [11, 16, 22]. A straight line with slope $-\beta$ is indicative of self-similarity and the Hurst parameter H is given by

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

The workload is said to self-similar if $0.5 < H < 1$, with larger values of H indicating an increase in burstiness. (see [11, 16, 22] 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 2 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 indeed reduce the burstiness in web workloads.[8] However, the Hurst parameter of the resulting workload continues to be in the range $0.5 < H < 1$, indicating that the workload continues to be self-similar with 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 [11]. 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

[8]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.



(a) No Cache (b) Infinite Cache

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

of the mass is present in the tail, making the distribution more heavy-tailed (see [11] 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 2. Together, Figure 13 and Table 2 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:

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 (self-similarity) 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

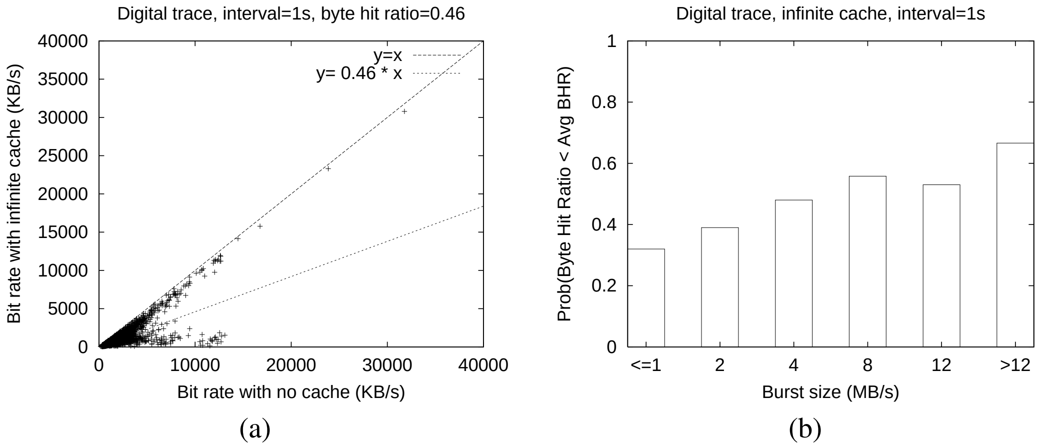


Figure 14: Effect of burstiness on the locality.

(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 the average size of a requested object increases sharply with burst size and this increase is significantly larger than that in the request rate (the average request size grows by a factor of 40, while the request rate only triples). This indicates that intense bursts are more likely to be caused by concurrent requests for large objects, rather than a large number of requests for small objects. Since large objects often exhibit poor locality,[9] the locality decreases with increasing burst size.

Thus, we have the following result:

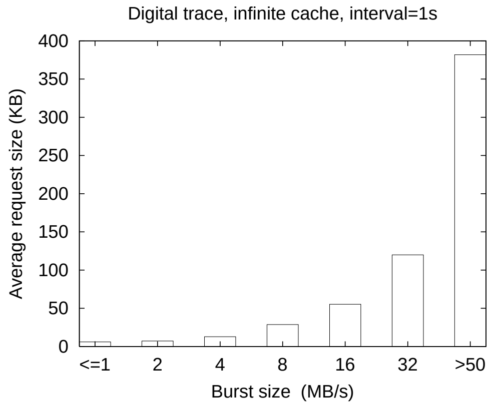
Result 5 *Intense bursts are more likely to be caused by concurrent requests for large objects (rather than a large number of requests for small objects). Large objects often exhibit poor locality, causing the locality exhibited by the workload to decrease during periods of intense bursts. Since bursts govern the tails of response time and bandwidth distributions, this reduced locality diminishes the impact of caching on reducing the tails of the distribution.*

4 Additional Considerations and Directions for Future Work

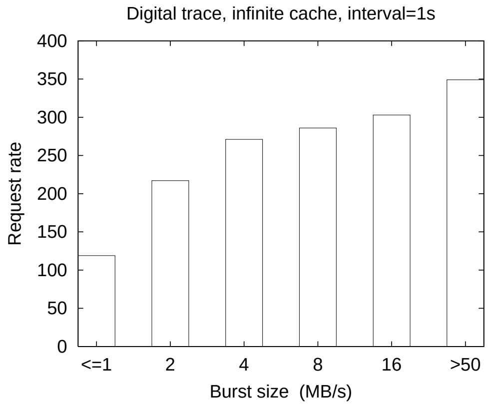
In this section, we consider the possible effects of the assumptions made in our simulation study on our results and discuss possible directions for future work.

- *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 a given workload, 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

[9]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 [4]. 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.

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 [5]. 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 [17]. 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 diminished 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 dynamic objects:* Since the traces used in our study were gathered in 1995-96, they contain only a small fraction ($< 7\%$) of dynamic and uncacheable objects. The fraction of such objects in more recent web workloads is reported to be at least 30% [15]. 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. We plan to validate this hypothesis when more recent trace workloads become publicly available.

- *Effect of intelligent prefetching:* It is well known that prefetching can be an effective technique for reducing client response times and improving hit ratios [10, 21, 25]. 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 (which primarily contribute to the long tails of web workloads). 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, 2, 4, 11, 12, 13]. 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 [15, 9, 23, 26]. 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 [6, 17, 32], cache replacement algorithms [5, 20], hierarchical proxy caching [29, 31, 33], prefetching techniques [14, 21, 25, 34] and protocol issues [3, 24] 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. 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. In particular, compared to over a 50% reduction in the average load, the savings in network and server capacity is only 20-35% for the 99^{th} percentile of the load distribution. We also found that while 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 the poor locality of large objects 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] M. Arlitt and C. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proceedings of the SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1996. http://www.cs.usask.ca/projects/discus/discus_reports.html.

[3] G Banga and J C. Mogul. Scalable Kernel Performance for Internet Servers under Reslistic Loads. In *Proceedings of the USENIX Annual Conference, New Orleans, LA*, June 1998.

[4] 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.

[5] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems, Montrey, CA*, December 1997.

[6] 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.

[7] 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.

[8] 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.

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

[10] M. Crovella and P. Barford. The Network Effects of Prefetching. In *Proceedings of IEEE INFOCOM Confernce, San Francisco, CA*, 1998.

[11] 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.

[12] 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.

[13] 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.

[14] L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web prefetching between low-bandwidth clients and proxies: potential and performance. In *Proceedings of ACM SIGMERTICS'99 conference, Atlanta, GA*, pages 178–187, May 1999.

[15] 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.

[16] 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.

[17] J. Gwertzman and M. Seltzer. World-Wide Web Cache Consistency. In *Proceedings of the 1996 USENIX Technical Conference*, January 1996.

[18] 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.

[19] J. L. Hennesy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] E. Nahum, T. Barzilai, and D. Kandlur. Performance issues in WWW servers. In *Proceedings of ACM SIGMERTICS'99 conference, Atlanta, GA*, pages 216–217, May 1999.

[25] V. Padmanabhan and J. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. In *Proceedings of the ACM SIGCOMM '96 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 22–36, July 1996.

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

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

[28] M. Seltzer, K. Smith, H. Balakrishnan, J. Chang, S. McMains, and V. Padmanabhan. File System Logging Versus Clustering: A Performance Comparison. In *Proceedings of the Winter 1995 USENIX Conference*, pages 249–264, January 1995.

[29] R. Tewari, M. Dahlin, H M. Vin, and J. Kay. Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet. In *Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS)*, June 1999.

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

[31] J. Yin, L. Alvisi, M. Dahlin, and C. Lin. Hierarchical Cache Consistency in a WAN. In *Proceedings of the Usenix Symposium on Internet Technologies (USEITS'99), Boulder, CO*, October 1999.

[32] 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.

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

[34] L. Zhang, S. Floyd, and V. Jacobson. Adaptive Web Caching. In *Proceedings of the NLANR Web Cache Workshop*, June 1997. http://ircache.nlanr.net/Cache/Workshop97/.