Workload Characterization in Web Caching Hierarchies

Guangwei Bai Carey Williamson
Department of Computer Science, University of Calgary
{bai,carey}@cpsc.ucalgary.ca

Abstract

This paper uses trace-driven simulation and synthetic Web workloads to study the request arrival process at each level of a simple Web proxy caching hierarchy. The simulation results show that a Web cache reduces both the peak and the mean request arrival rate for Web traffic workloads. However, the variability of the request arrival process may either increase, decrease, or remain the same after the cache, depending on the input arrival process and the configuration of the cache. If the input request arrival process is self-similar, then the filtered request arrival process remains self-similar, though with reduced mean. Furthermore, the superposition of Web request streams from multiple child caches results in a bursty aggregate request stream. Finally, we find that a Gamma distribution provides a flexible means of modeling the request arrival count distribution in hierarchical Web caching architectures.

1. Introduction

As the tremendous growth of the World Wide Web continues, multi-level Web proxy caching systems are being used to improve the performance of the Internet [15, 16, 18, 23]. Web caching proxies help by reducing Web server load, Internet bandwidth consumption, and the round-trip delays associated with Web object retrieval. In many cases, users perceive improved response times for document downloads.

The primary purpose of a proxy cache is to reduce the number of client requests that traverse the Internet to the origin Web server. Obviously, the presence of a Web proxy cache changes the Web workload seen by a Web server, since many requests are filtered (removed) from the Web server request stream when they are satisfied at the proxy. The same argument applies between the levels of caches in a multi-level Web proxy caching hierarchy.

The "filtering" effect of the cache manifests itself in two distinctly different ways. First, it removes many requests for highly-popular Web objects, making the object popularity profile seen by a higher-level cache or the origin server distinctly un-Zipf-like [23]. Second, it changes the structural characteristics of the request arrival process, since many of the requests during peak bursty periods are satisfied by the first level of cache. We refer to the changes in the document popularity profile as *frequency-domain* cache filter effects, and the changes in the request arrival process as *time-domain* cache filter effects. The frequency-domain effect has been fairly well-studied in the literature [10, 12, 14, 23], while the time-domain effect has received little attention [5]. It is the latter (time-domain) effect that is the focus of this paper.

The precise manifestation of the cache filter effect is highly dependent upon the architecture of the Web proxy caching system, the cache size used, and the cache replacement policy [5, 10, 23]. Since the goal in Web proxy caching is to improve overall "system-level" performance, it is important to design an appropriate caching system architecture. For this purpose, knowledge of Web workloads and a thorough understanding of cache filter effects are indispensable. This knowledge will provide valuable insight into caching system design and the mechanisms for efficient cooperation between caches at different levels.

The research questions addressed in this paper are:

- How significantly are the mean and the variance of the Web request arrival process transformed by the presence of a proxy cache?
- How sensitive are the cache filter effects to the characteristics of the input request workload?
- How can we model the aggregate Web request streams in a multi-level Web proxy caching hierarchy?

We address these questions using trace-driven simulations of a multi-level Web proxy caching hierarchy. The simulation experiments quantify the filter effects of a Web cache on the request arrival process, for synthetically-generated aggregate Web client workloads. For simplicity, we consider only a two-level Web proxy caching hierarchy, as shown in Figure 1. Assuming that λ_1 and λ_2 represent the Web request arrival processes (from clients) entering the child-level caches, we are interested in the filtered arrival processes λ_1' and λ_2' after these caches. We are also interested in how they multiplex to form λ_3, which itself is transformed into λ_3' before entering the Internet.

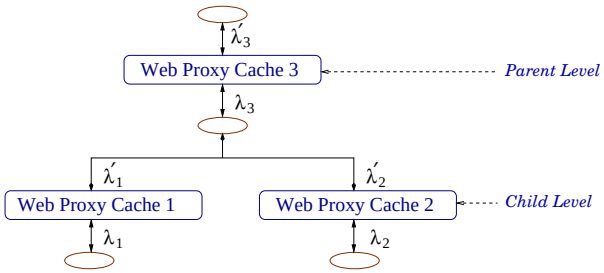


Figure 1. Example of Web caching hierarchy

The simulation experiments show the obvious result that a Web cache reduces the mean request arrival rate for Web traffic workloads. However, the variability (burstiness) of the request arrival process may either increase, decrease, or remain the same after the cache, depending on the input arrival process and the configuration of the cache. The study also demonstrates that the superposition of Web request streams from multiple child caches in a Web proxy caching hierarchy does not result in a smoothing of traffic. Rather, multiplexing bursty request streams tends to produce a bursty aggregate stream. Finally, we find that a Gamma distribution provides a flexible means of characterizing Web workloads in Web caching hierarchies.

The remainder of this paper is organized as follows. Section 2 discusses background information on Web proxy caching and related work on Web workload characterization and cache filter effects. Section 3 describes the experimental methodology for our study, and the synthetic Web proxy workloads used. Section 4 presents simulation results for Web cache filter effects on the request arrival process. Section 5 focuses on the superposition of self-similar Web workload streams in a multi-level caching hierarchy, and the modeling of aggregate Web workload in the time-domain. Finally, Section 6 concludes the paper and suggests directions for future research.

2. Background and Related Work

2.1. Web Proxy Caching

Web proxy caching is a technique used for improving Web performance on the Internet. Web proxy caches are located between Web clients (browsers) and Web servers. Proxies accept client requests and forward them to Web servers only as necessary. When a requested document is returned by a Web server, the proxy sends the document to the client and stores a copy of the document in its local cache, in the hope that the proxy can satisfy future client requests for the same document without contacting the origin server, thus reducing user-perceived response time.

Caching effectiveness is traditionally measured by two quantities: the (document) *hit ratio* is the percentage of the total requests that are satisfied directly by documents stored in the cache; and the *byte hit ratio* is the percentage of the total requested Web content bytes that are satisfied directly by documents stored in the cache. Both metrics are required since Web objects vary significantly in size. Other metrics such as user-perceived response time are dependent upon the hit ratio and the byte hit ratio, as well as network bandwidth, round-trip delay, and server load.

To enhance the performance of Web caching, multi-level Web caching hierarchies have recently received increasing research attention [10, 15, 18, 23]. In a hierarchical configuration, proxies at or near the end-user constitute the lowest level of the hierarchy, often with sibling-sibling relationships with one another. The lowest level proxies may have a child-parent relationship to a higher level proxy, usually a (geographically) regional proxy [23]. A regional proxy can in turn connect to a higher level proxy, such as a national proxy [18]. A request that cannot be satisfied by one proxy cache can be sent to a nearby sibling or to the parent using an Inter-Cache Protocol. Contacting the origin server to obtain the document serves as the last resort [23].

2.2. Web Workload Characterization

Web workload characterization studies have focussed on Web client [7], Web server [3], and Web proxy workload characteristics [2, 5, 18]. Common workload characteristics observed include a high degree of *one-time referencing*, a *Zipf-like document popularity distribution*, *heavy-tailed file and transfer size distributions*, and a *temporal locality property* in the document referencing behaviour [3, 9, 18]. Among these characteristics, the slope of the Zipf-like document popularity distribution is most relevant to Web caching performance [9]. Zipf's law expresses a power-law relationship between the popularity P of an item (i.e., its frequency of reference) and its rank r (i.e., relative rank among the referenced items, based on frequency of reference). This relationship is of the form $P = c/r^{\beta}$, where c is a constant, and β is often close to 1. When the slope is steep, requests are highly concentrated on a small subset of the Web content, and caching works well.

2.3. Web Proxy Cache Performance

Several recent research papers have explored the relationships between Web workload characteristics and Web proxy caching performance, particularly in caching hierarchies [10, 12, 14]. However, most of this research focuses on the frequency-domain aspect of the Web cache filter effect. For example, Doyle *et al.* [14] refer to this as the "trickle down" effect, and conduct a detailed simulation study to quantify its impact. Che *et al.* [12] pro-

pose a frequency-based caching hierarchy, where the lower-level cache handles requests for high-frequency items, and the higher-level cache handles requests for low-frequency items. Busari and Williamson [10] propose a "heterogeneous" Web proxy caching hierarchy that uses different caching policies at different levels of a caching hierarchy.

Few papers explicitly address the structural changes in the request arrival process in multi-level Web caching hierarchies. Our own previous work [5] studied time-domain cache filter effects using an empirical Web proxy workload, but only for a single-level Web proxy cache. This current paper extends our work to caching hierarchies, and generalizes our results to a broader set of (synthetically-generated) Web workload characteristics.

3. Experimental Methodology

Our experimental methodology has two main steps. First, we generate a set of synthetic Web proxy workloads to use in our study. We validate these workloads to ensure that they have the intended workload characteristics, and are similar to empirical Web proxy workloads used in our earlier studies [5, 9, 10, 18, 23]. Second, we conduct a set of trace-driven simulation experiments, using an application-level Web proxy caching simulator and the generated workloads. The output "miss" streams from the Web proxy cache simulations are used to quantify the filter effect of the cache, as a function of cache size and cache replacement policy.

The workload generation and cache simulation steps are described in more detail in Section 3.1 and 3.2, respectively.

3.1. Workload Generation

There are two reasons for the use of synthetic Web proxy workloads in our study, rather than empirical workloads. First, synthetic workload generation offers greater control over workload characteristics, and allows us to study the sensitivity of our results to particular workload characteristics. Second, it allows us to generate traces that are as long or as short as needed for our study, without having to worry about non-stationary behaviour, which can be significant in empirical Web proxy workloads [5].

In the workload generation step, there are two workload parameters of interest: Zipf slope, and request arrival process. The Zipf slope refers to slope of the Zipf-like document popularity profile in the input Web workload. This slope affects the magnitude of the Web cache filtering effect, since a steep Zipf slope tends to produce a high cache hit ratio, while a flat Zipf slope does not. The request arrival process refers to the timestamps generated for the Web client requests. We consider three different arrival processes: an (unrealistic) deterministic arrival process that is simple to analyze; an (unrealistic) Poisson arrival process that is also easy to analyze; and a self-similar arrival process that represents a realistic Web request arrival process.

Table 1. Web workload characteristics

Item	Trace 1	Trace 2
Total Requests	837,972	893,943
Total Documents	299,513	300,000
Unique Documents	35.74%	33.56%
One-timer Documents	209,504	209,693
One-timers	69.95%	69.90%
Zipf Slope	**-0.75**	**-0.80**
Total Bytes of Docs (MB)	3,706	3,712
Smallest Doc Size (bytes)	0	32
Largest Doc Size (MB)	39.475	39.475
Mean Doc Size (bytes)	12,977	12,976
Total Transferred Bytes (MB)	9,304	8,960
Mean Transfer Size (bytes)	11,370	10,264

In our work, the *ProWGen* (Proxy Workload Generation) [11] tool is used to synthesize Web proxy workloads. ProWGen captures the salient characteristics of Web proxy workloads: one-time referencing, Zipf-like document popularity, heavy-tailed file size distribution, and temporal locality. These characteristics affect Web proxy cache performance [9, 11], and are easy to analyze using the *WebTraff* tool [19]. By design, the two synthetic workloads differ in the Zipf-like document popularity profile, which influences the cache hit ratio [8, 9, 22]. Table 1 summarizes the characteristics of the synthetic traces used.

For each of the synthetic traces shown in Table 1, three arrival-time time series were generated, using deterministic, Poisson and self-similar request arrival processes, respectively. In fact, two instances of the self-similar case are studied, so that we can generalize our traffic characterization and modeling results. Note that the generation of the traffic arrival process (i.e., the timestamps on the Web document requests) is independent of the techniques used to generate the Web document requests (i.e., ProWGen's file popularity and temporal locality models).

The resulting synthetic workloads are used to investigate cache filter effects in a two-level Web caching hierarchy.

3.2. Web Proxy Cache Simulation

In the Web cache simulation step, two experimental factors are used: cache size, and cache replacement policy. The cache size determines the maximum number of Web content bytes that can be held in the cache at one time. The cache replacement policy determines what objects to remove from the cache when more space is needed to store an incoming object. Five cache replacement policies are considered: removing objects at random (RAND), remov-

ing objects in the order in which they arrived (First-In-First-Out, FIFO), removing objects based on recency of use (Least-Recently-Used, LRU), removing unpopular objects (Least-Frequently-Used, LFU), and removing large objects (Greedy-Dual-Size, GDS).

In our simulation experiments, the synthetic Web workload (a timestamped series of Web document requests) is provided as input to the Web proxy cache simulator. The network topology modeled is shown in Figure 1. The simulator generates as output the cache hit ratio for the experiment, and a timestamped series indicating the requests that result in cache misses. The latter output constitutes the *filtered request stream* used for traffic analysis.

The experiments use cache hit ratio and byte hit ratio as the primary performance metrics for cache performance, and the mean and variance of the request arrival process as the primary means of characterizing cache filter effects. The caching simulation results follow in Section 4.

4. Simulation Results: Cache Filter Effects

This section focuses on analysis and understanding of the cache filter effects in the time-domain. Section 4.1 makes general observations about cache filter effects, while the effects of cache configuration parameters (Section 4.2) and input workload characteristics (Section 4.3) follow.

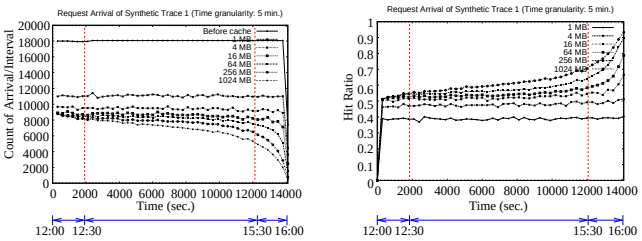
4.1. General Observations

The first experiment is designed to provide an intuitive understanding of the filter effect of a cache, and a qualitative assessment of its impact. For this experiment, a single workload (Trace 1) is provided as input to the Web proxy cache simulator. For simplicity, we assume a deterministic request arrival process, with an (arbitrary) average rate of 60 requests per second. This represents a trace duration of about 4 hours for Trace 1.

Figure 2 illustrates the general impacts of a proxy cache on the Web workload, using two time series plots. Figure 2(a) shows the request arrival count process for the original and filtered request streams, with request counts cumulated over 5 minute intervals. Figure 2(b) shows the corresponding cache hit ratio results for this trace.

Figure 2(a) clearly shows that the presence of the Web cache reduces both the peak and the mean rate of the request arrival process. The larger the cache size, the more pronounced the filter effect. These results are as expected.

Figure 2(b) shows the average document hit ratio in the cache for different cache sizes. These results are plotted using the average cache hit ratio over each 5 minute interval of the trace. The cache is initially empty at the starting point of the trace, and proceeds to fill as the simulation progresses, invoking the cache replacement policy when



(a) Time Series (b) Hit Ratio

Figure 2. Illustration of cache filter effects

needed to manage the contents of the cache. As expected, the cache hit ratio increases with cache size.

There are non-stationary behaviours evident in these two graphs, particularly for large cache sizes. A brief warmup period occurs initially before the cache hit ratio begins to stabilize. Some "end effects" are also visible, wherein the cache hit ratio increases at the end of the trace, since most of the Web content (particularly the popular content) fits in the cache when the cache size is large. The increasing hit ratio produces a corresponding drop in the request rate of the filtered request stream. For modest cache sizes, however, the hit ratio and the filtered request arrival process appear to be stationary for most of the trace (e.g., 3-hour portion between the vertical lines). We focus only on the stationary portion in our subsequent analyses.

4.2. Effect of Cache Configuration Parameters

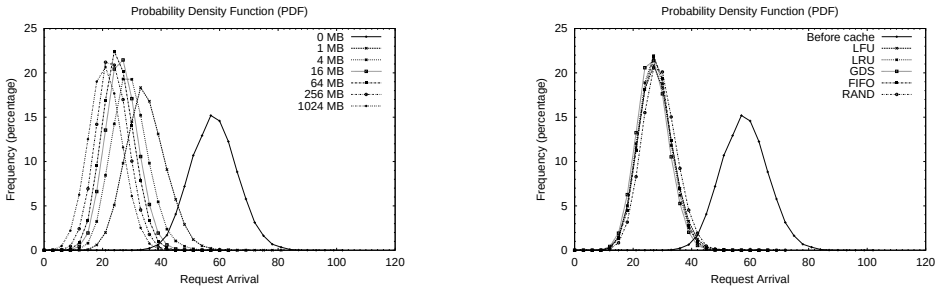
The second experiment illustrates the effect of cache configuration parameters on the cache filter effect. In particular, we focus on the impact of cache size and cache replacement policy. For simplicity, these experiments assume a Poisson arrival process for Web requests.

The characteristics of the filtered arrival process are illustrated in Figure 3. The primary impact of the cache is to reduce the mean arrival rate, shifting the request arrival count distribution to the left (see Figure 3(a)). In general, the filter effect increases with cache size, as expected.

Figure 3(b) shows the impact of cache replacement policy on the workload characteristics. Examining the plot shows that the filter effects of each policy are similar. We thus use the LFU replacement policy as a canonical example in most of the remaining experiments.

4.3. Effect of Input Arrival Process

The next set of experiments studies the influence of the input request arrival process. We consider three cases, namely deterministic, Poisson, and self-similar arrival processes. In all three cases, we use 60 requests per second as the (arbitrary) average request arrival rate, representing a



(a) Effect of Cache Size (LFU Policy) (b) Effect of Cache Replacement Policy (8 MB Cache)

Figure 3. Characteristics of the filtered arrival process as a function of cache size (Trace 1)

trace duration of about 4 hours. We characterize the filtered request stream at the 1 second granularity.

Case 1: Deterministic Arrival Process

The first experiment deals with the simplest case: a deterministic request arrival process. Exactly 60 requests arrive, equally spaced, in each one second of the trace. This trivial example simply provides a baseline for comparison.

Table 2 summarizes the statistical characteristics of the filtered request arrival process. In these experiments, the cache replacement policy is always LFU, while cache size is varied from 1 MB to 1 GB. The corresponding cache hit ratios are shown in the bottom row of Table 2.

Clearly, the presence of the cache reduces the mean arrival rate in the filtered request stream. The larger the cache, the greater this filtering effect. Note that in this (unrealistic) case, the presence of the Web cache *increases* the variance of the filtered request stream (since the input arrival process is deterministic, while the output stream is not). The variance of the filtered stream is similar at each cache size considered, implying that the variance-to-mean ratio of the filtered request stream increases with cache size.

Case 2: Poisson Arrival Process

The next experiment assumes that the input arrival process is Poisson. That is, the inter-arrival times between requests are exponentially distributed and independent. There is some evidence that the Poisson arrival process characterizes some aspects of Internet user behaviour [3, 21], though it does not provide an adequate characterization of aggregate Web traffic [3, 13].

Table 3 summarizes the statistical characteristics of the filtered request arrival process, for an LFU replacement policy, and cache sizes ranging from 1 MB to 1 GB. The cache hit ratios are also shown in Table 3. Note that the cache hit ratios are the same here as they were in Table 2, since only the relative ordering of requests (not their arrival time) matters to the Web caching simulator.

Again, the presence of the cache reduces the mean arrival rate in the filtered request stream. The larger the cache, the greater this filtering effect. However, the impact of the Web cache on the variance of the filtered request stream is less pronounced. The variance-to-mean ratio of the filtered request stream tends to increase with cache size, since the mean rate drops significantly, while the variance decreases slowly. For small cache sizes, the filtered request stream is reasonably well-characterized as a Poisson process; for large cache sizes it is not.

Case 3: Self-Similar Arrival Process

Recent research in network traffic measurement has challenged the Poisson assumptions in traditional network traffic models [1, 13, 17, 21]. Leland *et al.* [17] showed that Ethernet traffic is bursty across many time scales, and can be described statistically as *self-similar*. The term self-similar means that the statistical characterization of the traffic is essentially invariant with time scale; the same statistical properties are observed at time scales of milliseconds, seconds, minutes, hours, and more.

Our next experiment considers a self-similar request arrival process. Several models for self-similar stochastic processes exist, including Fractional Gaussian Noise and Fractional-ARIMA processes. In our study, we use the *synTraff* traffic modeling toolkit developed in prior work [6]. It uses a three-step modeling approach based on Fractional-ARIMA processes to generate *monofractal* traffic. Using this toolkit, we generate time series processes that are distributionally self-similar. The Hurst parameter is $H = 0.70$.

Table 4 summarizes the statistical characteristics of the filtered request arrival process. As observed previously, the mean arrival rate decreases as the cache size is increased. The variance of the filtered request stream also decreases, but not as quickly as the mean. More importantly, further analysis shows that the output stream maintains its self-similar properties (see Section 5.1 for fuller evidence of this). As observed earlier, the variance-to-mean ratio of the filtered request stream tends to increase with cache size.

Table 2. Simulation results for different cache sizes (Trace 1, Deterministic, LFU Policy)

Statistics	Before Cache	Cache Size (MB)					
		1	4	16	64	256	1024
Mean	60.23	36.88	31.45	28.71	27.31	25.37	23.03
Std. deviation	0.43	4.84	4.60	4.01	4.00	4.31	4.78
Request hit ratio	-	38.84%	47.84%	52.66%	55.47%	59.10%	62.70%

Table 3. Simulation results for different cache sizes (Trace 1, Poisson, LFU Policy)

Statistics	Before Cache	Cache Size (MB)					
		1	4	16	64	256	1024
Mean	60.10	36.81	31.38	28.65	27.26	25.33	23.00
Std. deviation	7.82	6.77	6.07	5.43	5.31	5.39	5.62
Request hit ratio	-	38.84%	47.84%	52.66%	55.47%	59.10%	62.70%

4.4. Summary of Results

This section focused on the time-domain analysis of cache filter effects, demonstrating the relationships between input workload characteristics, cache configuration parameters, and the characteristics of the output filtered request stream. In general, increasing the cache size significantly reduces the peak and mean arrival rate for the filtered request stream, but the impact on the variance is less pronounced. In fact, the variance-to-mean ratio of the filtered arrival process increases. If the input arrival process was self-similar, then the output process remains self-similar.

We use the self-similar arrival model as the basis for our study of traffic characteristics in a two-level Web proxy caching hierarchy in the next section.

5. Simulation Results: Caching Hierarchy

This section addresses the larger challenge of characterizing workloads throughout a Web proxy caching hierarchy. We start by characterizing the input workloads λ_1 and λ_2 offered to the child-level caches in Figure 1, and proceed to analyze the filter effects as the workload progresses to the Internet as λ_3'. Section 5.1 describes the input workloads and the filter effects of the first-level caches. Section 5.2 studies the aggregation of the filtered request streams to form λ_3. Finally, Section 5.3 shows how to model the aggregate stream λ_3 with a Gamma distribution, using knowledge of the input workloads and cache filter effects.

5.1. Workload Modeling and Analysis

As in Section 4, we use synthetically-generated Web proxy workloads to represent the input workloads λ_1 and λ_2 for our simulation. To demonstrate the generality of our analysis, we consider "heterogeneous" input workloads, in the sense that the workloads λ_1 and λ_2 differ in Zipf slope, Hurst parameter, and mean arrival rate.

The characteristics of the synthetic Web proxy workloads with self-similar arrival processes are illustrated in Figure 4. These workload traces are provided as input to the Web cache simulator at the first level of the Web caching hierarchy. In this case study, we use the GDS replacement policy, with an 8 MB cache size for both caches at the first level of the caching hierarchy. The filter effects of the child caches produce workloads λ_1' and λ_2', which we call the filtered workloads.

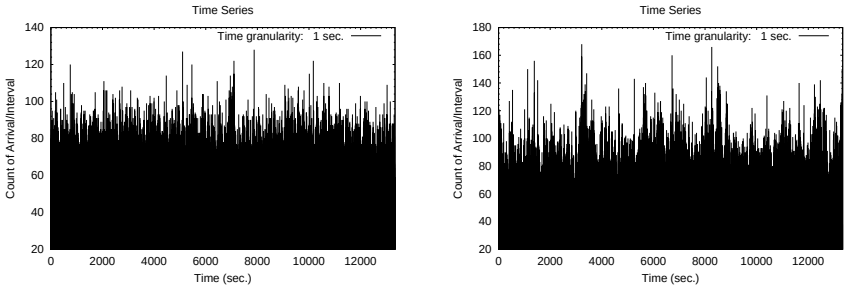
The time series of request arrival processes for the filtered request streams λ_1' and λ_2' are depicted in Figure 5. Other than the end effects evident in these plots, the filtered request streams appear to be stationary. We restrict our attention to the stationary portion of these traces.

Figure 6 is used to test for self-similarity in the filtered request arrival process. We use the standard statistical analysis techniques, namely the autocorrelation function, the variance-time plot, and the rescaled adjusted range statistic (R/S) [17]. The hyperbolic decay of the autocorrelation function in Figure 6(a) is indicative of self-similarity. The variance-time plot in Figure 6(b) has a slope significantly flatter than -1 (the solid line in the graph). This graph shows a slowly-decaying variance for the filtered time series, another indication of self-similarity. Finally, Figure 6(c) shows an R/S pox plot for this data set. The slope of this scatter plot can be used to estimate the Hurst parameter H characterizing the degree of self-similarity in this data set. The R/S plot provides a Hurst parameter estimate of $H \approx 0.699$ (very close to 0.70, the degree of self-similarity of the input workload).

All these observations suggest that the arrival process of the filtered workload λ_1' is self-similar. The same observations apply for filtered request stream λ_2'. Its analysis is omitted for space reasons.

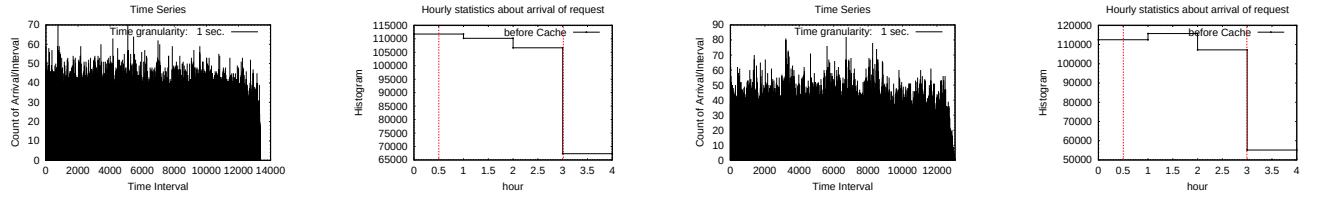
Table 4. Simulation results for different cache sizes (Trace 1, Self-Similar, LFU Policy)

Statistics	Before Cache	Cache Size (MB)					
		1	4	16	64	256	1024
Mean	62.87	38.50	32.79	29.88	28.27	26.05	23.49
Std. deviation	12.24	9.03	7.98	7.12	6.94	7.02	7.14
Request hit ratio	-	38.84%	47.84%	52.66%	55.47%	59.10%	62.70%



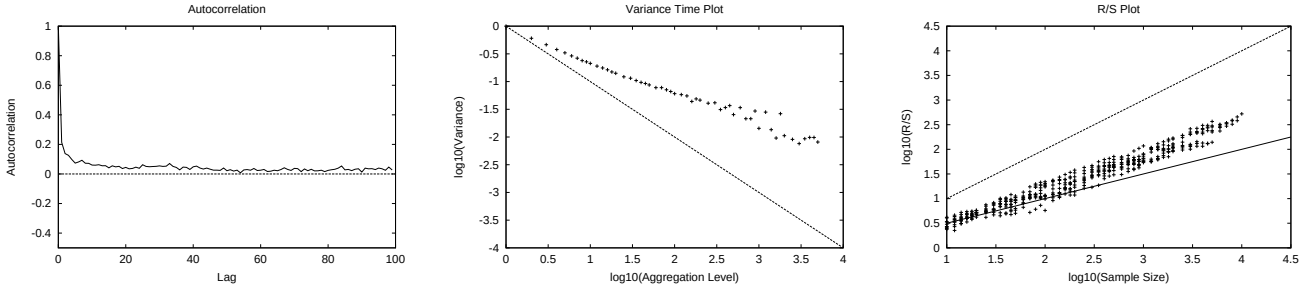
(a) Trace 1: H=0.70, Zipf slope=0.75 (b) Trace 2: H=0.80, Zipf slope=0.80

Figure 4. Synthetic self-similar workload traces used in simulations



(a) λ_1': Interval Size=1 second (b) λ_1': Interval Size=1 hour (c) λ_2': Interval Size=1 second (d) λ_2': Interval Size=1 hour

Figure 5. Time series plots of request arrival processes for filtered workloads λ_1' and λ_2'



(a) Autocorrelation Function (b) Variance-Time Plot (c) R/S Pox Plot

Figure 6. Evidence of self-similarity in filtered workload λ_1'

5.2. Superposition of Web Workloads

Our investigations so far demonstrate that the first-level cache changes the structural characteristics of the workload. However, it does not remove the self-similar property of the workloads. This section focuses on the superposition of Web workload streams (with self-similar arrival) in time-domain. Understanding the statistical multiplexing behaviour is important since many networks rely on it.

In this part of the study, the workload traces λ_1' and λ_2' are statistically multiplexed in the time-domain, resulting in an aggregate workload λ_3. Here we consider portions of each trace that cover the identical time period in order to get a stationary aggregate workload trace. We assume that λ_1' and λ_2' are independent.

The characteristics of the aggregate workload are illustrated in Figure 7. Figure 7(a) shows the marginal distribution (i.e., frequency histogram, or probability density function, PDF) of the traffic arrival process from Figure 8(a). As expected, the mean arrival rate for the aggregate arrival process is equal to the sum of that of the two individual count processes. Figure 7(b) shows the cumulative distribution function for the arrival process, while Figure 7(c) shows a log-log complementary distribution (LLCD) plot that illustrates the tail behaviour of the distribution.

Figure 8 presents the results from the examination of self-similarity for the aggregate workload. Again, we use the autocorrelation function, the variance-time plot, and the rescaled adjusted range statistic (R/S). Figure 8(a), (b) and (c) show the properties of self-similarity.

The R/S plot provides a Hurst parameter estimate of $H \approx 0.76$, suggesting that the aggregate arrival count process is self-similar. These observations are in close agreement with theoretical results [20]: the Hurst parameter of the aggregate stream should be the maximum of the Hurst parameters of the two input streams. In addition, the variance-to-mean ratio of the aggregate process should be the weighted average of those from the input streams [20].

These analyses demonstrate that the superposition of Web traffic streams does not smooth the traffic. Multiplexing bursty data streams tends to produce a bursty aggregate stream. The aggregate Web workload is then forwarded to the next level of the Web proxy caching hierarchy.

5.3. Modeling of Aggregate Workload

This section discusses a parameterizable mathematical model for characterizing the aggregate Web request arrival count distribution. Our previous work has shown that the Web request arrival count (before and after a proxy cache) can be well characterized by a Gamma distribution [5]. The issue at hand is whether it is suitable for modeling aggregate Web workload in a Web caching hierarchy.

Following the approach in [5], we propose two Gamma probability density functions $f_1'(x)$ and $f_2'(x)$ for the Web request arrival count λ_1' and λ_2', namely

$$f_1'(x) = \frac{(\frac{x-\mu_1}{\beta_1})^{\gamma_1 - 1} e^{(-\frac{x-\mu_1}{\beta_1})}}{\beta_1 \, \Gamma(\gamma_1)} \tag{1}$$

and

$$f_2'(x) = \frac{(\frac{x-\mu_2}{\beta_2})^{\gamma_2 - 1} e^{(-\frac{x-\mu_2}{\beta_2})}}{\beta_2 \, \Gamma(\gamma_2)} \tag{2}$$

where x denotes the arrival count per interval and Γ is the Gamma function

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt \tag{3}$$

In our experiments, we assume that the *location* parameter for the Gamma distribution is $\mu_1 = \mu_2 = 0$. Let M and D denote the mean and standard deviation, respectively, of the request arrival count process in a filtered workload (λ_1' or λ_2'). According to the Maximum Likelihood Estimates, we have the following estimates for the shape parameter γ and the scale parameter β of the gamma distribution:

$$\hat{\gamma} = (\frac{M}{D})^2 \tag{4}$$

and

$$\hat{\beta} = \frac{D^2}{M} \tag{5}$$

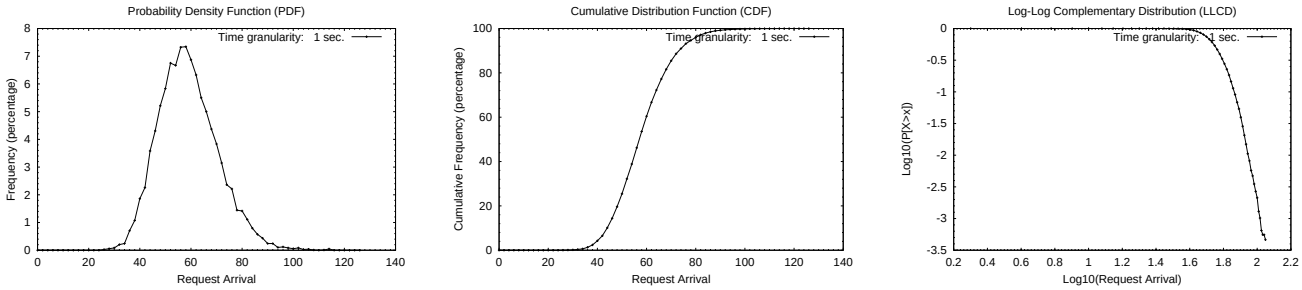
As an example, we consider an 8 MB cache size at each child proxy, each using the GDS cache replacement policy. From the statistical analyses, we obtain $M_1 = 29.885$, $D_1 = 7.115$ for λ_1' and $M_2 = 30.570$, $D_2 = 8.928$ for λ_2'.

According to Equation 4 and Equation 5, the Gamma parameters can be estimated as $\hat{\gamma}_1 = 17.642361$, $\hat{\beta}_1 = 1.693934$ and $\hat{\gamma}_2 = 11.724181$, $\hat{\beta}_2 = 2.607432$. So, the Probability Density Functions (PDF) of λ_1' and λ_2' are plotted in Figure 9.

Assuming that λ_1' and λ_2' are independent, the cumulative request count in the ith time interval for their superposition is governed by:

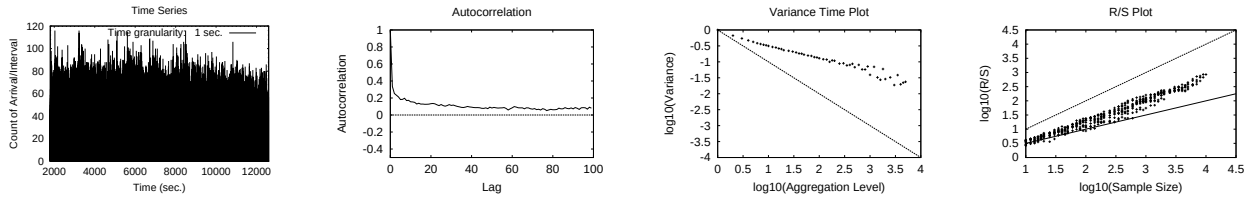
$$P_{\lambda_3}\{X = k\} = \sum_{i=0}^{k} [P_{\lambda_1'}\{X = i\} \times P_{\lambda_2'}\{X = (k-i)\}] \tag{6}$$

Figure 10 illustrates the characteristics of the request arrival count distribution for the aggregate workload given by the superposition of two request arrival processes. The simulation results are also shown for model validation. Figure 10 shows that the Gamma distribution provides a very good visual fit of the distribution, for both the body and the tail of the distribution. This result demonstrates that



(a) PDF (b) CDF (c) LLCD

Figure 7. Characteristics of aggregate request arrival process λ_3



(a) Time series (b) Autocorrelation Function (c) Variance-Time Plot (d) R/S Pox Plot

Figure 8. Evidence of self-similarity for aggregate request arrival process λ_3: $H \approx 0.76$

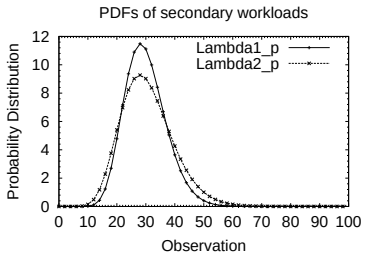


Figure 9. PDF of filtered workloads λ_1' and λ_2'

a Gamma distribution provides a simple, flexible, and relatively robust means of characterizing the aggregate Web request arrival count distribution. However, the parameters for the Gamma distribution, e.g. the *shape* parameter γ, the *scale* parameter β, depend upon the input Web workload characteristics and the cache parameters used.

This model can be used to estimate traffic characteristics in a Web caching hierarchy, if input workload characteristics and cache configuration parameters are known.

6. Summary and Conclusions

This paper uses trace-driven simulation to study the request arrival process at each level of a multi-level Web proxy caching hierarchy. The simulation experiments quantify the filter effects of a Web cache on the request arrival process, for synthetically-generated Web client workloads.

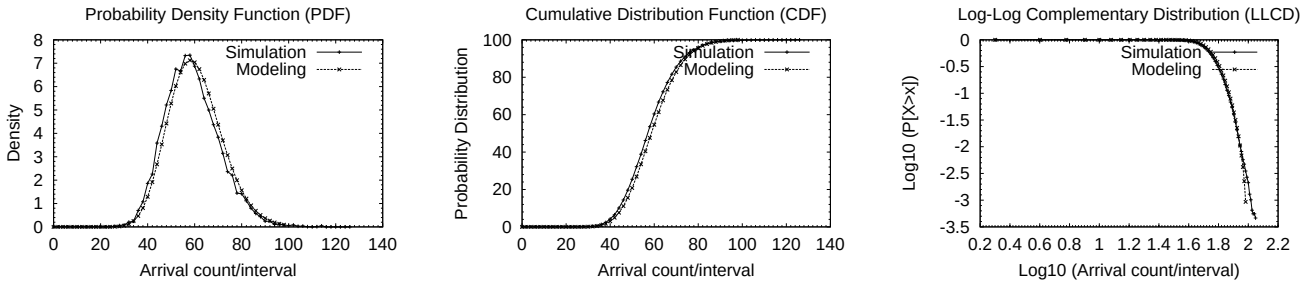
The simulation results show the obvious result that a Web cache reduces the mean request arrival rate for Web traffic workloads. However, the variability of the request arrival process may either increase, decrease, or remain the same after the cache, depending on the input arrival process. For a self-similar request arrival process, the filtered request arrival process remains self-similar, though with reduced mean. Furthermore, the superposition of Web request streams from multiple child caches in a Web proxy caching hierarchy remains bursty. Finally, we find the Gamma distribution is suitable for modeling the request arrival process in a Web caching hierarchy.

References

[1] R. Addie, M. Zukerman and T. Neame, "Fractal Traffic: Measurements, Modeling, and Performance Evaluation", *Proceedings of IEEE INFOCOM*, Vol. 3, pp. 977-984, April 1995.

[2] V. Almeida, M. Cesario, R. Fonseca, W. Meira Jr., and C. Murta, "Analyzing the Behavior of a Proxy Server in Light of Regional and Cultural Issues", *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[3] M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 631-645, October 1997.



(a) PDF (b) CDF (c) LLCD

Figure 10. Gamma distribution model of aggregate workload λ_3

[4] M. Arlitt and C. Williamson, "Trace-Driven Simulation of Document Caching Strategies for Internet Web Servers", *Simulation Journal*, Vol. 68, No. 1, pp. 23-33, January 1997.

[5] G. Bai and C. Williamson, "Time-Domain Analysis of Web Cache Filter Effects", *Proceedings of SPECTS'02*, San Diego, CA, pp. 195-205, July 2002.

[6] R. Balakrishnan and C. Williamson, "The *synTraff* Suite of Traffic Modeling Toolkits", *IEEE MASCOTS*, San Francisco, CA, pp. 333-340, August 2000.

[7] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications", *World Wide Web*, Vol. 2, No. 1, pp. 15-28, January 1999.

[8] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", *Proceedings of IEEE INFOCOM*, New York, NY, pp. 126-134, March 1999.

[9] M. Busari and C. Williamson, "On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics", *Proceedings of IEEE INFOCOM*, pp. 1225-1234, Anchorage, AL, April 2001.

[10] M. Busari and C. Williamson, "Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy", *Proceedings of IEEE MASCOTS*, pp. 379–388, Cincinnati, OH, August 2001.

[11] M. Busari and C. Williamson, "ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches", *Computer Networks*, Vol. 38, No. 6, pp. 779-794, June 2002.

[12] H. Che, Z. Wang, and Y. Tung, "Analysis and Design of Hierarchical Web Caching Systems", *IEEE INFOCOM*, pp. 1416-1424, Anchorage, AL, April 2001.

[13] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 835-846, December 1997.

[14] R. Doyle, J. Chase, S. Gadde, and A. Vahdat, "The Trickle-Down Effect: Web Caching and Server Request Distribution", *Proceedings of the Web Caching Workshop*, Boston, MA, June 2001.

[15] L Fan, P. Cao, J. Almeida and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *IEEE/ACM Transactions on Networking* Vol. 8, No. 3, pp. 281-293, June 2000.

[16] A. Feldmann, R. Cáceres, F. Douglis, G. Glass and M. Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments", *IEEE INFOCOM*, pp. 107-116, April 1999.

[17] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, pp. 1-15, February 1994.

[18] A. Mahanti, C. Williamson and D. Eager, "Traffic Analysis of a Web Proxy Caching Hierarchy", *IEEE Network*, Vol. 14, No. 3, pp. 16-23, May/June 2000.

[19] N. Markatchev and C. Williamson, "*WebTraff*: A GUI for Web Proxy Cache Workload Modeling and Analysis", *IEEE MASCOTS*, Fort Worth, TX, October 2002.

[20] A. Patel and C. Williamson, "Effective Bandwidth of Self-Similar Traffic Sources: Theoretical and Simulation Results", *Proceedings of the IASTED Conference on Applied Modeling and Simulation*, Banff, AB, pp. 298-302, July 1997.

[21] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pp. 226-244, 1995.

[22] C. Roadknight, I. Marshall, and D. Vearer, "File Popularity Characterization", *Proceedings of the Second Workshop on Internet Server Performance (WISP'99)*, Atlanta, GA, May 1999.

[23] C. Williamson, "On Filter Effects in Web Caching Hierarchies", *ACM Transactions on Internet Technology*, Vol. 2, No. 1, pp. 47-77, February 2002.