

1999-10-10

Temporal Locality in Web Request Streams: Sources, Characteristics, and Caching Implications

Jin, Shudong; Bestavros, Azer. "Temporal Locality in Web Request Streams: Sources, Characteristics, and Caching Implications", Technical Report BUCS-1999-014, Computer Science Department, Boston University, October 10, 1999. [Available from: <http://hdl.handle.net/2144/1791>] <https://hdl.handle.net/2144/1791>

"Downloaded from OpenBU. Boston University's institutional repository."

TEMPORAL LOCALITY IN WEB REQUEST STREAMS

Sources, Characteristics, and Caching Implications *

Shudong Jin and Azer Bestavros

Computer Science Department
Boston University
111 Cummington St, Boston, MA 02215
{jins,bestavros}@cs.bu.edu

OCTOBER 1999

Abstract

Temporal locality of reference in Web request streams emerges from two distinct phenomena: the *popularity* of Web objects and the *temporal correlation* of requests. Capturing these two elements of temporal locality is important because it enables cache replacement policies to adjust *how* they capitalize on temporal locality based on the relative prevalence of these phenomena. In this paper, we show that temporal locality metrics proposed in the literature are unable to delineate between these two sources of temporal locality. In particular, we show that the commonly-used distribution of reference interarrival times is predominantly determined by the power law governing the popularity of documents in a request stream.

To capture (and more importantly quantify) both sources of temporal locality in a request stream, we propose a new and robust metric that enables accurate delineation between locality due to popularity and that due to temporal correlation. Using this metric, we characterize the locality of reference in a number of representative proxy cache traces. Our findings show that there are measurable differences between the degrees (and sources) of temporal locality across these traces, and that these differences are effectively captured using our proposed metric. We illustrate the significance of our findings by proposing and evaluating a novel Web cache replacement policy—called GreedyDual*—which exploits both long-term popularity and short-term temporal correlation in an *adaptive* fashion. Our trace-driven simulation experiments show the superior performance of GreedyDual* when compared to other Web cache replacement policies.

Keywords: Temporal locality of reference; Temporal correlation; Web access characterization and modeling; Zipf law; Cache replacement policies; Performance evaluation.

*This work was partially supported by a NSF research grant CCR-9706685 and by a NSF ANIR grant.

1 Introduction

Web access patterns exhibit a number of unique properties that have been identified and characterized. Examples of such properties include the Zipf-like profile of document¹ popularity [17, 10], the heavy-tailed distribution of document sizes [4], the correlation between document size and popularity [17], the self-similar characteristics of traffic resulting from Web accesses [4, 11], and the temporal, spatial, and geographical locality of reference exhibited in a request stream [2]. The prevalence of some of these properties has motivated the development of many protocols (and optimizations thereof) that exploit such properties. For example, knowledge of document size distributions was used in [22] to enable novel load balancing across hosts in a Web server cluster. Another example is Web document prefetching mechanisms [20, 33], which leverage on spatial locality characteristics.

Motivation: Two of the most important properties exhibited in Web access patterns are (1) the *highly skewed popularity* of Web documents, and (2) the strong *temporal locality of reference* exhibited in request streams. Studies of Web access pattern characteristics have identified the presence of both of these properties. Web document popularity was found to follow a Zipf-like distribution, whereby the frequency of document access is proportional to the rank of the document. Temporal locality of reference—the property that recently requested documents are likely to be requested again—was also observed in Web request streams [4, 5, 12, 21, 28].

Both of these properties have implications with respect to caching and replication protocols. On the one hand, the highly skewed popularity of Web documents suggests the use of *long-term* frequency information in replacement (placement) strategies for caching (replication) protocols. Examples of this approach include the server-initiated replication strategies proposed in [8] and the frequency-based cache replacement strategies such as LFU. On the other hand, the temporal locality of reference in Web request streams suggest the use of *short-term* residency information (e.g. time since last access) in replacement strategies for caching protocols. LRU cache replacement strategies and its generalization [12, 24] leverage on this property.

The relationship and distinction between popularity and temporal locality properties in Web access patterns is often blurred. This is partially the result of the *causal* relationship between these two properties—highly popular documents tend to be referenced frequently, and thus exhibit stronger locality of reference patterns than those of less popular documents. Popularity—while an important contributor to temporal locality—is not the only catalyst. In particular, *temporal correlation* of repeated references to documents is another important contributor to locality of reference. An important question is whether these properties reflect fundamentally different phenomena that can be exploited independently.²

Paper Contribution and Scope: In this paper we provide an affirmative answer to this question. We show that the distribution of reference interarrival times (commonly used to characterize temporal locality [4, 10, 12, 14, 28]) is insensitive to temporal locality induced through temporal correlation of references. To that end, we quantify the inherent relationship between Zipf-like popularity profiles and the distribution of reference interarrival times.

To capture both aspects of temporal locality, we propose a new and robust metric that enables accurate delineation between locality due to popularity and that due to temporal correlation. Using this metric, we characterize the locality of reference in a number of representative proxy cache traces.

¹In this paper, we use the term “document” to refer to an object, file, etc.

²This independence can be illustrated by a simple example. References to a very popular document scatter randomly, thus locality of this document is strong, but the references do not exhibit temporal correlation. Similarly, a document that is unpopular over a long time scale may be referenced many times over shorter time scales.

Our findings show that there are measurable differences between the degrees of temporal locality across these traces, and that these differences are effectively captured using our proposed metric.

Our findings suggest that cache replacement policies must leverage on *both* popularity and temporal correlation properties. Furthermore, such policies must be able to adjust (in an on-line fashion) the relative merits of *long-term* frequency information versus *short-term* residency information in making eviction decisions. To that end, we present GreedyDual* (GD*), an adaptive cache replacement policy that uses our metric. We demonstrate GD*'s superiority through extensive trace-driven simulations.

2 Related Work

Temporal Locality in Web Request Streams: Denning and Schwartz [18] established the fundamental properties that characterize the phenomenon of *locality*. Such properties are the catalyst for well-established practices in the design of caching systems in hierarchical memory structures [37]. In order to apply these practices to the design of Web caching and prefetching systems, it's important to characterize the degree of locality present in typical Web request streams.

Early characterizations of Web access patterns suggested the presence of strong temporal locality of reference [2, 4, 12, 21, 28]. However, more recent studies have concluded that this temporal locality is weakening [6]. One reason for this trend was attributed to effective client caching. To understand this, it suffices to note that the request stream generated by a client using an efficient caching policy is precisely the set of requests that missed in the client cache. Such a request stream is likely to exhibit weak temporal locality of reference—in particular, a recently accessed object is *unlikely* to be accessed again in the future.³

Stack Distance Model: In [30], Mattson et al introduced the concept of stack distances as a means for analyzing the behavior of demand-paged memory systems and for evaluating the performance of memory management schemes. Stack distance refers to the number of unique references separating consecutive references to the same document. Stack distance strings are equivalent in information content to the reference string, but are more easily handled by mathematical models.

In [38], Spirn proposes the use of a Markov stack distance model to capture program behavior. A Markov stack distance model enables the prediction of future reference distances based on the most recently generated distances. As such, this model is unable to capture the long-range dependencies among references [38] and hence is unable to capture the bursty nature of page faults, which tend in real programs to occur in clusters (as a result of working set changes).

In [2], Almeida et al used the marginal distribution of stack distance strings to characterize temporal locality. Their analysis of Web request streams, revealed that the marginal distributions of stack distances followed a lognormal distribution. Moreover, it revealed the existence of long-range dependencies in stack distance strings, which they attributed to spatial locality of reference.

While the Stack Distance Model provides means for characterizing the degree of temporal locality that exists in a request stream, it is not capable of delineating the cause of such locality—namely locality due to popularity and locality due to the temporal correlation of references. Capturing (and more importantly quantifying) these two elements of temporal locality is important because it allows cache replacement policies (for instance) to adjust how they capitalize on temporal locality based on

³Assuming very efficient client caching, it follows that repeated requests to an object at a caching proxy are likely to be from multiple clients, and thus are reflective of popularity and *not* of temporal correlation of references. This was termed *geographical locality of reference* in [8]. This observation is supported by our findings later in this paper regarding the relative contributions of popularity and temporal correlation to locality properties.

the relative prevalence of these properties in the request stream. In particular, a cache replacement that favors long-term frequency information (e.g. LFU) over short-term residency information (e.g. LRU) is likely to be more effective for a request stream whose temporal locality is due mostly to the skewed long-term popularity distribution of documents. The opposite would be true for a request stream whose temporal locality is due mostly to short-term temporal correlation of document requests. This point is further explained later in this paper.

Reference Interarrival Model: Using an independent reference model [13], Breslau et al [10] showed that the Zipf-like [42] popularity profile of documents in Web request streams can asymptotically explain other properties (namely, cache efficiency and temporal locality). In particular, they showed that the probability of referencing an object t units of time after it has been last referenced is roughly proportional to $1/t$. Thus, the distribution of reference interarrival times⁴ could be used to model temporal locality. In this paper, we call this the reference interarrival model.

As we will show later, the reference interarrival model cannot fully capture the temporal locality exhibited in Web request streams. It also does not delineate locality due to popularity and locality due to the temporal correlation of references. Rather, because of the strong relationship between popularity and reference interarrival time distribution, this model tends to disguise reference correlation, which—as established in [2]—is one catalyst for temporal locality in Web request streams.

Exploiting Temporal Locality Properties: Locality of reference properties have been exploited in a number of Web caching, replication, and prefetching protocols and systems. Generally speaking, such protocols can be classified as: (1) Server-based (i.e. at the server or a proxy thereof) [7, 8, 15], (2) Client-based (i.e. at the client or a proxy thereof) [1, 9], or (3) Network-based (i.e. in the network, transparent to both the server and the client) [23, 12, 28, 14, 33, 20]. An important reason for this classification is that Web reference characteristics are likely to be different in each of the above categories. To understand this, it suffices to note that at each of the above categories, the request streams being multiplexed through a cache (or replica) are significantly different. For client-based caches, the request streams are *from* a limited and possibly homogeneous community of users (e.g. students in a University, or subscribers to a local ISP, etc.). For server-based caches (or replicas), the request streams are *to* the limited set of documents offered by the server. For network-based caches, neither of these constraints could be assumed. Furthermore, misses in client-based caches are hits in network-based caches, and misses in those are hits in server-based caches (or replicas).

If the underlying determinant of temporal locality is different for each of the above categories, it follows that an effective cache replacement strategy must be able to delineate between the *different causes* of temporal locality and to quantify their relative significance (in order to effectively capitalize thereon). In the Section 3 of this paper, we present a metric that enables such a capability. In Section 4, we sketch and evaluate a cache replacement strategy that effectively uses this metric.

3 Characterizing Temporal Locality of Reference

3.1 Traces Used in our Characterization

In this paper we used traces from DEC [19] and NLANR [31]. Some of the characteristics of these traces are shown in Table 1.

Preprocessing of DEC Traces: Our preprocessing of the DEC traces followed the same procedures described in [10, 12]. In particular, we have excluded non-cache-able requests, including cgi-bin requests

⁴In other words, the probability of a given interval of time between consecutive requests.

Trace	All requests	Unique requests	HR_∞	BHR_∞
DEC: 29/8-4/9, 1996	3,543,968(44.9G)	1,354,996(21.9G)	48.7%	35.8%
NLANR site RTP: 4/6-17/6, 1999	9,113,027(91.4G)	3,249,549(45.2G)	64.3%	50.7%
NLANR site SD: 4/6-17/6, 1999	9,082,461(129.7G)	3,549,609(61.5G)	60.9%	52.6%
NLANR site UC: 4/6-17/6, 1999	8,983,585(113.1G)	2,459,366(47.3G)	72.6%	58.2%

Table 1: Traces used in this paper

and queries. In addition, in our experiments, we count a request as a hit if the last modification times of the cached object and the actual reply to users are the same when both are known, or if the object size has not changed when both last modification times are unknown. In this paper we only present the results we obtained from the first week of the DEC trace (results from the other weeks were identical).

Preprocessing of NLANR Traces: Our preprocessing of the NLANR traces was more elaborate. The NLANR traces include many IMS (If-Modified-Since) and REFRESH requests with a reply code of “304” (Not Modified). In order to include such requests in the workload, we had to find the sizes of the documents of such requests. We do so through a 2-pass scanning of the entire trace.⁵ In addition to this preprocessing, we have also excluded non-cache-able requests, including cgi-bin requests and queries. In this paper we only present the results we obtained from the three two-week NLANR traces from sites RTP, SD, and UC (hereafter called the RTP, SD, and UC traces).

3.2 Characterizing Popularity

Numerous studies [6, 10, 17] have shown that a Zipf-like distribution (i.e. a power law) can model the relationship between the popularity of a document and its popularity rank.⁶ Namely, this relationship can be expressed as a power law: $P \sim \rho^{-\alpha}$, $0 < \alpha < 1$, where P is the document’s likelihood of access and ρ is its popularity rank. Thus the value of α could be used to characterize the Zipf-like popularity of Web documents in a request stream. Table 2 shows a least-square fit of the values of α for our trace set.

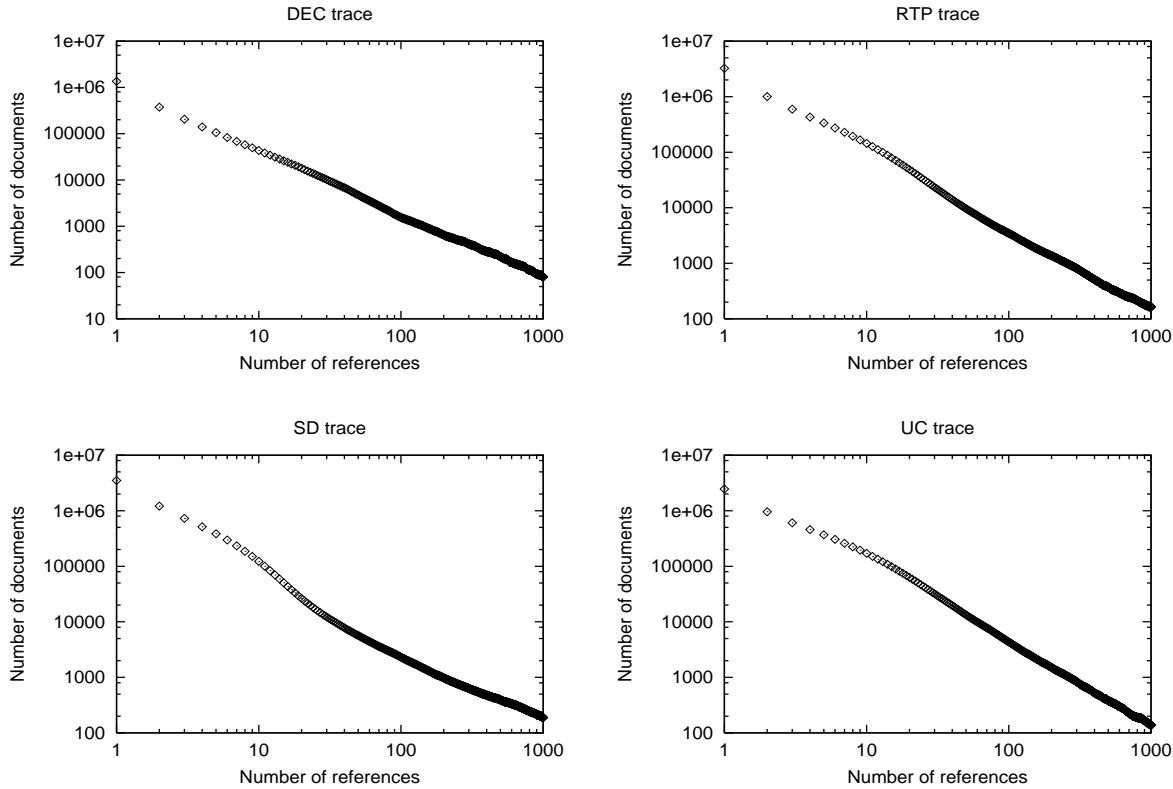
Trace	DEC trace	RTP trace	SD trace	UC trace
α	0.77	0.71	0.73	0.66

Table 2: Values of α (power law parameter) governing document popularity in the various traces.

Another way of conveying the popularity profile of documents in a request stream is to characterize the number of documents referenced at least k times—denoted by $D(k)$ —as a function of k . This can be expressed as: $D(k) \sim k^{-1/\alpha}$, which corresponds to transposing the x-axis and y-axis of the $P \sim \rho^{-\alpha}$ distribution. Figure 1 shows the plots of this distribution for our trace set. The log-log scale plots are close to straight lines with slopes that are close to $-1/\alpha$. These results are in line with previous studies on popularity distributions [2, 4, 10, 17].

⁵This process was 96%-successful in identifying cache-able requests. The remaining 4% were IMS and Refresh requests for which we were unable to identify the document sizes.

⁶In the remainder of this paper, we use the terms “Zipf-like distribution” and “power law” interchangeably.

Figure 1: Number of documents referenced $\geq k$ times.

3.3 Characterizing Reference Recency

In [10], Breslau et al argued that a combination of Zipf-like popularity profiles and an independent reference model can explain asymptotic properties of Web request streams, including temporal locality properties. They proposed that temporal locality be captured quantitatively by relating the probability of a future access to the time elapsed since the last access. Namely, *the probability that a document will be referenced at time t after it is last referenced is roughly proportional to $1/t$* .

To examine the expressiveness of this reference interarrival model in capturing temporal locality, we plot the fraction of references as a function of the time elapsed t since the last reference measured in 5-minute intervals. Figure 2 shows these distributions for the traces in Table 1.

For the DEC trace, the log-log scale plots in Figure 2 are nearly straight lines with slope quite close to -1.0 except for the existence of diurnal spikes [12, 21]. This confirms the applicability of the reference interarrival model (that the probability of access is proportional to $1/t$) and the observations in [12, 28]. For the NLANR traces, the plots in Figure 2 display similar properties,⁷ except that the slopes of the bodies are in the $-0.65 \sim -0.73$ range. The variations in the values of α across the NLANR traces can be explained by noting that the traces reflect different workloads. These variations aside, the values of α for the NLANR traces are clearly less than that measured for the DEC trace. The lower value of α in the NLANR traces (vintage 1999) compared to the DEC traces (vintage 1996) may be symptomatic of the weakening in temporal locality over time documented in [6].

⁷The “dip” at the extreme right-hand-side of the plots is due to the limited length (in time) of the traces we analyzed. The relationship between popularity and reference interarrivals we prove in Appendix A gives an analytical explanation for this “dip”.

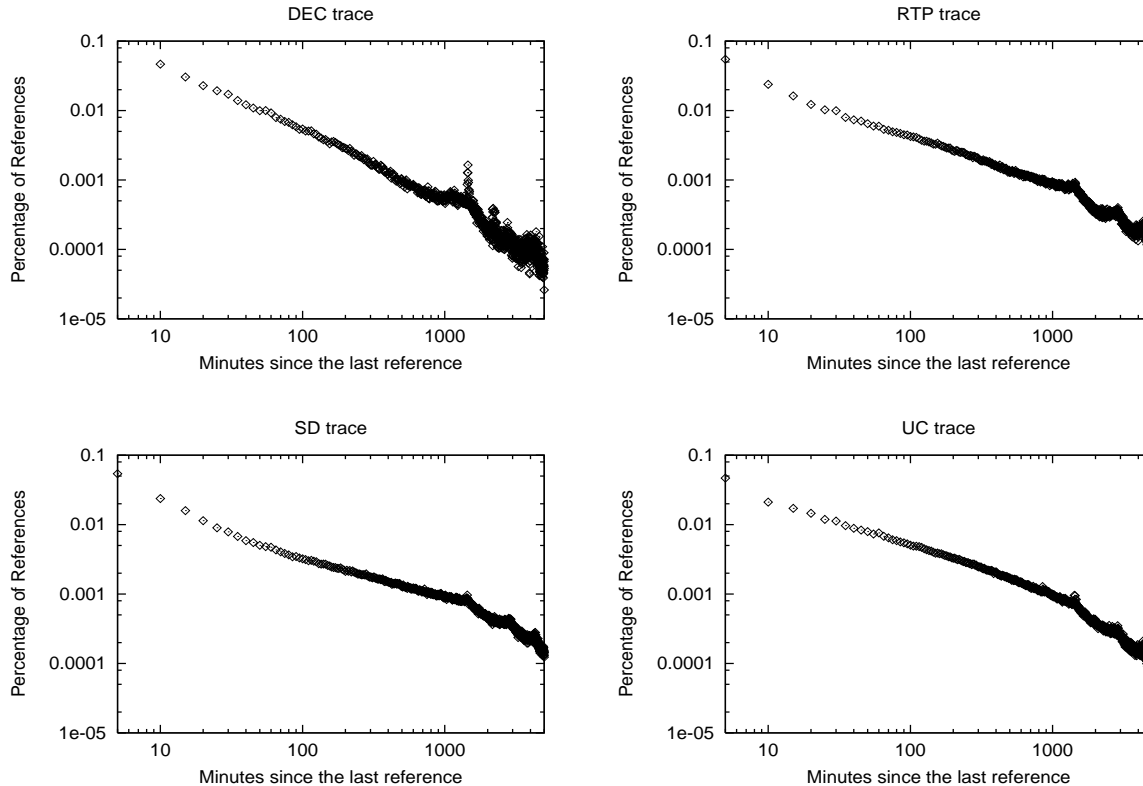


Figure 2: Probability distribution of reference interarrival times.

The distribution of reference interarrivals is reflective of *both* popularity and temporal correlation. In particular, the skewed distributions evident in Figure 2 may well be a reflection of the skewed popularity profile illustrated in Figure 1. Highly popular documents tend to be referenced frequently, and thus will exhibit shorter reference interarrival times; less popular documents, on the other hand, tend to be referenced infrequently, and thus will exhibit longer reference interarrival times. In the following subsection we establish quantitatively this relationship.

3.4 Popularity as a Dimension of Temporal Locality

The skewed popularity profile of documents in Web request streams is not the only source of temporal locality. To explain this point, consider the following two request streams: ‘‘XAXBXCXD XEF...’’ and ‘‘GGHHIIJJKLL...’’. Obviously, both streams exhibit temporal locality properties. In the first stream, the locality is due to the popularity of X, whereas in the second stream it is due to the correlation in time of the 1st and 2nd requests for G, H, I, J, K, and L. Temporal locality due to popularity is preserved under reordering. Thus, in a random permutation of the first stream, reference interarrival time is still proportional to the probability of access. Temporal locality due to correlation in time is *not* preserved under reordering.

Given the above two dimensions of temporal locality, an important question is whether the characterization of temporal locality using the reference interarrival model is capable of quantifying the effects of popularity and temporal correlation, independently.

To answer this question, we consider the changes in the distributional characteristics of reference interarrivals in Figure 2 when the traces are subjected to a random permutations. The left-hand-side

plots in Figure 3 are similar to those in Figure 2, with the exception that here we measure distance between two consecutive references using the number of intervening references instead of the absolute interarrival time. This difference tends to mask the diurnal spikes shown in Figure 2. The right-hand-side plots in Figure 3 are obtained by applying a random permutation to our traces, thus eliminating correlation in time while preserving the popularity profile of documents in the trace.

Comparing the left-hand-side and right-hand-side plots in Figure 3 reveals *no* significant changes in the distributions except for being “smoothed” as a result of trace scrambling.⁷ Table 3 shows the slopes of these plots using a least-square fit. The fact that there is little change as a result of scrambling suggests that the distribution of reference interarrival times is predominantly determined by the distribution of document popularity in the trace and, thus, *cannot* effectively quantify the degree of temporal correlation in the request stream.

Trace	DEC trace	RTP trace	SD trace	UC trace
Original	0.95	0.71	0.65	0.73
Scrambled	0.84	0.68	0.62	0.69

Table 3: Slopes of the curves in Figure 3.

The strong relationship between popularity and reference interarrival distributions is evident in the relationship between the slopes of the log-log scale plots in Figure 1 and Figure 3 (right). This relationship is formalized in the following Theorem.

Theorem 1 *If the distribution of document popularity in a request stream asymptotically follows a power law with parameter α , where $0.5 \ll \alpha \leq 1$, then the distribution of reference interarrivals in a random permutation of that request stream can be characterized asymptotically using a power law with parameter $(2 - \frac{1}{\alpha})$.*

Proof: See Appendix A.

The above Theorem suggests that a less skewed popularity profile tends to result in weaker temporal locality. Moreover, since $(2 - \frac{1}{\alpha}) < \alpha$, it follows that temporal locality resulting from skewed popularity is limited. Two special cases are: (1) When $\alpha = 1$, the slope of reference interarrival distribution is also unity. This is verified by the work [10]. (2) When α is close to 0.5, theoretically, the slope of reference interarrival distribution approaches 0.

It is important to note the difference between the above asymptotic relationship and the distributions obtained from realistic request streams. In particular, as shown in our proof, the right-hand-side plots in Figure 3 do not follow “strictly” straight lines. Thus, since the slopes in Table 3 were computed using a least-square fit over the range of distances $0 < x < 1,000,000$, it follows that the measured slopes are larger than those predicted using the $(2 - \frac{1}{\alpha})$ formula. If we further restrict the distances used in our least-square fit to the range $0 < x < 100,000$ in order to avoid the corruption attributed to the limited trace length, we obtain slopes that are almost identical to those predicted using the $(2 - \frac{1}{\alpha})$ formula.

To summarize, the reference interarrival model captures well the temporal locality of reference induced by the skewed popularity profile of documents in a Web request stream, but is unable to quantify locality due to temporal correlation of references.

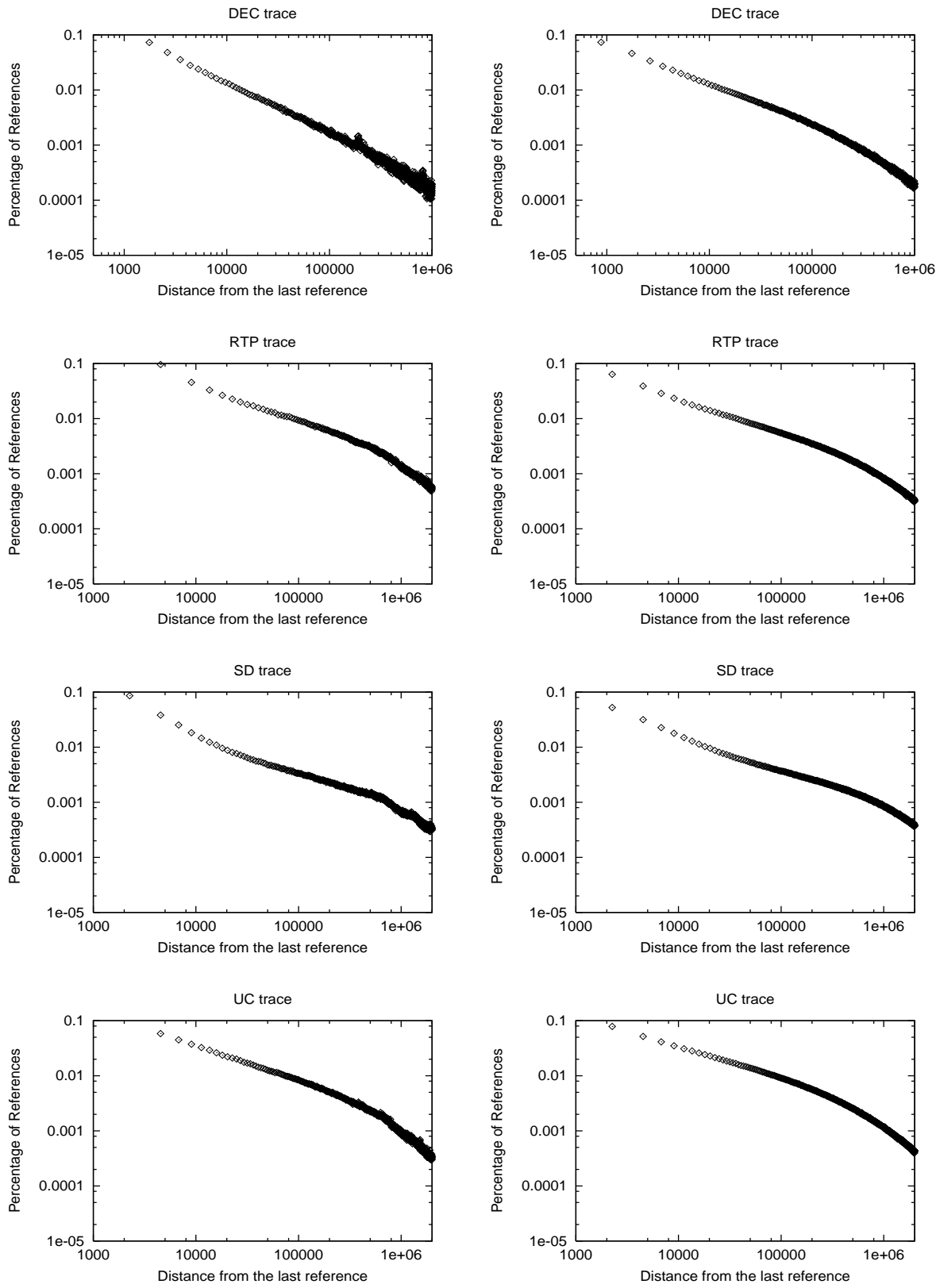


Figure 3: Probability distribution of reference interarrivals. The left plots are for the original traces, and the right plots are for the scrambled traces.

	DEC trace	RTP trace	SD trace	UC trace
Derived: $2 - \frac{1}{\alpha}$	0.70	0.59	0.61	0.49
Estimated slope for $x < 100,000$	0.71	0.62	0.64	0.55

Table 4: Derived and estimated slopes of the curves in Figure 3.

3.5 Temporal Correlation as a Dimension of Temporal Locality

The strong relationship between popularity and temporal locality—as evidenced by the closeness of the slopes in Table 3—often disguises an important aspect of temporal locality, namely the temporal correlation of repeated references to the same documents. This has led to (for example) the inadequate conclusion in [10] that a Zipf-like popularity distribution, together with an independent reference model, is enough to explain temporal locality.

To characterize the level of temporal correlation in a request streams, we must “equalize” the effect of popularity. Thus, we consider the probability distribution of reference interarrivals for *equally popular* documents. For such a distribution, we expect the effect of popularity to be eliminated.

Consider the distribution of reference interarrivals for documents after they appear k times in the request stream.⁸ Figure 4 (left) shows the plots when $k=1, 2, 4,$ and 8 for the RTP trace. The results for the other traces and for other values of k are similar. Comparing the distributions in Figure 4 (left) to those for the RTP trace in Figure 3 (left), we notice a weakening in the measured locality of reference (as evidenced by the lesser sloping on the log-log scale). This weakening is expected since the plots in Figure 4 (left) exclude the skew introduced by popularity. Thus, these plots capture the effect of temporal correlation but not of popularity.

Figure 4 (right) shows the distribution of reference interarrivals for documents referenced k times, when the trace is scrambled. The results indicate that the slope has all but disappeared, which is expected due to the masking of popularity effects. Here, neither popularity nor temporal correlation is present. Comparing the left-hand-side to the right-hand-side plots in Figure 4, we observe various degrees of temporal correlation in request streams.

It is worth noting that since each of the distributions in Figure 4 is for a specific value of k , the overall number of samples is significantly less than the number of samples available when we consider the distribution of reference interarrivals at all levels of popularity (i.e. for all k 's), which is given by 3 (left). This reduction in samples and the limited trace length explain the drop at the extreme right of the plots in Figure 4 (left), especially those for larger values of k .

Above study indicate, the distribution of reference interarrivals for “equally popular” documents is a good estimator of the degree of temporal correlation in the request stream. The degree of temporal correlation can be quantified by β , the slope of log-log scaled distribution of reference interarrivals for equally popular documents. Thus, for k -popular documents, the probability that the reference interarrival time equals t is roughly proportional to $t^{-\beta}$. Table 5 gives the ranges of β . It indicates that the value of β is rather stable for different values of k 's, but that it varies significantly across traces.⁹

Summary: Temporal locality in a request stream can be characterized using a pair (α, β) , where α is the parameter of the power law characterizing the popularity of documents in the request stream

⁸Note that we assume that there is a warm-up period, i.e., we get the samples from the second half of the request stream.

⁹Recall from our earlier discussion that proxy cache traces represent different client and server populations and thus is expected to exhibit different locality characteristics.

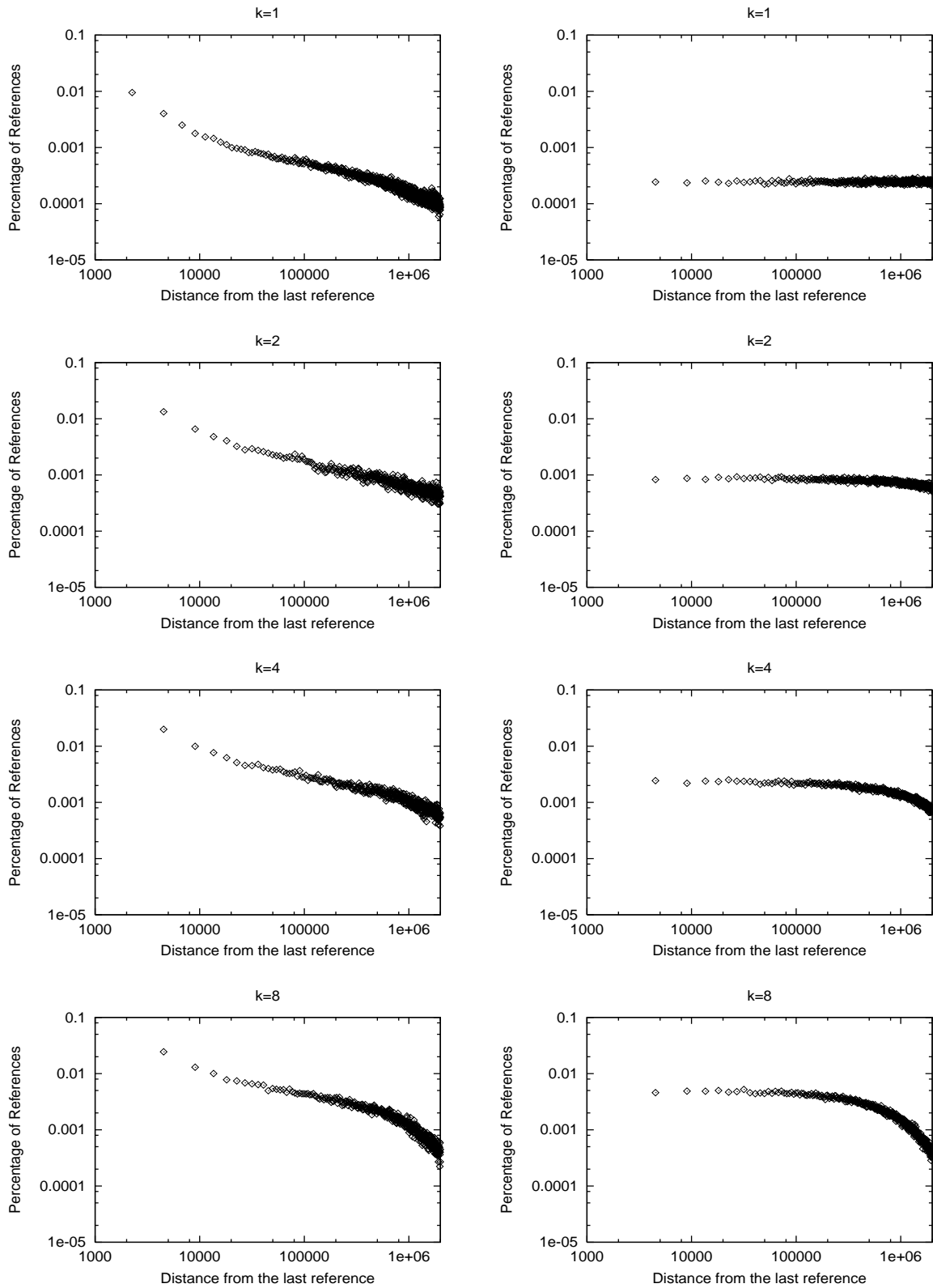


Figure 4: The reference interarrival probability distribution for the RTP trace. The left plots are for the original traces, and the right plots are for the scrambled traces.

Traces	DEC trace	RTP trace	SD trace	UC trace
$k = 1$	0.61	0.51	0.39	0.50
$k = 2$	0.63	0.49	0.41	0.50
$k = 4$	0.63	0.47	0.40	0.46
$k = 8$	0.65	0.46	0.41	0.43

Table 5: The values of β identify the degree of temporal correlation in the request streams. β is estimated with a least-square fit from the bodies of the plots in Figure 4 (left).

and β is the parameter of the power law characterizing the distribution of interarrivals of requests to equally-popular documents.

4 Implications for Cache Replacement Policies

By exploiting the locality properties in Web request streams, Web caching, replication, and prefetching protocols can potentially achieve significant savings [1, 5, 21, 39] in network traffic and in perceived document transfer latencies.

4.1 Previous Work

Table 6 classifies sixteen replacement policies according to whether they exploit temporal locality and reference frequency, and whether they are sensitive to the variable *cost/size* of documents, which is one of the salient aspects of Web caching (namely, unlike traditional memory systems, document sizes as well as miss penalties are highly variable).

Cost/Size	Aspect of Temporal Locality Being Exploited			
	None	Recency-based	Frequency-based	Both
Fixed	FIFO,Random	LRU	LFU	LRU-K,LFU-DA,LRFU
Variable	LFF,Latency	GD,GDS	Hybrid	LNC-W3,GDSF,GDSP,LRV

Table 6: Taxonomy of existing Web cache replacement policies

Recency-Based Policies: Cache replacement algorithms in traditional memory systems deal with uniform *cost/size* objects. LRU is the most widely used cache replacement algorithm, as it has been shown to be superior to other policies, e.g. FIFO and Random. Since Web traffic also exhibits locality, LRU is adopted widely in Web servers, client applications, and proxy servers. An example is the variance employed in the Squid Internet object cache [34]. The GreedyDual-Size or GDS algorithm [12] enables the incorporation of the variability of *cost/size*. GDS is a generalization of the GreedyDual or GD algorithm [41], which deals with uniform-size variable-cost objects. It was shown to be online optimal in terms of its competitive ratio and superior to LFF, which evicts the largest file and does not capture locality.

Frequency-Based Policies: The basic frequency-aware replacement algorithm is LFU. It always evicts the document with the lowest reference count. LFU is online-optimal under a purely independent reference model. A straightforward generalization of LFU for dealing with variable *cost/size* documents is the normalized-cost LFU, which uses $\frac{\text{frequency} \times \text{cost}}{\text{size}}$ as the key. The Hybrid algorithm described in [40] is an example of such a policy, and was shown to outperform LFF, LFU, and LRU.

Recency/Frequency-based Policies: Three examples of policies that incorporate both recency and frequency information under a uniform *cost/size* model are the LRU-K [32], the LFU-DA [3] and the LRFU [27]. LRU-K [32] maintains the last K reference times to each document to compute the average reference rate. LFU-DA [3] is a frequency-based algorithm with dynamic aging. Simulations with large traces indicate LFU-DA obtains the highest byte hit rate. LRFU [27] combines LRU and LFU in a hybrid policy. Generalizations for variable *cost/size* include LNC-W3 [35, 36], GDSF [3], GD-LFU in [26], GDSP [25], and LRV [28]. The LNC-W3 algorithm [35, 36] is a generalization of LRU-K. The GDSF, GD-LFU, and GDSP algorithms [3, 26, 25] augment GDS with a frequency metric.

4.2 Overview of GreedyDual* Cache Replacement

Notation: Let p be a document. Let $s(p)$ be the size of p , and $c(p)$ be the cost to fetch it. Each document has a relative frequency value, denoted by $f(p)$. Let the *utility* value of $u(p)$ represent the normalized value of p . The replacement algorithm tries to maximize the cost saving under the restriction of total cache size. In addition, let β be the parameter of the power law $T \sim t^{-\beta}$ characterizing reference correlation. Recall that reference correlation is measured by the distribution of reference interarrivals for equally-popular documents as explained in Section 3.

From GDS to GD*: Without reference correlation, a greedy algorithm can compute $u(p)$ and sort the documents in decreasing order, then keeps as many documents as possible in this order. With reference correlation, the GDS algorithm [12] takes $c(p)/s(p)$ as $u(p)$, and uses an inflation value L to age the documents. On retrieval or on a hit, the key of a document $H(p)$ is set to $L + u(p)$; on each eviction, L is set to the value $H(p)$ of the evicted document p . GreedyDual* (GD*) redefines the utility value $u(p)$ and the dynamic aging mechanism to reflect the strength or weakness of locality due to temporal correlation versus that due to popularity.

Utility Value: In GD*, the utility value $u(p)$ reflects the normalized expected cost saving if the document stays in the cache. Obviously $u(p)$ is proportional to $c(p)/s(p)$. Moreover, it is also proportional to the long-term frequency if the reference pattern is stable. Since the past reference count is a good approximation of the frequency, $u(p)$ should be roughly proportional to $\frac{f(p) \times c(p)}{s(p)}$, where $f(p)$ is approximated by the reference count so far.

Dynamic Aging: Since Web traffic exhibits reference correlation GD* uses a dynamic aging mechanism similar to that used in GDS. Namely, each document p has an $H(p)$ value, meanwhile the algorithm keeps an inflating L value to age the documents in the cache. When the L value catches up with $H(p)$, then p will be the candidate for eviction. On each hit or when fetching from the server, the algorithm resets the H value of a document to its *base value* plus L .

Now the only remaining problem is setting the base value. The base value for a document must reflect both the document utility and the degree of reference correlation. Since reference interarrivals for equally popular documents follow $T \sim t^{-\beta}$, the maximal length of time for p to stay in the cache should be proportional to $u(p)^{1/\beta}$. We assume the cache is in a steady state, when L increases steadily, and the time for a document to stay in the cache is roughly proportional to its base value. Therefore, in GD*, the base value is set to $u(p)^{1/\beta}$. We illustrate how it works through an example. Assume $\beta = 0.5$. Assume $u(p_1)$ is twice $u(p_2)$ due to differences in p_1 and p_2 's retrieval costs, their sizes, or their relative frequencies. Given the power law $T \sim t^{-\beta}$, it follows that document p_1 at time $4t$ after the last reference to it is as competitive as document p_2 at time t after the last reference to it. Thus, GD* obtains equal marginal cost saving from caching either.

Summary: Our GreedyDual* (GD*) algorithm captures *both* popularity and temporal correlation. The frequency in the base value formula captures long-term popularity, while β controls the rate of

aging. A smaller β means weaker reference correlation, therefore a larger base value, which means documents are aged more slowly. The GD* algorithm and details of its efficient implementation are given in Appendix B.

4.3 Performance Evaluation

We used trace-driven simulations to evaluate GD* against a number of cache replacement strategies.

Experimental Setup and Metrics: We compared GD* with LRU, GDS, and LFU-DA (LFU with dynamic aging [3]). Other policies such as LFF, Hybrid [40], and LRV [28] were not considered since they were shown to under-perform GDS [12].

Both GDS and GD* describe a family of algorithms. To complete the specification of a member of this family, we need to define what constitutes the *cost* of a miss (i.e. miss penalty). To that end, we adopt two models. Under the *constant cost* model, we assume that documents have the same retrieval cost. The resulting algorithms are termed GDS(1) and GD*(1). Under the *packet cost* model, we assume that document retrieval costs are proportional to document size.¹⁰ The resulting algorithms are termed GDS(packets) and GD*(packets).

In our experiments, we considered two main performance metrics—Hit Rate (HR) and Byte Hit Rate (BHR). The constant cost model aims at optimizing HR, whereas the packet cost model aims at optimizing BHR. The use of the constant cost model is appropriate if the purpose of caching is to improve the performance as perceived by the clients of the cache. This would be the case if the cache is deployed close to a client population (e.g. an organizational caching proxy) to reduce response times. The use of the packet cost model is appropriate if the purpose of caching is to improve the utility of the cache or to reduce the overall traffic between the cache and Web servers. This would be the case if the cache is deployed by an ISP to optimize the performance of its networks.

The traces used to drive our simulations were the ones presented in Table 1. HR_∞ and BHR_∞ are the hit rates when cache size is infinite; they represent an upper bound for HR and BHR, respectively. For GD* we estimated the β in an on-line fashion using a least-square fit.

In our experiments, we varied the cache size from a size of less than 1% to a size of about 20% of the total number of unique bytes in the trace.

Performance Under Constant Cost Assumption: Figure 5 shows HR and BHR for the different traces. In each plot, the x-axis (in logarithmic scale) represent the cache size and the y-axis represents the HR (left plots) or BHR (right plots). To establish how each of the algorithms approaches the upper bounds for HR and BHR, the range of the y-axis is set to $[0-HR_\infty]$ (left plots) and $[0-BHR_\infty]$ (right plots).

The results in Figure 5 show that LRU and LFU-DA perform significantly worse than GDS(1) and GD*(1).¹¹ The poor performance of LRU and LFU-DA can be explained by noting that these policies do not consider *size* as a factor in evaluating the utility of a document. The results in Figure 5 also indicate that the performance of GD*(1) is consistently better than that of GDS(1), especially when the cache size is small.¹² This suggests that the advantage of taking temporal correlation into consideration is more crucial for small caches.

¹⁰Namely, the number of *packets* transferred = $2 + \frac{size}{536}$.

¹¹For example, for the DEC trace when cache size is 1GB, LRU's HR is 35.4% and LFU-DA's HR is 37.5% while both GDS(1) and GD*(1) obtain a HR of about 44%—a 20%-26% improvement.

¹²For example, for the DEC trace when cache size is 0.3% of the total size of unique documents, the HR for GD*(1) is 30.8% compared to 26.9% for GDS(1)—a 15% improvement.

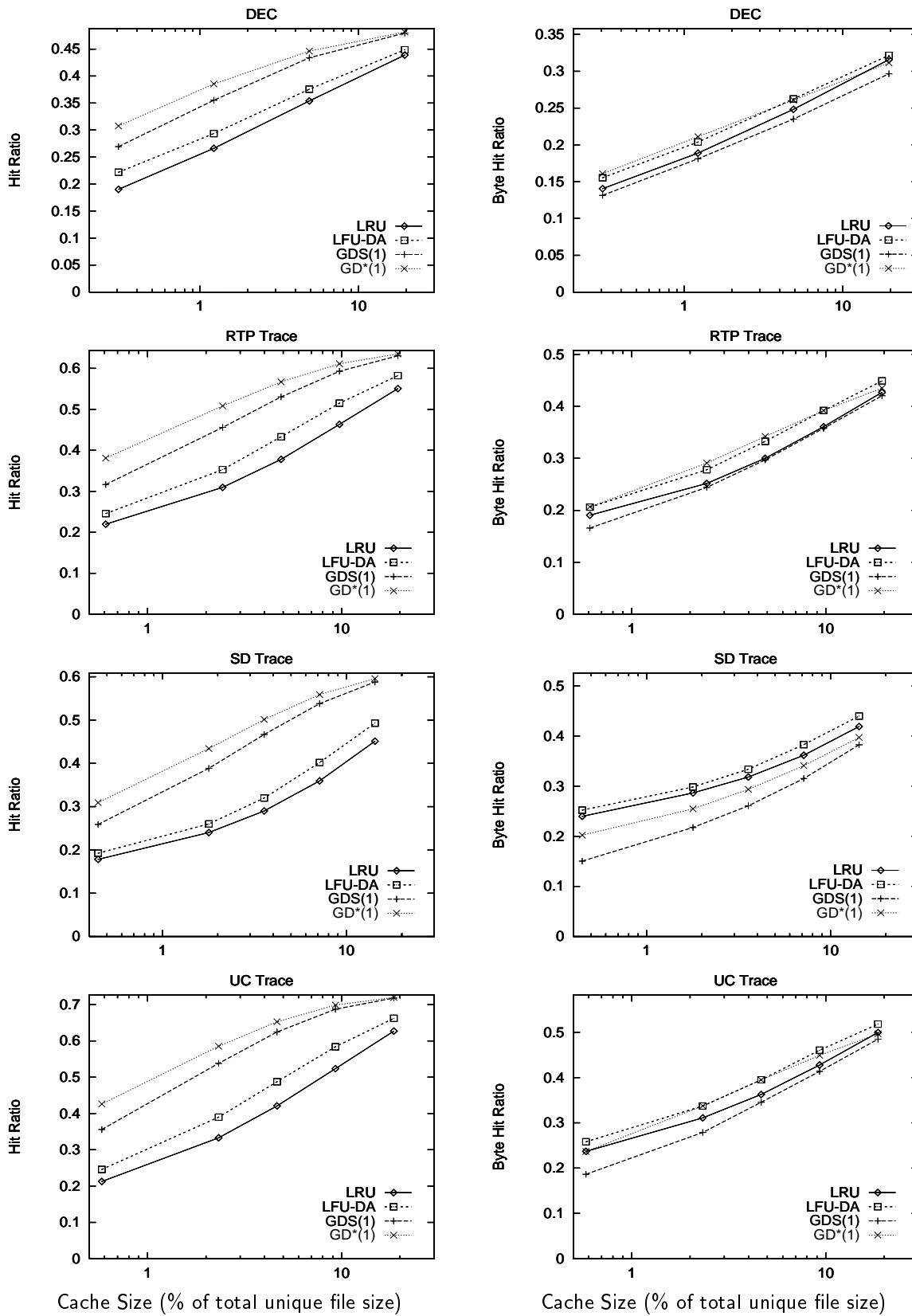


Figure 5: Hit rates when *cost* of document retrieval is fixed

In terms of BHR, the performance of GDS(1) was always the worst. This is not surprising since GDS(1) favors small documents, thus sacrificing BHR [6]. Interestingly, even though the main objective of GD*(1) is the optimization of HR, the BHR of GD*(1) remains competitive with those of LFU and LFU-DA. This can be explained by noting that the incorporation of reference frequency into GD* allows it to retain frequently referenced documents, including large ones.

There are two conclusions from this simulation. First, frequency-based policies consistently outperform recency-based policies, e.g., LFU-DA outperforms LRU and GD*(1) outperforms GDS(1). Second, when HR is the main objective, GD*(1) obtains higher HR than GDS(1), without significantly compromising BHR. Actually, GD*(1) is competitive with LRU and LFU-DA, which aim at maximizing BHR.

Performance Under Packets Cost Assumption: Figure 6 shows the hit rates of LRU, LFU-DA, GDS(packets), and GD*(packets) for the different traces.

The results in Figure 6 indicate that the hit rates achieved by GDS(packets) and LRU are very close. This can be explained by noting that when the cost is proportional to document size, GDS(packets) is nearly equivalent to LRU. The slight advantage of GDS(packets) is due to its sensitivity to document size. Figure 6 also shows that for both HR and BHR, LFU-DA outperforms GDS(packets) and LRU. This is consistent with other studies, which confirmed the advantage of frequency-based policies over recency-based policies [3, 10, 28, 40].

The results in Figure 6 indicate that GD*(packets) consistently outperforms all other policies with respect to both HR and BHR. Table 7 illustrates an instance of the improvements and savings achieved through the use of GD* under the UC trace when cache size is 1GB.¹³

	Performance Improvement Achieved through GD*					
	HR	(Gain%)	BHR	(Gain%)	Packets	(Gain%)
LRU → GD*	33.3% → 50.1%	(50.4%)	31.1% → 37.6%	(20.9%)	68.8M → 86.1M	(25.1%)
GDS → GD*	36.5% → 50.1%	(37.3%)	31.4% → 37.6%	(19.9%)	71.1M → 86.1M	(19.7%)
LFU-DA → GD*	39.0% → 50.1%	(28.4%)	33.7% → 37.6%	(11.6%)	79.0M → 86.1M	(8.9%)

Table 7: Example of performance gains achieved through the use of GD*.

Sensitivity of GD* to Temporal Correlations: To evaluate the effect of β on the efficiency of GD*, we manually set the value of β used in GD*(packets) from 0.125 to 2. Figure 7 shows the resulting hit rates. Both HR and BHR are maximized when β is close to 0.5, which is very close to the actual value of β (between 0.46 and 0.51) for the RTP traces as shown in Table 5. This confirms that $u(p)^{1/\beta}$ is an appropriate quantification of the base value for a document.

Another observation from Figure 7 is that the effect of β is more pronounced when the cache is small—when the cache is large, the hit rates are closer to their upper bounds, so the sensitivity of GD* to β decreases.

¹³For the UC trace, 1 GB of cache represents approximately 2.5% of the total size of the unique documents.

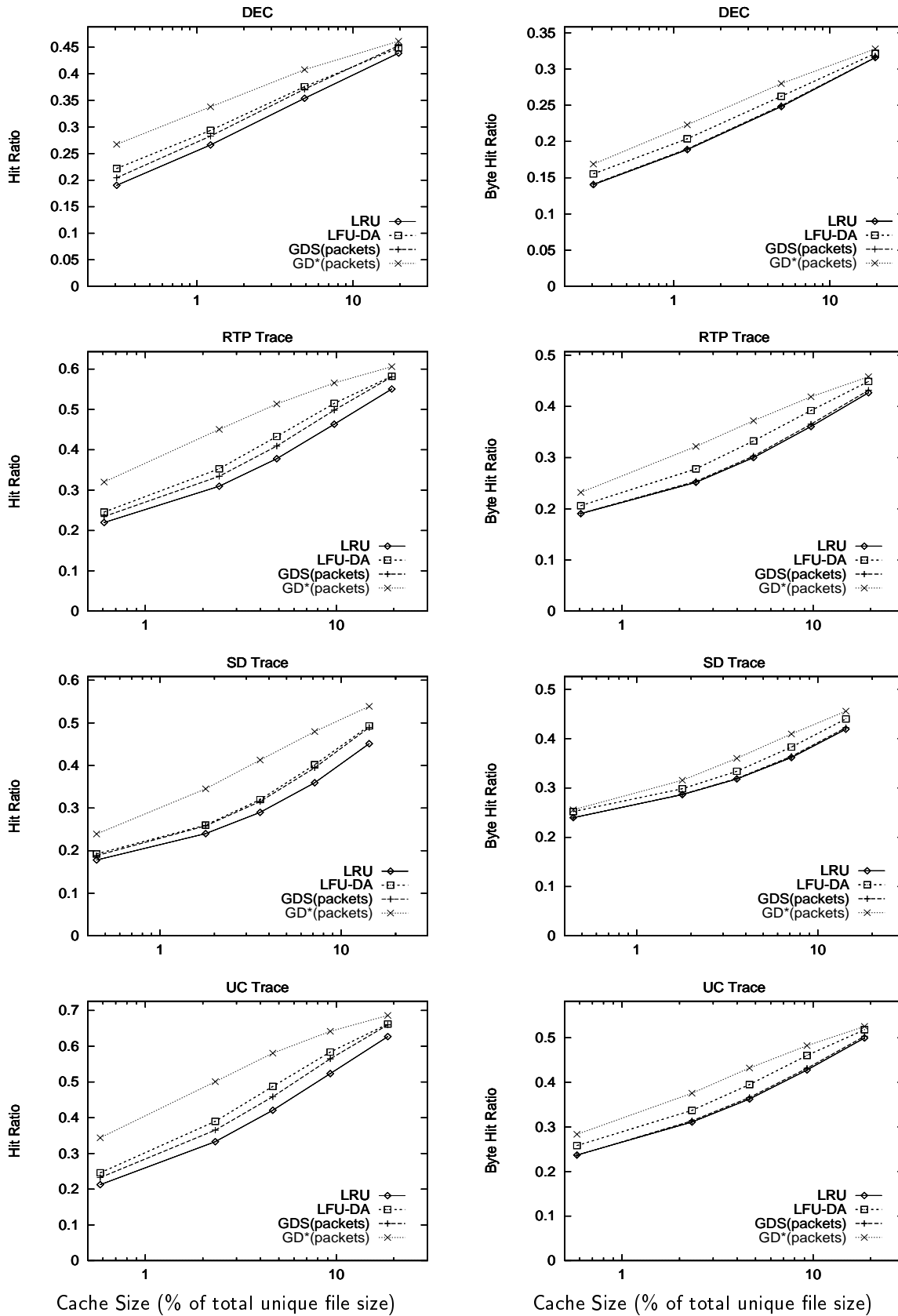
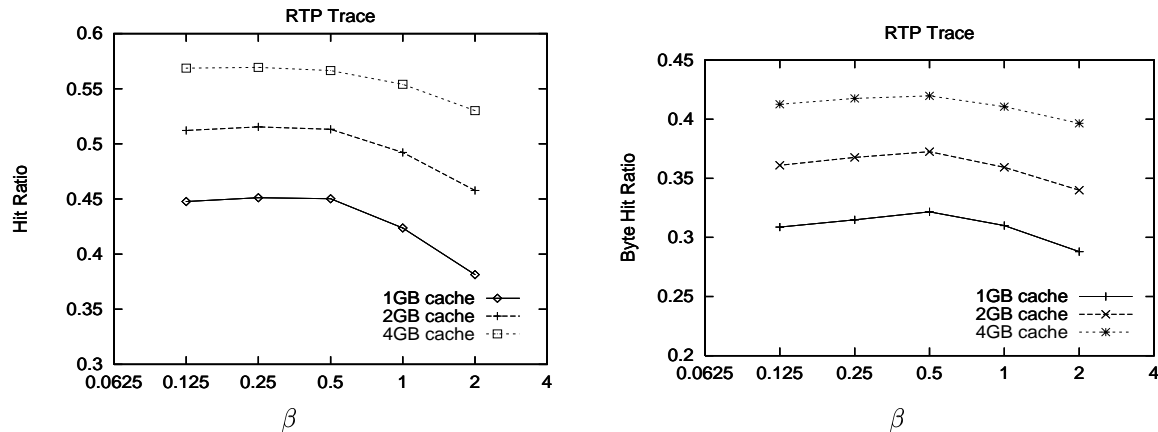


Figure 6: Hit rates when *cost* of document retrieval is equal to the number of packets transferred

Figure 7: Hit rates of GD*(packets) with different settings of β

5 Conclusion

In this paper, we have shown that there are two phenomena that contribute to temporal locality in Web request streams—namely popularity and temporal correlation of references. We have shown how to capture both of these dimensions using the parameters of two power laws—one characterizing popularity and the other characterizing the interarrival of references for equally-popular documents.

The implications of our characterization on Web cache replacement policies are two-fold. First, our study helps understand the relative importance of long-term frequency and short-term temporal correlation on replacement policies. Second, since popularity is the predominant cause of temporal locality (as our characterization indicates), a long-term frequency-based replacement policies (such as LFU) should be used as it is known to be online-optimal under an independent reference model. However, such a policy should be seasoned by allowing the replacement policies to be sensitive to the level of temporal correlation in the request stream.

We demonstrate this two-pronged approach to capturing temporal locality by proposing a novel cache replacement policy, which capitalizes on and adapts to the relative strengths of *both* long-term popularity and short-term temporal correlation. Simulation results show the superiority of this approach, and thus highlight the benefit of delineating between the two sources of temporal locality: popularity and temporal correlation of references.

References

- [1] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. In *Proceedings of the Fourth International World Wide Web Conference*, December 1995.
- [2] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing Reference Locality in the WWW. In *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS'96)*, December 1996.
- [3] Martin Arlitt, Ludmila Cherkasova, John Dille, Rich Friedrich, and Tai Jin. Evaluating Content Management Techniques for Web Proxy Caches. In *Proceedings of the 2nd Workshop on Internet Server Performance*, May 1999.
- [4] Martin Arlitt and Carey Williamson. Web server workload characteristics: The search for invariants. In *Proceedings of ACM SIGMETRICS'96*, May 1996.
- [5] Martin Arlitt and Carey Williamson. Internet Web servers: Workload characterization and performance implications *IEEE/ACM Transactions on Networking*, 5(5):631-644, October 1997.
- [6] Paul Barford, Azer Bestavros, Adam Bradley, and Mark Crovella. Changes in Web Client Access Patterns: Characteristics and Caching Implications. *World Wide Web*, 2(1): 15-28, 1999.
- [7] Azer Bestavros. WWW Traffic Reduction and Load Balancing Through Server-Based Caching. *IEEE Concurrency: Special Issue on Parallel and Distributed Technology*, 5(1):56-67, Jan-Mar 1997. IEEE Press.
- [8] Azer Bestavros and Carlos Cunha. Server-initiated document dissemination for the WWW. *IEEE Data Engineering Bulletin*, 19(3): 3-11, September 1996.
- [9] Azer Bestavros, Robert Carter, Mark Crovella, Carlos Cunha, Abdelsalam Heddaya, and Sulaiman Miradad. Application Level Document Caching in the Internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.
- [10] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of Infocom'99*, April 1999.
- [11] Mark Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of ACM SIGMETRICS'96*, May, 1999.
- [12] Pei Cao and Sandy Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems*, December 1997.
- [13] E. G. Coffman and P. J. Denning. *Operating systems theory*. Prentice-Hall, 1973.
- [14] Edith Cohen and Haim Kaplan. Exploiting regularity in Web traffic patterns for cache replacement. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, May 1999.
- [15] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Evaluating server-assisted cache replacement in the Web. In *Proceedings of ESA '98*, 1998.
- [16] Edith Cohen and Haim Kaplan. Caching documents with variable sizes and fetching costs: An LP-based approach. In *Proceedings of Symposium on Discrete Algorithms*, 1999.
- [17] Carlos Cunha, Azer Bestavros, and Mark Crovella. Characteristics of WWW Client-based Traces. Technical Report BUCS95-010, April 1995.
- [18] P. Denning and S. Schwartz. Properties of the working set model. *Communications of the ACM*, 15(3):191-198, 1972.
- [19] Digital Equipment Corporation. <ftp://ftp.digital.com/pub/DEC/traces/proxy/>.
- [20] Li Fan, Pei Cao, and Quinn Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *Proceedings of ACM SIGMETRICS'99*, May, 1999.

-
- [21] Steven D. Gribble and Eric A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace In *Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems*, December 1997.
- [22] Mor Harchol-Balter, Mark E. Crovella, and Cristina D. Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, Special Issue on Software Support for Distributed Computing, September 1999.
- [23] A. Heddaya and S. Mirdad. WebWave: Globally load balanced fully distributed caching of hot published documents. In *Proc. 17th IEEE Intl. Conference on Distributed Computing Systems, Baltimore, Maryland, USA*, May 1997.
- [24] Sandy Irani. Page replacement with multi-size pages and applications to Web caching. In *Proceedings of the 12th annual ACM symposium on Theory of computing*, 1997.
- [25] Shudong Jin and Azer Bestavros. Popularity-aware GreedyDual-Size algorithm for Web access. Technical Report BUCS99-009, August 1999.
- [26] Balachander Krishnamurthy and Craig E. Wills. Proxy cache coherency and replacement—towards a more complete picture. In *Proceedings of the 19th IEEE ICDCS'99*, June 1999.
- [27] Donghee Lee, Jongmoo Choi, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. On the existence of a spectrum of policies that subsumes the LRU and LFU policies In *Proceeding of the 1999 ACM SIGMETRICS Conference*, May 1999.
- [28] P. Lorenzetti, L. Rizzo and L. Vicisano. Replacement policies for a proxy cache. Technical Report LR-960731, Univ. di Pisa.
- [29] Evangelos Markatos. Main memory caching of web documents. In *Proceedings of the Fifth International Conference on the WWW*, 1996.
- [30] R. Mattson, J. Gecsei, D. Slutz, and I. Traiger. Evaluation techniques and storage hierarchies. *IBM Systems Journal*, 9:78–117, 1970.
- [31] National Laboratory for Applied Network Research. <ftp://ircache.nlanr.net/Traces/>.
- [32] Elizabeth J. O'Neil, Patrick E. O'Neil, Gerhard Weikum. The LRU-K Page Replacement Algorithm For Database Disk Buffering. In *Proceedings of ACM SIGMOD*, 1993: 297-306.
- [33] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predicative prefetching to improve World Wide Web latency. In *Proceedings of ACM SIGCOMM*, 1996.
- [34] Squid Internet Object Cache. <http://squid.nlanr.net/Squid>.
- [35] Peter Scheuermann, Junho Shim, and Radek Vingralek. A case for delay-conscious caching of Web documents. In *Proceedings of the 6th International WWW Conference*, 1997.
- [36] Junho Shim, Peter Scheuermann, and Radek Vingralek. Proxy cache design: algorithms, implementation and performance. *IEEE Transactions on Knowledge and Data Engineering*, 1999.
- [37] Alan Jay Smith. Cache memories. *Computing Surveys*, 14(3):473–530, September 1982.
- [38] Jeffrey Spirn. Distance string models for program behavior. *IEEE Computer*, 13(11), November 1976.
- [39] W. Williams, M. Abrams, C. R. Standbridge, G. Abdulla and E. A. Fox. Removal policies in network caches for World-Wide Web documents. In *Proceedings of the ACM SIGCOMM*, 1996.
- [40] R. Wooster and M. Abrams. Proxy caching that estimates page load delays. In *Proceedings of the 6th International WWW Conference*, 1997.
- [41] Neal E. Young. On-line caching as cache size varies. In *Proceedings of Symposium on Discrete Algorithms*, 1991.
- [42] G. K. Zipf Human behavior and the principles of least effort. Addison-Wesley, Cambridge MA, 1929.

A Relationship between Popularity and Reference Interarrivals

Theorem: If the distribution of document popularity in a request stream asymptotically follows a power law with parameter α , where $0.5 \ll \alpha \leq 1$, then the distribution of reference interarrivals in a random permutation of that request stream can be characterized asymptotically using a power law with parameter $(2 - \frac{1}{\alpha})$.

Proof: Let k denote the number of references to a page and $N(k)$ be the number of pages referenced k times.

Let $F(t)$ denote the number of instances whereby two references to the same document are separated by t units of time, where $0 < t < 1$ is the normalized time spanning the randomly-permuted request stream.¹⁴ Notice that $F(t)$ is only a constant multiply of the probability distribution of interarrival times.

Let $P(t, k)$ denote the number of instances whereby two references to the same k -popular document are separated by t units of time, where $k \geq 1$ (i.e. we are only interested in documents referenced at least twice).

We make the following two observations:

1. $N(k) \sim k^{-1-\frac{1}{\alpha}}$, where α is the parameter of the power law governing the Zipf-like distribution of document popularity. This observation follows by noting that $N(K)$ can be obtained by transposing the x-axis and y-axis of the popularity distribution and taking the derivative of the resulting function. Let $N(k) = Ck^{-1-\frac{1}{\alpha}}$.
2. $P(t, k) \sim ke^{-kt}$. The interarrivals of independent (i.e. randomly scattered) references to an object follow an exponential distribution with a mean of $1/k$. Normalizing it, we get $P(t, k) = \frac{ke^{-kt}}{1-e^{-k}}$. Since $1-e^{-k}$ approaches unity so fast when k increases, it follows that we can take $P(t, k) \approx ke^{-kt}$, without affecting the asymptotic property we are attempting to establish.

For a document accessed k times, the number of intervals appearing in the reference stream is $(k-1)$. From the definition of $P(t, k)$, we get that the total number of t -wide intervals is given by $(k-1)P(t, k) = k(k-1)e^{-kt}$. Thus, the total number of such intervals for all documents in the reference stream is $N(k)(k-1)P(t, k) = Ck^{-\frac{1}{\alpha}}(k-1)e^{kt}$.

We compute $F(t)$ by summing up the value of $P(t, k)$ for all k 's:

$$\begin{aligned} F(t) &= \sum_k N(k)(k-1)P(t, k) \\ &= \sum_k Ck^{-\frac{1}{\alpha}}(k-1)e^{kt}, k \geq 2. \end{aligned} \tag{1}$$

¹⁴In other words, $t = 0$ denotes the beginning of the trace and $t = 1$ denotes the end of the trace.

Replacing the summation in equation 1 with an integral, we obtain the following expression for $F(t)$.

$$\begin{aligned} F(t) &\approx \int_2^\infty C k^{-\frac{1}{\alpha}} (k-1) e^{kt} dk \\ &= C \frac{\Gamma(2 - \frac{1}{\alpha}, 2t) - t\Gamma(1 - \frac{1}{\alpha}, 2t)}{t^{2-1/\alpha}} \end{aligned} \quad (2)$$

In equation 2, $\Gamma(a, x) \equiv \int_x^\infty t^{a-1} e^{-t} dt$ is the incomplete Gamma function. Comparing it to $\Gamma(2-1/\alpha, 2t)$ when t is small, $t\Gamma(1-1/\alpha, 2t)$ is insignificant for any $0.5 \ll \alpha \leq 1$. Moreover, the complete Gamma function $\Gamma(2 - \frac{1}{\alpha}) \equiv \Gamma(2 - \frac{1}{\alpha}, 0)$ is analytic since $2 - \frac{1}{\alpha} > 0$. When t is small, $\Gamma(2 - \frac{1}{\alpha}, 2t)$ is close to $\Gamma(2 - \frac{1}{\alpha})$, a constant. Therefore in this case:

$$\begin{aligned} F(t) &\approx C \frac{\Gamma(2 - 1/\alpha, 2t) - t\Gamma(1 - 1/\alpha, 2t)}{t^{2-1/\alpha}} \\ C &\approx \frac{\Gamma(2 - 1/\alpha)}{t^{2-1/\alpha}} \sim \frac{1}{t^{2-1/\alpha}}, t \ll 1. \end{aligned} \quad (3)$$

When t is not very small, the second item is non-negligible, so there will be a visible ‘‘dip’’ in the log-log scale plot (see footnote ⁷). Figure 8 displays $F(t)/C$ for different values of α .

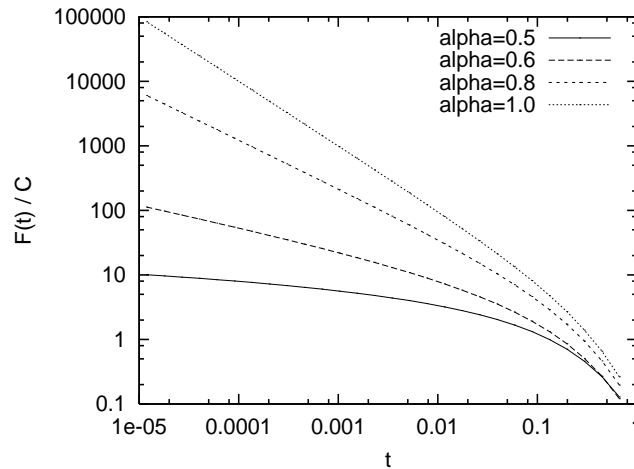


Figure 8: $F(t)/C$ with parameter α .

Note: The interarrival probability distribution function shown in Figure 3 has the same shape as the curves in Figure 8. The log-log scaled plots exhibit nearly straight line when t is small. This demonstrates the inherent quantitative relationship between power-law popularity distribution and interarrival distribution spelled out in Theorem A.

B The GreedyDual* Cache Replacement Algorithm

```

Algorithm GreedyDual*

 $L \leftarrow 0.0$ 
for each request for document  $p$  do
  if  $p$  is in cache
    then  $H(p) \leftarrow L + (\frac{f(p) \times c(p)}{s(p)})^{1/\beta}$ 
  else fetch  $p$ 
    while there is not enough free cache for  $p$ 
      do  $L \leftarrow \min\{H(q) | q \text{ is in cache}\}$ 
      Evict the minimum  $q$ 
     $H(p) \leftarrow L + (\frac{f(p) \times c(p)}{s(p)})^{1/\beta}$ 

```

Implementation Details

Our implementation of GreedyDual* maintains a priority queue with key $H(p)$. Handling either a hit or a replacement requires $O(\log n)$ time. Thus, it has the same overhead as GreedyDual-Size.

In our implementation, the constant β is estimated in an online fashion. This is done by keeping the number of references at different intervals for equally popular documents and computing β using a least-square fit. The value of β for a given proxy cache was found to be stable over time.

All frequency-aware replacement policies face a common problem. They must keep track of reference counts, in order to guarantee the accuracy of frequency information. Obviously, it is unrealistic for an algorithm to keep all the reference counters of evicted documents. However, it is possible to only keep a subset of the the reference counters. In our simulation, the space for reference counters was subject to the following constraints: (1) less than 1% of the cache keeps the reference counters of evicted documents, and (2) the total number of counters is less than 512K. This is done through the use of an $O(1)$ replacement algorithm.