Characterizing Reference Locality in the WWW*

Virgílio Almeida†§ (virgilio@bu.edu) Azer Bestavros† (best@bu.edu) Mark Crovella† (crovella@bu.edu) Adriana de Oliveira‡ (dri@dcc.ufmg.br)

TR-96-11

Department of Computer Science
Boston University
Boston, MA 02215

Abstract

As the World Wide Web (Web) is increasingly adopted as the infrastructure for large-scale distributed information systems, issues of performance modeling become ever more critical. In particular, locality of reference is an important property in the performance modeling of distributed information systems. In the case of the Web, understanding the nature of reference locality will help improve the design of middleware, such as caching, prefetching, and document dissemination systems. For example, good measurements of reference locality would allow us to generate synthetic reference streams with accurate performance characteristics, would allow us to compare empirically measured streams to explain differences, and would allow us to predict expected performance for system design and capacity planning.

In this paper we propose models for both temporal and spatial locality of reference in streams of requests arriving at Web servers. We show that simple models based only on document popularity (likelihood of reference) are insufficient for capturing either temporal or spatial locality. Instead, we rely on an equivalent, but numerical, representation of a reference stream: a stack distance trace. We show that temporal locality can be characterized by the marginal distribution of the stack distance trace, and we propose models for typical distributions and compare their cache performance to our traces.

We also show that spatial locality in a reference stream can be characterized using the notion of self-similarity. Self-similarity describes long-range correlations in the dataset, which is a property that previous researchers have found hard to incorporate into synthetic reference strings. We show that stack distance strings appear to be stongly self-similar, and we provide measurements of the degree of self-similarity in our traces. Finally, we discuss methods for generating synthetic Web traces that exhibit the properties of temporal and spatial locality that we measured in our data.

Keywords: Self-similarity; Long-range dependence; Distance strings; Reference locality; Caching; Performance modeling.

*This work has been partially supported by NSF (grants CCR-9308344 and CCR-9501822) and by CNPq-Brazil.

†Computer Science Department, Boston University, Boston, MA 02215, USA.
‡Depto. de Ciencia da Computacão, Universidade Federal de Minas Gerais, Belo Horizonte, MG 30161, Brazil.
§On sabbatical at Boston University from Universidade Federal de Minas Gerais.

1 Introduction

The principle of locality of reference has very important consequences for computer systems design. Reference streams exhibiting *temporal* locality can benefit from caching; and reference streams exhibiting *spatial* locality can benefit from prefetching. The application of these principles in, *e.g.*, memory systems is well understood [15, 31].

In order to apply these principles to the design of World Wide Web (Web) caching and prefetching systems, it's important to characterize the degree of temporal and spatial locality present in typical Web reference streams. Good measurements of these properties would allow us to generate synthetic reference streams with accurate performance characteristics, would allow us to compare empirically measured streams to explain differences, and would allow us to predict expected performance for system design and capacity planning. Previous research in the locality properties of symbolic reference streams have shown the benefits of transforming the reference stream into an equivalent, but numerical, representation: a *stack distance* stream [25]. This transformation preserves all of the information of the original trace except for the specific reference names, and so it is invertible, allowing reconstruction of the original trace (albeit with synthetic names). Two properties of the stack distance stream can be used to measure the notions of temporal and spatial locality: *marginal distribution,* and *correlation structure* [25, 30]. The marginal distribution measures the likelihood that two references to the same object are separated by some number of intervening references in the stream. The correlation structure of the stream measures the likelihood that we can predict future references based on the stream of past references.

In this paper we use quantitative methods based on stack distance to measure both the temporal and spatial locality properties for Web reference streams. We characterize reference streams collected at four Web servers, propose a model for marginal distribution of stack distance (and thus temporal locality) and show typical ranges of the relevant parameters. We also propose a model for spatial locality of our reference streams based on the correlation structure of stack distances. Using these measurements we show how to generate synthetic reference streams of arbitrary length whose performance properties mimic those of our empirically measured data.

A novel feature of our approach is the use of statistical *self-similarity* to capture the correlation structure of the reference stream. That is, we show evidence that Web stack distance streams are statistically self-similar and that this property can be used to explain aspects of the reference stream that are the result of spatial locality. The presence of self-similarity in the reference stream means that correlations between object references can occur at widely varying timescales (*long-range dependence*). A timeseries with long-range dependence appears much more "bursty" than series with only short-range or no dependence — which agrees with generally observed characterizations of reference locality. Long-range dependence in the stack distance series is also related to previously noted fractal properties of cache miss patterns [34], and we interpret prior work in fractal reference patterns in terms of our observations.

This method of measuring spatial locality also provides an interpretation for the notion of spatial locality when reference streams are composed of symbols rather than numbers. We consider object A to be close to object B if there is a high likelihood that a reference to A will immediately follow a reference to B. That is, the "distance" between objects is the inverse of the transition probability of object references considered as a first-order Markov chain. While this definition does not correspond to our usual notions of a norm (it is not symmetric), it does capture the important property of spatial locality for the purposes of prefetching: the likelihood that prefetching will be successful.

Our methods focus on the locality properties of reference streams without regard to document size. Thus in our simulations we consider caches to be sized to hold a fixed number of documents rather than a fixed amount of data. This assumption stems from the fact that we are interested in locality, which is a property of reference streams independent of any cache size.[2]

In the remainder of this section we describe the data on which our measurements and trace simulations are based. In section 2, we characterize the popularity of Web documents and show that a cache reference model based on popularity is not expressive enough to capture the temporal and spatial locality of reference properties exhibited in actual Web traces. Evidence of these locality of reference properties is presented in section 3. In sections 4 and 5, we characterize these properties and present statistical models that capture these characteristics. In section 6, we examine related work. We conclude in section 7 with a summary and directions for future work.

Data Collection

The Web is a large-scale distributed information system based on a client-server architecture. Thus, the workload viewed from a server standpoint consists of a number of requests originated at many different clients. In order to analyze reference locality, we examined the access logs for different Web servers, namely: the NCSA Web server located at the National Center for Supercomputing Applications (NCSA), the SDSC Web server at the San Diego Supercomputer Center (SDSC), the EPA Web server located at Research Triangle Park, NC and the Web server at the Computer Science Department at Boston University (BU). In the case of the NCSA, the data refers only to one server (Costello). The SDSC and EPA logs are available at the Internet Traffic Archives [19], the NCSA log was obtained after contacting the staff at the site and the BU logs were collected at the departmental Web server.

The logs have one line of information per request processed by the server. Each line contains the name of the host making the request, the timestamp the request was made, the filename of the requested object and size in bytes of the reply. Table 1 summarizes the statistics about the logs of the four Web servers.

Item	**NCSA**	**SDSC**	**EPA**	**BU**
Access Log Duration	1 day	1 day	1 day	2 weeks
Access Log Start Date (in 1995)	Dec 19	Aug 22	Aug 29	Oct 08
Total Requests	46,955	28,338	47,748	80,518
Unique Requests	4,851	1,267	6,518	4,471

Table 1: Summary of Access Log Data

2 Characterizing Document Popularity

The highly uneven popularity of various Web documents is a well-documented phenomenon that has been exploited in previous caching, replication and dissemination studies. In these studies, it was shown that "Popular files are very popular" [5, 18]. In [12], Cunha, Bestavros, and Crovella characterized the popularity of *Web documents requested by clients* and confirmed the applicability of Zipf's law [36, discussed in [23]] to Web documents. Zipf's law was originally applied to the

[2] Since we have shown in previous work [12] that document size is correlated with expected reference frequency, this assumption is somewhat limiting for direct use in cache performance analysis. For that reason we are currently extending this work to include document sizes in our analysis. Nonetheless we believe that the current methods provide a simple and effective way to capture both the spatial and temporal locality of Web reference streams.

relationship between a word's popularity in terms of rank and its frequency of use. It states that if one ranks the popularity of words used in a given text[3] (denoted by ρ) by their frequency of use (denoted by P) then

$$P \sim 1/\rho.$$

Note that this distribution is a parameterless hyperbolic distribution. *i.e.,* ρ is raised to exactly -1, so that the n^{th} most popular document is exactly twice as likely to be accessed as the $2n^{\mathrm{th}}$ most popular document.

Our data shows that Zipf's law applies quite strongly to *Web documents serviced by Web servers.* This is demonstrated in Figure 1 for all 2135 documents accessed in the BU trace. The figure shows a log-log plot of the total number of references (Y axis) to each document as a function of the document's rank in overall popularity (X axis). The tightness of the fit to a straight line is strong ($R^2 = 0.99$), as is the slope of the line: -0.85 (shown in the figure). Thus the exponent relating popularity to rank for Web documents is very nearly -1, as predicted by Zipf's law.

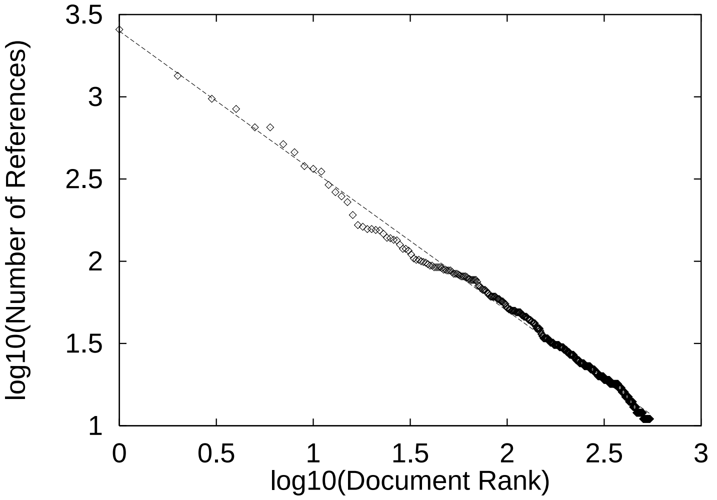

Figure 1: Zipf's Law Applied to Web Documents

Our measurements suggest that Zipf's law applies, not only to single requests for documents, but also to request *strides.* A request stride is a sequence of requests from the *same* client where the time between successive requests in that stride is not larger than a given *stride threshold.* Figure-2 shows a log-log plot of the total number of references (Y axis) to strides of length k (for $k = 1, 2, 3$) as a function of the stride's rank in overall popularity (X axis). The trace used for this plot is the BU trace and the stride threshold is set at 10 seconds. Again, the tightness of the fit for the family of curves in Figure-2 to straight lines slope close to -1 suggest that Zipf's law does apply to sequences of requests.

Given the above observations, one possible access model [17, 2] is to generate document requests according to the popularity profile of documents. We call this model the *Zipf-based model.* To test the validity of this model we measured the miss rate at the server when requests are generated

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

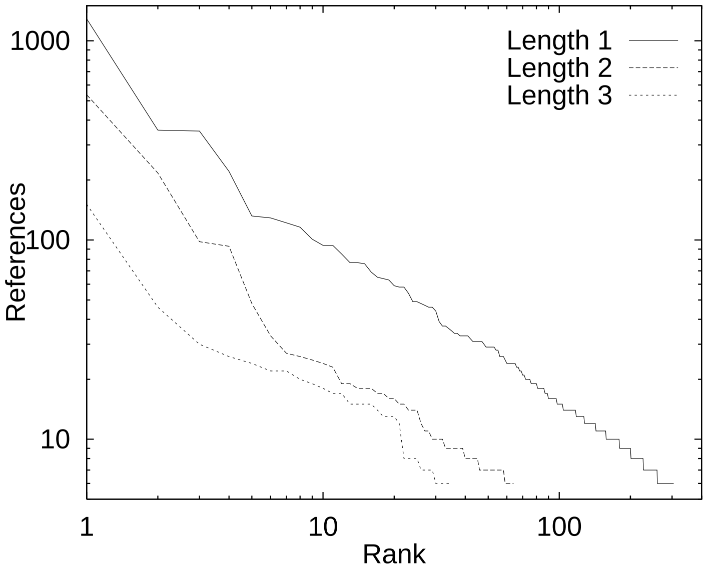

Figure 2: Zipf's Law Applied to Sequences of Web Documents

synthetically, to satisfy the observed popularity profile of documents at BU, and compared these measurements with the measurements obtained when the actual BU trace is used to drive the simulation. The synthetic trace was generated by applying a random permutation to the actual trace for BU. The result of this comparison is shown in Figure 3.

Obviously, using popularity as the basis of a cache model for Web documents does not yield acceptable results. Figure 3 shows that for a cache size of 400 items, the miss rate that results from a Zipf-based cache model is nearly 33% less than the actual miss rate.

The main weakness of a Zipf-based model is that it does not capture the potential locality of reference properties that may be present in actual traces. In particular, while the Zipf-based synthetic trace preserves the general popularity profile of documents, it fails to capture the temporal and spatial locality of reference properties. In the next section we show evidence that these properties indeed exist in Web traces, and in sections 4 and 5, we provide simple, yet expressive statistical models thereof.

3 Evidence of Reference Locality

3.1 Temporal Locality

Temporal locality implies that recently accessed documents are more likely to be referenced in the near future. Based on the concepts proposed in [25, 32], we define a stack distance model that captures the temporal locality relationships present in a request stream. Let us consider a reference stream $R_t = r_1, r_2, ..., r_t$, where r_t denotes the name of the object requested at virtual time t. Our unit of time will be one request, so that virtual time t indicates that t requests have already arrived at a server. Let us also define the LRU stack S_t, which is an ordering of all objects of a server by recency of usage. Thus, at virtual time t, the LRU stack is given by:

$$S_t = \{o_1, o_2, \ldots, o_N\} \tag{1}$$

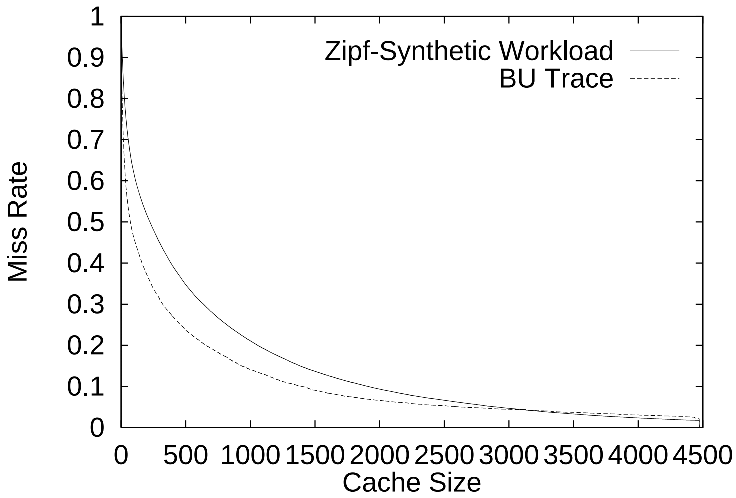

Figure 3: Miss rate for a Zipf-based synthetic workload and for an actual trace

where $o_1, o_2, \ldots, o_N$ are objects of the server and o_1 is the most recently accessed object, o_2 the next most recently referenced, etc. If o_1 is the most recently accessed document, then $r_t = o_1$. Whenever a reference is made to an object, the stack must be updated. Considering that $r_{t+1} = o_i$, then the stack becomes $S_{t+1} = \{o_i, o_1, o_2, ..., o_{i-1}, o_{i+1}, ..., o_N\}$. Suppose now that $S_{t-1} = \{o_1, o_2, ..., o_N\}$ and $r_t = o_i$. Then, we could say that request r_t is at distance i in stack S_{t-1}. Let d_t denote the stack distance of the document referenced at time t. We then have the following relation:

$$\text{if } r_t \;=\; o_i \text{ then } d_t \;=\; i \tag{2}$$

Thus, for any request string $R_t = r_1, r_2, ..., r_t$ there is a corresponding distance string $\delta = d_1, d_2, ..., d_t$. We can then consider that the distance string and the request string are equivalent in terms of reference information. Since the distance string reflects the pattern in which Web users request documents from a server rather than the actual identity of the documents, it will be used as our model for object reference pattern.

The marginal distribution of distance probabilities is an indication of temporal locality because it measures the number of intervening references between two references to the same object. Small stack distances result from frequent references to a document. In other words, the documents referenced most frequently tend to be those close to the top of the stack.

As a measure of the temporal locality present in our traces, we compare the average stack distance found in the BU trace with that of the scrambled BU trace. The results, shown in table 2, indicate that the average stack distance is apparently greater for the scrambled trace, meaning that temporal locality has been decreased.

3.2 Spatial Locality

The existence of spatial locality of reference could be established by comparing the total number of unique sequences observed in a trace and the total number of unique sequences that would be found in a random permutation of such a trace. Notice that a random permutation of a trace (henceforth

Item	BU-original	BU-scrambled
Mean Stack Distance	479.798	645.586
Standard Deviation	941.430	968.840

Table 2: Evidence of Temporal Locality

called a *scrambled* trace) *preserves* the popularity profile of the various documents in that trace, but that it *destroys* the spatial locality of reference that may exist in such a trace (by uncorrelating the sequence of references). If references are indeed correlated, then one would expect the total number of unique sequences observed in a trace to be much less than the total number of unique sequences observed in a scrambled trace.

Figure 4 shows the total number of unique sequences (Y axis) of length k (X axis) observed in the first week (the first 32,839 references) of the BU trace, which we call the BU1 server trace. Three cases are shown. The top curve shows the total number of unique k-long sequences observed in a scrambled BU1 trace. The middle curve shows the total number of unique k-long sequences observed in the BU1 trace. Since traces arriving at the server reflect an interleaving of a number of individual client traces, we plot in the bottom curve the total number of unique k-long sequences observed when separately considering individual clients in the BU1 trace.

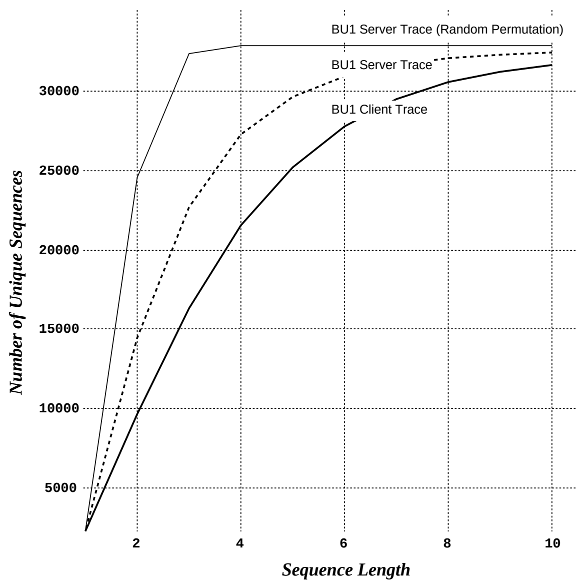

Figure 4: Unique sequences observed in the BU trace

All three curves show an increase in the total number of unique sequences as k increases,[4] with the randomized trace showing the steepest increase and the client trace showing the slowest increase. These results confirm the existence of considerable spatial locality in client access patterns. For example, while up to 32,327 different sequences of length 3 are observed in the scrambled BU1

[4]The reason that all three curves level-off is related to the limited length of the trace. In particular, it is impossible to observe any more than $l-k$ unique k-long sequences in a trace of length l. For our experiments $l = 32,839$—hence the observed ceiling.

trace, only 16,114 (more than 50% less) different sequences of length 3 are observed in the BU1 client traces. This spatial locality is "diluted" to some extent (but still very much evident) in the BU1 server trace—22,680 (about 30% less) different sequences of length 3 are observed in the BU1 server trace.

4 Characterizing Temporal Locality

Since it is clear that real traces and scrambled traces differ significantly in the aggregate properties of their stack distance traces, we examine the marginal distributions of stack distances.

As discussed in Section 1, the marginal distribution of stack distance captures the temporal locality present in the trace. That is, if D is a random variable corresponding to stack distance (with distribution function F_D) and $M(C)$ is the miss rate of a cache that can hold C files (using the same replacement algorithm that was used to generate the stack distance trace), then:

$$P[D > C] = 1 - F_D(C) = M(C)$$

Thus, knowledge of the complementary distribution function $1 - F_D$ provides enough information to predict the performance of a cache of any size for the given trace.

To estimate distributions for the four traces we consider, histograms are shown in Figure 5. This figure shows that most of the stack distances are close to 1 (the traces show good temporal locality) and yet there are long tails to the distributions. We considered a number of long-tailed distributions to model this data. In particular we rejected the Pareto distribution ([11]) because although the tails are long, they do not seem to follow a power-law. The data were found to be best fit by a *lognormal* distribution. Data that follows a lognormal distribution has the property that its logarithm is normally distributed. The resulting distribution has a very long tail but is not *heavy-tailed* in the strict sense of following a power-law in the tails.

To illustrate the use of lognormal distributions to model stack distances, Figure 6 shows the empirically measured distributions and the fitted distributions. Each plot shows a histogram of the base 10 logs of the stack distance data; in addition the plot shows the shape of the corresponding fit of a lognormal distribution to the data. While the data sets may appear to vary from normal to some degree, in reality these differences have been magnified by the log-transformation process. In fact the fit to the data is quite close as we will show below.

The parameters used to generate the fitted distributions shown in Figure 6 are given in Table 3. The $\hat{\mu}$ and $\hat{\sigma}$ columns show the mean and standard deviation *of the base 10 log* of the datasets. The table shows that for the traces we studied, the $\hat{\sigma}$ varies only slightly. Most of the variation between the datasets occurs in the $\hat{\mu}$ parameter.

Trace	$\hat{\mu}$	$\hat{\sigma}$
BU	1.829	0.947
NCSA	1.730	0.836
SDSC	1.568	0.827
EPA	2.150	0.921

Table 3: Parameters of the Lognormal Distribution for Traces

The ability of lognormal distributions to predict actual miss rates in finite caches is shown in Figure 7. As discussed above we plot $P[D > C]$ since it is equivalent to miss rate. In each plot,

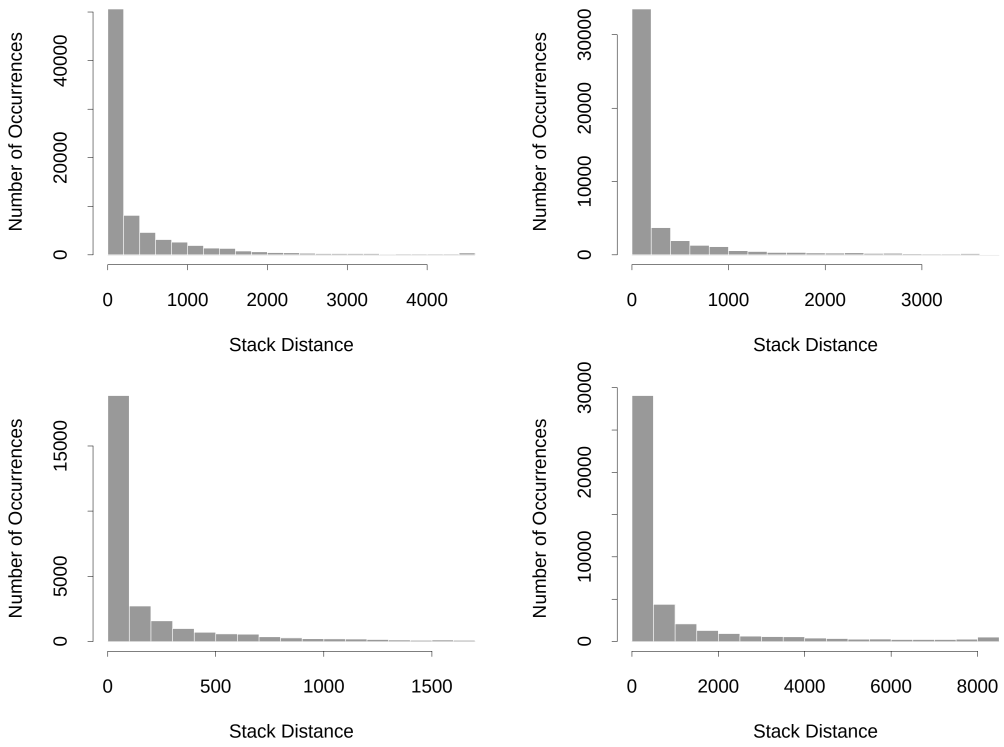

Figure 5: Empirical Stack Distance Distributions (Upper: BU, NCSA; Lower: SDSC, EPA)

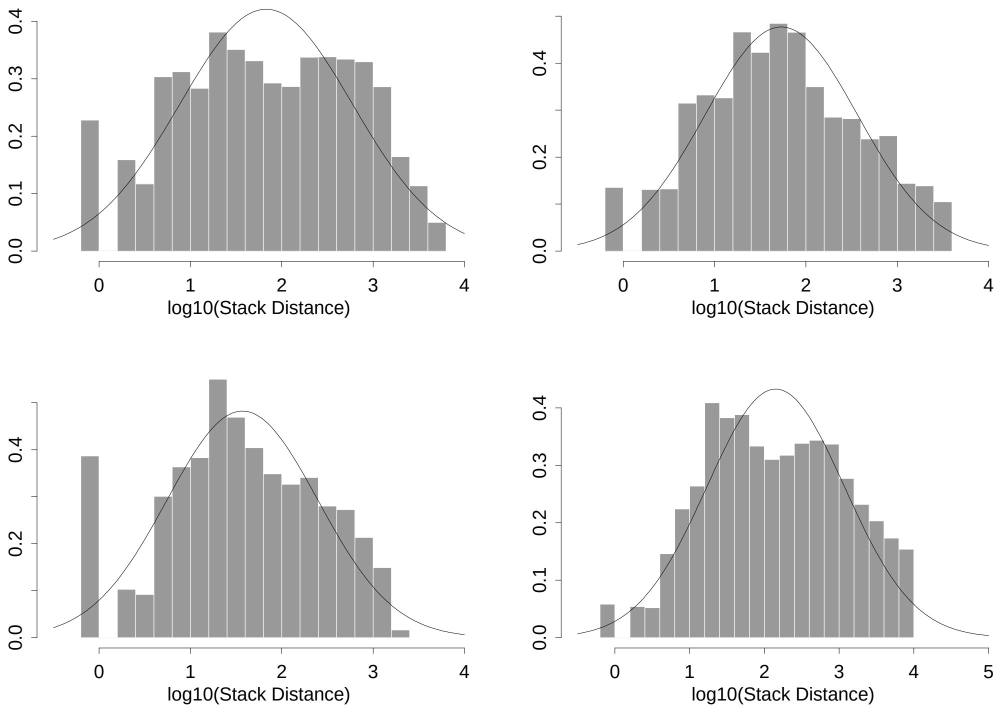

Figure 6: Lognormal Distributions of Stack Distance Traces (Upper: BU, NCSA; Lower: SDSC, EPA)

we show the predicted miss rate using the lognormal stack distance model and the actual miss rate obtained from the data. The agreement between model and data in each case is good, except in the case of the EPA trace, which has very poor locality.

A comparison of the traces in terms of their fitted parameters is shown in Figure 8. This figure shows how temporal locality (measured by miss rate) varies across the traces, and the corresponding parameter values that capture that locality. The figure shows that as the temporal locality of a trace decreases, the mean of lognormal distribution increases (from about 1.6 up to 2.2) and the standard deviation of the lognormal distribution also tends to increase, although to a lesser degree (from 0.83 up to 0.95).

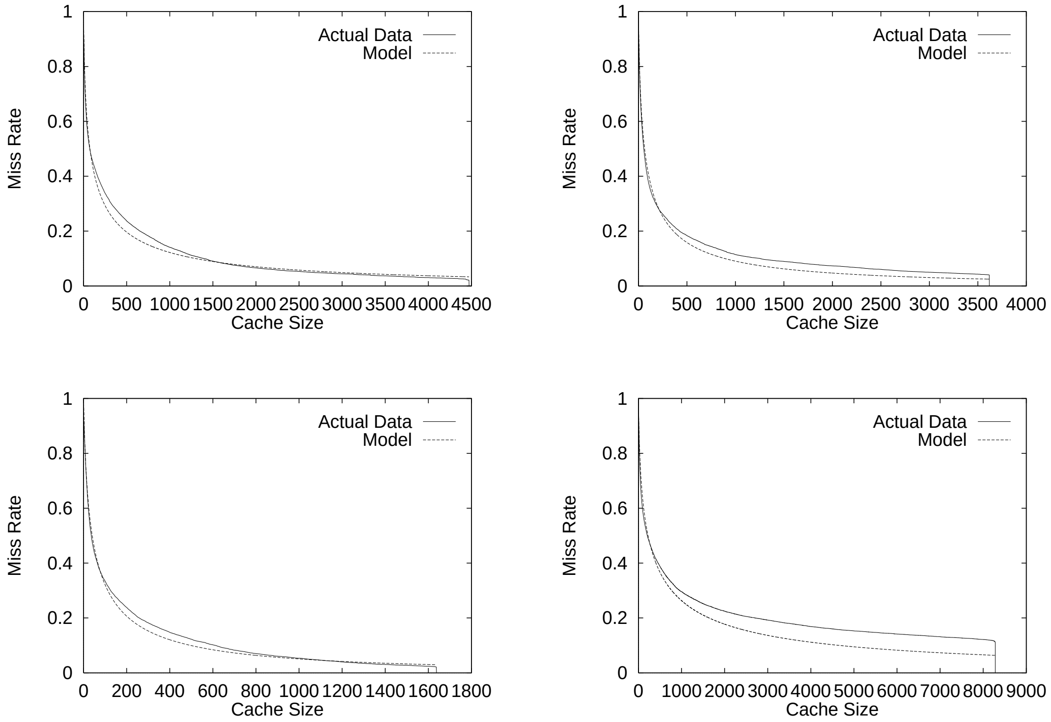

Figure 7: Actual and Predicted Miss Rates using Lognormal Models (Upper: BU, NCSA; Lower: SDSC, EPA)

5 Characterizing Spatial Locality

Having characterized the temporal locality present in our traces, we now present models for spatial locality. Previous work has shown that the correlation structure of the stack distance stream is useful for modeling spatial locality [30]; however previous attempts to model the correlation structure of reference strings have used models with only short-range dependence, such as finite first-order Markov chains. As pointed out in [32], such short-range dependent models are not capable of capturing the long-term structure of reference strings, which includes phase change behavior.

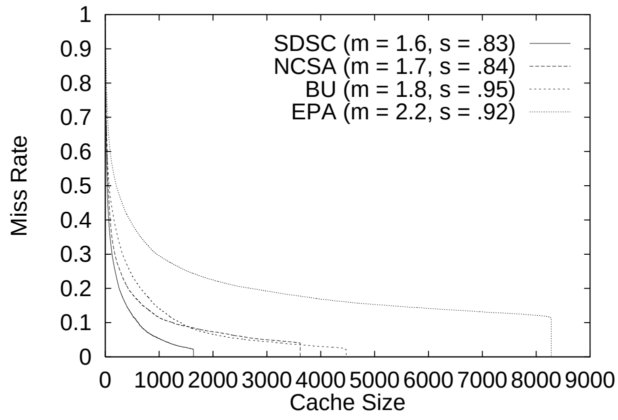

Figure 8: Comparison of Temporal Locality and Parameters

5.1 Measuring Stack Distance Self-Similarity

To capture the long-range dependence in stack distance strings we employ the statistics of self-similarity. We say that a series Y_t is H-self-similar if for any positive stretching factor c, the rescaled process with time scale ct, which is $c^{-H} Y_{ct}$, is equal in distribution to the original process Y_t. That is,

$$Y_t =_d c^{-H} Y_{ct} \quad \text{for all } c, t > 0. \tag{3}$$

This means that the series Y_t has fractal-like properties: detail is present at all scales. As a result, using only the single parameter H (related to the fractal dimension) we can capture variation at a wide range of scales.

Equation 3 applies to a motion process, such as Brownian motion (for which $H = 0.5$). For stationary processes like our stack distance traces, an equivalent definition is used, which relates the stationary process to its corresponding motion process (formed by taking successive sums of the stationary process):

> **Definition.** Let $\{X_t\}$ be a stationary process with continuous time parameter t. Then we say that X_t is H-self-similar if for any positive aggregation factor m the series obtained by m-aggregating X_t is equal in distribution to X_t. That is:
>
> $$X_t =_d m^{-H} \sum_{i=(t-1)m+1}^{tm} X_i \quad \text{for all } m \in N, t > 0 \tag{4}$$

Because of the correlations present on all scales that is implied by Equation 4, stationary self-similar processes also have the property of *long-range dependence.* A long-range dependent series typically has "bursts" (values above or below the mean) that occur on all time scales. That is, regardless of the scale at which the series is viewed, bursty patterns can be observed. This property is in strong contrast to short-range dependent models (such as Markovian processes or Poisson processes) which appear smooth at long timescales.

The property of burstiness at all timescales is noticeably present in stack distance series; it is this property that has been difficult to capture in the past using only short-range dependent models. An example of this effect is shown in Figure 9. The figure compares the BU stack distance trace with a trace formed by first randomly scrambling the BU dataset before calculating stack distances. Note that both traces consist of exactly the same set of file references; however in the case of the scrambled dataset the spatial locality has been destroyed.

The figure shows the two stack distance traces at varying levels of detail. In the bottom graphs each point in the series is formed by summing 5 consecutive points (in non-overlapping blocks) from the original traces. Moving up the next series is formed by summing 5 points from the lower series, and so on until at the top level each data point consists of the sum of 625 data points from the original stack distance series. In each plot we have only plotted 128 points so as to show equivalent levels of detail; at the top, this corresponds to the entire series, at lower levels, the plots are over ranges that are selected arbitrarily. When summing data in this way, short range dependences are averaged out; if the series is long-range dependent, bursts will still remain. This can be seen by comparing the scrambled trace (on the right) with the original trace (on the left). The bursts that are present in the original trace are evidence of very long-range correlations in the original dataset, which corresponds to long periods of very large stack distances – as would be caused by phase changes in file referencing behavior.

The degree of self-similarity in a series is captured by the parameter H, which takes values between 0.5 and 1.0. If $H = 0.5$, the series has no long-range dependence, and acts like Gaussian noise (whose motion process is Brownian motion). As $H \rightarrow 1$, the burstiness of the series becomes more pronounced at high levels of aggregation. To estimate the H parameter for our series, we use four methods. Three methods, the variance-time plot, the R/S plot, and the periodogram, are graphical. These methods are useful for assessing whether assumptions about the data are correct (such as stationarity) and for providing a single estimate of H. The fourth method is the maximum likelihood estimator for H, called the *Whittle* estimator, which provide confidence intervals as well. Interested readers are referred to [4, 22] for a discussion of these methods in more detail.

Plots of the graphical estimators for the BU trace are shown in Figure 10. On the left is the variance-time estimator; linearity on this plot is evidence of long-range dependence, and the slope of the line yields an estimate of H, which in this case is 0.82 (the line corresponding to $H = 1/2$ is also plotted for reference. In the center is a plot of the R/S statistic; again, linearity is evidence of long-range dependence, and the slope yields an estimate of H: 0.78. In this plot the lines corresponding to $H = 1/2$ and $H = 1$ are plotted for reference. On the right is the periodogram estimator; significant scatter is typical in this plot, but a least-squares fit to the points also yields and estimate of H, in this case 0.87. Each of these plots shows evidence of self-similarity, and the graphical results for the other three stack distance traces show similar linearity.

In addition to the graphical estimators, we also used the Whittle MLE estimator to estimate H for our datasets. In all cases we found reasonable agreement between the estimators, and values of H significantly above 1/2. The results for our four datasets are shown in Table 4. The values of H range from about 0.85 for the BU dataset to about 0.65 for the EPA dataset.

We considered the possibility that the process of transforming the original traces into stack distance strings may have been responsible for introducing dependence into the data. To answer this question we randomly rearranged each dataset, calculated the corresponding new stack distance series, and estimated the H values of the resulting series. The results are shown in Table 5. These estimates are all very close to 1/2, indicative of no long-range dependence in the series. As a result, we conclude that the presence of long-range dependence in these timeseries reflects actual correlations among the symbol pattern in the original reference traces.

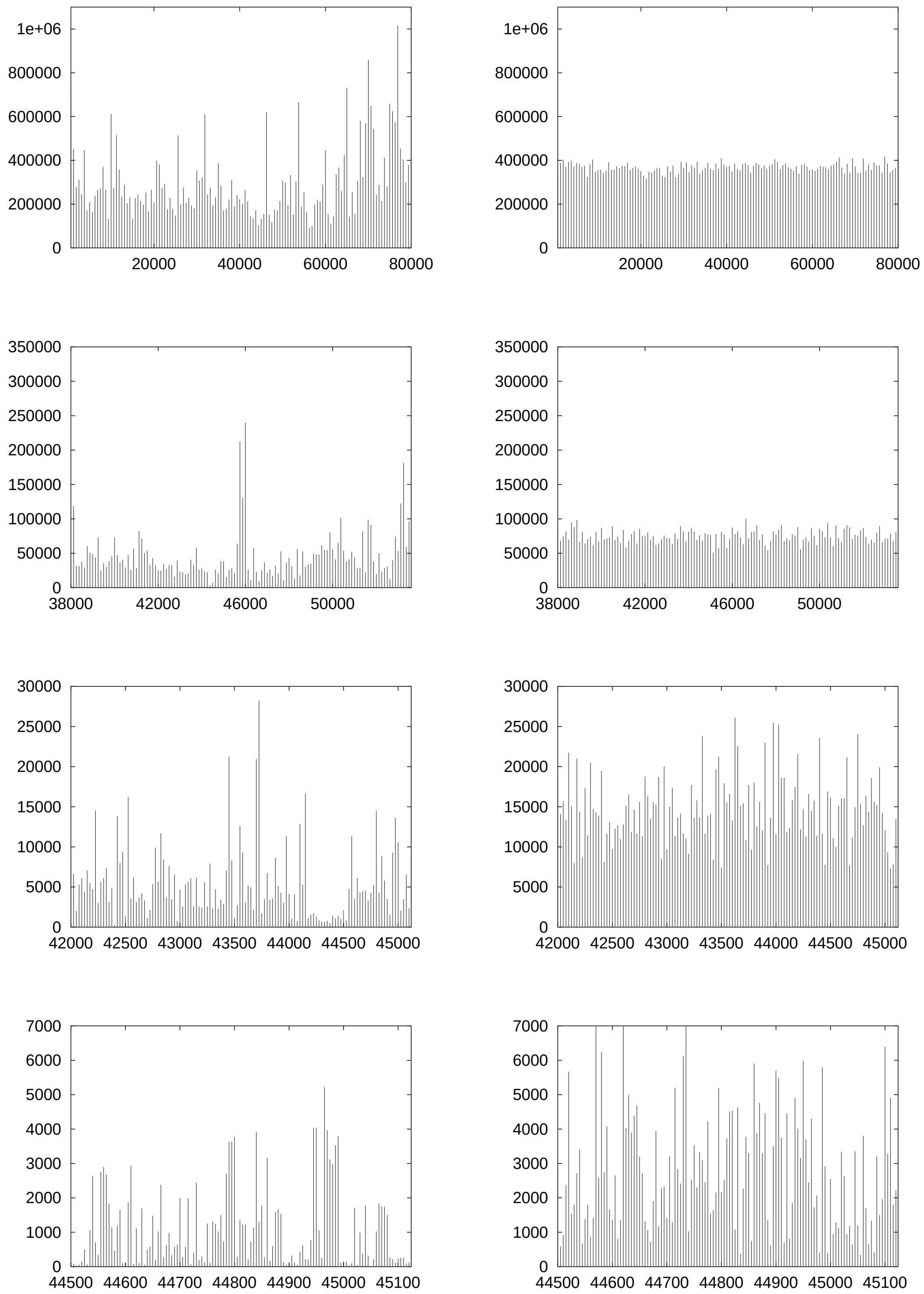

Figure 9: Stack Distance Scaling Behavior of Original (left) and Scrambled (right) BU Traces

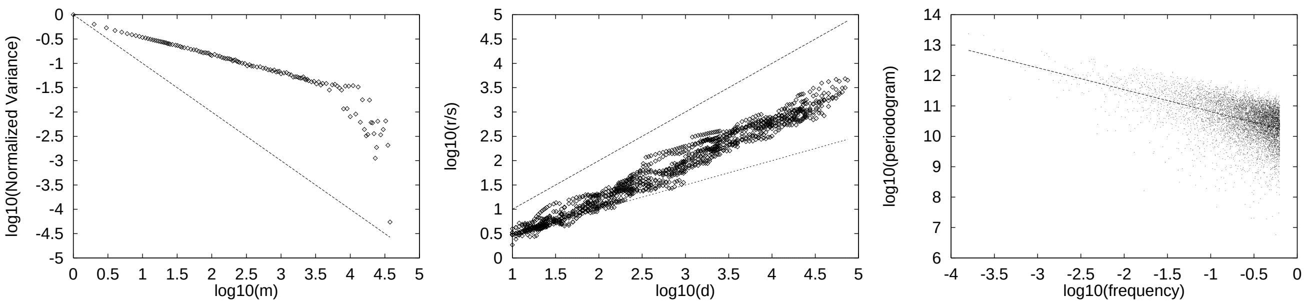

Figure 10: Graphical Estimators of H for BU Trace

	V-T	R/S	Period.	Whittle MLE	(95% conf.)
BU	.82	.78	.87	.85	(0.84, 0.87)
NCSA	.71	.74	.74	.74	(0.73, 0.77)
SDSC	.71	.68	.69	.68	(0.66, 0.71)
EPA	.64	.66	.66	.65	(0.64, 0.67)

Table 4: Estimates of H for Datasets Used in This Study

	V-T	R/S	Period.	Whittle MLE
BU	.50	.55	.50	.50
NCSA	.50	.51	.51	.49
EPA	.52	.54	.50	.50
SDSC	.51	.55	.47	.50

Table 5: Estimates of H for *Scrambled* Datasets

5.2 Relationship to Fractal Cache Miss Patterns

Another way to look at Figure 9 is that, progressing from top to bottom, we are "zooming in" on the data. The fact that the plots on the left of the figure show roughly similar degrees of burstiness at all scales is evidence of fractal-like behavior of the stack distance trace.

In fact, a statistically self-similar series X_t like the one shown on the left of Figure 9 has an almost sure fractal dimension (in the sense of either the Hausdorff $\dim_H$ or box-counting dimension $\dim_B$) of [16]

$$\dim(X_t) = 2 - H.$$

Thus the fractal dimension of such a series is $3/2$ for Gaussian noise ($H = 1/2$), and decreases to 1 as the correlation present in the series increases ($H \to 1$). The corresponds with intuition: a series with more dependence is more "predictable" and fills space less completely.

The fractal properties of stack-distance series serve to place earlier work by Voldman *et. al.* [34] in a slightly broader context. Those authors found that patterns of cache *misses* often form random fractals. One of their results is that if cache miss events are plotted on the number line using the sequence number of the reference as the coordinate, the result is a fractal pattern of dimension between 0 and 1.

Our results are consistent with that work as follows. Cache miss patterns can be seen to be the points of intersection between the stack distance trace and a line, parallel to the x axis, corresponding to some fixed cache size. The intersection of fractal X having $1 < \dim_H(X) < 2$ and a line L ($\dim_H(L) = 1$) is almost surely a new fractal X' with $\dim_H(X') = \dim_H(X) - 1$ [23]. This means that a cache miss fractal derived from a stack distance fractal such as our traces should have a dimension between 0 and 1. In fact we have observed this effect and have found that cache misses in our traces show fractal characteristics that are consistent with the patterns previously observed by Voldman *et. al.*

5.3 Generating Synthetic Web Reference Traces

To generate synthetic Web reference traces it is important to mimic as closely as possible both the temporal and spatial locality present in real traces. To do so using the results in this paper suggests the following process:

1. Select parameters μ and σ reflecting temporal locality, and H reflecting spatial locality. These may be selected based on empirical measurement of traces that are to be imitated, or from the range of values measured in the traces reported here.

2. Generate a stack distance trace with marginal distribution determined by μ and σ and long-range dependence determined by H.

3. Invert the stack distance trace to form a sequence of file names.

On first glance, step 2 seems difficult. Fortunately recent results have shown that marginal distribution and long-range dependence can be considered as separate problems for trace generation [21]. That work shows that, given a long-range dependent series X_t with $H = H_X$ and marginal cumulative probability function $F_X(x)$, we can generate another long-range dependent series Y_t with $H = H_X$ and arbitrary marginal cumulative probability function $F_Y(x)$. This is achieved by individual transformation of the elements of X_t as follows:

$$Y_t = F_Y^{-1}(F_X(X_t)) \tag{5}$$

The authors in [21] show that this transformation preserves long-range dependence with the same value of H.

Thus, our approach to step 2 is as follows. First, generate a self-similar series with given H. This can be done in a number of ways; a fast approach is described in [28]. Then apply the transformation of Equation 5 to obtain a series with the proper lognormal marginal distribution.

We have begun to use the method for generating web reference traces and are currently evaluating the properties of the resulting traces. Initial experiments involving models of only temporal locality properties are encouraging.

6 Related Work

In this section we review previous research directly[5] related to our work, namely work in reference locality modeling, use of fractals for modeling program behavior, and work in exploiting Web reference locality.

6.1 Reference Locality Modeling

Denning and Schwartz [15] established the fundamental properties that characterize the phenomenon of locality in hierarchical structures of memory. In [25], the authors studied stack algorithms and introduced stack distance as a means for analyzing behavior of demand-paged memory systems. They also discussed the significance of stack distance strings for evaluating performance of memory management schemes. Then, Spirn [32] proposes the use of distance string models to represent program behavior. Distance strings are equivalent in information content to the reference string, but are more easily handled by mathematical models. The distance string model and the LRU stack model are equivalent, since it is possible to determine the LRU stack at each time from the knowledge of the distance string. However, the proposed distance string model does not capture the long-range dependencies among references. As pointed out by the author, the model does not provide an accurate characterization of page faults, which tend in real programs to occur in clusters. When changes in locality occur gradually, page by page, it is relatively simple to model the reference locality. However, in a real program, the locality set of execution is completely disrupted by the execution of a new phase of the program. Usually, the abrupt changes in locality generate large distances in distance string model. Markov distance string models are capable of predicting bursts based on the most recent generated distances. Thus, the work in [32] argues that Markov models are not able to capture the behavior of real programs, which exhibit some form of long-range dependencies. The author also points out that those models, based on the distance probabilities, are not suitable for simulation studies of program behavior, because they lack the ability to mimic long-range effects.

6.2 Application of Fractals to Reference Modeling

The use of fractal geometry and self-similarity to analyze behavior of computer systems was pioneered by the work of Voldman et al [34]. The authors looked at memory reference traces of three software environments and found that the distributions of intermiss distances are hyperbolic. The cache misses are grouped over time in clusters, which are statistically self-similar. As a result of the analysis, they conjecture that the fractal dimension of a given cache miss distribution is a measure of the complexity of the underlying software.

[5]There is a vast literature on characterizing and exploiting the locality of reference properties exhibited in computer systems. Covering this literature is beyond the scope of this paper.

Based on the work in [34], paper [33] models the patterns generated by a processor accessing memory as random walk with fractal dimension. The author proposes that in an asymptotic behavior the number of misses that a program encounters in an infinite cache is given by a hyperbolic function of the fractal dimension. The paper also presents an approximate analytical model for cache misses. The model has two distinct phases, one called forced mode, which represents the cold start process. The fractal mode phase models the stable phase of a cache. However, the model is not easily used because of the empirical process of obtaining and adjusting its parameters from plots of the traces. Also, the proposed model does not represent multiprogramming environments, where context switches occur frequently.

Our paper extends the work referenced in [34, 33, 32] to the context of large distributed systems and also introduces new parameters that fully describe locality aspects of a reference stream that arrives at a Web server.

6.3 Exploiting Web Reference Locality

Previous studies in caching, replication, dissemination, and prefetching protocols for large distributed information systems have recognized and exploited the various locality of reference properties examined in this paper. In the remainder of this section we present a brief review of these protocols.

Most of the caching studies for large distributed information systems concentrate on client-based caching, whereby recently/frequently accessed information is cached at the client (or at a proxy thereof) in anticipation of future accesses to the same information. Traditionally, this was done in the realms of distributed file systems [20], but more recently, there have been some attempts at extending these techniques to distributed information systems such as FTP and HTTP.

Caching to reduce the bandwidth requirements for the FTP protocol on the NSFNET has been studied in [14]. In this study, a hierarchical caching system that caches files at Core Nodal Switching Subsystems is shown to reduce the NSFNET backbone traffic by 21%. The effect of data placement and replication on network traffic was also studied in [3], where file access patterns are used to suggest a distributed dynamic replication scheme. A more static solution based on fixed network and storage costs for the delivery of multimedia home entertainment was suggested in [27]. Multi-level caching was studied in [26], where simulations of a two-level caching system is shown to reduce both network and server loads. In [9], a dynamic hierarchical file system, that supports demand-driven replication is proposed, whereby clients are allowed to service requests issued by other clients from the local disk cache. A similar cooperative caching idea was suggested in [13].

Glassman [17] presents one of the earliest attempts for caching on the Web, whereby *"satellite relays"* (proxy caches) are organized into a tree-structured hierarchy with cache misses in lower relays percolating up through higher relays until the requested object is found. The performance of this caching system for a single relay with a rather small cache size indicated that it is possible to maintain a *"fairly stable"* 33% hit rate. The performance of such a system for large cache sizes was extrapolated using a Zipf-based model. In particular, it was estimated that with an infinite-size cache, the maximum achievable hit rate is 40%. Another early attempt at characterizing Web access patterns for the purpose of engineering Web caching systems is the work of Recker and Pitkow [29], in which a model for Web information access is proposed based on two metrics borrowed from psychological research on human memory—namely the frequency and recency rates of past accesses.

The first comprehensive study of client-based caching for the Web was conducted by our Oceans group.[6] In that study [7], the effectiveness of session caching, host caching, and LAN proxy caching

[6] `http://cs-www.bu.edu/groups/oceans/`

were established using a unique set of 5,700 client traces (almost 600,000 URL requests), which were obtained by intrumenting Mosaic as detailed in [12]. This study concluded that LAN proxy caching—while effective in reducing response time by as much as a half—is ultimately limited by the low level of sharing of remote documents amongst clients of the same site. This finding agrees with Glassman's predictions [17] and was further confirmed for general proxy caching by Abrams *et al* [1].

Williams *et al* [35] present a taxonomy (and compare the performance) of a number of proxy cache replacement policies. Their work suggests that proxy cache management should consider document sizes when making replacement decisions. This was also confirmed by Bolot and Hoschka [10], who showed the usefulness of using information about document size and network load in the replacement algorithms of Web caches, based on time series analysis techniques of Web traffic. Similar results about the inadequacy of classic LRU cache replacement are presented in [1].

In [5], Bestavros proposed an alternative to client-based caching, which attempts to capitalize on the global knowledge amassed at servers about *geographical* locality of reference—the property that an object accessed by a client at a site is likely to be accessed again in the future by a client in a *"nearby"* site. To that end, he proposed a popularity-based dissemination protocol, which allows servers to cache documents at proxies that are closer to clients, in such a way that the distance between a client and a document server (or proxy thereof) is inversely proportional to the popularity of that document. A similar philosophy was sketched in [18], whereby the placement of popular documents was based on geographical information (such as the distance in actual miles between servers and clients).

In [24], Markatos examines the potential performance gains from using *Main Memory Web Caches*. He shows that a small amount of main memory could yield substantial improvement in performance if cache management methods that are sensitive to document popularity and to user preferences for small documents [12] are employed.

Exploiting the spatial locality of reference properties exhibited in Web access patterns was investigated in two recent prefetching studies of our Oceans group. In [8], Bestavros and Cunha propose a protocol that allows a client to prefetch Web documents based on Markov models built from the client's previous access patterns. Preliminary results indicate that client-initiated prefetching is extremely effective for access patterns that involve frequently-traversed documents, but not effective at all for access patterns that involve newly-traversed documents. For such access patterns, Bestavros proposed an alternative protocol [6] that allows prefetching to be initiated by servers (or by clients as a result of hints generated by servers) based on a Markov model built by analyzing the server's access logs.

7 Conclusion

In this paper we have measured the locality present in reference traces arriving at four servers in the World Wide Web, and shown statistical models that capture the locality properties of our datasets. These models can be used to compare different reference streams, predict cache performance, and generate synthetic reference streams with realistic properties.

We first showed that simple methods for generating reference streams based only on document popularity capture neither the temporal nor the spatial locality present in a trace. We showed that some method that measures locality explicitly is needed in order to generate traces with the necessary characteristics. As a result, to measure locality directly, we have relied on the technique of transforming reference strings into distance strings, and studying the statistical properties of those distance strings.

We then measured the temporal locality properties of the reference traces by characterizing the distribution of stack distances in each trace. We've shown that stack distances have long tails, and that a reasonable model for stack distance distribution is the *lognormal* distribution. Typical parameters for the lognormal distributions fitted to our data were given, and the resulting models were shown to exhibit cache miss rates that are close to the original traces over the entire range of cache sizes.

To model spatial locality, we argued that models incorporating long-range dependence were necessary in order to capture the long-term behavior of reference patterns. We showed that stack distance traces seem to show properties of self-similarity, and we measured the relevant parameters for each of our traces. The resulting insight into the nature of correlation in stack distance series allowed us to interpret previous work in fractal cache miss patterns in a broader context.

Ongoing work is extending these models of locality to incorporate explicit modeling of document size. In addition, although our work is based on traces obtained from servers, we are also interested in the locality properties of client reference traces, and how those properties change when streams are captured close to the client, as compared to close to the server.

8 Acknowledgements

The authors thank John Dilley for comments that improved the content and presentation of this report.

References

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

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of ACM SIGMOD conference*, San Jose, CA, May 1995.

[3] Swarup Acharya and Stanley B. Zdonik. An efficient scheme for dynamic data replication. Technical Report CS-93-43, Brown University, Providence, Rhode Island 02912, September 1993.

[4] Jan Beran. *Statistics for Long-Memory Processes.* Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1994.

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

[6] Azer Bestavros. Using speculation to reduce server load and service time on the www. In *Proceedings of CIKM'95: The* 4^{th} *ACM International Conference on Information and Knowledge Management*, Baltimore, Maryland, November 1995.

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

[8] Azer Bestavros and Carlos Cunha. A prefetching protocol using client speculation for the www. Technical Report TR-95-011, Boston University, CS Dept, Boston, MA 02215, April 1995.

[9] Matthew Addison Blaze. *Caching in Large Scale Distributed File Systems*. PhD thesis, Princeton University, January 1993.

[10] Jean-Chrysostome Bolot and Philipp Hoschka. Performance engineering of the world wide web: Application to dimensioning and cache design. In *Proceedings of the Fifth Interntional COnference on the WWW*, Paris, France, 1996.

[11] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1996.

[12] Carlos A. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of www client-based traces. Technical Report TR-95-010, Boston University Department of Computer Science, April 1995.

[13] Michael D. Dahlin, Randolph Y. Wang, Thomas E. Anderson, and Dacid A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *First Symposium on Operating systems Design and Implementation (OSDI)*, pages 267–280, 1994.

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

[15] P. Denning and S. Schwartz. Properties of the working set model. *Communications of the ACM*, 15(3):191–198, 1972.

[16] Kenneth Falconer. *Fractal Geometry*. John Wiley & Sons, Ltd., 1990.

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

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

[19] The Internet Town Hall. The internet traffic archive. available as http://www.town.hall.org/Archives/pub/ITA/, 1995.

[20] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[21] Changcheng Huang, Michael Devetsikiotis, Ioannis Lambadaris, and A. Roger Kaye. Modeling and simulation of self-similar variable bit rate compressed video: A unified approach. In *Proceedings of ACM SIGCOMM '95*, pages 114–125, 1995.

[22] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2:1–15, 1994.

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

[24] Evangelos Markatos. Main memory caching of web documents. In *Proceedings of the Fifth Interntional COnference on the WWW*, Paris, France, 1996.

[25] R. Mattson, J. Gecsei, D. Slutz, and I. Traiger. Evaluation techniques and storage hierarchies. *IBM Systems Journal*, 9:78–117, 1970.

[26] D. Muntz and P. Honeyman. Multi-level caching in distributed file systems or your cache ain't nuthing but trash. In *Proceedings of the Winter 1992 USENIX*, pages 305–313, January 1992.

[27] Christos H. Papadimitriou, Srinivas Ramanathan, and P. Venkat Rangan. Information caching for delivery of personalized video programs on home entertainment channels. In *Proceedings of the International Confrence on Multimedia Computing and Systems*, pages 214–223, May 1994.

[28] Vern Paxson. Fast approximation of self-similar network traffic. Technical Report LBL-36750, Lawrence Berkeley Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, 1995.

[29] Mimi M. Recker and James E. Pitkow. Predicting document access in large, multimedia repositories. Technical Report Technical Report VU-GIT-94-35, Georgia Tech—Graphics, Visualization, and Usability Center, August 1994.

[30] G. Shedlar and C. Tung. Locality in page reference strings. *SIAM Jounral of Computing*, 1(3), September 1972.

[31] Alan Jay Smith. Cache memories. *Computing Surveys*, 14(3):473–530, September 1982.

[32] Jeffrey Spirn. Distance string models for program behavior. *IEEE Computer*, 13(11), November 1976.

[33] Dominique Thiebaut. On the fractal dimension of computer programs and its application to the prediction of the cache miss ratio. *IEEE Transactions on Computers*, 38(7), July 1989.

[34] Jean Voldman, Benoit Mandelbrot, Lee W. Hoevel, Joshua Knight, and Philip L. Rosenfeld. Fractal nature of software-cache interaction. *IBM Journal of Research and Development*, 27(2):164–170, 1983.

[35] Stephen Williams, Marc Abramsand Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. Removal policies in network caches for world-wide web documents. http://ei.cs.vt.edu/ succeed/96WAASF1/, 1996.

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