The Relevance of Long-Range Dependence in Disk Traffic and Implications for Trace Synthesis

Bo Hong
Storage Systems Research Center
University of California, Santa Cruz
hongbo@cse.ucsc.edu

Tara M. Madhyastha
Dept. of Computer Engineering
University of California, Santa Cruz
tara@cse.ucsc.edu

Abstract

Accurate disk workloads are crucial for storage systems design, but I/O traces are difficult to obtain, unwieldy to work with, and unparameterizable. I/O traces are often bursty and difficult to characterize. Although good models of I/O workloads would be extremely useful, such bursty traces cannot accurately be modeled using exponential or Poisson arrival times. Much experimental evidence suggests that I/O traces are self-similar, which researchers have hoped might help to model bursty traces. In this paper, we show that self-similarity at large time scales does not significantly affect disk behavior with respect to response times. This allows us to generate synthetic arrival patterns at relatively small time scales, improving the accuracy of trace generation. The relative error of our method, with input parameters suitable for the workload, ranges from approximately 8% to 12%.

1. Introduction

Performance analysis and architecture of storage systems depend heavily upon traces and simulations. Unfortunately, I/O traces are difficult to obtain, extremely large and unwieldy, and cannot be parameterized. Thus, system researchers often resort to benchmarks, whose accuracy usually depends heavily on the underlying workload models.

Ideally, one would like to monitor any disk workload and model it accurately (with respect to some important performance metrics) with some small number of parameters. This vision is far from reality; however, this paper identifies parameters that can capture request interarrival burstiness.

We consider an I/O trace that consists of a set of timestamped values each containing a disk offset, a read/write flag, and a length. This low-level description accommodates a primitive application-level or SCSI I/O interface. We wish to model these streams accurately enough so that accesses synthesized from the model cause a storage hierarchy to behave "similarly" to the real trace. For a single disk drive, similar behavior is measured by checking whether the distributions of response times as well as queue lengths resemble those created by the original workload.

We show that long-range dependence in I/O traffic does not significantly affect disk behavior. This allows us to generate synthetic interarrival patterns at relatively small time scales, improving the accuracy of trace generation.

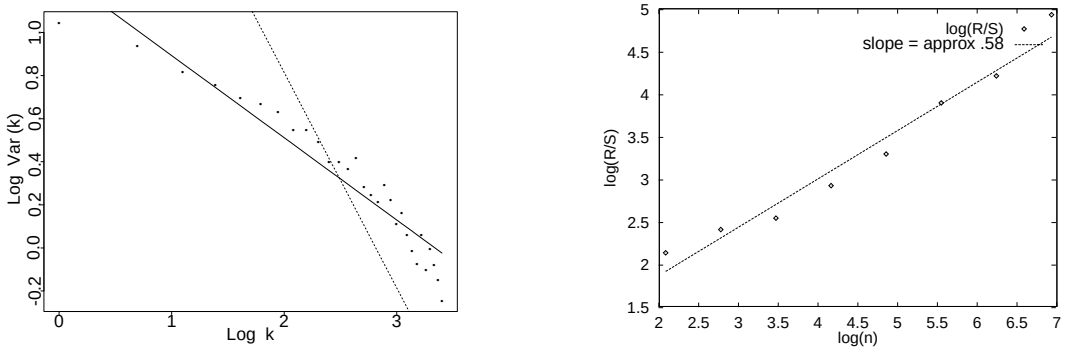
The paper is organized as follows. We describe related work in Section 2. We introduce self-similarity and the Hurst coefficient as a way to approximate long-range dependence and show that long-range dependence has little effect upon disk response times in Section 3. Binomial multifractals can generate bursty traffic; Section 4 describes this model. In Section 5 we describe a novel I/O request synthesis technique using multifractal models. We evaluate our method in Section 6 and conclude with directions for future work in Section 7.

2. Related Work

Generating realistic disk traces is a difficult and unsolved problem [5]. Much experimental evidence shows that disk I/O, file, network, and Web traffic shares some common properties, such as burstiness and long-range dependence, with self-similar and multifractal processes [4, 6, 7, 10]. These properties cannot accurately be modeled using exponential or Poisson arrival times.

Several researchers have used self-similarity to model bursty traces, particularly network traces. Chen *et al.* [2] examined ATM variable bit rate traffic and found that the higher the Hurst coefficient, a measure of self-similarity, the burstier the traffic. Multifractal models, or generalizations of self-similar traffic models, have been shown to model some kinds of traffic more effectively than self-similar models [3]. Wang *et al.* [19] proposed to use binomial multifractals to model bursty disk traffic. The model

Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2005)
0-7695-2318-8/05 $20.00 © 2005 IEEE



(a) Variance plot

(b) R/S plot

Figure 1. Estimating the Hurst parameter.

is parsimonious, depending only upon a single parameter, bias p, which can be estimated from real traces. Wang *et al.* [18] further proposed a two-dimensional multifractal model that characterizes both the temporal and spatial behaviors of data accesses and captures the spatio-temporal correlations using the joint entropy of the two-dimensional disk request arrival events (time and space).

Grossglauser and Bolot [8] demonstrated that it was not useful to model long-range dependence in network traffic at time scales disproportionate to the performance metrics under observation. Neidhardt and Wang [12] showed that queuing behavior depends not only on the Hurst coefficient, but a combination of system parameters. Our approach is to investigate whether this is true for I/O traffic, and whether this fact is useful for improving multifractal synthesis techniques.

3. Relevance of Long-Range Dependence in Disk Traffic

Some researchers have claimed that bursty I/O workloads have a structure that might help to model them called *self-similarity*. Informally, in this context, to say a time series is self-similar implies that it looks qualitatively the same at different time scales. Self-similar traffic has the property of *long-range dependence*: the data set exhibits a slow decay in its autocorrelation function. This correlation structure is significant because self-similar traffic may be more bursty than that generated by other sources. However, we show here that long-range dependence, as measured by the Hurst coefficient, has little effect upon disk response times.

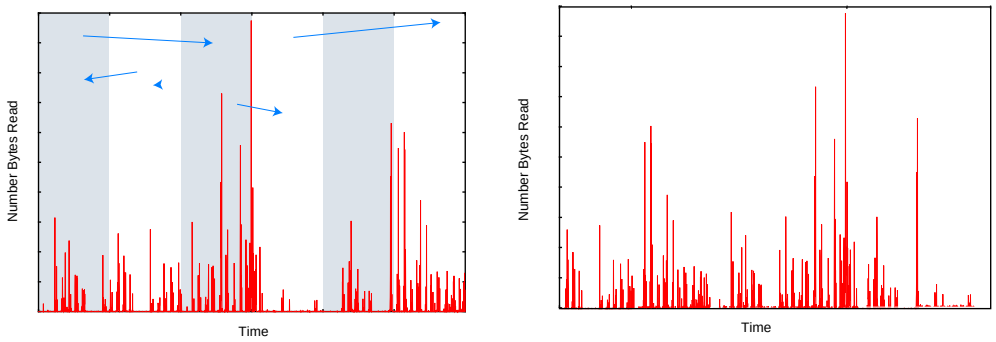
3.1. Self-Similarity

A more rigorous definition of self-similarity from [1] is as follows: Let Y_t be a stochastic process with continuous time parameter t. Y_t is called self-similar with self-similarity parameter H, if for any positive stretching factor c, the rescaled process with time scale ct, $c^{-H}Y_{ct}$, is equal in distribution to the original process Y_t. The parameter H is also known as the Hurst coefficient, and a value of H between $\frac{1}{2}$ and 1 indicates the degree of self-similarity. Generally speaking, the Hurst coefficient is a predominant way to quantify long-range dependence in a stochastic process. A comprehensive treatment of the Hurst parameter is presented by Meakin [11].

There are several exploratory analytic tools that are used to estimate H; two such methods are applied to a small UNIX workstation disk trace [14] in Figures 1(a) and 1(b). The first method, shown in Figure 1(a), is the variance plot. We plot the logarithm of the variance of an aggregated (averaged) series against the logarithm of the aggregation level. The slope of this plot should be equal to $H-1$. The second method, shown in Figure 1(b), is the R/S plot. This method plots (in logscale) the R/S statistic, or the *rescaled adjusted range* against $\log n$. The rescaled adjusted range is the data range normalized by the standard deviation. The precise definition of how to calculate this statistic can be found in [1, 17]. If there is long-range dependence in the process, the slope of the curve generated by this plot provides an estimate of H; if not, $\log R/S$ should be randomly scattered around a straight line with a slope of 0.5 [1].

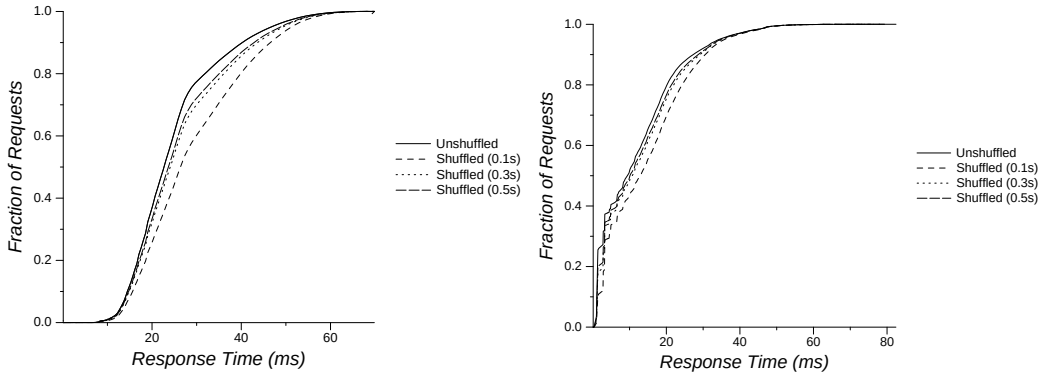
3.2. Long-Range Dependence in Disk Traffic

Our hypothesis is that previous events cannot affect disk behavior beyond a certain threshold, determined by system parameters, so modeling long-range dependence at large time scales is unnecessary. To test this hypothesis, we study how disk response time changes as we gradually destroy long-range dependence in the traces by shuffling increasingly smaller intervals. This approach is identical to the experimental approach taken by [8], and is illustrated in Figure 2. Figure 2(a) shows a trace that has been divided into



(a) Original/unshuffled trace (b) Shuffled trace

Figure 2. Shuffling traces removes long-range dependence.



(a) Cello (b) Snake

Figure 3. Response time distributions for traces shuffled using small intervals.

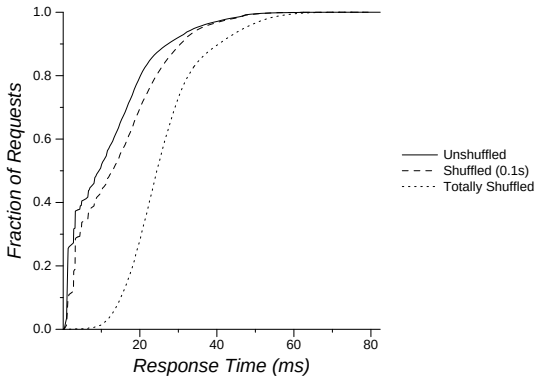


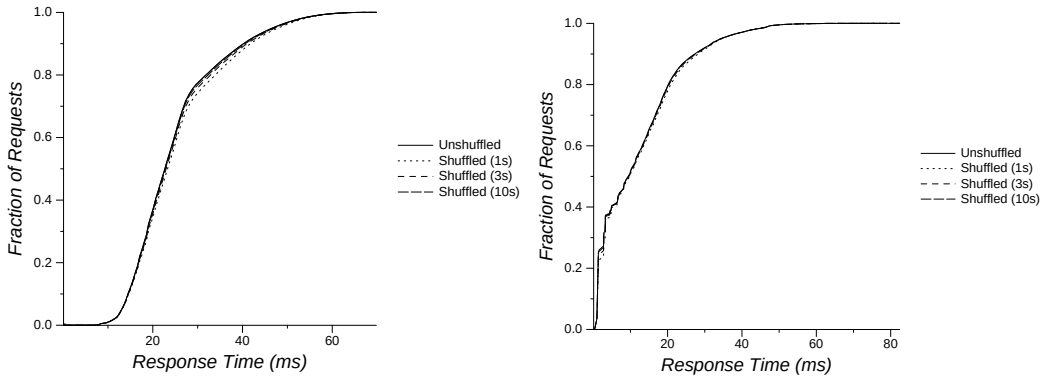
Figure 4. In the limit, all temporal locality is lost when we shuffle a snake trace.

six intervals. These intervals are then randomly rearranged to create a new trace (Figure 2(b)). Within each interval, the temporal relationships are preserved, but the new trace has no long-range dependence beyond the interval width.

Our selected workloads, described in more detail in [14], are the cello news disk trace (HP2204A) and the snake usr2 disk trace (HP97560) gathered between 05/30/92 and 06/06/92. The average I/O loads on these disks are small: approximately three requests for the cello news disk and one request for the snake usr2 disk per second. However, the maximum queue length can be very large: over 1000 requests cello and over 60 requests on snake. In general snake is more bursty than cello, and the logical sequentiality (percentage of requests that are at adjacent disk addresses or addresses spaced by the file system interleave factor) of cello and snake is 2% and 29%, respectively.

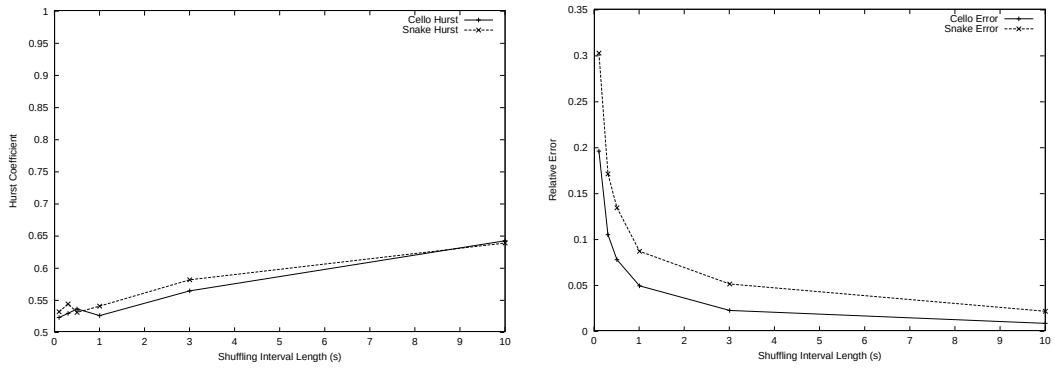
We examine the numerical metric of disk performance that was previously used to validate disk models [15]: the root mean squared (RMS) horizontal distance between the cumulative distribution functions (CDF) of I/O response times. The distributions of queue lengths of traces shuffled at intervals more than one second are similar to those of the real traces [9], and are not presented separately.

We vary the shuffle interval length from 10 seconds to 0.1 second and use both the shuffled and original traces to drive the Pantheon disk simulator [20]. Shuffling traces using intervals smaller than 0.5 second results in significantly skewed disk response time distributions, as shown in Fig-



(a) Cello (b) Snake

Figure 5. Response time distributions for traces shuffled using large intervals.



(a) Hurst coefficient (b) Relative error (RMS)

Figure 6. Effect of shuffling.

ures 3(a) and 3(b). In the limit, we destroy all temporal locality by randomizing all events and the obtained response time distribution curve extremely differs from the original one, as shown in Figure 4. In contrast, although at interval sizes of one second and above there is virtually no long-range dependence left in the shuffled traces, the relative errors are very small, as shown in Figures 5(a) and 5(b).

Shuffling removes long-range dependence in I/O traces. The Hurst coefficient is a measure of long-range dependence; as intuition dictates, the smaller the time interval, the fewer long-range correlations are preserved and the lower the Hurst coefficient, as shown in Figure 6(a). In comparison, the original Hurst coefficient is 0.89 for cello and 0.79 for snake, respectively.

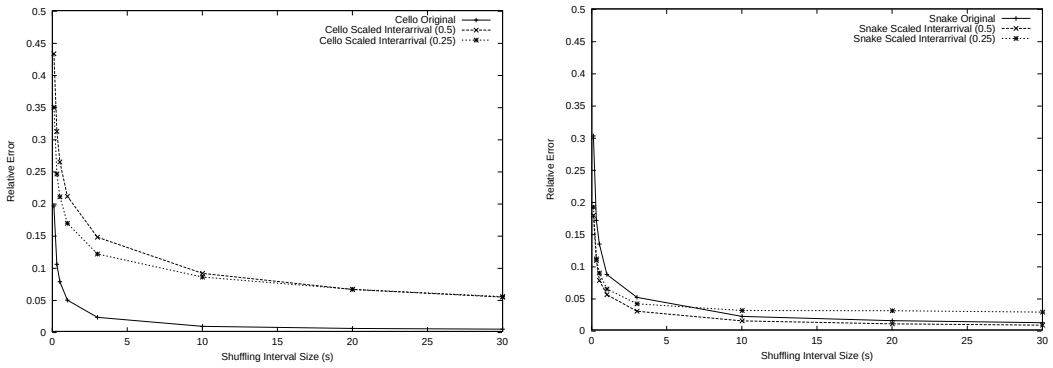
Despite the lack of long-range dependence, particularly indicated by the fluctuation of the Hurst coefficient at small intervals (less than one second), the relative error, measured by RMS [15], for the shuffled traces is relatively small and decreases with the increase of the shuffle interval length, as shown in Figure 6(b). In general the relative error is larger for snake than for cello: at one second it is approximately 5% for cello and 9% for snake. The error for snake can be bounded under 5% by using shuffle intervals larger than five seconds.

3.3. Choosing an Interval

To better understand how to select an appropriate interval length to bound the error for each trace, we study the relative error as a function of workload burstiness. We multiply the traced interarrival times by a factor of 0.5 or 0.25 to artificially increase the burstiness of the traces, creating more disk request queuing.

Figures 7(a) and 7(b) show the effect of scaling on relative error for cello and snake, respectively. In general, as the shuffle interval length increases, the relative error decreases. However, the interval length necessary to maintain the same relative error does not automatically increase as we scale down the traced interarrival time, and the curves are quite different for the two workloads.

Cello has fewer long “gaps” between activities than snake. Almost all of the cello requests have interarrival times less than 0.1 second; in contrast, 63% of the snake requests arrive after the previous one within 0.1 second,



(a) Cello (b) Snake

Figure 7. Effect of shuffling on relative error for scaled traces.

and 7% of the snake requests have interarrival times longer than 10 seconds (see Figure 10). Shuffling has less effect on the request interarrival time distribution for cello than for snake. Cello is less sequential than snake so shuffling does not perturb access spatial locality for cello as much as for snake. Thus, the error caused by shuffling the original trace is lower for cello than for snake.

When the traced interarrival times are shortened, queue lengths increase. For cello, this improves average seek time thanks to request scheduling optimization. Shuffling changes this queuing behavior and causes higher errors than in the original trace. Snake has a disk cache and is more sequential but less busy than cello, resulting in much lower average queue length. Therefore, this queuing effect is not as important for snake. In addition, shortening request interarrival times can, to some degree, preserve more temporal relations in the workload under the same shuffle interval length. Thus, errors for the shuffled scaled snake traces are actually lower than those for the shuffled original trace at small intervals, and increase with larger shuffle intervals.

3.4. Modern Traces

The traces described here were from 1992. A reconfigured cello was re-traced in 1999, but we have not yet been able to repeat our experiments on those traces. However, to generalize our results on long-range dependence to modern traces, we study the characteristics of new cello news disk traces, obtained from a Seagate ST19171W disk, for one week (09/09/1999 to 09/15/1999). The I/O load has increased to about 16 requests per second but the maximum queue length is still in the same range as cello in 1992, from 700 to 1300. The logical sequentiality of cello (1999) is less than 1%. The Hurst coefficient is 0.89, similar to that of the 1992 cello trace. Experiments on the scaled 1992 cello traces (Figure 7) approximate the characteristics of the modern traces with respect to the Hurst coefficient and mean interarrival time. We believe that the shuffle interval required to minimize errors for modern traces may be slightly longer, but otherwise the behavior is the same.

We can conclude from our experiments that long-range dependence does not significantly affect disk behavior with respect to response times. For the purpose of performance evaluation, we need only consider I/O activities at time scales related to the system we are evaluating. For measuring disk response times and queuing behaviors, an appropriate interval length is estimated empirically to be between 3 and 10 seconds for the cello and snake workloads.

4. Binomial Multifractals

Self-similarity is a measure of fractal-like scaling behaviors over multiple time scales, characterized by the single Hurst parameter. In contrast, multifractals are a generalization of monofractal self-similar processes that allow for time-dependent scaling laws, and are based on multiplicative schemes. They have a bursty appearance similar to that of real I/O traffic. We introduce binomial multifractals, for the purpose of modeling I/O traffic, below. A rigorous introduction to binomial measures and multifractals can be found in [13].

4.1. Property of Self-Similarity

We first define a self-similar binomial measure on the unit interval in a recursive construction. Figure 8 shows the first two stages of the construction, which starts with the uniform probability measure μ_0 on the unit interval I = [0, 1] with mass 1 (Figure 8a). At the first stage (Figure 8b), I is split into two equal-length subintervals I_0 = [0, 1/2] and I_1 = [1/2, 1] and the masses $m_0 = p$ $(p > 1/2)$ and $m_1 = 1 - m_0 = 1 - p$ are spread uniformly on them. The density on I_0 and I_1 is $2p$ and $2(1-p)$, respectively. At the second stage (Figure 8c), I_0 is split into two equal-length subintervals I_{00} = [0, 1/4] and I_{01} = [1/4, 1/2] and

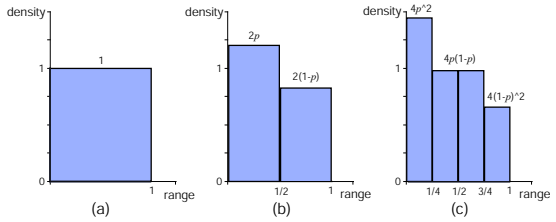


Figure 8. Recursive process in binomial measure generation. Start from (a) with a uniform probability measure, divide the mass with probability p in (b), divide again in (c).

the masses $m_{00} = p^2$ and $m_{01} = p(1-p)$ are spread uniformly on them; I_1 is split into two equal-length subintervals I_{10} = [1/2, 3/4] and I_{11} = [3/4, 1] and the masses $m_{10} = p(1-p)$ and $m_{11} = (1-p)^2$ are spread uniformly on them; The density on I_{00}, I_{01}, I_{10} and I_{11} is $4p^2$, $4p(1-p)$, $4p(1-p)$ and $(1-p)^2$, respectively. This construction continues recursively. Formally, at stage n, $n \in N$, each interval $I_{\varepsilon_1\varepsilon_2\cdots\varepsilon_{n-1}}$ in stage $n-1$ is split into two equal-length subintervals $I_{\varepsilon_1\varepsilon_2\cdots\varepsilon_{n-1}\varepsilon_n}$ with mass $m_{\varepsilon_1}m_{\varepsilon_2}\cdots m_{\varepsilon_{n-1}}m_{\varepsilon_n}$, $\varepsilon_i = 0,1$. Therefore, $\mu(I_{\varepsilon_1\varepsilon_2\cdots\varepsilon_n}) = m_{\varepsilon_1}m_{\varepsilon_2}\cdots m_{\varepsilon_n}$. This defines a sequences of measures μ_n on the unit interval I, which converge weakly towards a probability measure μ, the binomial measure. From the procedure of construction, it is clear that μ is strictly self-similar, as shown in Figure 9.

We extend this construction by randomizing the allocation of the mass in the recursive subdivisions. In this case, we may randomly choose the left multiplier as m_0 or m_1 (each with probability of 0.5), instead of always choosing m_0. Such randomization leads to binomial multifractals and makes them difficult to repeat, analyze, and predict. Real I/O traces could resemble more closely multifractals because of the unpredictability in workloads.

4.2. Property of Burstiness

Roughly speaking, the Hurst coefficient H describes global burstiness. However, local burstiness in disk I/Os is more interesting in practice. Multifractals can represent local burstiness, as described by the local Hölder exponent and multifractal spectrum of binomial measures.

For any $x \in [0,1)$, there is a unique subinterval $I_{\varepsilon_1\varepsilon_2\cdots\varepsilon_n}$ containing it in stage n. Let us denote it as $I^{(n)}(x)$. For convenience, we assume $m_0 > m_1$. For some x, the density on $I^{(n)}(x)$, defined as $\frac{\mu(I^{(n)}(x))}{|I^{(n)}(x)|} = \frac{m_{\varepsilon_1}m_{\varepsilon_2}\cdots m_{\varepsilon_n}}{2^{-n}}$, tends to infinity when $n \to \infty$, as shown by the points in the leftmost subintervals in Figures 8 and 9. Here $\mu(I^{(n)}(x))$ is the mass on $I^{(n)}(x)$ and $|I^{(n)}(x)|$ is the length of $I^{(n)}(x)$. The coarse graining in this interval has the property of burstiness. We can use a singularity exponent, the Hölder exponent, $\alpha(x)$

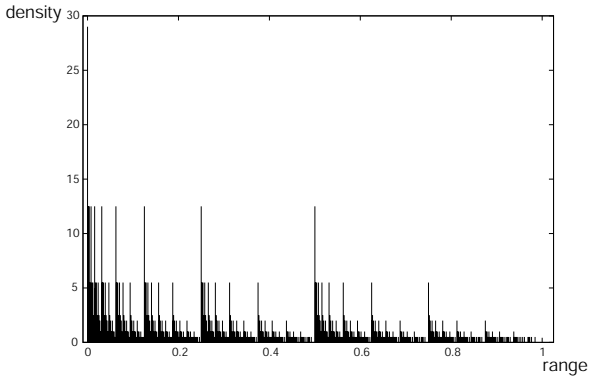


Figure 9. Probability density function for a sample binomial multifractal with bias 0.7.

to describe how fast the value approaches infinity:

$$\begin{aligned}\alpha(x) &= \lim_{n\to\infty}\alpha^{(n)}(x)\\ &= \lim_{n\to\infty}\frac{\log_2\mu(I^{(n)}(x))}{\log_2|I^{(n)}(x)|}\\ &= \lim_{n\to\infty}\frac{\log_2 m_{\varepsilon_1}m_{\varepsilon_2}\cdots m_{\varepsilon_n}}{\log_2 2^{-n}}\\ &= -\lim_{n\to\infty}\frac{\log_2\prod_{i=1}^{n}m_{\varepsilon_i}}{n}. \qquad (1)\end{aligned}$$

The multifractal spectrum $f(\alpha)$ describes the global distribution of Hölder exponent $\alpha(x)$:

$$\begin{aligned}f(\alpha) &= \lim_{n\to\infty}f^{(n)}(\alpha)\\ &= \lim_{n\to\infty}\frac{\log_2 N^{(n)}(\alpha)}{n}, \qquad (2)\end{aligned}$$

where $N^{(n)}(\alpha)$ denotes the number of subintervals $I^{(n)}$ with the Hölder exponent value of α.

At stage n, $\frac{n!}{i!(n-i)!}(= N^{(n)}(\alpha))$ subintervals have the same mass of $m_0^{n-i}m_1^i$. Therefore,

$$\begin{aligned}\alpha^{(n)} &= -(\log_2 m_0^{n-i}m_1^i)/n\\ &= -(i/n)\log_2 m_1 - (1-i/n)\log_2 m_0 \qquad (3)\\ &= (i/n)\alpha_{max} + (1-i/n)\alpha_{min}, \qquad (4)\end{aligned}$$

where $\alpha_{min} = -\log_2 m_0 = -\log_2 p$ and $\alpha_{max} = -\log_2 m_1 = -\log_2(1-p)$. According to the Stirling's formula,

$$\frac{n!}{i!(n-i)!} \sim (2^{-n})^{-E(i/n)}, \qquad (5)$$

where $E(i/n) = -(i/n)\log_2(i/n) - (1-i/n)\log_2(1-i/n)$. Combining above equations, we can find

$$\begin{aligned}f(\alpha) &= -\frac{\alpha_{max}-\alpha}{\alpha_{max}-\alpha_{min}}\log_2\left(\frac{\alpha_{max}-\alpha}{\alpha_{max}-\alpha_{min}}\right) \qquad (6)\\ &\quad -\frac{\alpha-\alpha_{min}}{\alpha_{max}-\alpha_{min}}\log_2\left(\frac{\alpha-\alpha_{min}}{\alpha_{max}-\alpha_{min}}\right),\end{aligned}$$

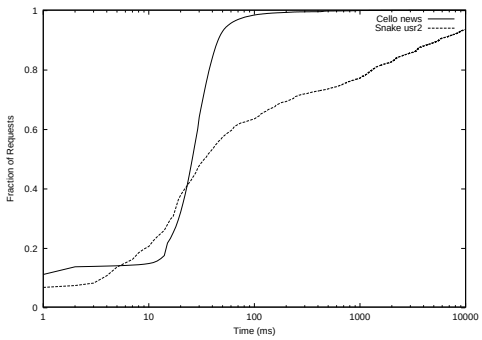


Figure 10. Cumulative distribution functions of request interarrival times.

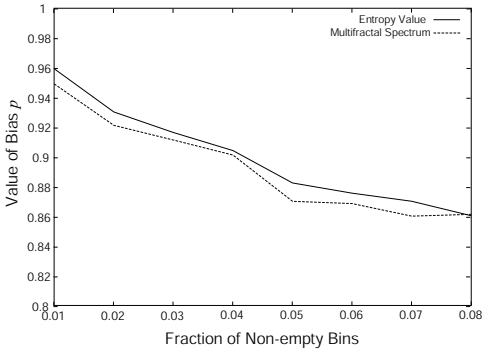


Figure 11. Estimation of p using the entropy method and the spectrum method.

where $\alpha_{min} \leq \alpha \leq \alpha_{max}$. This function has the same form as the entropy function, which provides us a way to estimate m_0 (bias p).

5. Multifractal I/O Request Synthesis

Multifractals represent local bursty I/O behaviors more accurately than other means of generating self-similar traffic. Here we introduce a method to use multifractals to model I/O request interarrival times at small time scales.

5.1. Estimation of Bias

The parameter bias p in binomial multifractals (or m_0 in binomial measures) describes the local burstiness behavior, which can be estimated from real traces. There are several ways to estimate bias p and we only introduce the two we used in our experiments.

The first way to estimate p is from the multifractal spectrum $f(\alpha)$ of binomial multifractals. We know that $f(\alpha)$ has the same shape as an entropy function. Bias p determines the location of the curve and how it is stretched. We can find the best fitting bias by visually judging how well the practical curve fits the theoretical ones.

The second way to estimate p is from the entropy value of real traces. Wang *et al.* [19] proposed this method because of its robustness and efficiency.

Assume that S is an information source that emits independent symbols from alphabet $\{s_0, s_1, \cdots, s_{k-1}\}$ with probabilities $\{p_0, p_1, \cdots, p_{k-1}\}$. By definition, $\sum p_i = 1$. The average amount of information we obtain by observing the output of S is called *entropy* [16] and is defined as:

$$E(p_0, \cdots, p_{k-1}) = -\sum_{i=0}^{k-1} p_i \log_2 p_i. \quad (7)$$

The disk traces can be viewed as a discrete time sequence Y_t, whose length can be normalized to be 1. For the purpose of model fitting, we can aggregate it at level n:

$$Y_t^{(n)}(k) = \int_{k2^{-n}}^{(k+1)2^{-n}} Y_t dt, \quad (8)$$

where $k = 0, 1, \cdots, 2^n - 1$. At level n, the sequence $Y_t^{(n)}$ can be considered as a distribution of an information source with alphabet $\{s_0, s_1, \cdots, s_{2^n-1}\}$, whose entropy is given by

$$E_p^{(n)} = -\sum_{k=0}^{2^n-1} \frac{Y_t^{(n)}(k)}{\int_0^1 Y_t dt} \log_2 \frac{Y_t^{(n)}(k)}{\int_0^1 Y_t dt}. \quad (9)$$

If we plot the value $E_p^{(n)}$ against n, the points should form a line with slope $E_p^{(1)}$ for a self-similar process like binomial measures, as proved in [19]. Thus, we can estimate $E_p^{(1)}$ from these points, and bias p using Equation 10:

$$E_p^{(1)} = -p \log_2 p - (1-p) \log_2 (1-p). \quad (10)$$

5.2. Verification of Estimation of Bias p

Not every I/O trace can be fit to a multifractal distribution. We empirically qualify the necessary characteristics of an I/O trace for accurate estimation of bias p.

We divide each trace interval into x bins and aggregate the requests within each bin. We estimate bias p for each trace interval by using its entropy value and Equation 10. Choice of an appropriate bin size is crucial to the success of this method; if it is too large, bursty requests are aggregated, destroying local burstiness. If the bin size is too small, the fraction of empty bins is too large and there are not enough samples to estimate p accurately. In intervals with little I/O activity, it may simply not be possible to estimate p.

From Section 3, we should choose an trace interval between 3–10 seconds to keep the error under 5%. We also need to choose a bin size that yields a reasonable percentage of non-empty bins without over-aggregating. Figure 10

CALCULATE-P
INPUT: length l, trace interval w.
OUTPUT: bias p.
ALGORITHM:

for each i from 1 to $log_2 l$
  calculate the entropy value $E^{(i)}$ of w using Equation 9
  $array[i] \leftarrow E^{(i)}$
end for
estimate bias p from entropy values in *array* using linear regression
return p

Figure 12. Bias p estimation algorithm.

shows typical cumulative distribution functions of request interarrival times for cello and snake. The percentage of requests with interarrival times less than 10 ms is 15% for cello and 20% for snake; these percentages are relatively small. Based on this observation, we select a bin size of 10 ms to avoid over-aggregating requests. We choose an trace interval size of 5.12 seconds so that the number of bins within an interval is a power of 2.

To determine what fraction of non-empty bins is necessary to obtain a good estimate of p, we calculate p using both the entropy method and the spectrum method for selected data sets with certain percentages of non-empty bins, as shown in Figure 11. We could not exhaustively test all the data because the estimation of p using multifractal spectra is a visual test. Therefore, we selected a subset of data sets as follows. Because the data sets with the same percentage of non-empty bins might have different aggregation ratios (number of requests in the interval / number of non-empty bins), we used the histogram of aggregation ratios to further round out our sample data sets. For example, if 20% of the trace intervals with 5% non-empty bins have an aggregation ratio of 1.6 (rounding to the nearest tenth), 20% of our samples have those characteristics.

We require that the fraction of non-empty bins be at least 3% for meaningful estimation of p. If the fraction of non-empty bins is smaller than 3% we can use any distribution, for example, a uniform distribution, to fit the data.

5.3. Multifractal Interarrival Synthesis Algorithm

We propose a new algorithm to synthesize the patterns of disk request interarrival times based on real traces. The approach is to divide the real trace into non-overlapping intervals with equal time length and fit each trace interval to a binomial multifractal distribution. The parameter bias p is calculated using the entropy method (Equations 8–10), as shown in Figure 12, where w is a real trace interval and $\log_2 l$ is the maximum aggregation level in Equation 9.

IMPROVED-BINOMIAL-MULTIFRACTAL-GENERATION
INPUT: bias p, length l, initial mass m, common request size r.
OUTPUT: a timestamped request sequence $((t_1, m_1), (t_2, m_2), \ldots, (t_n, m_n))$.
ALGORITHM:

1. Initialize the stack and push pair (l, m) onto the stack; initialize the logical clock c.
2. If the stack is empty, return; otherwise, go on to Step 3.
3. Pop a pair (l_i, m_i) from the stack. If $l_i = 1$, distribute the mass m_i in requests of size r in the time interval $(c, c+1)$, advance c by 1, and then go back to Step 2; if $0.5 \times r < m_i < 1.5 \times r$, try to combine the top item (l_{i+1}, m_{i+1}) in the stack to generate output, advance c by l_i or $l_i + l_{i+1}$ accordingly, and then go back to Step 2. Each generated request (t_k, m_k) has a size associated with a logical timestamp.
4. Flip a coin. If head, push pairs $(l_i/2, m_i \times p)$ and $(l_i/2, m_i \times (1-p))$ into the stack; if tail, push them in reverse order. Go back to Step 2.

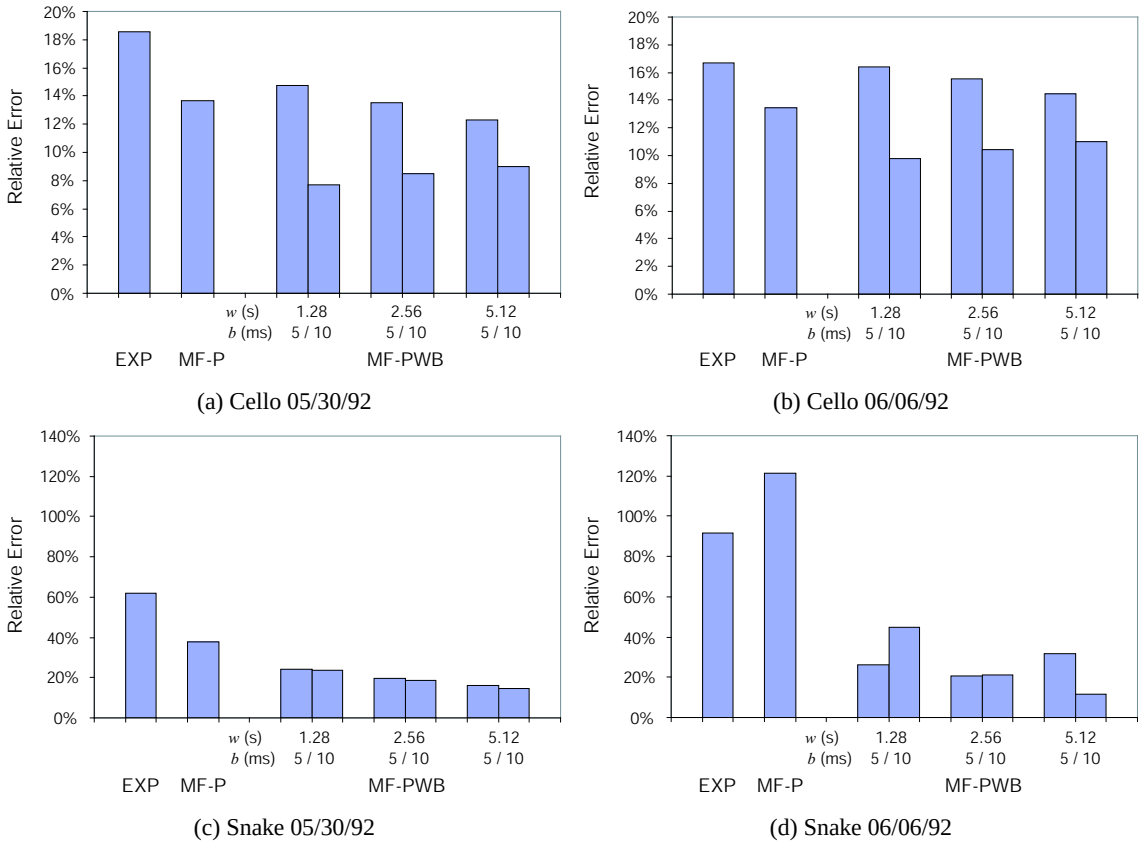
Figure 13. Binomial multifractal I/O request generation.

SYNTHETIC-TRACE-GENERATION
INPUT: interval length s, bin size b, original trace file f, common request size r.
OUTPUT: synthetic trace file.
ALGORITHM:

for each non-empty interval w in f
  if the fraction of non-empty bins $< 3\%$, $p = 0.5$
  else p = CALCULATE-P(s/b, w)
  resolution = 1 ms
  length = s/resolution
  mass = aggregated request size in interval w
  IMPROVED-BINOMIAL-MULTIFRACTAL-GENERATION $(p, length, mass, r)$
  map local timestamps to real timestamps
end for

Figure 14. Synthetic I/O trace generation algorithm.

The key idea of synthesis is to use the aggregated request size (mass) from the original trace interval and redistribute the mass in time according to the calculated bias. Figure 13 shows our algorithm for multifractal trace generation, based on [19]. The input of initial mass m indicates



(a) Cello 05/30/92

(b) Cello 06/06/92

(c) Snake 05/30/92

(d) Snake 06/06/92

Figure 15. Relative error of interarrival time and request size syntheses.

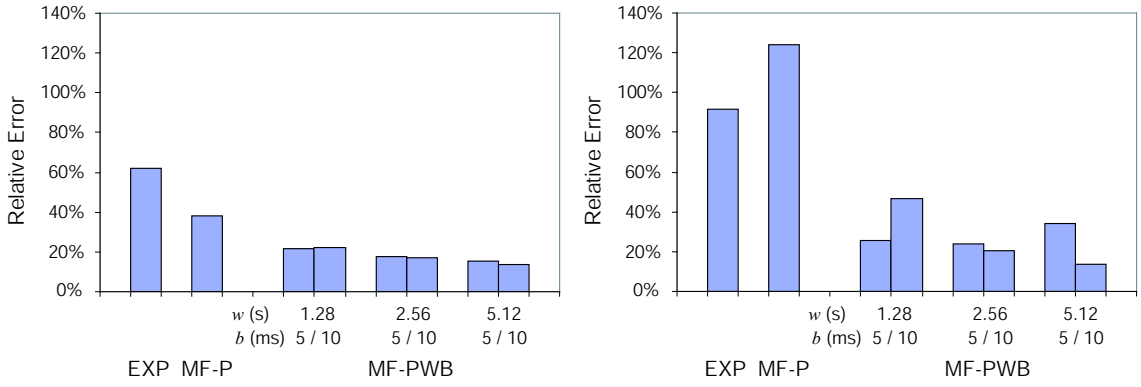
the amount of data accessed in the interval while bias p defines the burstiness of data accesses. Length l is determined by the time length of a real trace interval and the desired granularity of synthetic request timestamps. Binomial multifractal generation is an iterative process and $log_2 l$ defines the maximum iteration level. This algorithm uses a logical clock c to keep track of the time advance in trace generation. Due to the multiplicative nature of the generation, the original algorithm from [19], which proceeds the iteration until the maximum iteration level is reached, often creates many small requests, which can induce synthetic errors as high as 800%. To avoid this, we use the knowledge that the size of 70–80% of disk requests in the cello and snake traces, generated under HP-UX, are 8 KB [14], and define the common request size r of 8 KB as an input to the algorithm, which stops unnecessary higher-level iterations when the mass being distributed becomes too small.

Figure 14 shows how to use IMPROVED-BINOMIAL-MULTIFRACTAL-GENERATION to synthesize a trace. This algorithm divides the real trace f into non-overlapping intervals and treats them as independent of each other because of the irrelevance of long-range dependence beyond the interval length s. Consequently, the trace synthesis process in each generation interval is also independent. The algorithm also takes as input a selected bin size b for the purpose of calculating p, as described in Section 5.2. The iteration of binomial multifractal generation in an interval stops when the resolution of one millisecond is reached.

6. Simulation Results

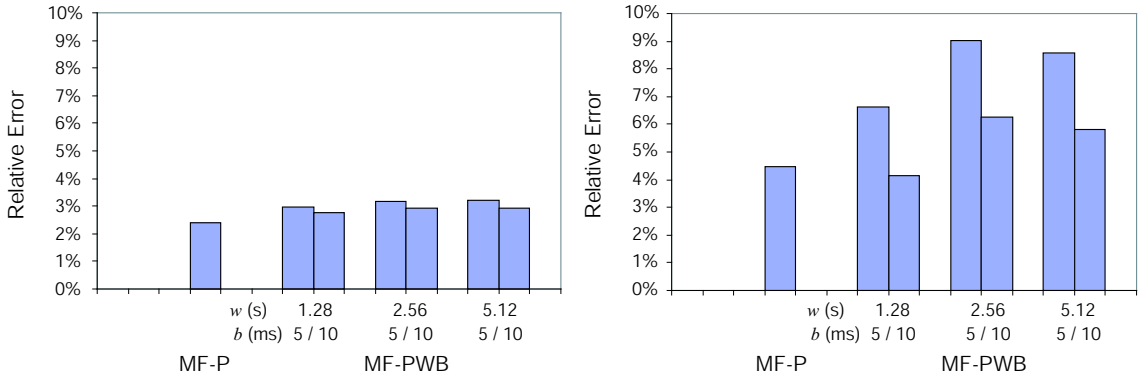
We use our proposed I/O trace generation method, as described in Figure 14, to generate synthetic workloads. Because we do not attempt to synthesize request starting locations or read/write operations, we retain the same sector identifiers and operations from the original trace, preserving spatial locality in the workload. Therefore, our baseline for comparison is the original trace. We use both original and synthetic traces to excise the Pantheon disk simulator [20] and evaluate the synthesis accuracy by measuring the relative error (RMS) between the cumulative distribution functions of disk response times for the original and synthetic traces.

To measure the improvement obtained by using syntheses based on fine-grained trace parameters, we compare our method, MF-PWB, to a simple exponential interarrival model, EXP, and a variant of the method proposed by Wang *et al.* [19], MF-P. In MF-P, bias p is calculated over the en-



(a) Snake 05/30/92 (b) Snake 06/06/92

Figure 16. Relative error of interarrival time synthesis.



(a) Snake 05/30/92 (b) Snake 06/06/92

Figure 17. Relative error of request size synthesis.

tire trace, but we use the algorithm in Figure 13 to adjust the generated trace for the most common request size. The key difference between MF-P and MF-PWB is that MF-PWB uses more parameters and does not model long-range dependence at time scales greater than *w*, which is the trace interval length. Another parameter of MF-PWB is the bin size *b*.

Figures 15(a)–15(d) show the relative errors, measured by RMS of response time distributions, of EXP, MF-P and MF-PWB with different parameters for the cello and snake traces in 05/30/92 and 06/06/92. We selected these specific days because they have the maximum and minimum mean response times for the snake traces, which in general are more bursty and harder to model than cello. Note that EXP generates synthetic interarrival times only. The relative error of MF-PWB ranges from 7.7% to 44.6%, depending upon the trace itself and the parameters *w* and *b*; the error of MF-P ranges from 13.5% to 121.6%; and the error of EXP ranges from 16.7% to 91.6%, which is at least twice the error of MF-PWB.

In general, MF-PWB reproduces interarrival patterns more accurately than MF-P; computing *p* at smaller time intervals generally translates into more accurate synthesis. This improvement is significant for snake but less significant for cello if we select poor values of *b*. However, the results for cello and snake are still comparable.

To better illustrate the quality of synthetic request arrival times and sizes, we isolate the effect of each. Figures 16(a) and 16(b) show the relative errors, caused by interarrival time synthesis, of EXP, MF-P and MF-PWB for snake. Only arrival times are synthetic; we obtain all of the other request parameters from the original trace. The errors are almost the same as those from traces with both synthetic request arrival times and sizes, as shown in Figures 15(c) and 15(d).

Figures 17(a) and 17(b) show the relative errors, caused by request size synthesis, of MF-P and MF-PWB for snake. Here only request sizes are synthetic. We exclude EXP in comparison because it does not generate synthetic request sizes. Request size synthesis accounts for less than 10% of the synthesis error, and MF-P is slightly better than MF-PWB for that component of synthesis. The results of synthesis error analysis for cello are similar [9]. Thus, for both

cello and snake the majority of synthesis errors by MF-P and MF-PWB comes from synthetic arrival times.

7. Conclusions and Future Work

For the purpose of performance evaluation, we need only consider I/O activities at time scales related to the system we are evaluating. For measuring disk response times and queuing behaviors, we determined that a interval length of five seconds bounded the error to less than 5% for two workloads, one random and one sequential.

However, accurately capturing I/O burstiness is extremely important. We demonstrated a method of synthesizing interarrival times using binomial multifractals that exploits the fact that long-range dependence is unnecessary beyond certain time scales. Using this method, we synthesized traces with a relative error that ranged from approximately 8% to 12% on random and sequential workloads.

We are currently working on methods that can automatically determine the appropriate interval length, and on combining this model for temporal locality with a similar one for spatial locality in I/O workloads.

Acknowledgments

We thank Mengzhi Wang and Christos Faloutsos for their help with binomial multifractal synthesis. We are also grateful to John Wilkes for his suggestions on this work and providing us traces and the Pantheon software. We thank Eitan Bachmat and Kimberly Keeton for their helpful comments on our work. Our shepherds Julian Satran and Robert Chadduck helped us finalize the paper. This work was supported in part by the National Science Foundation under grants NSF CCR-0093051 and NSF IDM-0083130.

References

[1] J. Beran. *Statistics for Long-Memory Processes*. Chapman & Hall, New York, NY, 1994.

[2] F.-M. Chen, J. Mellor, and P. Mars. On burstiness of self-similar traffic models. In *Proceedings of the European Conference on Networks and Optical Communications 1996 (NOC '96)*, pages 146–153, 1996.

[3] H. Chen, H. Cai, and Y. Li. The multifractal property of bursty traffic and its parameter estimation based on wavelets. In *Proceedings of IEEE TENCON'97. IEEE Region 10 Annual Conference*, volume 2, pages 791–794, December 1997.

[4] M. E. Crovella and A. Bestavros. Self–similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[5] G. R. Ganger. Generating representative synthetic traces: An unsolved problem. In *Proceedings of the 21st International Computer Measurement Group Conference (CMG95)*, pages 1263–1269, Dec. 1995.

[6] M. E. Gómez and V. Santonja. Analysis of self-similarity in I/O workload using structural modeling. In *Proceedings of the 7th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '99)*, pages 234–243, College Park, MD, October 1999. IEEE.

[7] S. Gribble, G. S. Manku, E. Brewer, T. J. Gibson, and E. L. Miller. Self-similarity in file systems: Measurement and applications. In *Proceedings of the 1998 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 141–150, Madison, WI, June 1998.

[8] M. Grossglauser and J.-C. Bolot. On the relevance of long-range dependence in network traffic. *IEEE/ACM Transactions on Networking*, 7(5):629–640, 1999.

[9] B. Hong. Techniques for synthetic I/O workload generation. M.sc. thesis, University of California at Santa Cruz, Sept. 2002.

[10] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, pages 1–15, 1994.

[11] P. Meakin. *Fractals, scaling and growth far from equilibrium*. Cambridge University Press, 1998.

[12] A. L. Neidhardt and J. L. Wang. The concept of relevant time scales and its application to queuing analysis of self-similar traffic (or is Hurst naughty or nice?). In *Measurement and Modeling of Computer Systems*, pages 222–232, 1998.

[13] R. H. Riedi. Introduction to multifractals. Technical Report 99-06, Rice University, Sept. 1999.

[14] C. Ruemmler and J. Wilkes. Unix disk access patterns. In *Proceedings of the Winter 1993 USENIX Technical Conference*, pages 405–420, San Diego, CA, Jan. 1993.

[15] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–29, Mar. 1994.

[16] C. E. Shannon and W. Weaver. *Mathematical Theory of Communication*. University of Illinois Press, 1963.

[17] M. Taqqu. Time series with long-range dependence. http://math.bu.edu/people/murad/methods/index.html.

[18] M. Wang, A. Ailamaki, and C. Faloutsos. Capturing the spatio-temporal behavior of real traffic data. In *IFIP International Symposium on Computer Performance Modeling, Measurement, and Evaluation*, Rome, Italy, 2002.

[19] M. Wang, N. H. Chan, S. Papadimitriou, C. Faloutsos, and T. Madhyastha. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *Proceedings of the 18th International Conference on Data Engineering (ICDE '02)*, pages 507–516, Feb. 2002.

[20] J. Wilkes. The Pantheon storage-system simulator. Technical Report HPL-SSP-95-14, Storage Systems Program, Computer Systems Laboratory, Hewlett-Packard Laboratories, Palo Alto, CA, May 1996.