Characterization of Self-Similarity in Memory Workload: Analysis and Synthesis

Qiang Zou
School of Computer Science
Southwest University
Chongqing 400715 China
qzou@swu.edu.cn

Jianhui Yue
Dept. of Elec. & Computer
University of Maine
Orono, ME 04469 USA
jyue@eece.maine.edu

Bruce Segee
Dept. of Elec. & Computer
University of Maine
Orono, ME 04469 USA
segee@eece.maine.edu

Yifeng Zhu
Dept. of Elec. & Computer
University of Maine
Orono, ME 04469 USA
zhu@eece.maine.edu

***Abstract*—This paper studies self-similarity of memory I/O accesses in high-performance computer systems. We analyze the auto-correlation functions of memory access arrival intervals with small time scales and present both pictorial and statistical evidence that memory accesses have self-similar like behavior. For memory I/O traces studied in our experiments, all estimated Hurst parameters are larger than 0.5, which indicate that self-similarity seems to be a general property of memory access behaviors. In addition, we implement a memory access series generator in which the inputs are the measured properties of the available trace data. Experimental results show that this model can accurately emulate the complex access arrival behaviors of real memory systems, particularly the heavy-tail characteristics under both Gaussian and non-Gaussian workloads.**

I. Introduction

Accurately characterizing memory access behavior in computation-intensive workloads is essential to understanding the performance of the memory system. Researchers in both academia and industry have developed various benchmark suites, such as commercial workloads [15], desktop applications [16], multimedia applications [8], and XQuery applications [19] to test and evaluate the memory systems. However, for many benchmarks, it takes weeks or even months to complete a single run on cycle accurate execution-driven simulators such as M5 [2] and SimpleScalar [47]. In addition, a memory system has a large design space to be explored, such as close/open bank model, address mapping schemes, and transition control between different power states, and accordingly it becomes increasingly more challenging to run benchmarks multiple times in order to obtain comprehensive and fair evaluation. Synthetic benchmarks provide an improved methodology to speed up the evaluation process. The most critical issue in designing a synthetic benchmark is to accurately characterize the memory access behavior. In this paper, we focus our study on analyzing and modeling of memory I/O access arrivals.

Analysis of memory system access characteristics and patterns has received considerable attention in the past few years. Several studies have investigated the basic characteristics of memory accesses, such as cache miss rates, I/O intensity, and impacts of page size, in SPEC CPU benchmark [4], [6], [7], [12]. Eeckhout *et al* [44] model the access sequence as a Statistical Flow Graph (SFG), in which basic blocks and their mutual transition probability are statistically identified. Joshi *et al* [45] and Bell *et al* [46] model memory accesses as a mixed sequence of constant and variable strides. Ganesan *et al* [43] propose to extract the memory level parallelism (MLP) from the real benchmark to estimate memory access burstiness and they consider the variations of the time intervals between consecutive burstiness of on-chip cache misses when modeling the burstiness of memory accesse. Li [25] studies the scaling properties of SPEC2000 integer benchmarks and proposes an method to estimate the short-term and long-term execution characteristics of large programs. However, none of these studies statistically examine in detail the burstiness of memory access requests, particularly the widely popular phenomena of random fluctuations in request arrival rates at different time scales.

In this paper, we study several sets of memory I/O traces collected in the SPEC CPU 2000 and SPEC CPU 2006 benchmark suites. We demonstrate the existence of self-similar like phenomenon in memory I/O workloads over small time scales. We refer to this property as "self-similar like behavior" because memory traces studied in this paper are only in small time scales. A truly self-similar process should exhibit the self-similarity at all time scales. We analyze the correlations of inter-access times and study the self-similarity in memory workloads. To the best of our knowledge, little research work on this topic has been reported in the literature.

This paper makes the following three contributions:

- Our study shows that there are evident correlations between inter-access time intervals in traces collected in almost all benchmarks in SPEC2000 and SPEC2006, and exceptionally strong correlations in some of them. This suggests that further study of the self-similarity is needed to understand the statistical phenomena of memory I/O accesses.
- We examine both integer and floating-point type of memory accesses, and present visual and mathematical evidence to show that memory accesses exhibit self-similar like behavior over small time scales.
- We propose a mathematical model based on the α-stable process to accurately synthesize the memory access series, particularly the heavy-tail characteristics under Gaussian and non-Gaussian workloads.

The rest of this paper is organized as follows. Section II summarizes related research works. Section III studies the correlation of inter-access times and discusses the necessity of studying self-similarity in memory I/O workloads. Section IV presents the visual evidence of self-similarity in memory workload. Section V shows the estimates of Hurst parameters. Section VI proposes an α-stable model to synthesize the memory access series and compares the workloads synthesized by the proposed model with real traces. Section VII concludes this paper.

II. Related Work

The characteristics of self-similarity and long-range dependence in data traffic was initially found in computer network traffic [26]. Since then extensively research work have been done to investigate this important nature in computer and network systems. For example, many studies have concluded that network behavior exhibits the long-range dependence, scaling phenomenon, and heavy-tailed distribution [24], [27], [28], [33]. Willinger *et al* [28] show that the self-similarity can be attributed to the ON/OFF behavior of traffic sources within their system. Crovella *et al* [33] verify the existence of self-similarity in web traces.

Self-similarity has also been studied in the context of file and disk I/O workloads [29], [30], [31], [37], [38], [40]. For example, Gribble *et al* [29] give visual and statistical evidences to demonstrate that high-level file system events exhibit self-similar behavior for a short-term time scale of approximately a day. Gomez *et al* [30], [31] show that disk-level I/O requests are self-similar in nature, and propose a structural modeling to synthesize disk-level I/O arrival patterns. Recently, Kavalanekar *et al* [40] capture twelve sets of storage traces from Exchange, software build and release, Live Maps, MSN storage, security authentication, and display advertisement platform servers in Microsoft Corporation. They analyze and confirm the self-similarity of block-level I/O in both time and space. Additionally, Riska *et al* [37], [38] demonstrate that disk drives in a wide range of computing environments exhibit high variability, strong burstiness, and long-range dependence.

Intensive research work has been done to study the characteristics of memory workloads [6], [7], [12], [18], [22], [43], [48], [49]. For example, based on microarchitecture-independent metrics such as the memory level parallelism (MLP), Ganesan *et al* [43] build a model of the burstiness of memory accesses under the workloads of SPEC CPU 2006 and ImplantBench by considering the frequency of on-chip cache misses. However, no research has been done to study or confirm the self-similarity nature of memory access workloads, to the best of our knowledge. In this paper, we deploy Leland's theory and analysis techniques [26] to analyze and examine the existence of self-similarity in memory system workloads. This research work is important since it can provide guidance to researchers how to correctly model and evaluate the I/O arrivals and burstiness in memory systems.

TABLE I
PROCESSOR PARAMETERS

Parameter	Value
Frequency	2 GHz
L1 I-cache	32 KB
L1 D-cache	32 KB
L2 Cache	2 MB
L2 Cache Line Size	64 Bytes

TABLE II
DRAM PARAMETERS

Parameter	Value
Frequency	667 MHz
tRP: Row Precharge time	12 ns
tRCD: Row active to row active delay	12 ns
tRAS : Row Activation time	27 ns
tCAS: Delay to access a certain column	8 ns
IDD0: Active precharge current	85 mA
IDD2P: Precharge powerdown current	7 mA
IDD2N: Precharge standby current	40 mA
IDD3N: Active standby current	55 mA
IDD3P: Active powerdown current	30 mA
IDD4R: Burst read current	135 mA
IDD4W: Burst write current	135 mA
IDD5: Burst refesh current	210 mA
Vdd: Supply voltage	1.8v
#Ranks per DIMM	2
Rank capacity	256 MB
#Banks per Rank	8
#Rows per Bank	16,384
#Columns per Row	1,024
Channel Width	8 Bytes
Memory Scheduling Algorithm	FCFS

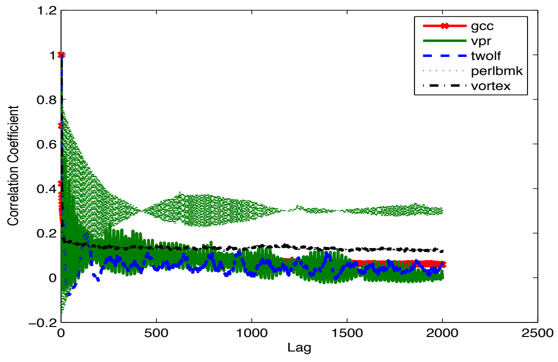
III. Memory Workload Measurement

In this section, we briefly introduce memory I/O workloads to be analyzed in this paper and then study the correlation of arrival intervals or inter-access times (IAT) to characterize the memory access behavior.

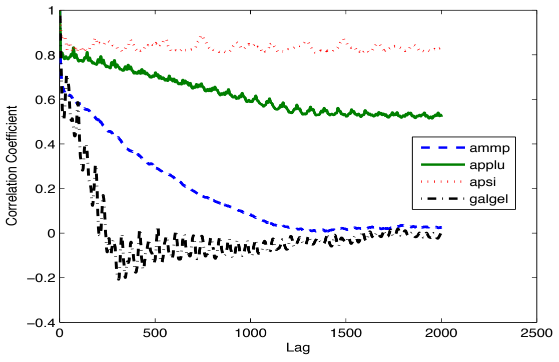
A. Collection of memory access traces

In this paper, we choose SPEC CPU 2000 and 2006 as our target workloads. Both SPEC 2000 and 2006 are the standardized computation-intensive benchmark suite widely used in both academia and industry to comprehensively and fairly evaluate the performance of CPUs, memory systems, and compiler techniques. These benchmarks are developed by using platform-neutral C/C++ or Fortran languages and thus they can run on a wide variety of computer architectures. Both benchmark suites include integer applications and float-point applications, and the detailed description of each application is given in Ref. [5] and [7], respectively. SPEC 2006 benchmark suite was released to further exercise the tested platforms. However, in both academia and industry, SPEC 2000 is still widely used.

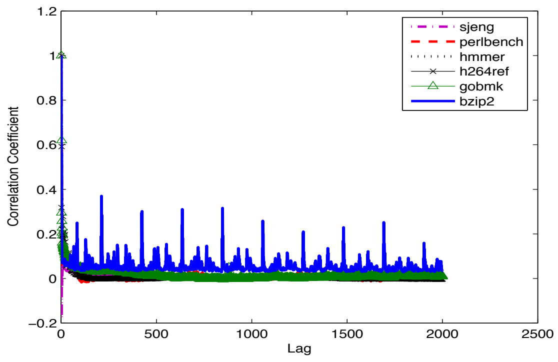
We have collected the memory access trace of the SPEC 2000 and 2006 benchmark suites using an execution-driven processor simulator called M5 [2]. We have integrated a cycle-level DRAM simulator named DRAMsim [3] into M5 in order to accurately simulate the memory system. Table I and II show



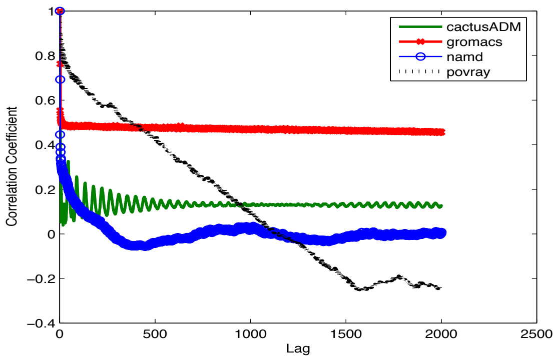
(a) SPECint2000 applications



(b) SPECfp2000 applications



(c) SPECint2006 applications



(d) SPECfp2006 applications

Fig. 1. Auto-correlation functions (ACFs) of memory accesses for the SPECint2000 applications (*gcc*, *vpr*, *twolf*, *perlbmk*, and *vortex*), the SPECfp2000 applications (*ammp*, *applu*, *apsi*, and *galgel*), the SPECint2006 applications (*sjeng*, *perlbench*, *hmmer*, *h264ref*, *gobmk*, and *bzip2*), and the SPECfp2006 applications (*cactusADM*, *gromacs*, *namd*, and *povray*).

the parameters of the processor and Micron DDR2 memory [1] used in our simulation experiments.

TABLE III
SUMMARY OF MEMORY ACCESS TRACES.

	SPEC 2000	Trace duration (nanosec.)	SPEC 2006	Trace duration (nanosec.)
Integer	*gcc*	646,034,651	*perlbench*	100,293,948
	vpr	126,198,640	*bzip2*	113,789,777
	twolf	121,289,348	*astar*	275,606,430
	perlbmk	165,041,932	*mcf*	1.7406e+009
	vortex	122,437,501	*gobmk*	99,584,340
	gzip	153,188,589	*hmmer*	49,996,087
	mcf	168,405,856	*sjeng*	85,787,863
	parser	131,240,186	*xalancbmk*	205,785,089
	gap	91,373,966	*h264ref*	61,018,687
	bzip	359,328,264	*omnetpp*	624,006,166
Floating point	*ammp*	224,507,172	*cactusADM*	89,682,223
	applu	485,690,806	*gromacs*	81,601,154
	apsi	538,127,920	*namd*	47,286,401
	galgel	373,450,978	*povray*	59,257,422

This paper randomly selects 14 applications from SPEC 2000 and 14 applications from SPEC 2006. We run these applications in M5 and save the memory accesses into trace files. Selected applications of SPEC 2000 include ten integer benchmarks (*gcc*, *vpr*, *twolf*, *perlbmk*, *vortex*, *gzip*, *mcf*, *parser*, *gap* and *bzip*), and four floating-point benchmarks (*ammp*, *applu*, *apsi* and *galgel*). From the SPEC 2006, we select ten integer benchmarks, including *perlbench*, *bzip2*, *astar*, *mcf*, *gobmk*, *hmmer*, *sjeng*, *xalancbmk*, *h264ref* and *omnetpp*, and four floating-point benchmarks, including *cactusADM*, *gromacs*, *namd* and *povray*. All memory access timestamps were recorded in nanoseconds. The basic information of these memory traces are summarized in Table III.

B. Correlation study

In order to identify the statistical characteristics and gain a deep understanding of memory access behaviors, in this paper, we first study the similarity of inter-access times in memory I/O streams over different time spans by using autocorrelation functions (ACF). The detailed introduction of this mathematical tool can be found in Ref. [39].

Given a set of observations $X = (X_t : t = 1, 2, \ldots, N)$, the correlation coefficient at *lag* k is defined as

$$c_k = \frac{1}{N-k}\sum_{i=1}^{N-k}(X_i - \bar{X})(X_{i+k} - \bar{X}) \qquad (1)$$

where $\bar{X}$ is the expectation of the time series X. Then the auto-correlation function $ACF(k)$, with a *lag* of k, is defined as

$$ACF(k) = \frac{c_k}{c_0}. \qquad (2)$$

In the following, we use auto-correlation functions (ACF) to study the characteristics in inter-access times for both the integer and floating-point memory traces summarized in Table III from a time dependence perspective. Due to space limitation, we only present the analytical results of *gcc*, *vpr*, *twolf*, *perlbmk*, *vortex*, *ammp*, *applu*, *apsi*, *galgel* for SPEC2000, and *perlbench*, *bzip2*, *gobmk*, *hmmer*, *sjeng*, *h264ref*, *cactusADM*, *gromacs*, *namd*, *povray* for SPEC2006, as shown in Figure 1.

If the correlation coefficient of inter-access times quickly decreases to zero, it can be concluded that the memory access traffic is expected to be smooth instead of bursty and little or no correlations exist between the inter-access times. In this case it is reasonable to model the inter-access time as a sequence of random variables with independently and identically distribution (IID). On the contrary, if the correlation coefficient does not approach to zero quickly, then there exists some degree of correlations between inter-access times and such memory traffic is expected to be bursty instead of smooth. As a result, the inter-access time cannot be modeled as a simple IID random process and further study of auto-similarity is then necessary in order to correctly model the memory traffic.

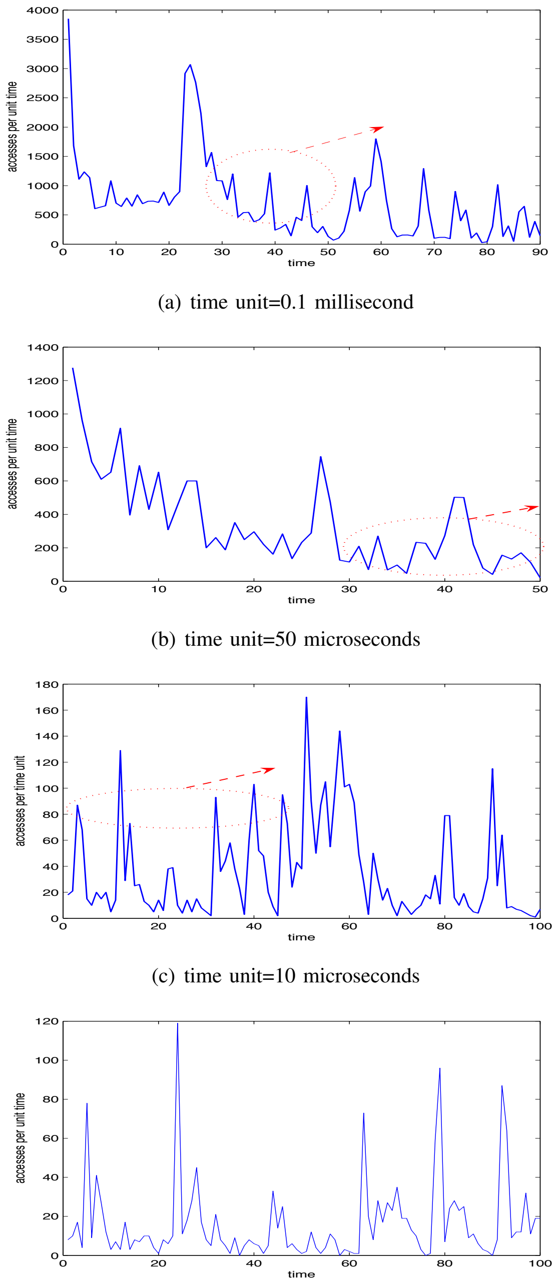
Figure 1(a)-(d) respectively plot the auto-correlation coefficient of memory accesses of studied integer benchmarks and floating-point benchmarks as the *lag* parameter increases gradually from 0 to 2000. The results show that there are evident correlations in all studied traces for all auto-correlation functions of the inter-access times, exceptionally strong correlations in some traces such as *applu*, *apsi* and *gromacs*. This indicates that independently and identically distributed (IID) processes might not be appropriate in characterizing the memory accesses. Therefore, it is necessary to study and investigate the self-similarity of the memory traffic. This important observation motivates the research work of this paper.

IV. Self-similarity in Memory Workloads

In this section, we present the visual and intuitive evidence to demonstrate the existence of self-similarity in studied memory traces.

A. Visual Examination of Self-similarity

Informally, self-similarity means scale invariance because there exists the same or similar statistical properties such as mean, variance and marginal distribution across all time scales. Existing works focused on network traffic [24], [26], [27], [28], disk I/O workload [30], [34], [35], [40], usually examine the self-similarity of traces for long-term time scales, such as days, months or even years. Due to the fact that the SPEC benchmarks are memory I/O intensive and the memory activities are bursty and need to be collected at the accuracy of nanoseconds, most memory I/O traces studied in this paper last less than one second. It is far shorter than the time scale used in disk or network I/O studies. However, the time-stamp to record the memory traces accurate to nanoseconds which are also far shorter than the time-stamp accuracy of milliseconds



(a) time unit=0.1 millisecond

(b) time unit=50 microseconds

(c) time unit=10 microseconds

(d) time unit=5 microseconds

Fig. 2. Pictorial demonstration of self-similarity: memory workload (memory accesses per time unit for the *gcc* trace) on four different time scales.

or even seconds in network packet or disk I/O traces. Thus, in this paper, we consider the scale of memory traces as a notion of "long-term" time scale corresponds to the nanosecond-level time-stamp.

For each set of memory access traces summarized in Table III, we compute the memory access arrival rates (i.e., the number of memory accesses per time unit) at four different choices of time units. Figure 2 depicts a sequence of simple plots of the memory access arrival rates for the *gcc* trace. Figure 2(a) is plotted with a time unit of 0.1 milliseconds. We concentrate on a randomly chosen small region, as indicated by the circle and arrow, then increase the time resolution, and obtain Figure 2(b). From Figure 2 (a) to (d), each successive plot is a small portion of the previous one by increasing the time resolution. Obviously there are many sharp activity "spikes" in each plot. The presence of "spikes" in each plot indicate that the burstiness of memory access, i.e., each spike corresponds to a "burst" in memory access workload. From Figure 2 (a) to (d), we observe the scale-invariant feature of memory workload: each burst interval consists of bursty subintervals. So it is not easy to distinguish among the four plots in a distributional sense because all plots look intuitively very "similar" to one another. This is a strong intuitive evidence of the self-similarity of memory accesses.

B. Theory of self-similarity

The theory behind self-similar processes is briefly summarized as follows. A more thorough description can be found in [26], [27], [28]. This section only outlines the basics that will be used in Section V. The description of self-similarity given below closely follows Beran *et al* [24].

Let $X = (X_t : t = 1, 2, \ldots)$ be a covariance stationary stochastic process with constant mean $\mu = E[X_t]$, and finite variance $\sigma^2 = E[(X_t - \mu)^2]$. For the process X_t, the autocorrelation function $ACF(k)$ depends only on k and is defined as follows.

$$ACF(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{E[(X_t - \mu)^2]}, \ for\ k \geq 0. \quad (3)$$

The process X_t is said to exhibit self-similarity if

$$\lim_{k \to \infty} \frac{ACF(k)}{k^{-\beta}} = c < \infty, \ for\ 0 < \beta < 1. \quad (4)$$

Note that, in the equation above, $ACF(k)$ is non-sumable, i.e., $\sum_k ACF(k) = \infty$. We say that such an autocorrelation function decays hyperbolically and the corresponding process X_t is long-range dependent. In contrast, the autocorrelation function of a Poisson process decays exponentially and is sumable; that is $\sum_k ACF(k) = 0$. Such a process is said to be short-range dependent.

The process X_t is said to be exactly second-order self-similar with the *Hurst parameter* H ($0.5 < H < 1$), if X_t has an autocorrelation function of the form

$$ACF(k) = \frac{1}{2}[(k+1)^{2-\beta} - 2k^{2-\beta} + (k-1)^{2-\beta}]. \quad (5)$$

For the process X_t, its aggregated process $X^{(m)}$ is given as $X_k^{(m)} = \frac{1}{m}\sum_{j=0}^{m-1} X_{km-j}$, for $k \geq 1$. And

$$Var(X^{(m)}) = \sigma^2 m^{-\beta}, \ for\ 0 < \beta < 1. \quad (6)$$

The process X_t is said to be asymptotically second-order self-similar with the Hurst parameter H ($0.5 < H < 1$), if

$$\lim_{m \to \infty} ACF^{(m)}(k) = ACF(k) \quad (7)$$

V. Estimating the Hurst Parameter

In this section, we use rigorous statistical techniques to estimate the Hurst parameter of memory access workloads, and mathematically demonstrate the presence of self-similar behavior in all studied memory workloads. The Hurst parameter noted H measures the self-similar degree of a time-series, and a value in the range $(0.5, 1)$ indicates self-similarity [31]. The larger the Hurst estimate is, the higher the degree of auto-similar property is. Two techniques that we employ are well-known graphical tools, namely *variance-time plots* [26], [29] and *R/S analysis* (Pox plot) [26], [29], as discussed below.

A. Variance-time plots

As introduced in the prior section, for an asymptotically second-order self-similar process X_t, the relation between the variance of the aggregated process $X^{(m)}$ and the block size m is defined by Equation (6). Taking the logarithm of both sides of the equation results in the relation

$$log(Var(X^{(m)})) \approx a - \beta log(m), \quad (8)$$

where a is a constant, and $m \to \infty$ [29]. Thus, we can plot the curve of $log(Var(X^{(m)}))$ versus $log(m)$, for various values of m. The curve will be a linear series of points with slope $-\beta$, and using a linear regression method we can obtain an estimate of β. Slopes between -1 and 0 correspond to Hurst parameters H between 0.5 and 1. This plot is called a *variance-time plot*, and we can calculate the Hurst parameter H using the following equation

$$H = 1 - \frac{\beta}{2}. \quad (9)$$

Figure 3 illustrates the variance-time plots for the integer benchmarks (e.g., *vortex* for SPEC2000 and *h264ref* for SEPC2006), and the floating point benchmarks (e.g., *apsi* for SPEC2000 and *cactusADM* for SPEC2006). The results show that all four plots are linear and they have the Hurst parameter of 0.987, 0.651, 0.892 and 0.723, respectively. This verifies the self-similar nature of these memory access workloads. Especially, the curve in plot (a) is more linear than those in plot (b), (c) and (d). This observation implies that there is a higher degree of self-similarity for memory trace *vortex*.

We generate the variance-time plots for all traces listed in Table III and then estimate their Hurst parameter from the plots, as shown in Table IV. The results show that all Hurst parameters are significantly larger than 0.5, indicating that all studied memory workloads exhibit self-similarity. It is surprising to find that the estimated Hurst parameters for all SPEC2000 integer memory traces are approximately the same, such as 0.987 for *vortex*, 0.937 for *twolf*, 0.995 for *gcc* and 0.993 for *mcf*, and 0.996 for both *perlbmk* and *parser*. This indicates that these SPEC2000 integer benchmarks have the similar level of self-similarity.

B. R/S-Analysis and Pox plots

Another commonly used method to estimate the Hurst parameter is R/S (rescaled adjusted range) analysis, also called

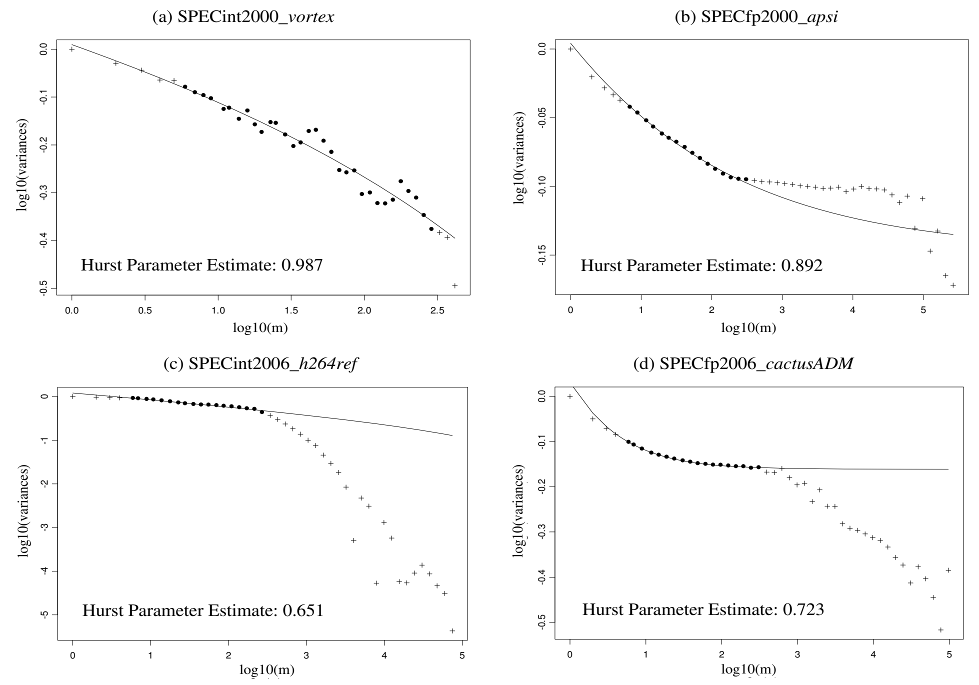


Fig. 3. Variance time plots for the integer benchmarks (e.g., *vortex* for SPEC2000, and *h264ref* for SPEC2006), and the floating point benchmarks (e.g., *apsi* for SPEC2000, and *cactusADM* for SPEC2006). The variable m represents the aggregation level in microsecond.

TABLE IV
ESTIMATE OF *H* USING VARIANCE-TIME PLOT AND POX PLOT (R/S) FOR THE SPEC2000 AND SPEC2006 BENCHMARKS.

Type	Benchmarks	Variance-time plot	Pox plot (R/S)
SPECint 2000	*gcc*	0.995	0.859
	vpr	0.872	0.699
	twolf	0.937	0.609
	perlbmk	0.996	0.507
	vortex	0.987	0.790
	gzip	0.906	0.795
	mcf	0.993	0.586
	parser	0.996	0.814
	gap	0.819	0.618
	bzip	0.994	0.786
SPECfp 2000	*ammp*	0.861	0.653
	applu	0.629	0.696
	apsi	0.892	0.652
	galgel	0.934	0.716
SPECint 2006	*perlbench*	0.805	0.675
	bzip2	0.757	0.639
	astar	0.694	0.518
	mcf	0.615	0.572
	gobmk	0.923	0.608
	hmmer	0.787	0.641
	sjeng	0.679	0.586
	xalancbmk	0.729	0.611
	h264ref	0.651	0.701
	omnetpp	0.850	0.583
SPECfp 2006	*cactusADM*	0.723	0.593
	gromacs	0.692	0.738
	namd	0.918	0.587
	povray	0.863	0.569

Pox plot. For a given set of observations $(X_t : t = 1, 2, \ldots, n)$ with a mean $\bar{X}(n)$, a variance $S^2(n)$, all observations are placed into K disjoint subsets, with each subset containing an average of n/K observations. Then the rescaled adjusted range statistic is given by [26]

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)}[max(0, W_1, W_2, \ldots, W_n) \\ -min(0, W_1, W_2, \ldots, W_n)]. \quad (10)$$

where $W_k = X_1 + X_2 + \ldots + X_n - k \cdot \bar{X}(n)$, for $k \geq 1$.

If X_t is self-similar (long-range dependent), then the following equation holds

$$E[\frac{R(n)}{S(n)}] \approx b \cdot n^H, \quad (11)$$

where $n \rightarrow \infty$, H is the Hurst parameter of X_t, and b is a constant. This empirical law is known as the Hurst effect.

Taking the logarithm of both sides of the equation results in the following relation

$$log(E[\frac{R(n)}{S(n)}]) \approx H \cdot log(n) + c, \quad (12)$$

where c is a constant, and $n \rightarrow \infty$. Thus we can plot $log(E[\frac{R(n)}{S(n)}])$ versus $log(n)$ for varying values of n, and obtain the estimate of the Hurst parameter H. This plot should result in a roughly linear graph with a slope equal to the Hurst parameter H. Such a plot is known as a *Pox plot*. So a least-squares linear fit can be used to estimate the Hurst parameter.

Figure 4 shows the Pox plots of the same integer and floating-point benchmarks studied in Figure 3. Following a least-square linear fit, the Hurst parameter is estimated as 0.790, 0.701, 0.652 and 0.593 for *vortex*, *h264ref*, *apsi*, and *cactusADM*, respectively. All estimated Hurst parameters are larger than 0.5, indicating that the inter-access time in these

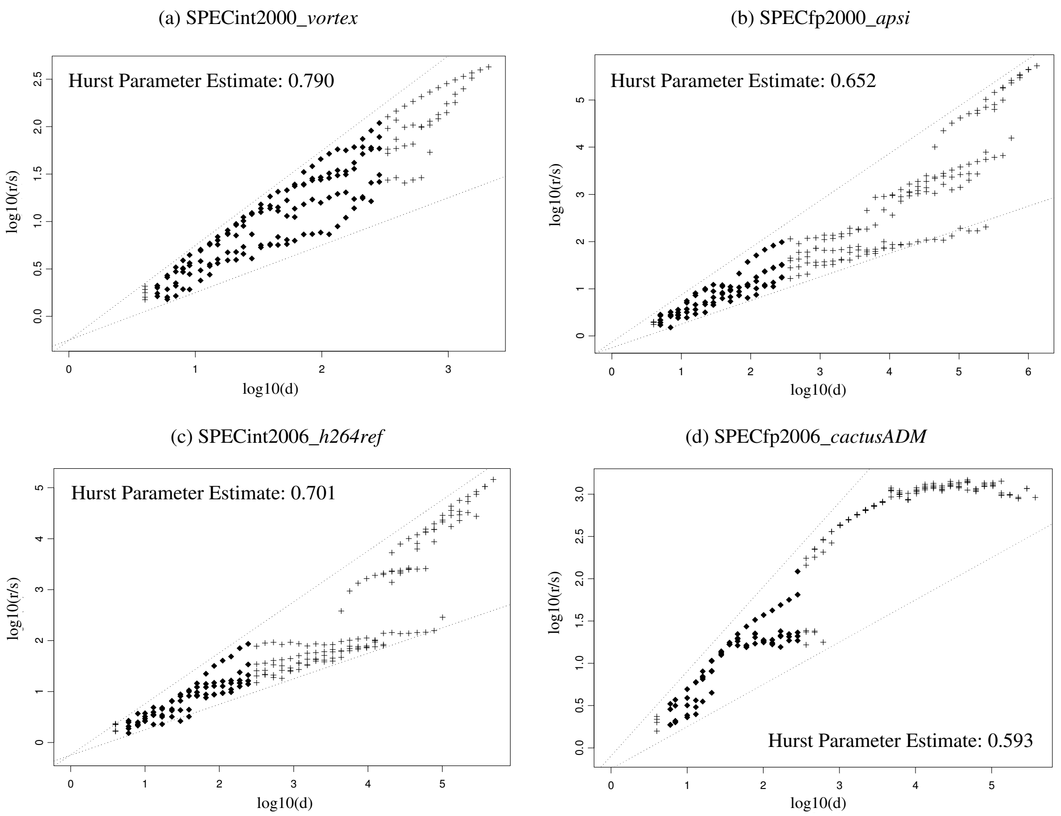


Fig. 4. Pox Plots for the integer benchmarks (e.g., *vortex* for SPEC2000, and *h264ref* for SPEC2006), and the floating point benchmarks (e.g., *apsi* for SPEC2000, and *cactusADM* for SPEC2006). The variable n represents the size of non-overlapping block at which R/S statistic is computed.

workloads are self-similar, which validates the results of Pox plot analysis and increases the confidence of the estimation accuracy.

We use the described R/S analysis technique to estimate the Hurst parameters of all memory traces that are collected in the SPEC 2000 and 2006 benchmark suites. The estimated Hurst parameters, listed in Table IV, are significantly larger than 0.5, confirming again the presence of self-similarity in the studied memory workloads.

The difference between the two measured *H* estimates for some integer memory traces (e.g., *twolf*, *mcf* and *perlbmk*) is large, especially for *perlbmk*. On one hand, this observation cannot be easily explained. Taking the R/S-Analysis estimate of *perlbmk* as an example, the low value of the R/S-Analysis estimate (0.507) perhaps is a result of the existence of some regular memory accesses in *perlbmk*, which causes the correlation coefficients of inter-access times fluctuate regularly in Figure 1(a). On the other hand, the difference verifies the common wisdom that there is no single estimator that can provide a definitive answer [41], although both R/S-Analysis and variance-time plots can qualitatively demonstrate the existence of self-similarity.

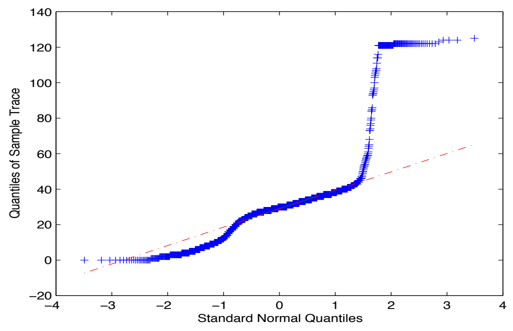
In summary, both the R/S-Analysis and variance-time plots consistently confirm that the inter-access times of all studied workloads exhibit self-similarity. This indicates that the memory I/O accesses in the SPEC2000 and SPEC2006 benchmarks tend to be very bursty, instead of smooth. If a model is required to characterize memory I/O arrivals, certainly a sequence of independently and identically distributed random processes is inappropriate.

VI. Synthesizing Memory Workload Based on Alpha-stable Process

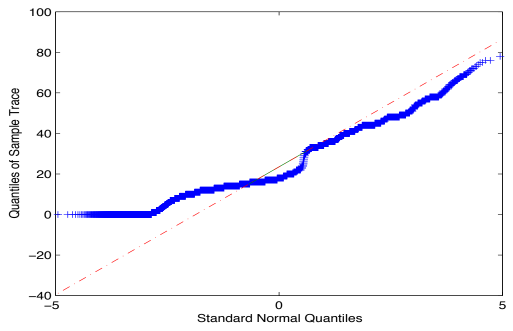
Previous sections have shown both visual and statistical evidence to verify the existence of self-similar nature of memory accesses. In this section we presents a mathematical model that can be used to generate synthetically memory access workloads while preserving the self-similar property.

A. Why use the alpha-stable?

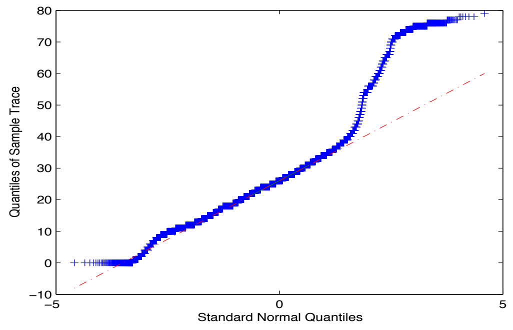
Many techniques have been proposed to synthesize self-similar traffics [10], [11], [23], [29], [30], [31], [35], [36]. For example, two successful methods include Fractional Auto-Regressive Integrated Moving Average (FARIMA) and Fractional Brownian Motion (FBM). FARIMA [23] was first used to generate synthetic Variable Bit Rate (VBR) video traces. However, FARIMA is not intrinsically bursty. The FBM model used by several researchers [10], [11], [36] is easy to construct and can model the self-similarity under the Gaussian condition, but not the non-Gaussian condition.



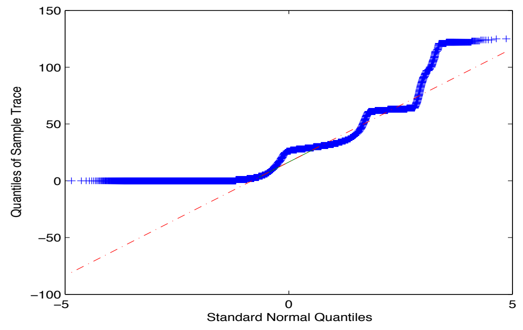
(a) SPECint2000 (*vortex*)



(b) SPECfp2000 (*apsi*)



(c) SPECint2006 (*h264ref*)



(d) SPECfp2006 (*cactusADM*)

Fig. 5. Examine the Gaussian property of *SPECint* (e.g., *vortex* for SPEC2000, *h264ref* for SPEC2006) and *SPECfp* (e.g., *apsi* for SPEC2000, *cactusADM* for SPEC2006) workloads through QQ plots of sample data versus standard normal.

To emulate the burstiness in storage systems, Ref. [32] shows that I/O burstiness can exhibit either Gaussian or non-Gaussian. Thus it is important to identify the Gaussian or non-Gaussian property for a given workload. Otherwise, the real degree of access burstiness cannot be truthfully represented. In particular, when we synthesize the self-similarity memory workloads, we also need to take the Gaussian property into considerations to avoid miss-presenting the burstiness.

For both integer and floating-point benchmarks, we use the normal quantile plots (QQ plots) to measure the Gaussian property [36]. The QQ plots of the *vortex*, *h264ref*, *apsi* and *cactusADM* traces, given in Figure 5, show that *apsi* are Gaussian, but others are non-Gaussian. In Figure 5 (b), all of the scatter points corresponding to an access event given in the traces evidently follow a straight line, indicating that *apsi* is Gaussian. In Figure 5 (a), (c) and (d), all of the scatter points evidently don't fall into a straight line but an increasing curve, indicating that *vortex*, *h264ref*, and *cactusADM* are non-Gaussian. The results above show interestingly that some self-similar memory workloads have the Gaussian property while other memory workloads do not. Therefore, the model used to capture the access burstiness in memory traces should be able to represent both the Gaussian and non-Gaussian properties. The α-stable process can meet this requirement well [32].

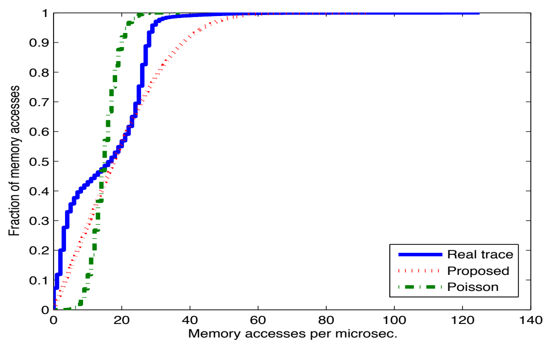
For a set of observations $X = (X_t : t = 1, 2, \ldots, n)$ with a mean μ, a variance $2\sigma^2$, the process X_t is said to be an alpha-stable process if its stable distribution is defined by its characteristic function [42]:

$$E[e^{i\theta X}] = \begin{cases} e^{-\sigma^\alpha|\theta|^\alpha(1-i\beta sign\theta\tan\frac{\pi\alpha}{2})+i\mu\theta}, & \alpha \neq 1 \\ e^{-\sigma|\theta|(1+i\beta sign\theta\ln|\theta|)+i\mu\theta}, & \alpha = 1 \end{cases} \tag{13}$$

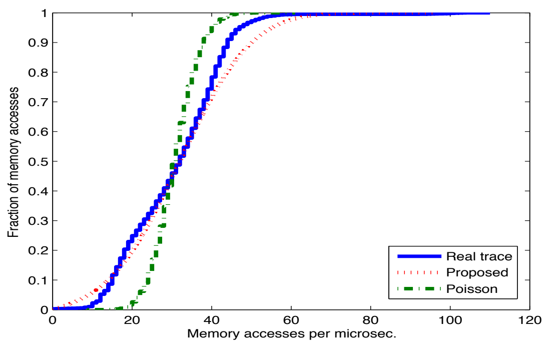
where $sign\theta$ is an indicative function, $0 < \alpha \leq 2$, $\sigma > 0$, $-1 \leq \beta \leq 1$, and $\mu \in R$. The characteristic exponent α measures the degree of burstiness in the memory workload, and β represents the degree of heavy tail in the memory workload.

If $\alpha = 2$, then $\beta \tan \pi = 0$ and β is then meaningless. In this case, it is the characteristic function of a Gaussian stochastic process, i.e., $E[e^{i\theta X}] = exp\{-\sigma^2\theta^2 + i\mu\theta\}$. Otherwise, it is one class of non-Gaussian functions. Therefore, as the value of parameter α changes, the α-stable process is able to flexibly represent a stochastic process under both the Gaussian and non-Gaussian conditions.

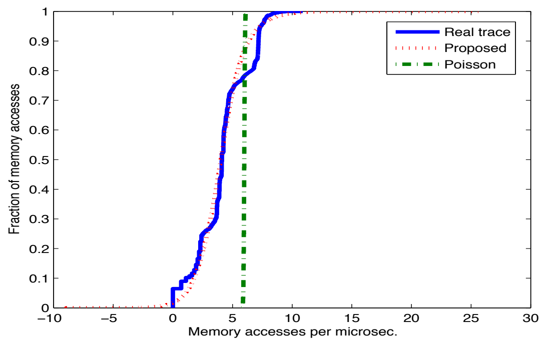
Ref. [32] has developed a model based on the α-stable process to accurately build a synthetic disk I/O workload in storage systems. In this paper, we extend this α-stable model to synthesize memory access workloads. Specifically, the inputs in the α-stable model are the measured properties of the available trace data, including the degree of self-similarity in the memory I/O workload, the degree of memory I/O burstiness, the degree of heavy tail in the memory I/O workload. This model allows us to conveniently turn the memory workload model for different environments.



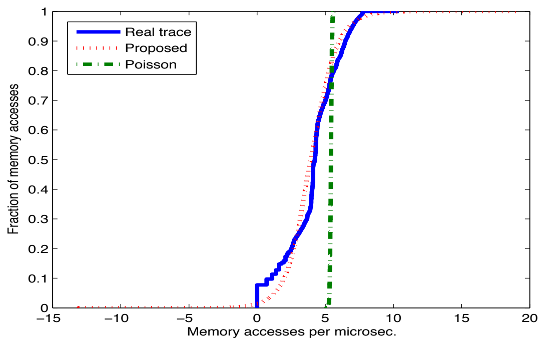
(a) SPECint2000 (*bzip*)



(b) SPECfp2000 (*applu*)



(c) SPECint2006 (*perlbench*)



(d) SPECfp2006 (*gromacs*)

Fig. 6. Comparison of CDFs between synthetic memory trace and real trace for SPECint2000 (e.g., *bzip*), SPECfp2000 (e.g., *applu*), SPECint2006 (e.g., *perlbench*) and SPECfp2006 (e.g., *gromacs*).

B. Experiment results

For each benchmark, we compute the access arrival rate, i.e., the number of accesses per time unit. We place all access arrival rates into a group of stochastic numbers. The maximum-likelihood method is used to estimate the parameters of the α-stable process corresponding to the memory traces needed to be measured. Table V summarizes the estimates of α-stable parameters. In Table V, each row includes the memory trace name and the estimates of four α-stable parameters. Due to the fact that the α-stable distribution degenerates to a Gaussian stochastic process if $\alpha = 2$, with mean value μ and variance $2\sigma^2$, β will be meaningless [32]. Accordingly in Table V, a slash will fill in the place of β if $\alpha = 2$. This measurement results prove the validity of Figure 5(b) again, i.e., the *apsi* workload is Gaussian.

Our model can faithfully emulate the burstiness of memory I/O activities in all studied benchmarks. The *cumulative distribution functions* (CDFs) are used to intuitively compare the synthetic workloads through both our proposed and Poisson methods and trace results. The cumulative distribution functions (CDFs) of the synthetic and real traces for *bzip*, *applu*, *perlbench* and *gromacs*, are illustrated in Figure 6, the X-axis shows the memory access numbers per microsecond (thereinto, the X-axis in Fig. 6(c) and (d) show the logscale), and the Y-axis denotes the percentage of the arrival rates. A point $(x; y)$ in the cumulative distribution curve indicates that y% of access

TABLE V
ESTIMATES OF THE PARAMETER OF α-STABLE DISTRIBUTION BASED ON MAXIMUM-LIKELIHOOD ESTIMATE.

Trace type	Memory benchmarks	α-stable parameter			
		α	β	σ	μ
SPECint 2000	*gcc*	2.00	\	3.67	277
	vpr	0.51	0.32	2.18	1.50
	twolf	2.00	\	0.52	1.22
	perlbmk	0.86	0.23	33.1	300
	vortex	1.38	1.00	32.1	41.2
	gzip	0.70	1.00	0.51	0.36
	mcf	0.76	0.26	1.29	26.1
	parser	2.00	\	14.2	26.0
	gap	0.67	0.38	18.2	156
	bzip	2.00	\	11.5	17.0
SPECfp 2000	*ammp*	1.66	0.52	5.13	25.5
	applu	2.00	\	9.96	32.0
	apsi	2.00	\	8.91	18.0
	galgel	1.26	0.53	7.14	16.6
SPECint 2006	*perlbench*	0.725	1.00	33.3	28.33
	bzip2	1.195	1.00	26.41	36.70
	astar	1.402	0.395	50.27	19.8
	mcf	0.725	0.56	3.81	102.6
	gobmk	0.6756	0.6967	18.42	49.34
	hmmer	2.00	\	31.97	51
	sjeng	1.07	1.00	138.9	-110.6
	xalancbmk	2.00	\	10.58	73.2
	h264ref	0.7775	0.617	15.49	55.55
	omnetpp	0.62	0.49	10.84	20.17
SPECfp 2006	*cactusADM*	0.898	1.00	15.35	0.648
	gromacs	0.89	1.00	42.35	22.1
	namd	0.49	0.56	20.6	39.6
	povray	2.00	\	15804.6	9925

rates are less than or equal to an arrival rate of x.

The memory workload synthesized by the α-stable model very closely matches the real trace data, especially for *perlbench* and *gromacs*, as shown in Figure 6. It is evident that it is difficult for the Poisson method to accurately capture the memory I/O burstiness which can be precisely characterized by the α-stable method.

A quantitative approach to evaluate the improvement is to analyze the error. A *trimmed mean* [9] is widely used to measure the central tendency and it is less sensitive to outliers that are far away from the mean. A trimmed mean is calculated by discarding a certain number of highest and lowest outliers and then computing the average of the remaining measurements. Since statistically a trimmed mean is usually more resilient and robust than a simple average mean, we use the trimmed mean to evaluate the matching degrees between each real workload and its corresponding synthetic workload. The trimmed means of errors and comparison results are summarized in Table VI.

As can be seen from Table VI, for almost all of traces studied in this paper, the trimmed mean of error between the real workload and the α-stable synthetic workload is minimum, with the exception in which the trimmed mean of error between the *hmmer* trace and the α-stable synthetic workload is 41.9, with the increase of 15 percent of 36.4, the trimmed mean of error between the *hmmer* trace and the Poisson synthetic workload. Nevertheless, comparing with the matching degree of the Poisson synthetic workload, the matching degree of the α-stable synthetic workload for the *hmmer* workload is still reasonably good.

TABLE VI
THE TRIMMED MEANS OF ERRORS FOR THE SPEC2000 AND SPEC2006 BENCHMARKS.

Type	Benchmarks	Poisson	Proposed	Improvement (%)
SPECint 2000	*gcc*	61.22	57.28	6
	vpr	6.79	6.43	5
	twolf	3.04	0.65	79
	perlbmk	8.20	5.04	38
	vortex	98.51	95.86	3
	gzip	4.81	2.18	54
	mcf	4.74	4.63	3
	parser	21.67	17.26	20
	gap	51.84	46.94	9
	bzip	16.35	14.10	14
SPECfp 2000	*ammp*	14.64	7.67	48
	applu	19.38	12.75	34
	apsi	18.06	15.32	15
	galgel	17.65	11.93	32
SPECint 2006	*perlbench*	365.28	158.99	56.5
	bzip2	55.26	32.57	41
	astar	30.2	19.7	34.8
	mcf	87.13	72.59	16.7
	gobmk	160.81	64.75	59.7
	hmmer	36.4	41.9	−15
	sjeng	501.1	276.7	44.8
	xalancbmk	61.23	29.52	51.8
	h264ref	115.5	44.2	61.7
	omnetpp	57.26	40.81	28.7
SPECfp 2006	*cactusADM*	64.11	35.72	44.3
	gromacs	183.01	115.01	37.2
	namd	392.42	158.02	60
	povray	149.47	143.64	4

As shown in Table VI, for *bzip*, *applu*, *perlbench* and *gromacs*, the trimmed means of errors between the real trace and the synthesized workload through the Poisson method are 16.35, 19.38, 365.28, and 183.01, respectively, and the trimmed means of errors between the real trace and the synthesized workload through the α-stable model with these parameter values in Table V are 14.10, 12.75, 158.99, and 115.01, respectively. Accordingly, our proposed model can reduce the trimmed mean of error of the Poisson models by 14%, 34%, 56.5% and 37.2%, respectively. So, the synthetic workloads generated by the α-stable method are more accurate than the synthetic workloads synthesized by the Poisson method.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we studied the self-similarity phenomena of the memory access in the widely used SPEC 2000 and 2006 benchmark suites. We examine the auto-correlation functions of inter-access times (arrival intervals) for all of memory access traces collected in SPEC2000 and SPEC2006. Results show that there are evident correlations between memory accesses in both the integer and floating-point benchmarks. Therefore, a sequence of independent and identically distributed random variables is inappropriate to characterize and model memory I/O accesses. This motivates us to further study the self-similarity in memory workloads, which can provide useful insight into analysis of memory workloads, and design of memory benchmarks and synthetic workloads.

We have shown visual evidence that the memory accesses are consistent with self-similar like behavior in small time scales. We have used rigorous statistical techniques, including variance-time plot and R/S analysis (Pox plot), to estimate the Hurst parameter of memory access traces. In our experiments, all estimated Hurst parameters are significantly larger than 0.5, indicating that self-similarity seems to be a general property of memory access behaviors. As a result, when characterizing the memory I/O workloads or designing synthetic benchmark to evaluate a memory system, the self-similarity, an intrinsic nature in memory I/Os, should be taken into consideration to correctly preserve or emulate the I/O burstiness.

In addition, based on the α-stable process, we implement a memory access series generator in which the inputs are the measured properties of the available trace data. Experimental results show that this model can faithfully capture the complex I/O arrival characteristics of memory workloads, particularly the heavy-tail characteristics under both Gaussian and non-Gaussian workloads.

REFERENCES

[1] M. Inc.. Micron 512mb: Ddr2 sdram data sheet. http://www.micron.com.
[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu and et al. The m5 simulator: Modeling networked systems. IEEE Micro, 26(4):52-60, 2006.
[3] D. Wang, B. Ganesh, N. Tuaycharoen and et al. Dramsim: a memory system simulator. SIGARCH Computer Architecture News, 33(4):100-107, 2005.
[4] J. Henning. SPEC CPU2000: Measuring CPU Performance in the New Millennium. IEEE Computer, 33(7), July 2000.

[5] J. Henning. SPEC CPU2006 Benchmark Descriptions. ACM SIGARCH Computer Architecture News, 34(4):1-17, September 2006.

[6] A. Jaleel. Memory Characterization of Workloads Using Instrumentation-Driven Simulation–A Pin-based Memory Characterization of the SPEC CPU2000 and SPEC CPU2006 Benchmark Suites. VSSAD Technical Report 2007.

[7] S. Sair and M. Charney. Memory Behavior of the SPEC CPU2000 Benchmark Suite. IBM Thomas J. Watson Research Center Technical Report RC-21852, October 2000.

[8] Z. Xu, S. Sohoni, R. Min and Y. Hu. An Analysis of the Cache Performance of Multimedia Applications. IEEE Transactions on Computers, 53(1):20-38, January 2004.

[9] Z. J. Liu. Computational Science Technique and Matlab. Science Press. Beijing, China, 2001.

[10] Norros. On the use of Fractional Brownian Motion in the theory of connectionless networks. IEEE Journal on Selected Areas in Communications (JSAC), 15: 200-208, 1997.

[11] Z. Kurmas, K. Keeton and K. Mackenzie. Synthesizing Representative I/O Workloads Using Iterative Distillation. In Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS). Orlando, Florida, 2003.

[12] D. Ye, J. Ray and D. Kaeli. Characterization of File I/O Activity for SPEC CPU2006. ACM SIGARCH Computer Architecture News, 35(1):112-117, March 2007.

[13] SPEC CPU2000 published results. http://www.spec.org/cpu2000/results.

[14] SPEC CPU2006 published results. http://www.spec.org/cpu2006/results.

[15] L. A. Barroso, K. Gharachorloo and E. Bugnion. Memory system characterization of commercial workloads. In Proceedings of the 25th International Symposium on Computer Architecture (ISCA). Barcelona, Spain, June 1998.

[16] D. Lee, P. Crowley, J. Baer, T. Anderson, and et al. Execution Characteristics of Desktop Applications on Windows NT. In Proceedings of the 25th International Symposium Computer Architecture (ISCA). Barcelona, Spain, June 1998.

[17] Y. Chen, W. Li, J. Lin and et al. Memory Characterization of Emerging Recognition-Mining-Synthesis Workloads for Multi-Core Processors. In Proceedings of the Workshop for Computer Architecture Evaluation of Commerical Workloads (CAECW'08).

[18] J. Lin, Y. Chen, W. Li and et al. Memory Characterization of SPEC CPU2006 Benchmark Suite. In Proceedings of the Workshop for Computer Architecture Evaluation of Commerical Workloads (CAECW'08).

[19] H. Liu, R. Li, Q. Gao and et al. Characterizing Memory Behavior of XML Data Querying on CMP. In Proceedings of the Workshop for Computer Architecture Evaluation of Commerical Workloads (CAECW'08).

[20] M. Charney and T. Puzak. Prefetching and Memory System Behavior of the SPEC95 Benchmark Suite. IBM J. Research and Development, 41(3):265-286, May 1997.

[21] J. Gee, M. Hill, A. J. Smith. Cache Performance of the SPEC Benchmark Suite. UC Berkeley, Technical Report: CSD-91-648, 1991.

[22] W. Korn, M. S. Chang. SPEC CPU2006 sensitivity to memory page sizes. In ACM SIGARCH newsletter, Computer Architecture News, Volume 35, No. 1, March 2007.

[23] M. W. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In Proceedings of the ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications. London, UK, September 1994.

[24] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger. Long-range dependence in variable-bit-rate video traffic. IEEE Transactions on Communications, 43:1566-1579, Mar. 1995.

[25] Tao Li. Using A Multiscale Approach to Characterize Workload Dynamics. In Proceedings of the Workshop on Modeling, Benchmarking and Simulation (MoBS). Madison, Wisconsin, June 2005.

[26] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Transactions on Networking, 2(2):1-15, Feb. 1994.

[27] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. IEEE/ACM Transactions on Networking, 3(3):226-244, 1995.

[28] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. IEEE/ACM Transactions on Networking, 5(1):71-86, 1997.

[29] S. Gribble, G. Manku, and E. Brewer. Self-similarity in high-level file systems: Measurement and applications. In Proceedings of the ACM SIGMETRICS'98. Madison, WI. June 1998.

[30] M. Gomez and V. Santonja. Self-Similarity in I/O Workload: Analysis and Modeling. In Proceedings of the 1st IEEE International Workshop on Workload Characterization (WWC'98). Dallas, Texas, November 1998.

[31] M. Gomez and V. Santonja. Analysis of Self-similarity in I/O Workload Using Structural Modeling. In Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS). College Park, Maryland, October 1999.

[32] Q. Zou, D. Feng, Y. Zhu and et al. A Novel and Generic Model for Synthesizing Disk I/O Traffic Based on The Alpha-stable Process. In Proceedings of the 16th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). Baltimore, Maryland, September 2008.

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

[34] B. Hong and T. Madhyastha. The Relevance of Long-Range Dependence in Disk Traffic and Implications for Trace Synthesis. In Proceedings of the IEEE Conference on Mass Storage Systems and Technologies (MSST). Monterey, CA, April 2005.

[35] M. Wang and T. Madhyastha and et al. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In Proceedings of the 18th International Conference on Data Engineering (ICDE). San Jose, CA, February 2002.

[36] C. Stathis and B. Maglaris. Modelling the self-similar behaviour of network traffic. Computer Networks, 34:37-47, 2000.

[37] A. Riska and E. Riedel. Long-Range Dependence at the Disk Drive Level. In Proceedings of the Third International Conference on the Quantitative Evaluation of Systems (QEST). University of California, Riverside, CA, September 2006.

[38] A. Riska and E. Riedel. Disk Drive Level Workload Characterization. In Proceedings of the 2006 USENIX Annual Technical Conference. Boston, MA, June 2006.

[39] J. Zhang, A. Sivasubramaniam, H. Franke and et al. Synthesizing Representative I/O Workloads for TPC-H. In Proceedings of the Tenth International Symposium on High Performance Computer Architecture (HPCA-10). Madrid, Spain, February 2004.

[40] S. Kavalanekar, B. Worthington, Q. Zhang and V. Sharda. Characterization of Storage Workload Traces from Production Windows Servers. In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC). Seattle, WA, September 2008.

[41] T. Karagiannis, M. Faloutsos and R. Riedi. Long-Range Dependence: Now you see it, now you don't! In Proceedings of the GLOBECOM. Taipei, Taiwan, November 2002.

[42] G. Samorodnitsky and M. Taqqu. Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance. New York, 1994.

[43] K. Ganesan, J. Jo, and L. K. John. Synthesizing Memory-Level Parallelism Aware Miniature Clones for SPEC CPU2006 and ImplantBench Workloads. In Proceedings of the 2010 International Symposium on Performance Analysis of Systems and Software (ISPASS). White Plains, NY, March 2010.

[44] L. Eeckhout, R. H. Bell Jr., B. Stougie and et al. Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies. In Proceedings of the 31st International Symposium on Computer Architecture (ISCA), 2004.

[45] A. Joshi, L. Eeckhout, R. H. Bell Jr., and Lizy K. John. Performance Cloning: A Technique for Disseminating Proprietary Applications as Benchmarks. In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC'06). San Jose, California, October 2006.

[46] R. H. Bell Jr., R. R. Bhatia, L. K. John and et al. Automatic Testcase Synthesis and Performance Model Validation for High Performance PowerPC Processors. In Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Austin, Texas, March 2006.

[47] D. C. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report CS-TR-97-1342. University of Wisconsin, Madison, June 1997.

[48] Y. Kim, M. Papamichael, O. Mutlu and et al. Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior. In Proceedings of the MICRO-43, Atlanta, GA, Dec. 2010.

[49] Y. Kim, D. Han, O. Mutlu, and et al. ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers. In Proceedings of the HPCA-16, Bangalore, India, Jan. 2010.