Using A Multiscale Approach to Characterize Workload Dynamics

Tao Li

Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida, 32611
E-mail: taoli@ece.ufl.edu

Abstract

Workload characterization has been essential ingredients for the design of computer systems. As computer architecture becomes adaptive and reconfigurable, its efficiency increasingly depends on the dynamic behavior of workloads.

Program execution manifests wildly varied changes at run time. As software becomes large and sophisticated, such variation can span a wide range of time scales. As current workload characterization methodologies have paid less attention to the scaling behavior of program, we are often at a loss as to how to interpret and forecast the changing of workload dynamics over time. To address this issue, we advocate a multiscale workload characterization methodology that can capture the dynamic nature of workloads across different time scales.

We apply this technique to study the scaling properties of the SPEC2000 integer benchmarks. The on-line program scaling estimator proposed in this paper allows one to capture both short-term and long-term execution characteristics of large program in-flight.

1. Introduction

Workload characterization is at the foundation of computer architecture design and optimization. By understanding workload behavior, both hardware and systems can be tuned to better suit the needs of the applications. There have been many studies [1, 2, 3, 4] that evaluated the performance of workloads and attempted to characterize it with various architectural features. A majority of these studies use aggregate metrics (e.g., average IPC, total cache misses) to represent the specific architectural characteristics for the entire program execution.

As computer architecture becomes adaptive and reconfigurable, its efficiency increasingly depends on the workload dynamic behavior. For example, reconfigurable microarchitecture periodically profiles program execution characteristics and uses them as the feedbacks to adapt its resource (e.g. instruction queues, branch predictors and caches) to meet the need of applications. Previous studies [5, 6] revealed that program execution manifests wildly varied changes at run time. As software becomes large and sophisticated, such variation can span a wide range of time scales. Current workload characterization methodologies, however, have paid less attention to reveal how the workload characteristics change across different time scales. As a result, we are often at a loss as to how to interpret and forecast the changes in workload behavior over time. Such limitations can become serious with the increasing workload execution time.

A better understanding of program multiscale behavior can benefit computer architecture design and performance evaluation in many aspects. For example, reconfigurable architecture can exploit program scaling properties that capture both small time period and large time period characteristics to fine-tune its multi-configuration hardware units across various time scales. In another scenario, for a program that shows the well-defined scaling properties (e.g., *self-similarity*), its large time scale behavior can be accurately projected by using its small time scale behavior.

The goal of this paper is to improve our current understanding of the changing nature of the workload dynamics over time. Differing from the previously proposed methodologies [1, 5] (e.g., aggregate measurement, phase characterization), this paper introduces the using of multiscale models and metrics to describe program dynamics. The discrete wavelet transforms are used to discover program behavior at multiresolution levels. The proposed multiscale workload characterization methodology allows one to "zoom in" and "zoom out" over a wide range of program execution periods to capture the workload dynamic behavior. This paper proposes an on-line program scaling estimator that allows one to capture both short-term and long-term execution characteristics of large software in-flight without storing huge traces. The proposed estimator can be integrated with current performance monitoring systems.

The remainder of this paper is organized as follows. Section 2 introduces the basic scaling models and metrics. Section 3 presents a wavelet-based scaling analysis method. Section 4 describes the experimental setup. Observations from the program scaling analysis are presented and discussed in Section 5. We first look at program second-order scaling statistics and then study its high-order scaling properties. Section 6 presents an on-line program scaling estimator that allows the performance monitoring systems to discover the changes in workload behavior on the long-run programs. Section 7 discusses the related work. We conclude this paper and point out some directions of our future work in the Section 8.

2. Background: Scaling Models

The notion of scaling can be loosely defined as the absence of special characteristic time or space scales. As a result, the whole and its parts can not be statistically distinguished from each other. As a critical phenomenon in the nature, scaling manifests itself in many real world objects (e.g. complex forms and patterns such as clouds, mountains and coastlines) [7].

Since our main focus in this paper is to characterize the scaling of workload behavior in the time domain, we present vari-

ous scaling models in the context of time series in the following subsections.

2.1 Self-similarity

The purest formal framework for scaling is that of exact self-similarity. Self-similarity means that the sample paths of the process $X(t)$ and those of a rescaled version $c^H X(t/c)$, obtained by simultaneously dilating the time axis by a factor $c > 0$, and the amplitude axis by a factor c^H, can not be statistically distinguished from each other (Figure 1). The Hurst parameter H, is used to measure the degree of self-similarity. The closer H is to 1, the stronger self-similarity the process exhibits.

Figure 1 Self-similarity: a dilated portion of the sample path of a process can not be statistically distinguished from the whole.

Exact self-similarity fulfils the intuition of scaling in a perfect way. However, the model is overly rigid. The remainder of this section details more flexible models that enable some deviations from the exact self-similarity.

2.2 Long-Range Dependence

For a time series $x(t)$ contains the data of interest, the basic features of this process are its mean $u_x = E[x]$, variance $\sigma_x^2 = E[(x-u_x)^2]$, and correlation function $r_x(k) = E[(x(t+k)-u_x)(x(t)-u_x)]$. The process $x(t)$ displays *long-range dependence* (LRD) if its correlation function $r_x(k)$ behaves like a power-law of the time lag k, i.e.

$$r_x(k) \sim c_r |k|^{2H-2} \quad \text{as} \quad |k| \to \infty$$

where c_r is a positive constant and the Hurst parameter $1/2 < H < 1$. In such a case, the correlations decay so slowly that they sum to infinity. When the Hurst parameter $H = 1/2$, the process manifests *short-range dependence* (SRD). The SRD is characterized by quickly decaying correlations.

3. Scaling Analysis Techniques

In this section, we introduce a wavelet-based methodology as the key tool for scaling discovery. The wavelet analysis [8] acts as a mathematical "microscope" which allows one to zoom in on fine structures of a signal or, alternatively, to reveal large scale structures by zooming out.

3.1 Discrete Wavelet Transform (DWT)

Consider a series $X_{0,k}, k = 0,1,2,...$, at the finest level of time scale resolution 2^{-n}. This time series might represent the measured workload dynamic characteristics (e.g., the number of instructions retired per cycle, cache misses per thousand cycles etc.) during each sampling interval. We can coarsen this event series by averaging (with a slightly unusual normalization factor) over non-overlapping blocks of size two

$$X_{1,k} = \frac{1}{\sqrt{2}}(X_{0,2k} + X_{0,2k+1}) \quad (1)$$

and generates a new time series X_1, which represents a coarser granularity picture of the original series X_0. The difference between the two, known as *details*, is

$$d_{1,k} = \frac{1}{\sqrt{2}}(X_{0,2k} - X_{0,2k+1}) \quad (2).$$

Note that the original time series X_0 can be reconstructed from its coarser representation X_1 by simply adding in the details d_1; i.e., $X_0 = 2^{-1/2}(X_1 + d_1)$. We can repeat this process (i.e., write X_1 as the sum of yet a coarser version X_2 of X_0 and the details d_2, and iterate) for as many scale as are present in the original time series

$$X_0 = 2^{-n/2} X_n + 2^{-n/2} d_n + ... + 2^{-1/2} d_1 .$$

We refer to the collection of details $d_{j,k}$ as the *discrete wavelet coefficients*. Note that for the purpose of illustration simplicity, we choose the Harr wavelet as the mother function here. The calculations of all $d_{j,k}$, which can be done iteratively using the equations 1 and 2, make up the so called *discrete wavelet transform* (DWT).

As wavelet transform divides data into a low-pass approximation and a high-pass detail at any level of resolution and analyzes each component with a resolution matched to its scale, the coefficients of wavelet decomposition can be directly used to study the scale dependent properties of the data. In the following subsections, we describe a set of wavelet-based scaling analysis techniques.

3.2 Energy Function and Logscale Diagram

Given a time series $X_{0,k}, k = 0,1,2,...$, and its discrete wavelet coefficients $d_X(j,.)$, the average energy at resolution level 2^j is then defined as:

$$E_j = \frac{1}{n_j} \sum_{k=1}^{n_j} |d_X(j,k)|^2 \quad (3),$$

where n_j is the number of wavelet coefficients. The *log-scale diagram* (LD) is the plot of E_j as a function of resolution level 2^j (together with the confidence intervals) on a $\log_2 - \log_2$ scale; i.e., $y_j = \log_2(E_j)$ as a function of scale j. Asymptotically, the

energy function is expected to be linear with time scale j for self-similar processes, i.e.

$$y_j = \alpha j + \log_2(E_0), \text{ where } \alpha = 2H - 1$$

The slope of the LD plot provides an estimate of the Hurst parameter H. If $H = 0.5$, $\alpha = 0$ (a flat slope points to a short-range dependent process) while $H > 0.5$ yields $\alpha > 0$ (a positive slope indicates a long-range dependent process).

Therefore, the LD plot allows the detection of scaling through observation of strict alignment (linear trend) of the confidence interval of the y_j within some octave range. If a strict alignment is detected, the scaling parameters can be estimated. The LD is a second-order statistics (sample variance) of the wavelet coefficients, hence it does not capture the higher-order properties of the processes.

3.3 Partition Function and Multiscale Diagram

Multifractal analysis compares the scaling of different wavelet moments q to estimate the local regularity in processes. Its main tool is the wavelet partition function $S(q, j)$, a generalization of the wavelet energy function (3), defined as

$$S(q, j) = \sum_k \left| 2^{-j/2} d_X(j,k) \right|^q \quad (4).$$

By computing the partition function, one can characterize the statistics of the local behavior of workload dynamics. This is because a wavelet is an oscillating function and the values of the wavelet coefficients are proportional to the size of the irregularity. The partition function raises the wavelet coefficients to an exponent and magnifies the importance of the largest coefficients that arise due to a local irregularity. On the contrary, it reduces the importance of small coefficients. Therefore, the smaller the wavelet coefficients for the scale j, the smaller the value of the partition function for $S(q, j)$ for large q. This permits to study the importance of the local irregularities at time scale j.

The *multi-scale diagram* (MD) is the plot of $S(q, j)$ as a function of resolution level 2^j on a $\log_2 - \log_2$ scale; i.e., $\log_2(S(q, j))$ as a function of scale j, for a range of values of the moments q.

A relatively smooth process (second-order process) that shows no particularly large local irregularity will have values of the $\log_2(S(2, j))$ larger than $\log_2(S(q, j))$ (for $q > 2$). On the opposite, an irregular process (high-order process) will tend to have larger values of $\log_2(S(q, j))$ for values of $q > 2$.

4. Experimental Setup

In this paper, we demonstrate the using of wavelet-based multiscaling analysis techniques to characterize the changing nature of workload dynamics over time. This section describes the experimental methodology, including the simulated machine architecture, the studied workloads and the collected benchmark traces. An on-line estimator without requiring the off-line trace analysis will be described in Section 6.

4.1 Machine Configuration

The statistics of workload dynamics are measured on the SimpleScalar 3.0 [9] *sim-outorder* simulator. The simulated microarchitecture configuration in this study is an 8-way superscalar model. The details of the machine configurations are summarized in Table 1.

Table 1 Processor configurations

Parameter	Configuration
Processor Width	8
ITLB	128 entries, 4-way, 200 cycle miss
Branch Prediction	combined 8K tables, 10 cycle misprediction, 2 predictions/cycle
BTB	2K entries, 4-way
Return Address Stack	32 entries
L1 Instruction Cache	32K, 2-way, 32 Byte/line, 2 ports, 4 MSHR, 1 cycle access
RUU Size	128 entries
Load/ Store Queue	64 entries
Store Buffer	16 entries
Integer ALU	4 I-ALU, 2 I-MUL/DIV
FP ALU	2 FP-ALU, 1FP-MUL/DIV
DTLB	256 entries, 4-way, 200 cycle miss
L1 Data Cache	64KB, 4-way, 64 Byte/line, 2 ports, 8 MSHR, 1 cycle access
L1 Cache	unified 1MB, 4-way, 128 Byte/line, 12 cycle access
Memory Access	100 cycles

4.2 Program Traces

We use the twelve SPEC2000 integer programs as the experimented workloads. To analyze the workload dynamics at large time scales, we use the reference input data sets and simulate 9 out of 12 workloads until completion. Benchmarks *mcf*, *parser* and *twolf* take extremely long time to complete. For these three benchmarks, we stop the simulation after the benchmark execution cycles reach the time scales that are comparable to those of other completely simulated benchmarks.

During the simulations, the statistics of the architectural characteristics of interest are collected and recorded in traces. The instructions-per-cycle (IPC) metric, which indicates how efficiently a microprocessor performs its functions, is mainly used in this study. Each collected trace contains a sequence representing the program IPC rate measured on every ten-thousand elapsed cycles. The program traces statistics are summarized in Table 2. One can see that these collected traces allow the study of the changing nature of workload dynamics over a period of hundreds of billion cycles.

5. Characterizing Program Scaling Behavior

This section provides a detailed characterization of the SPEC2000 integer benchmarks scaling behavior using the methodology described in Section 3.

Table 2 Benchmark traces description

Benchmark	Input	Duration (Cycles)
mcf	/ref/inp.in	570,689,841,862
gcc	/ref/166.i	33,578,085,795
crafty	/ref/crafty.in	337,250,101,460
gzip	/ref/input.graphic	52,867,265,321
bzip2	/ref/input.source	70,644,828,028
eon	/ref/chair.cook.ppm	93,485,005,275
gap	/ref/ref.in	355,758,277,267
parser	/ref/ref.in	247,035,615,983
perlbmk	/ref/splitmail.pl	49,931,474,883
twolf	/ref/ref	274,987,890,000
vortex	/ref/lendian1.raw	93,677,830,341
vpr	/ref/net.in	122,267,820,515

5.1 Visualization of the Traces

We begin our analysis of multiscaling characteristics with a demonstration of the time-varying behavior of two programs from SPEC2000, *crafty* and *gzip*. Figure 2 visualizes the traces across 5 orders of magnitude of time scales. Each plot within the figure represents the IPC rate, enumerated as the number of instructions retired per time unit. Successive plots are refinements of the previous plots; the top plot in each column has a time unit of 4.096×10^7 cycles, the second of 5.12×10^6 cycles, and so one. The left column illustrates the IPC dynamics of benchmark *gzip*, and the right column illustrates the IPC dynamics of benchmark *crafty*.

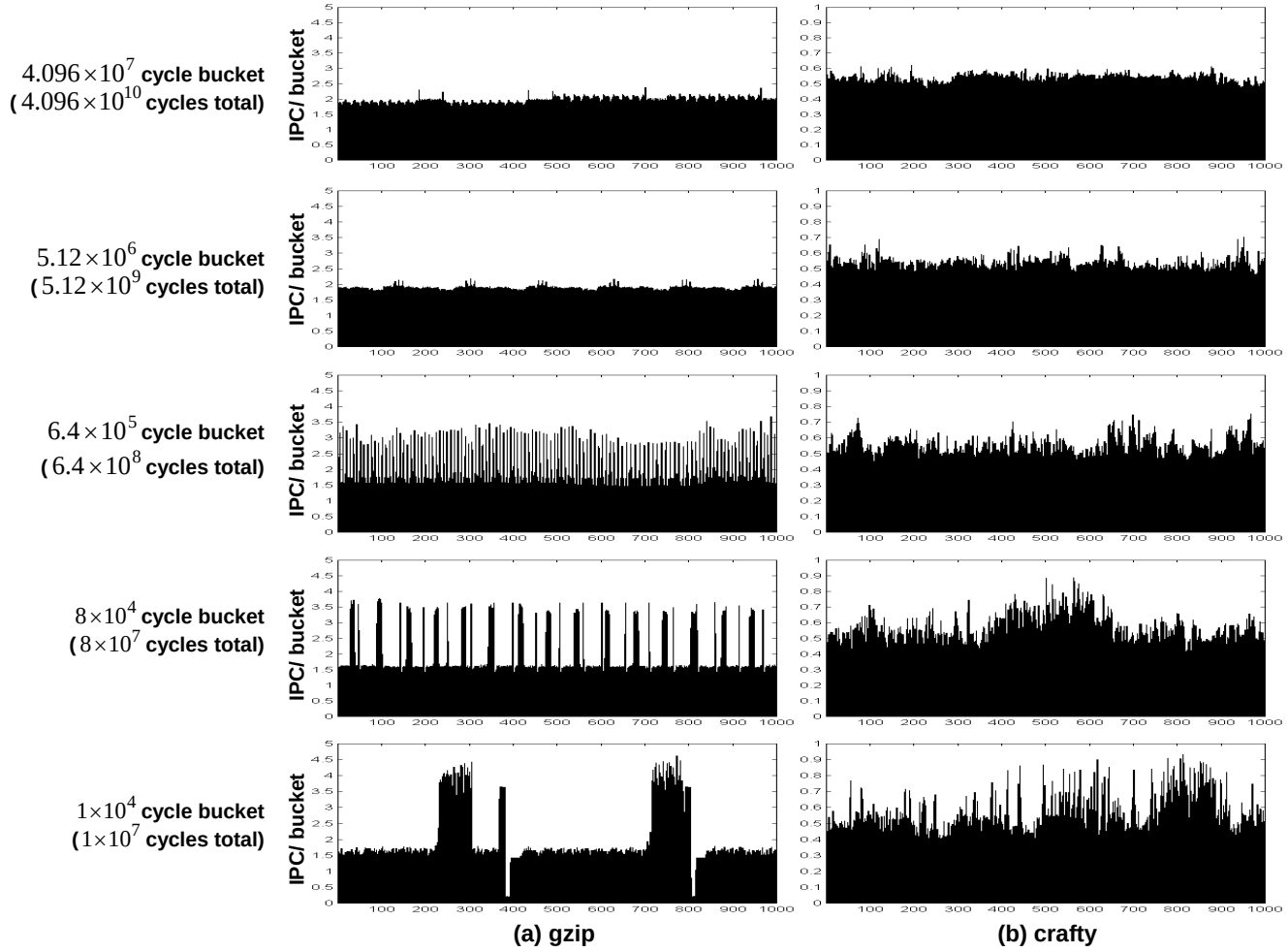

Figure 2 Visualization of the IPC dynamics: programs IPC per unit time (or "bucket") are plotted for 5 increasingly refined time units, varying by a total factor of 4,096. Column (a) shows benchmark *gzip*, and column (b) shows benchmark *crafty*. The x-axis of all graphs is the number of buckets; the y-axis shows the IPC during that bucket.

One can see that the IPC dynamics of both programs change across different time scales. For the *gzip* trace, the pronounced phases occur at fine time granularities (1×10^4 and 8×10^4 cycle buckets). For coarse granularities (6.4×10^5 cycle bucket), the trace manifests very bursty and unpredictable behavior. On the large time scales (5.12×10^6 and

4.096×10^7 cycle buckets**),** the trace becomes extremely smooth, although a small number of spikes can be seen to interrupt this smoothness. Therefore, the workload dynamics of *gzip* changes over its execution time. Characteristics observed at small time scales will not represent those at large time scales.

The IPC dynamics on benchmark *crafty*, on the other hand, appears bursty at all time scales, although the burstiness appears to smooth out at coarser time granularities. A closer investigation reveals that the trace exhibits scale invariance, i.e. the workload dynamics appear to be similar at all time scales. Viewing the collected traces as time series, we then use the multiresolution analysis techniques described in the Section 3 to investigate the program scaling behavior.

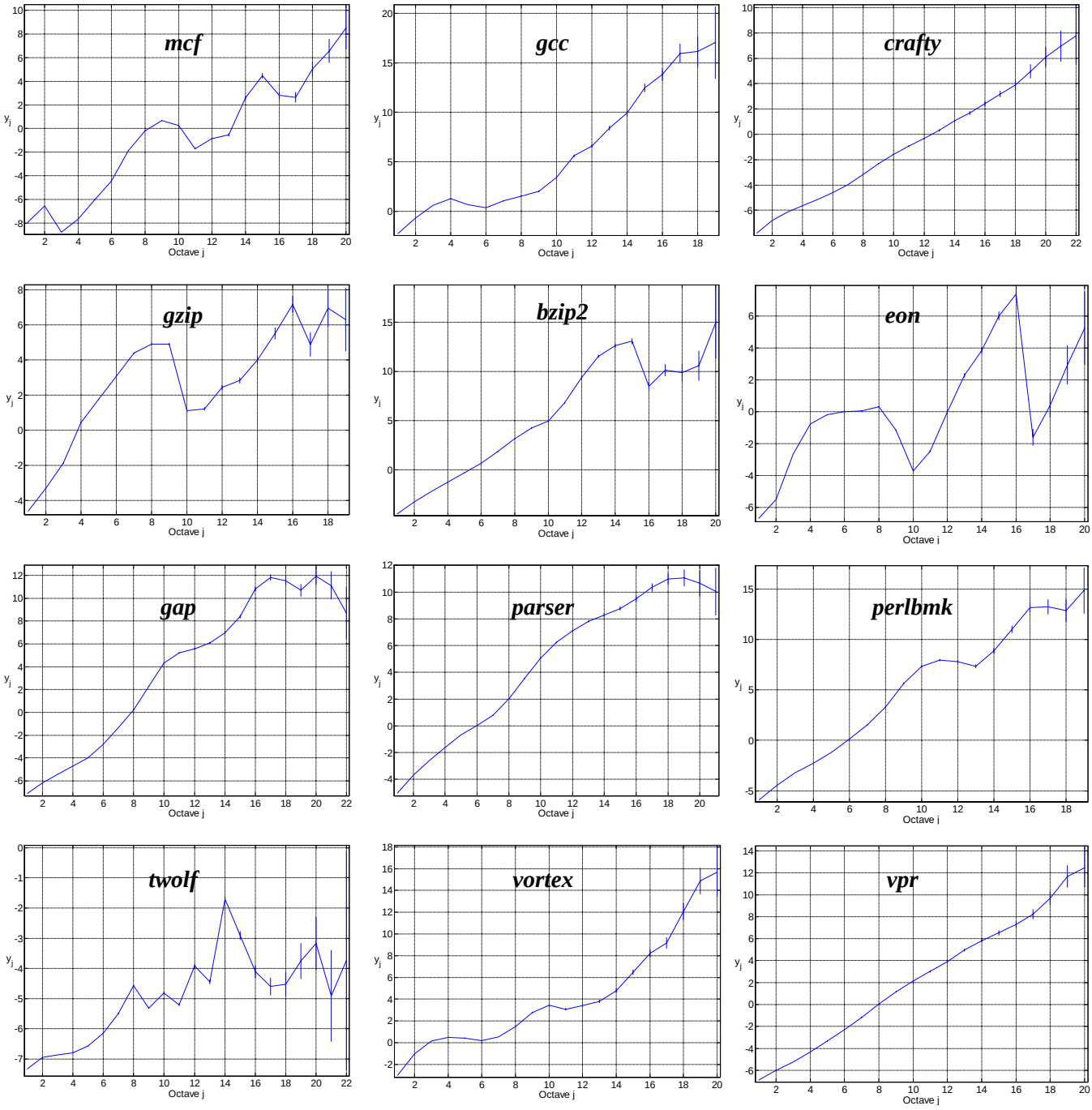

Figure 3 the LD plots of SPEC2000 integer benchmark traces

5.2 Second-order Scaling Analysis

In this section, we study the second-order scaling properties of the experimented workloads. The second-order scaling properties over the total length of the traces are analyzed.

Figure 3 presents the LD plots for the SPEC 2000 integer benchmarks. The x-axis shows time scales, varied from fine to coarse resolutions. The vertical bars at each octave give the 95% confidence intervals for the y_j. Confidence intervals about the y_j increase monotonically with j as one moves to larger and larger scales, as seen in each of the diagram in Figure 3.

A property of the LD concerns the theoretical possibility to distinguish the type of process present within some octave range based on the value of the slope of the logscale diagram. As described in Section 3, asymptotically self-similar (or, equivalently, long-range dependence) will result in a linear relation between y_j and the scale j. If the trace is exactly self-similar, a plot of y_j vs. j will show a linear relationship for all scales.

Looking at Figure 3, one can see that the SPEC2000 integer programs show heterogeneous scaling characteristics. For benchmarks *crafty*, *vpr*, and *parser*, there is a clear evidence of strict scaling across all octaves, i.e. the traces of these benchmarks exhibit self-similarity. More frequently, one sees the LD plots with scaling over a limited range of time scales, two or multiple different scaling regions, a flat slope regions and dips. A flat slope region in the LD plots indicates that the program dynamics statistics look like the uncorrelated "white noise" across the spanned time scales. A dip in the LD plots implies a periodic component. This is because the highly regular or periodic structures (e.g. loops) in time series reveal themselves in terms of small wavelet coefficients and subsequently the low values of the energy function.

For small time scales between octaves 1 and 6, the studied workloads seem to exhibit various characteristics. For example, periodicity occurs on benchmarks *mcf* and *gcc* by showing the dips in their LD plots. Octaves in [2,4] for *twolf* and octaves in [3,6] for *vortex* show evidence of SRD.

For time scales between octaves 6 and 16, the linear relation between y_j and scale j are observed on the majority of the studied workloads, implying that the corresponding traces are consistent with self-similarity or long-range dependence. Typical scaling regions extend several orders of magnitudes, with upper cutoffs that vary from one benchmark to another. *Mcf*, *gzip*, *eon* and *twolf* lack LRD characteristics on these time sales. For these benchmarks, the linear scaling properties are interrupted by the dips due to periodic execution spanned across these time scales.

Note that benchmark *vortex* trace, with some "bumps" around scales 9-11, exhibits some indications for departure from exact LRD. The *perlbmk* trace shows this departure much clearer: a non-trivial scaling behavior for small time scales (octaves below 10) and a distinctly different large-time scaling behavior (octaves above 13), with a "change-region" that shows up very clearly as a pronounced "knee" in the graphs, approximately at scales [10,13].

It is observed that the dips at large time scales occur on benchmarks *mcf*, *gzip*, *bzip2*, *eon*, *gap* and *twolf*, indicating the those programs contain periodic structures even at very large time scales. In general, the time-varying slope of the LD's on Figure 3 for the largest time scales (octave in [16 20]) reveal that these time-scales contain a mix of correlation and scaling.

5.3 High-order Scaling Analysis

To discover the high-order scaling properties of the collected traces, we use the partition function and multiscale diagram described in subsection 3.3. While the LD reveals the energy (variance) of the increments of a process, the partition function and MD give information about the distribution of large and small wavelet coefficients at each time scale. We plot $\log(S(q, j))$ as a function of the scale j, for a range of the moments q ($1 \le q \le 8$). Discriminating between second-order (smooth) and high-order (irregular) process can be done by determining for which values of q the $\log(S(q, j))$ is the largest. A second-order process will have larger $\log(S(q, j))$ for $q \le 2$ while a high-order process for $q > 2$.

Figure 4 presents the logarithm of the partition function $\log(S(q, j))$ for all benchmarks. It shows that at large time scales, most of the studied benchmarks show the second-order scaling behavior ($\log(S(q, j)) < \log(S(2, j))$ for $q > 2$). This indicates that the second-order scaling properties are sufficient to describe the workload dynamic behavior at large time scales. Benchmarks *gcc*, *parser*, *perlbmk*, and *vortex* have large irregularities at small time scales. This implies that these traces contain large wavelet coefficients at small time scales, hence large irregularities. *Bzip2* and *gap* contain irregularities even at coarser time scales. For those programs, although at large time scales, all traces are second-order processes, for the short time scales behavior of workloads, multiplicative processes described it well.

6. On-line Program Scaling Estimation

So far, our workload dynamics scaling analysis has heavily relied on the trace collection and off-line processing. The demand on huge trace storage and off-line processing makes this method incapable of handling large programs that run long time. Moreover, for many applications, such as resource allocation/adaptation and performance monitoring/prediction, run-time program scaling measurement is necessary.

The Hurst parameter H, which measures the degree of self-similarity, holds a central place in the description of scaling behavior. Its accurate measurement is therefore of considerable importance. In this section, we present an on-line program scaling estimator that allows one to characterize program scaling behavior in-flight.

To calculate H, (1) a discrete wavelet transform (DWT) of the input data is first performed to generate the details $d_{j,k}$ over the time scales. (2) Then, at each fixed octave j, the details are squared and averaged across time k to produce the energy function E_j. (3) Finally, the Hurst parameter H is determined from the scaling regions in the LD plot.

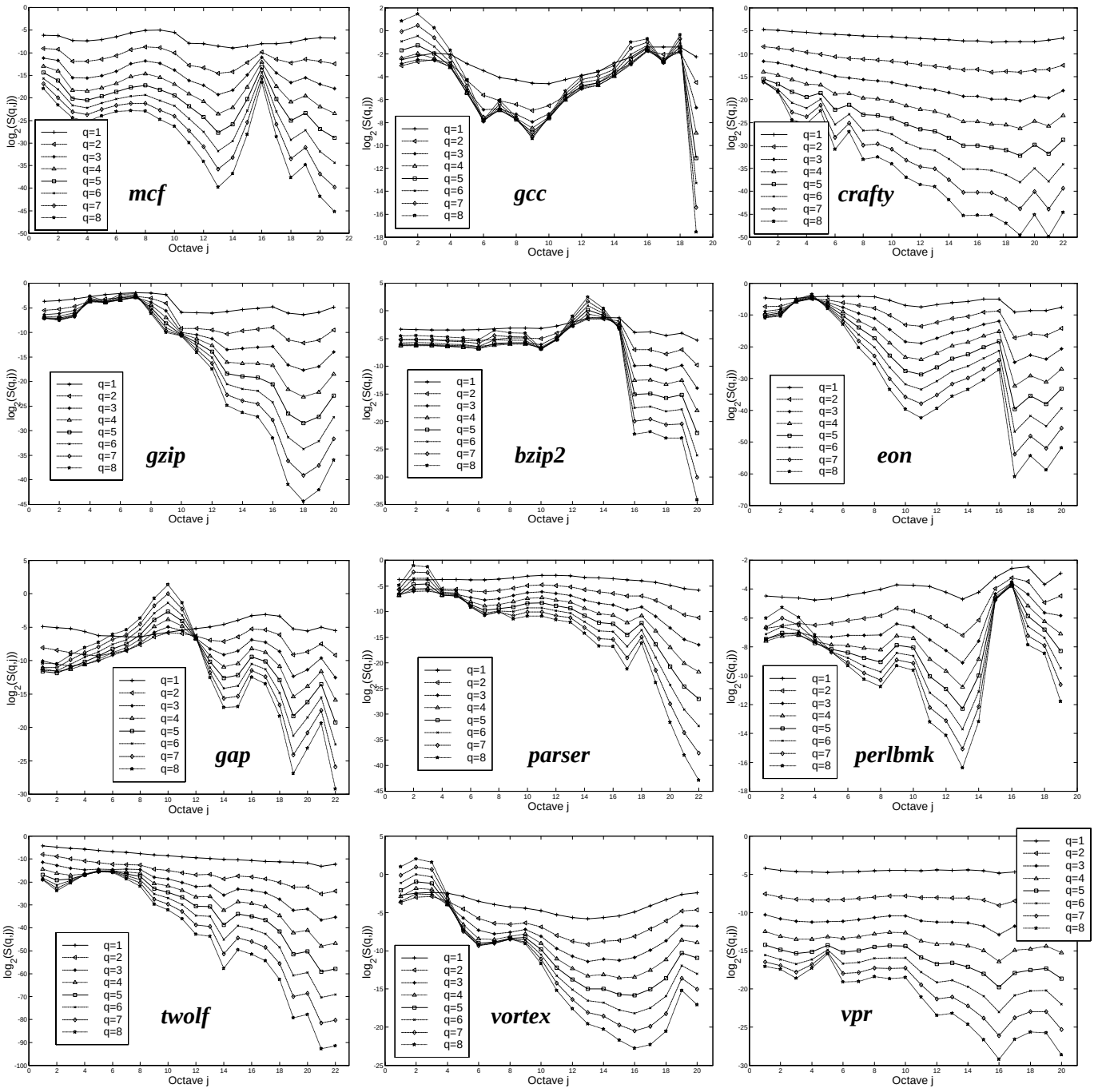

Figure 4 the MD plots

6.1 DWT and Filter-banks Design

The on-line discrete wavelet transform plays the key role in scaling estimation. Steps (2) and (3) are trivial once the details are computed. As described in Section 3, the DWT can be viewed as the multiresolution decomposition of an input sequence. The decomposition processes is implemented with two digital filters: high-pass filter G and low-pass filter H. After the process of filtering, downsampling operation is used to decimate half of the filtered results. To achieve multiresolution decomposition, the multilevel DWT can be implemented by the Mallat's pyramid algorithm [10]. Figure 5 shows the steps of Mallat's pyramid algorithm for DWT computation.

Figure 6 shows a three-level DTW using the pyramid algorithm and filter-banks. The output coefficients at each resolution level are calculated by using the low-pass coefficients of the previous level and are decimated by two.

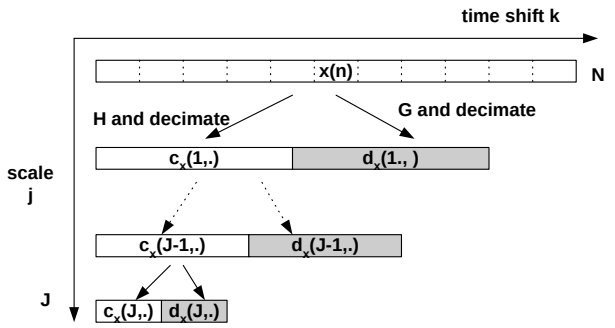

Figure 5 Mallat's pyramid algorithm for DWT computation

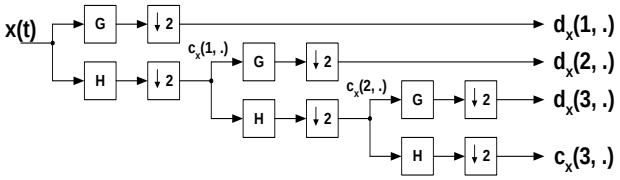

Figure 6 A three resolution level wavelet decomposition. At each level in the recursive structure, the high-pass filter (G) output $d_x(j,.)$ and the low-pass filter (H) output $c_x(j.,)$, occur at half the rate of the low-pass filter input $c_x(j-1,.)$

All filters in the pyramid tree shown in Figure 7 are constructed using FIR filters [11]. The structure of a FIR filter of length N is shown in Figure 7. As can be seen, each filter tap consists of a delay element, an adder, and a multiplier.

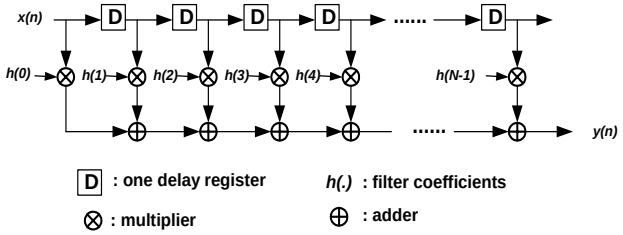

Figure 7 FIR Filter Structure

The filter coefficients are derived from the scaling functions of the corresponding wavelets. In this study, we use Daubechies wavelets to implement the DWT. This wavelet type is known for its excellent special and spectral localities [12]. The coefficients of the 3 wavelets from the Daubechies wavelets family are summarized in Table 3.

The Daubechies filters have the following relationship between its low-pass filter and high-pass filter coefficients:

$$g_{M-1-i} = (-1)^i h_i \quad (M > 2)$$

In this study, we use Daubechies 6-tap wavelet as the filter coefficients.

Table 3 Filter Coefficients of Daubechies Wavelets

		Daubechies Wavelets		
Coefficients i		M=6	M=8	M=10
i=0	*h(0)*	0.332671	0.230378	0.160102
i=1	*h(1)*	0.806892	0.714847	0.603829
i=2	*h(2)*	0.459878	0.630881	0.724309
i=3	*h(3)*	-0.135011	-0.027984	0.138428
i=4	*h(4)*	-0.085441	-0.187305	-0.242295
i=5	*h(5)*	0.035226	0.030841	-0.032245
i=6	*h(6)*		0.032883	0.077571
i=7	*h(7)*		-0.010597	-0.006241
i=8	*h(8)*			-0.012581
i=9	*h(9)*			0.003336
$\sum\|h(n)\|$		1.85118	1.865446	2.000938
$\max\{\|h(n)\|\}$		0.806892	0.714847	0.724309

6.2 Scaling Estimation Methodology

In a system designed to measure program dynamics on-line, there are two key components: the workload dynamics capture process and the on-line estimation itself.

Figure 8 illustrates the overall structure of scaling estimation and the interface between CPU and the modules. As can be see, performance counters and clock are used to generate the events of interest during program execution. The on-line estimator can be integrated into the operating system kernel or mapped with the highly efficient VISI architecture [23]. For the simulation based studies, the one-line scaling estimation module can be integrated into architectural simulators to provide estimates. In this work, we have integrated the proposed estimator into the SimpleScalar *sim-outorder* simulator.

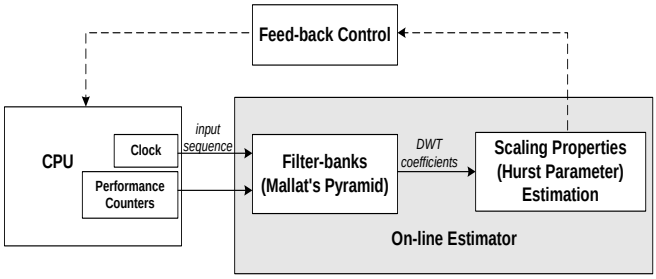

Figure 8 Overall Structure of Scaling Estimation

6.3 Performance

The performance of the estimator as a function of the length of data processed is demonstrated using execution driven simulations on the machine model described in Section 4. The IPC of program for every ten thousands elapsed cycles, generated by the *sim-outorder* simulator, is piped to the on-line estimator one at a time. The interval chosen between the actual estimations of H is every 2^8 data points.

Figure 9 shows the three examples of on-line estimation. The graphs illustrate typical behavior of the estimator in time. The dashed line shows the true Hurst parameter while the solid lines show examples of the one-line Hurst parameter estimates. As can be seen, there is a warm up period at the beginning of

the measurement run to wait for the octaves required for the analysis to become available. Figure 9 shows that the on-line estimator can accurately estimate the program scaling behavior.

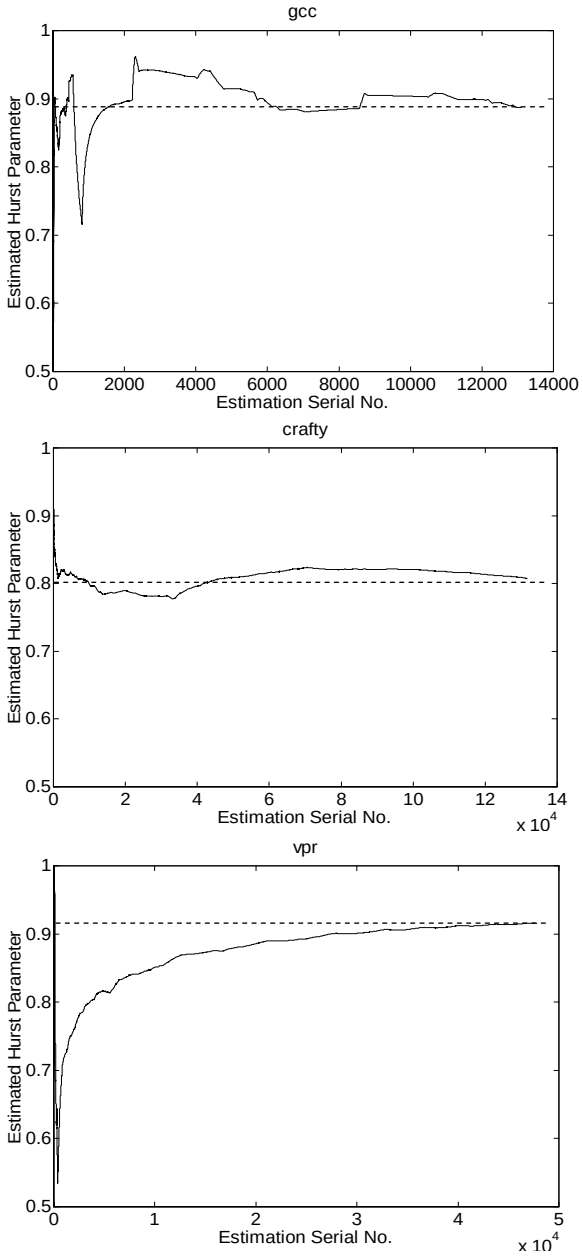

Figure 9 Three examples of on-line estimation. The dashed lines show the true Hurst parameter while the solid lines show examples of the on-line Hurst parameter estimates.

7. Related Work

The analysis of workload dynamic/phases behavior and then apply to run-time optimizations have recently received considerable attention [5, 6, 13, 14, 15]. For example, the SimPoint framework proposed by Sherwood and Calder [5] uses a set of representative instruction chunks to represent the entire program execution. In [24], variable length intervals are used to study hierarchical phase behavior of programs. Scaling models have been applied in the past to diverse fields, such as the study of strange attractors of certain dynamic systems [16] and modeling of traffic in modern communication networks [17]. Self-similarity, long-range dependence, and multifractal behavior have been studied and convincingly matched to real network traffic [18, 19, 20]. In [21], the authors present a first study to apply wavelet analysis to the computer architecture field. The VLSI architectures for implementing the DWT are summarized in [22, 23].

8. Conclusions and Future Work

The efficiency of advanced hardware and system optimizations increasingly depend on the dynamic behavior of workloads. Program execution manifests changing nature at run time. As software execution cycles become larger, such variation can span across a wide range of time scales. Understanding and characterizing the time-varying behavior of workload dynamics lays a foundation for an informed investigation on these long-run systems. Moreover, knowledge of program scaling properties is important to support tasks such as resource adaptation, performance monitoring, and anomaly detection at different time periods.

We show in this paper that various scaling properties (e.g., self-similarity, long-range dependence, and multiscaling) and metrics (e.g., energy and partition functions), combined with a wavelet-based analysis technique, can be used as a useful tool for unraveling the program dynamics over different time periods. We then apply this methodology to study the scaling behavior of SPEC2000 integer benchmarks, that is, to identify regions where the scaling property holds, to detect changes in scaling behavior, and to find ranges of time scales with more complex scaling patterns. An on-line program scaling estimator of the Hurst parameter is developed to allow the measuring of program scaling properties at run-time.

The compact and parsimonious models that exploit scaling phenomena can capture both long-term and short-term program behavior. In the future work, we plan to investigate the feasibility of using scaling models for benchmark statistics simulations and benchmark synthesis. The using of program scaling properties for resource adaptation, performance prediction, QoS control, and bottleneck and anomaly detections will also be investigated.

References

[1] A. Maynard, C. Donnelly, and B. Olszewski, Contrasting Characteristics and Cache Performance of Technical and Multi-user Commercial Workloads, In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 1994.

[2] K. Keeton, D. A. Patterson, Y. Q. He, R. C. Raphael and W. E. Baker, Performance Characterization of a Quad Pentium Pro SMP Using OLTP Workloads, In the Proceedings of the International Symposium on Computer Architecture, 1998.

[3] L. A. Barroso, K. Gharachorloo, E. Bugnion, Memory System Characterization of Commercial Workloads, In

the Proceedings of the International Symposium on Computer Architecture, 1998.

[4] L. Eeckhout, R. Bell, B. Stougie, K. D. Bosschere and L. K. John, Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies, In the Proceedings of the International Symposium on Computer Architecture, 2004.

[5] T. Sherwood, E. Perelman, G. Hamerly and B. Calder, Automatically Characterizing Large Scale Program Behavior , In the Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 2002.

[6] E. Duesterwald, C. Cascaval, and S. Dwarkadas, Characterizing and Predicting Program Behavior and its Variability, In the Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 2003.

[7] D. Sornette, Critical Phenomena in Natural Sciences, Springer-Verlag, 2000.

[8] I. Daubechies. Ten Lectures on Wavelets. Capital City Press, Montpelier,Vermont, 1992.

[9] D. Burger and T. M. Austin, The SimpleScalar Tool Set, Version 2.0, Technical Report 1342, Computer Sciences Department, University of Wisconsin–Madison, June 1997.

[10] S. Mallat, Multifrequency Channel Decompositions of Images and Wavelet Models, IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. 37, page 2091-2110, 1989.

[11] A. Oppenheim and R. Schafer, Discrete Signal Processing, New Jersy: Prentice Hall, 1999.

[12] I. Daubechies, Orthonomal bases of Compactly Supported Wavelets, Communications on Pure and Applied Mathematics, vol. 41, pages 906-966, 1988.

[13] T. Sherwood, S. Sair and B. Calder, Phase Tracking and Prediction, In the Proceedings of the Annual International Symposium on Computer Architecture, 2003.

[14] A. S. Dhodapkar and J. E. Smith, Managing Multi-configuration hardware via Dynamic Working Set Analysis, In Proceedings of the International Symposium on Computer Architecture, 2002.

[15] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, Memory Hierarchy Reconfiguration for Energy and Performance in General-Purpose Processor Architectures, In Proceedings of the International Symposium on Microarchitecture, 2000.

[16] D. Auerbach, B. O'Shaughnessy, and I. Procaccia, Scaling Structure of Strange Attractors, Physical Review A, vol. 37, issue 6, page 2234–2236, Mar. 1988

[17] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz, The Changing Nature of Network Traffic: Scaling Phenomena, ACM Computer Communication Review, vol. 28, page 5-29, Apr. 1998.

[18] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, On the Self-Similar Nature of Ethernet Traffic, IEEE/ACM Transactions on Networking, vol. 2, no. 1, page 1-15, Feb. 1994.

[19] A. Feldmann, A. C. Gilbert, and W. Willinger, Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN traffic, ACM Computer Communication Review, vol. 28, page 42-55, Sept. 1998.

[20] P. Abry and D. Veitch, Wavelet Analysis for Long-range Dependent Traffic, IEEE Transactions on Information Theory, 44 page 2-15, 1998.

[21] R. Joseph, Z. Hu, and M. Martonosi, Wavelet Analysis for Microprocessor Design: Experiences with Wavelet-Based dI/dt Characterization, In the Proceedings of the International Symposium on High Performance Computer Architecture, 2004.

[22] C. Chakrabarti, M. Viswanath, R. M. Owens, Architectures for Wavelets Transforms: A Survey, Journal of VLSI Signal Processing, vol. 41, page 171-192, Dec. 1996.

[23] P. Y. Chen, VLSI Implementation for One-Dimensional Multilevel Lifting-Based Wavelet Transform, IEEE Transactions on Computers, vol. 53, no. 4, Apr. 2004.

[24] J. Lau, E. Perelman, G. Hamerly, T. Sherwood, and B. Calder, Motivation for Variable Length Intervals and Hierarchical Phase Behavior, In the Proceedings of the International Symposium on Performance Analysis of Systems and Software, 2005.