RC24007 (W0607-058) July 14, 2006
Computer Science

IBM Research Report

Online Power and Performance Estimation for Dynamic Power Management

Karthick Rajamani, Heather Hanson, Juan C. Rubio,
Soraya Ghiasi, Freeman L. Rawson
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Online Power and Performance Estimation for Dynamic Power Management

ABSTRACT

Power management is at the forefront of challenges facing next-generation computing systems. Multiple mechanisms for dynamic regulation of processor power can be used aggressively to address power and thermal issues in real time; however, to do so effectively requires knowledge of the power and performance impact of the power-regulation choices. The problem is complex since the impact of each mechanism is both system- and workload-dependent.

In this paper, we present our methodology and models for online predictions driven by data from processor performance counters. They estimate power consumption and performance when multiple power-management features are used simultaneously. Our goal is to provide effective, efficient power and performance prediction capabilities, which allow OS schedulers and workload/power managers to gauge the power and performance impact of a state change before making it. We focus on near-term estimation over 10s of millisecond intervals, rather than long-term averages. This approach allows power managers to fine-tune the power state of the processor for scheduling changes and workload variations. It also offers fast response for controlling the demand on the power supply and the cooling subsystem.

We implement our methodology in a software service running on a Pentium M system and evaluate the power and performance models using the SPEC CPU2000 benchmarks. We also demonstrate their use with two new dynamic power management applications: Power Saver that optimizes for saving power within a specified bound on performance loss and a Performance Maximizer that optimizes for best performance while meeting a specified power limit.

1. INTRODUCTION

Power and thermal management are at the forefront of concerns facing modern computing systems. Increasing circuit densities, limitations in technology scaling with respect to voltage and power reductions, dense system packaging, and increasing cost of advanced cooling and packaging solutions have made power and thermal issues critical across all classes of computing systems. Researchers are trying to address problems ranging from chip-level thermal management, to intelligent management of data-center resources for balancing cost and capacity limitations against performance goals and dynamic provisioning capabilities.

Techniques like dynamic voltage and frequency scaling (DVFS) provide a way of regulating power consumption at the cost of altering system performance. Once proposed for battery-operated environments, such techniques are now prevalent across a wide spectrum of systems. Examples include PowerTune in Apple's IBM PowerPC 970 based systems [1], Intel's DBS (demand-based switching) for server power management [2], Linux's `cpufreq`-based drivers and Windows' power management that exploits ACPI-defined processor p-states. Their usage is still primarily limited to energy-saving, reducing power consumption when systems are under-utilized. However, there are a wide array of problems beyond energy savings which require intelligent power and thermal management solutions. For example, Intel's on-chip micro-controller, Foxton [3], tries to maximize chip computing capabilities in the face of variation in the manufacturing process and environment limitations by using just-in-time frequency management with real-time thermal and power consumption feedback from on-chip sensors.

The power management mechanisms most suitable for dynamic adaptation change characteristics of the system which have an impact on both the power and computing capabilities and, consequently, application performance. The most critical aspect of any dynamic power management solution is the ability to assess, in real-time, the impact on both performance and power of any mechanism the solution can apply at a given moment. *Knowledge of the quantitative impact of these actions is vital for dynamic power management solutions which must choose which mechanism to engage and to what extent in order to address a given situation.*

In this paper, we focus on predicting the performance and power impact of an execution state change for any running application in real-time. Guiding principles for prediction model development include ease of incorporation in software solutions at the kernel- or user-level and low run-time overheads. Our methodology uses platform-specific processor performance counters as generic activity monitors for power management and creates activity-based models for real-time performance and power estimation. Using performance counters for power estimation and activity monitoring has proven history. Our work extends their usage for real-time prediction and with models that predict across state changes. We also generalize the existing concept of p-states to include all the processor features which manage power and affect performance and can be altered at run-time. For the Pentium M platform used in this paper, our modified notion of p-states is defined by frequency, voltage, and clock throttle level.

This paper makes the following contributions:

- It gives a methodology for real-time performance and power prediction across p-state changes initiated by dynamic power management.
- It develops power-performance estimation solutions on a widely available processor, the Intel Pentium M processor. Both available mechanisms for processor execution state changes—DVFS and clock throttling—are analyzed.
- It evaluates the methodology with detailed experimental results on the Pentium M.
- It demonstrates their usefulness with two new dynamic power management applications.

The paper is organized as follows. The following section briefly discusses the related work. Section 3 presents the role of power and performance estimation, our methodology for it, and our platform and infrastructure for real-time estimations and evaluation. Section 4 presents our solutions for activity rate and performance estimations, their experimental evaluations and results. Section 5 presents our solutions for activity-based power estimation, their experimental evalua-

tion and results. Section 6 presents examples of our models being used in prototype implementations for new dynamic power management applications. Finally, we conclude with Section 7.

2. RELATED WORK

Industrial standards such as Advanced Power Management(APM) and the Advanced Configuration and Power Interface (ACPI) [4], define protocols for power management states. Commercial systems include a variety of power-management mechanisms including SpeedStep [5] and PowerNow [6]. Multiple p-states are available for most systems.

There have been a number of efforts over the years examining the implementation and effectiveness of dynamic voltage and frequency scaling for saving power and energy in embedded systems [7, 8, 9]. Performance-oriented explorations include attempts to quantify and/or reduce the performance lost when using DVFS for energy-saving. While our solution applies to this case, our estimation approach also facilitates performance optimization in larger systems constrained by power supply and cooling limitations rather than energy.

Ghiasi, et al., introduce an operating systems scheduler which dynamically places tasks on the processor that most closely matches the application's *ideal frequency setting* [10] in a multi-processor system with heterogeneous-frequency processors. While they use performance counter-based prediction to guide decisions, they focus on mitigating performance loss and do not model or predict power.

Counter-based power estimates have been developed for use in power and energy management for several platforms, including models by Bellosa, et al., [11, 12] and Contreras and Martonosi [13] for the Intel Xscale embedded systems processor and Bircher, et al., [14] for the Intel Pentium4 processor. Our work is distinct in the following ways:

- Prior power model evaluations focused on average power prediction. We investigate instantaneous power and performance, which is beneficial for online applications such as operating systems scheduling.
- We predict the impact of p-state changes, unlike much of prior work which predicts only for the current p-state.
- Counter-based models are inherently platform-specific, and our models are the first of which we are aware for the Pentium M system.

3. ONLINE ESTIMATION

The goals of this paper are to develop a methodology for deriving models to predict the power and performance impact of changes in processor p-states used in dynamic power management, and, to show how such models can be used by operating systems and workload managers to make power and performance trade-offs during system operation.

3.1 Exploiting Power/Performance Models

Power and performance models have wide applicability in systems software. They can be used to adapt to application phase changes, environmental changes, and even hardware failures. An operating system can use them to implement performance-aware power-savings strategies by projecting the performance of its power management actions, estimating the performance impact of each possible power-saving action before exercising it. In a virtualized environment, the hypervisor can provide specific power and performance guarantees to each OS partition while still operating within its overall power and thermal constraints.

Alternatively, the system can maximize performance while operating under power or thermal constraints. For a given power constraint, the OS can select the p-state that yields the maximum performance for the workload while still meeting the constraint. As workload behavior changes, the OS can continually adapt to provide the highest-performing safe p-state for the current application behavior.

Power constraints can also change dynamically due to the addition of new machines to a power or cooling domain, cooling unit failures, load-shedding during non-peak hours, or the imposition of a power cap to bound operating expenses. Such a change can be explicitly communicated or automatically detected by the OS. Our power and performance estimation models allow the OS to adapt the p-state to the changing conditions.

Systems can also use the models to enhance traditionally performance-centric decisions such as task scheduling and admission control by taking power and energy considerations into account. They allow the decisions to be to be guided by both power estimations and performance projections.

3.2 Model Considerations

Events which trigger power-performance adaptation like application phase changes, environmental changes, and hardware failures have a wide range in their monitoring, estimation and reaction timescales from 10^{-2} to 10^{1} seconds. Targeting a 10 millisecond timescale allows our solutions to be applicable across a wide range of power management solutions.

Wide applicability and good accuracy are obviously desirable, but low overheads and quick responsiveness are also vital for the real-time usage. We have found that activity-based models for power and performance estimation based on data collected from processor performance counters meet these requirements. Performance counters are a proven technology for processor event monitoring and are widely available in modern microprocessors. Counter data can be accessed with high enough frequency (100s of samples per second) with little performance impact.

The requirement that we use a very limited number of counters without the overhead of sampling or switching between counter types does limit the absolute accuracy of our predictions. Models with more counters, more frequent accesses, and long observation periods would increase prediction accuracy but limit the ability to use counters for real-time, short-term prediction vital to tracking workload behavior or unpredicted environment changes. We also require our models have minimal observation and learning overheads. Ideally, a single observation at any p-state should help us estimate the power and performance impact for any change in p-state.

3.3 Problem Formulation

Throughout the rest of the paper, we use the following notation.

- S: $S(f,t)$ denotes the processor p-state as a function of the frequency of operation f (with implicit associated voltage) and the effective duty cycle t. In the case of the Pentium M clock throttling selects the effective duty cycle.

- A: $A(f,t)$ denotes an activity rate of the processor at frequency f and effective duty cycle t.
- P: $P(f,t)$ denotes the power at frequency f and effective duty cycle t.
- Values without a $'$ are current values, values with a $'$ are new or future values. For example, f represents the current frequency and f' is the new frequency.

Our goal, given the observation of a performance metric at a particular $S(f,t)$, is to estimate what the values for the performance metric and power would be when $S(f,t)$ is changed. Since the instruction stream initiates activity and power consumption is a consequence of this activity, we decompose the problem into two steps, estimating activity at a new state $S(f',t')$ and then using the activity estimate to produce a power estimate:

1. Estimate $A \rightarrow A'$ given $S(f,t) \Rightarrow S(f',t')$.
2. Estimate P' given A'.

3.4 Infrastructure

The experimental infrastructure for model development and evaluation uses a Pentium M 755 (Dothan) with a Thermal Design Power (TDP) of 21 W and supporting both DVFS and clock throttling p-state change mechanisms. DVFS allows scaling from 600 MHz to 2000 MHz. We discretize the frequency scaling range into eight 200 MHz steps with corresponding voltage changes. Clock throttling provides 8 steps of roughly 12.5% increments in effective duty cycle.

The Radisys system board [15] has built-in sense resistors for tapping the two supply points that power the processor, as shown in Figure 1. The current, measured as the voltage across the sense resistor, and the supply voltage are filtered, amplified with a National Instruments SCXI-1125 module and then digitized with a NI PCI-6052E DAQ card to be periodically collected on a separate measurement computer. The power measurement is completely non-intrusive and the peak sampling capability of 333 K samples/s is more than adequate for the 10ms sampling intervals we are interested in for this work.

We have developed low-overhead drivers for changing $S(f,t)$ states by altering DVFS and clock throttling levels and for monitoring performance counters under Windows XP and Linux. The monitoring drivers collect the counters every 10 milliseconds with negligible performance impact. A GPIO signal is used to mark the beginning and end of the traces, and the marks are used to correlate the power and activity traces.

The Pentium M processor has only two simultaneously accessible programmable performance counters that can monitor any of 92 different events. For practical real-time application of our estimation models, we restrict our models to using just one or two activity rates chosen from the 92 monitorable events. The estimation models developed in Section 4 and Section 5 are tailored to the Intel Pentium M processor. However, the same approach used here can be applied to create models for a different processor.

3.5 Usage Considerations

We are exploring online estimation with a minimal number of event counters for practical use in commercial systems. Although we have built our estimation models upon user-accessible performance counters for this study, in practice this approach may be unacceptable. User-accessible counters may be reconfigured by system users, rendering the values stored in performance counters registers meaningless to our models. Introducing a small, dedicated set of performance counters for use by estimation models would eliminate this shortcoming. Building models with additional user-accessible performance counters can improve model accuracy, but at the expense of additional area in the processor core. Dedicated performance counters would also require additional area and wiring, so the number of available counters will remain restricted.

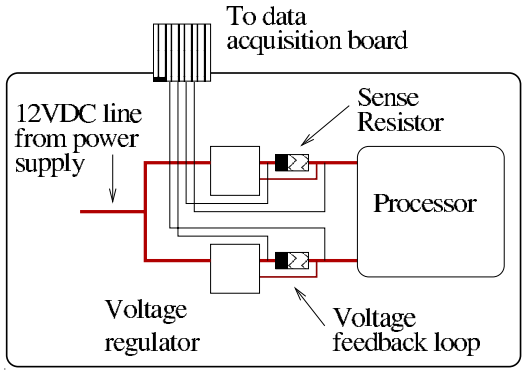


Figure 1: Experimental platform: system under test with sense resistors and data acquisition probes to measure processor power.

We found that three counters would be beneficial yet still within reasonable limits. However, because we use the Pentium M's existing performance counters, we are limited to the simultaneous use of at most two counters in this study. We demonstrate the use of performance and power estimation separately in this work using two counters per model, noting that a system with 3 counters could estimate power and performance simultaneously.

3.6 Model Development and Evaluation

Basic insights into how power or an activity rate changes with $S \rightarrow S'$ are obtained by observations made with a well-understood set of micro-benchmarks, collectively known as Memory Sensitive Loops (MS loops). Table 1 summarizes the four loops used for this study. All four perform simple array access operations. These loops are configured to operate with different data footprints to exercise each of the different memory hierarchy levels (L1, L2, and DRAM) intensively to study dependence of application behavior on the memory hierarchy. Our training data set consists of 12 data points (4 loops, 3 footprints each) per $S(f,t)$ setting. An advantage of using small, well-defined loops over other benchmarks is that their performance and power characteristics are relatively stable during the course of a run and across multiple runs. The loops are run at the highest real-time priority mode to minimize interference.

Once developed, our models are validated using a different dataset. We use the SPEC CPU2000 benchmark suite with the *reference* datasets for this purpose.

4. PERFORMANCE MODELING

It is important to know the performance impact of an $S(f,t)$ change before making it, because the impact might be intolerable or undesirable for the workload. We address this issue by including activity modeling as a core component of our approach for power and performance estimation across

Micro-benchmark	Description
DAXPY	The daxpy function from the Linpack benchmark [16]. This loop traverses two floating point arrays, scales each element of the first array by a constant adding it to the corresponding element of the second array.
FMA	Floating-point multiply-add. It reads two adjacent elements from a single array, computes their dot product and sums it up for all such pairs – results in a multiplication and an addition per pair. The result is kept in a local variable. The Pentium M's built-in hardware prefetching is most exercised for this loop.
MCOPY	Sequentially copies all elements of one array to a second one. This loop tests the bandwidth limits of the accessed memory hierarchy layers.
MLOAD_RAND	Random memory load. This loop performs random accesses over the elements of an array. It can be used for determining the latency of a memory hierarchy.

Table 1: Micro-benchmarks used to study characteristics and as training set for the models.

$S(f,t)$ changes. It has dual roles:

- Performance estimation: We present a methodology to estimate the change in instantaneous performance captured by Instructions Retired/cycle with change in $S(f,t)$.
- Power estimation: We provide models that facilitate the estimation of the activities used by the power models: Instructions Decoded/cycle (DPC) and Data Cache Unit Miss Outstanding Cycles/cycle (DCU) (refer section 5).

4.1 Development of Activity Models

Our model development approach is guided by observations which indicate how activity rates change with frequency and throttling level for each of the MS loops. In the Pentium M, the frequencies for the L1 and L2 cache levels scale with the processor frequency, while the speed of DRAM memory is independent of the processor frequency. Thus, to understand the impact of dominant usage of each level of the memory hierarchy, each MS loop is run at three memory footprints. The different characteristics of the loops give us insight on the difference in impact for different workload types.

Our activity model development is based upon the following steps:

1. Examine activity rate trends with $S(f,t)$ changes.
2. Identify regions of uniform trends, capture the trends in the models.
3. Simplify models to reduce the number of parameters.
4. Solve for remaining parameters.

In the model development discussion below, different characteristics or behavior are illustrated with specific, representative data. First we discuss the characteristics and the consequent models for the instruction rates - both IPC and DPC share these characteristics. DCU exhibits different behavior and is discussed separately.

4.1.1 Instruction Rate Characteristics

Key characteristics are first illustrated with representative data for IPC. Although we do not discuss the characteristics of DPC changes here, these observations are equally valid for DPC - both respond to frequency and throttling changes in a similar way for all regions of memory.

Figure 2 shows the variation in IPC versus the effective duty cycle as a result of throttling for each of the different frequency settings. While the behavior shown is for the DAXPY micro-benchmark, it is representative of the behavior for all the MS loops. The behavior for the 128K footprint is very similar to the 4K footprint and so not shown separately. The first observation is IPC variation with effective duty cycle is linear at all footprints – the impact of throttling on performance is linear with the duty cycle reduction and not very dependent on memory hierarchy usage.

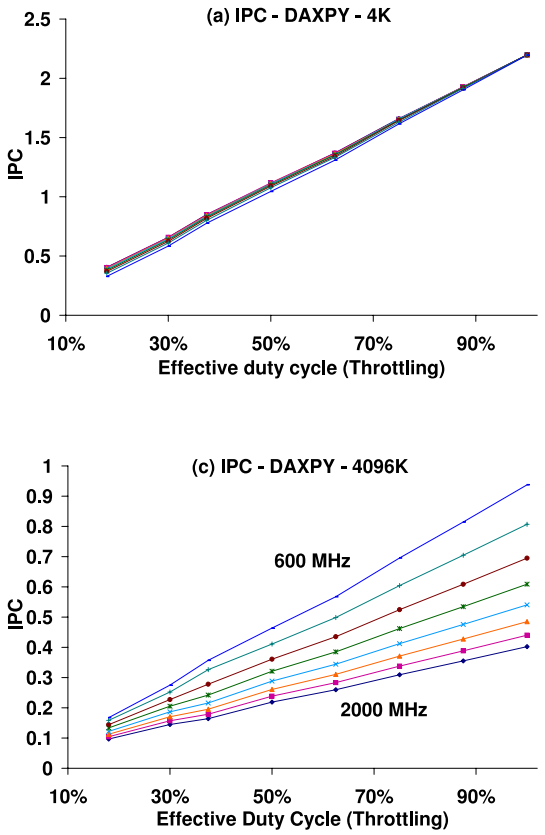


Figure 2: IPC vs throttling at different frequencies—DAXPY micro-benchmark.

IPC is not affected by frequency change when the data accesses are restricted to L1 (4KB) and L2 (128KB) - the behavior for the 128KB footprint is very similar to the 4KB footprint and so not shown. The separation of the lines at the 4096KB footprint, as seen in Figure 2 (c) suggests frequency has significant impact on IPC once DRAM accesses dominate.

The impact of frequency on IPC is further illustrated in Figure 3 that shows the variation in IPC with frequency at each throttling level (labeled with the effective duty cycle). (a) shows the independence of IPC with respect to frequency at the 128KB footprint. The 4KB footprint is not shown, but exhibits the same behavior. Figure 3(c) shows the near linear increase in IPC with the cycle time (or $1/f$) rather than with frequency. The R^2 of the least squares fit decreases with the duty cycle of throttling (shown just for the highest throttling and no throttling cases), indicating the linear fit is better when unthrottled.

The independence of IPC from frequency for 4KB and 128KB but the inverse relationship for the 4096KB footprint comes from which memory hierarchy levels are involved for each footprint. For the 4KB and 128KB footprints only the L1 and L2 cache levels are used and as they scale with processor frequency there is no change in IPC with frequency. With the 4096KB footprint, the DRAM is accessed and its latency does not scale with processor frequency. At higher frequencies, the fixed DRAM latency imposes a larger penalty in terms of processor cycles and so results in a lower IPC compared to when executing at a lower frequency.

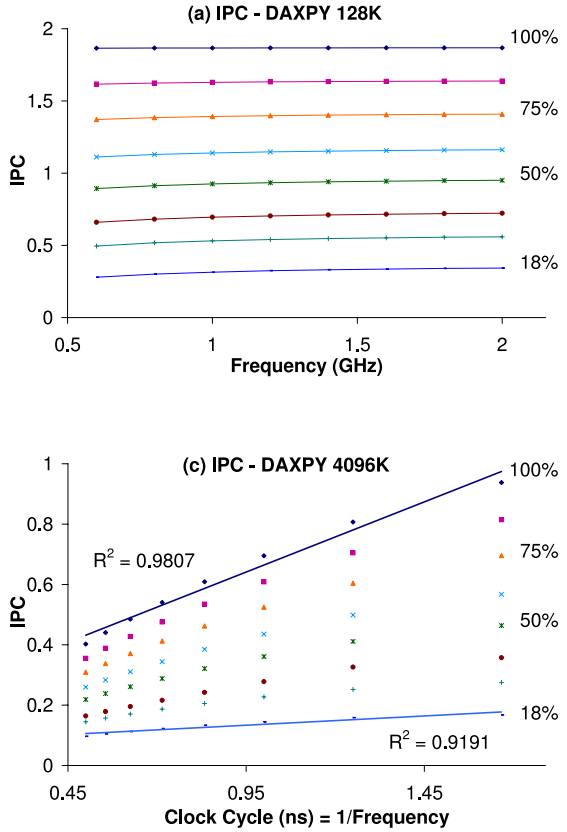


Figure 3: IPC vs frequency—DAXPY micro-benchmark. Note (c) is showing variation with $1/f$.

4.1.2 Models for Instruction Rates

When an activity rate A scales linearly with an effective operating level L (duty cycle, frequency or 1/frequency), then it can be represented as

$$A = a \cdot L + b \tag{1}$$

The two parameters in the equation, a and b, can be dependent upon the workload and the values for the other setting, e.g. the coefficient and constant for throttling are dependent upon the frequency of operation as shown in Figure 2(c).

When a is negligible, then A is independent of the value of L, so an observation at any value of L would be applicable for any other value of L.

$$A(L) \approx b \implies A(L') \approx A(L) \tag{2}$$

This is the model for IPC when the footprints are L1 or L2 resident (see Figure 3).

If b is negligible, then $A(L')$ can be inferred from $A(L)$ as

$$A(L) \approx b \cdot L \implies A(L') \approx A(L) \cdot (L'/L) \tag{3}$$

This captures the behavior of IPC with respect to throttling changes at all footprints (see Figure 2).

When neither a or b are negligible, one would need to take multiple observations at different values of L keeping other conditions the same to solve for both parameters. This may not be a feasible option if the application changes behavior often or if one is not allowed $S(f,t)$ changes just to learn the model.

As we desire to predict the behavior from a single observation to support fast real-time adaptations, we searched for an alternative approach. The constant b in the linear model reduces the impact of L on A. An allometric function provides an alternative that can be used to get a similar effect without the constant

$$A(L) \approx c \cdot (L)^d \implies A(L') \approx A(L) \cdot (L'/L)^d \tag{4}$$

The exponent d is restricted to the range between 0 and 1. This approximation can only be used when the errors between the linear model and this *generalize allometric* model are small.

Combining the equations derived for the three cases, our model can then be represented by the following generalized equation:

$$A(f',t') = A(f,t) \cdot \left(\frac{t'}{t}\right)^{\alpha} \cdot \left(\frac{f}{f'}\right)^{\beta} \tag{5}$$

where α is between 0 and 1 inclusive, and β s between -1 and 1 inclusive. The values of α and β are determined by which case corresponds to the behavior associated with a particular L change, where L may be f, or t.

To apply our generalized model to the activity rates we are interested in, recall that IPC is linear with the cycle time ($1/f$) in the 4096KB footprint region. The linear dependency on $1/f$ results in f/f' rather than f'/f.

With MS loops it is easy to identify when to treat IPC as a constant with respect to frequency and when to use the inverse relationship as the footprints give a good indication of the dominant memory hierarchy level in use. For a general application that may not be the case. We use the ratio of the L1 data miss cycles outstanding (DCU) to the instruction rate (IPC or DPC) ton insight into of the memory hierarchy usage. For DCU/IPC below a certain threshold, IPC is predicted as constant with respect to frequency. For DCU/IPC above that threshold, IPC is predicted to vary as a power between 0 to 1 of cycle time. The values of the threshold and exponent are solved using all the data for MS loops by minimizing the sum of the square of the normalized errors ($\Sigma(1 - y_{est}/y)^2$). We chose to use the normalized errors as the absolute error values vary over a wide range and using them would discard the importance of errors at small values of IPC and exaggerate the importance of errors at large val-

ues of IPC. Solving for the thresholds and exponents in this fashion yields Equation 6 for IPC and Equation 7 for DPC.

$$IPC' = \begin{cases} IPC \cdot (t'/t), & DCU/IPC < 1.21 \\ IPC \cdot (t'/t) \cdot (f/f')^{.81}, & DCU/IPC \geq 1.21 \end{cases} \quad (6)$$

$$DPC' = \begin{cases} DPC \cdot (t'/t), & DCU/DPC < 1.21 \\ DPC \cdot (t'/t) \cdot (f/f')^{.92}, & DCU/DPC \geq 1.21 \end{cases} \quad (7)$$

4.1.3 Model for DCU

Tracking the behavior of DCU with memory footprints, we find that with respect to throttling, the behavior is linear for L2 and DRAM accesses with small b that can be ignored and is unaffected within L1.

DCU is linear with cycle time within L1, almost constant with high amount of L2 accesses and linear with frequency with a large constant with dominant DRAM accesses.

Again, to identify the threshold for the transition from constant to power scaling with respect to frequency and the exponent we solve for them by minimizing the sum of square of the normalized errors. Doing so, we end up with Equation 8 for predicting DCU.

$$DCU' = \begin{cases} DCU \cdot (f/f'), & DCU < .001 \\ DCU \cdot (t'/t), & DCU \geq .001 \, and \\ & DCU/DPC < .005 \\ DCU \cdot (t'/t) \cdot (f'/f)^{.27}, & DCU \geq .001 \, and \\ & DCU/DPC \geq .005 \end{cases} \quad (8)$$

4.2 Evaluating Activity Models

For our evaluation, we initiate a change from S to S', estimate the activity rates (A) for that change and compare the measured activity rates ($A_m(S')$), versus estimated activity rates ($A_e(S')$).

All observations are taken with the SPEC CPU2000 benchmark. For each pair of activity rates {IPC, DCU} and {DPC, DCU}, the benchmark suite was executed three times – one where only frequency changes are initiated every 1 second for unthrottled operation, one where only throttling changes are initiated every second for 2000 MHz operation, and one where both frequency and throttling levels are changed every second. The timer for activity rate sampling was set with 33 ms periods; the actual duration of each sample period is subject to the Windows timer variability.

To remove the impact of changing application characteristics (and consequently, changing activity rates) from when we make the observation $A(S)$ to when we measure $A(S')$ we impose two conditions. One, the application has to be in a relatively steady state at the time of observation $A(S)$. We implement this by checking the recent history of the activity rates before changing $S(f,t)$. Two, we observe activity through a sequence of $S \rightarrow S' \rightarrow S$. The second observation at S should be comparable to the original $A(S)$ to ensure that the difference in activity rate is due to the change in state and not due to change in application behavior. If the repeated observation at S differs by more than 10% from the original S observation, we invalidate that sample. Only valid samples are included in the evaluation analysis.

4.3 Results

Table 2 summarizes the main results from the evaluation. There are three columns for each activity metric—*f and t*, *f*, and *t*—containing the statistics for the different runs as described above. The first row contains the average absolute percentage error across all the predictions for that run, the second row its standard deviation, and the third row the percentage of the population in the interval $[mean - stddev, mean + stddev]$. The next row contains the number of observations. All the other rows summarize the error distribution which is a *normal-like* distribution (unimodal, roughly symmetric about the mean). When only frequency or throttling levels are changed, the total run duration is shorter than when both are changed as the system runs unthrottled or at maximum frequency, correspondingly. This results in fewer number of observations when only one mechanism is used compared to when using both.

Table 2 shows the average error predicting across both frequency and throttling level changes is just under 10% for both the instruction rate metrics. For both, predicting impact of throttling level changes is more accurate than predicting frequency change impact.

The average error for predicting DCU changes is higher at 22.45%. Predicting the impact of throttling level change is less accurate than predicting frequency change impact because the linear approximation for DCU versus duty cycle relation has slightly higher errors than a linear approximation for the instruction rate versus duty cycle relations.

The relatively high proportion of the population within $[\mu - \sigma, \mu + \sigma]$ suggest that knowing the μ and σ values for these estimations provides a reasonable summary of the results. For a normal distribution, only 68% of the population would be within this range.

The values for the error prediction median and range (maximum over-prediction to maximum under-prediction) show that all predictions—but for DCU prediction with just frequency changes—are negatively centered. The fact that the medians are quite close to zero while the distribution means are more negative suggest that the distributions are negatively centered more from a larger magnitude of over-prediction errors than from a larger incidence of such errors.

We also found that the results did not change much when we used the same SPEC data set observations used in the validation to also derive the parameter values used in the activity-estimation models (i.e. fit the models to the test data set). This is primarily a fall-out of our model development approach that minimized the number of parameters that were dependent on the model input data set.

5. ACTIVITY-BASED POWER MODELING

In this section, we address model development and evaluation for the task of estimating the power from activity rates for each processor state $S(f,t)$.

5.1 Power Model Development

Pentium M performance counters defines 92 event types, but only two may be observed simultaneously. For real-time, low-overhead, online estimation, we use a maximum of two counters in the power model. We examined 13 counters likely to be useful for power prediction, including instructions completed, micro-operations retired, instructions decoded, bus transactions to memory, and several cache-related counters. We correlated counter values with measured power recorded during microbenchmarks executing at

	IPC Prediction			DPC Prediction			DCU Prediction		
	f and t	f	t	f and t	f	t	f and t	f	t
Mean absolute error (m)	9.53%	7.17%	4.89%	9.79%	8.58%	6.29%	22.45%	13.17%	16.26%
Std. dev. absolute error (s)	13.97%	14.7%	6.66%	16.74%	15.88%	9.35%	33.87%	9.72%	29.22%
Pop. within $[m-s, m+s]$	90.54%	91.55%	89.06%	97.28%	89.02%	93.98%	92.6%	83.98%	90.24%
Number of observations	2970	734	905	2826	774	963	2826	774	963
Error distr. mean (μ)	-2.63%	-2.94%	-1.14%	-3.46%	-3.22%	-1.34%	-5.61%	0.62%	-4.61%
Error distr. std. dev. (σ)	16.7%	16.08%	8.19%	19.08%	17.76%	11.19%	40.24%	16.35%	33.12%
Pop. within $[\mu-\sigma, \mu+\sigma]$	83.77%	88.28%	84.09%	86.02%	86.18%	83.8%	86.34%	68.6%	85.67%
Error distr. median	-0.61%	-0.32%	-0.35%	-0.36%	-0.18%	-0.24%	0.31%	2.53%	-0.41%
Maximum over-prediction	151%	179%	54%	168%	113%	84%	389%	76%	289%
Maximum under-prediction	54%	41%	47%	71%	47%	45%	82%	50%	82%

Table 2: Results for prediction.

frequency (MHz)	α	β	γ	R^2	S_e
600	0.34	0.88	1.70	0.94	0.04
800	0.54	1.28	2.28	0.94	0.07
1000	0.77	1.68	2.81	0.93	0.09
1200	1.06	2.16	3.44	0.93	0.13
1400	1.42	2.80	4.15	0.93	0.16
1600	1.82	3.44	5.00	0.93	0.22
1800	2.36	4.08	6.10	0.91	0.33
2000	2.93	4.64	7.47	0.90	0.46

Table 3: 1-Counter Model Coefficients and Regression Statistics

the highest user priority. We chose to use Decoded Instructions per cycle (DPC) in the power model for three reasons. First, DPC is well-correlated with power, with correlation factors ranging from 0.82 at 600 MHz to 0.93 at 2000 MHz. Second, DPC captures power-consuming activity due to speculative instructions that similar counters such as instructions completed (IPC) do not. Third, decoded or dispatched instruction events are widely available on other platforms so similar models may be constructed for many other systems.

While instruction rates are good indicators for power consumption, the system operating frequency and supply voltage also have a significant impact on power. The dynamic power equation:

$$Power_{dynamic} = \alpha C V^2 f \tag{9}$$

where C is the capacitance, α is the switching activity factor, V is the voltage and f is the frequency, shows the quadratic influence of voltage on the dynamic portion of power dissipation. Because voltage also has a technology-specific influence on leakage power, we chose to develop power estimation models specific to each frequency-voltage pair. We list only frequency in the following discussion as shorthand notation for the f, V pair. Figure 4(a) illustrates the benefit of developing frequency-specific models that fit a narrower distribution of points, which significantly reduces the fitting error.

We constructed a 1-counter power model, shown in Equation 10, to describe the relationship between measured power and DPC, including the reasonably linear effect of clock throttling, t.

$$Power_{1-c} = \alpha \cdot DPC + \beta \cdot throttle + \gamma \tag{10}$$

To build the model, we applied linear regression to power and counter data collected with micro-benchmarks and determined one set of coefficients for each frequency. Table 3 lists the 1-counter model coefficients. Figure 5(a) shows the estimated power versus the corresponding measured power and indicates model fit. Lower frequencies and more-throttled settings result in less workload variation than higher frequencies and less throttled conditions. The frequency-dependent variation is reflected in the regression statistics; the regression R^2 value decreases and the error sum of squares, S_e, increases with increasing frequency, indicating that the linear model better reflects the relationship between the DPC counter and power at lower frequencies. The 1-counter linear prediction deviates from a perfect fit by 2.3 watts,16%, in the worst-case, with an average of absolute-value error of 0.31 watts.

Figure 4(b) shows that memory hierarchy usage allows further subdivision within a given frequency. From the memory-related counters available on the Pentium M, we chose DCU_MISS_OUTSTANDING (DCU), which tracks the number of cycles of outstanding misses for the data cache unit. The ratio of DCU:DPC counters gives us an indication of the memory stall time per instruction. Very small ratio values indicate accesses primarily to the L1, which allow the core to work continuously and generally consume more power. Large ratio values indicate off-chip memory accesses, when the core will be stalled a significant amount of the time and typically use less power. Intermediate values indicate accesses to the L2 cache and a corresponding intermediate power range.

The counter ratio can be used to generate a piece-wise model corresponding to the three levels of the memory hierarchy, as shown in Equation 11. Model **A** is used for samples with a counter ratio less than or equal to *threshold 1*, model **B** is for samples with ratios between thresholds, and model **C** is for samples above *threshold 2*.

$$Power_{2-c} = \alpha \cdot DPC + \beta \cdot DCU + \gamma \cdot throttle + \delta. \tag{11}$$

We used linear regression in conjunction with two threshold values to build the **A**, **B**, and **C** model components. The model employs thresholds that resulted in the minimum total error, threshold 1 = 0.08 and threshold 2 = 1.08. Table 4 lists the coefficients for Equation 11. Like the 1-counter model, the 2-counter model contains separate predictors for each frequency.

Figure 5(b) illustrates the 2-counter model applied to the micro-bench training data. Prediction trends are similar to the 1-counter model; the regression fit is better for low frequencies and low throttle levels where there is little variation among benchmarks, and the model has more error at the higher frequencies and unthrottled conditions where appli-

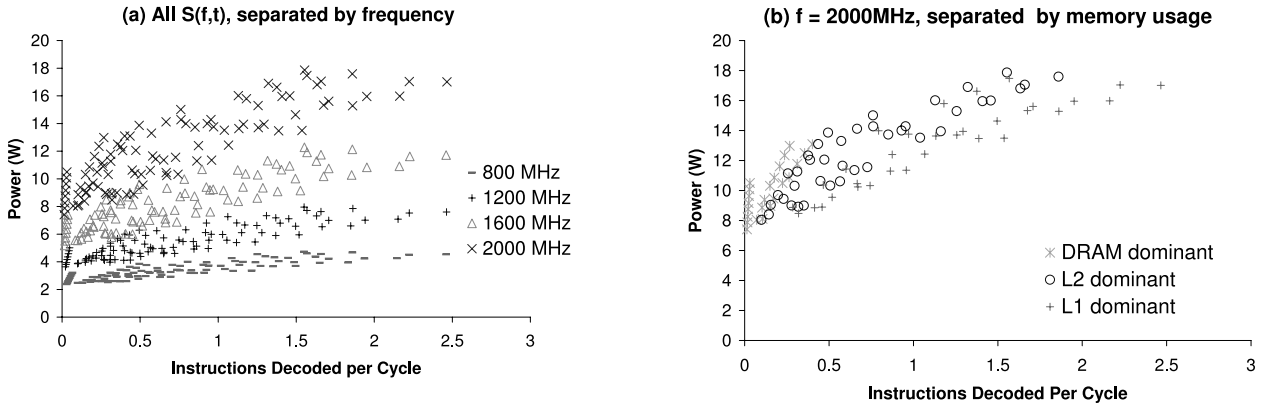
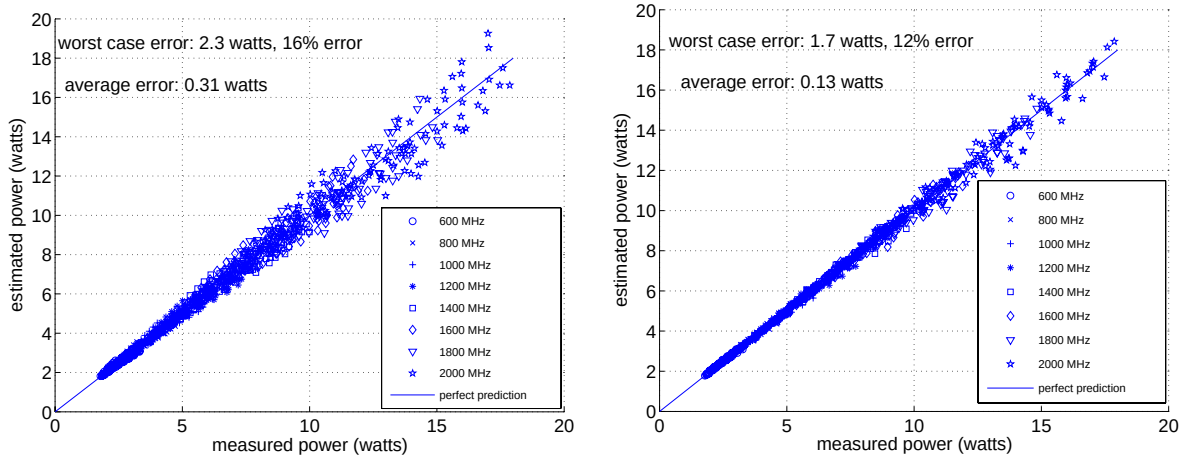


Figure 4: Power versus Decoded Instructions per Cycle



(a) 1-counter model (b) 2-counter model

Figure 5: Power models fit with MS loops training data

f	Model A (core-bound)						Model B (intermediate)						Model C (memory-bound)					
MHz	α	β	γ	δ	R^2	S_e	α	β	γ	δ	R^2	S_e	α	β	γ	δ	R^2	S_e
600	0.13	1.82	1.20	1.70	0.982	0.06	0.18	-0.20	1.36	1.69	0.996	0.03	0.65	0.028	0.64	1.67	0.986	0.03
800	0.22	2.96	1.68	2.30	0.979	0.10	0.31	-0.19	1.92	2.30	0.996	0.05	1.09	0.031	0.88	2.25	0.986	0.05
1000	0.31	4.07	2.32	2.83	0.977	0.14	0.45	-0.05	2.48	2.83	0.994	0.07	1.66	0.032	1.2	2.78	0.983	0.08
1200	0.34	5.40	3.20	3.48	0.979	0.17	0.66	0.22	2.96	3.46	0.996	0.08	2.41	0.031	1.52	3.37	0.987	0.09
1400	0.50	6.70	4.08	4.20	0.979	0.22	0.91	0.55	3.60	4.19	0.996	0.10	3.38	0.037	1.92	4.08	0.989	0.10
1600	0.59	8.37	5.12	5.11	0.978	0.30	1.26	1.05	4.08	5.07	0.994	0.15	4.60	0.043	2.40	4.83	0.990	0.13
1800	0.79	11.09	6.24	6.32	0.961	0.47	1.70	1.87	4.40	6.21	0.984	0.30	5.96	0.042	2.96	5.80	0.986	0.19
2000	0.76	13.13	7.68	7.81	0.947	0.66	2.12	2.88	4.64	7.60	0.973	0.42	7.51	0.049	3.36	7.03	0.981	0.46

Table 4: 2-Counter Model Coefficients and Statistics

cation behavior differences are more pronounced. However, the model is better able to capture workload memory behavior and the model fit degrades less at higher frequencies than the 1-counter model. Regression statistics in Table 4 show that the regression R^2 decreases slightly with frequency for models **B** and **C**. Model **A** fits the data less well than the other two components, due in part to a larger number of benchmarks that are classified as model **A** and also due to the wider variation in behavior among core-bound benchmarks than memory-bound benchmarks. Overall, the worst-case error for the two-counter model is about 1.7 watts, with an average absolute-value error of approximately 0.13 watts.

5.2 Model Evaluation

We evaluated each model by analyzing its accuracy for predicting power of the SPEC CPU2000 benchmark suite. All the SPEC benchmark programs were run with their *reference* input sets. Due to to execution time considerations, we evaluated a subset of the power management settings: 3

frequencies { f = 600 MHz, 1600 MHz, 2000 MHz}, and 3 throttle levels { t = 37%, 75%, 100%} for a total of 9 power management settings per benchmark. We collected power and performance-counter traces for each SPEC CPU2000 benchmark, then averaged their values for each 100ms of execution time to form a series of data samples for each benchmark.

For evaluation, we generated power estimates for each interval of the SPEC CPU2000 performance-counter data (for a total of nearly 200,000 samples) for the 1- and 2-counter models and compared the result with measured power data.

5.3 Results

The average absolute value errors are both in the range of the measurement error with 0.35 watts for the 1-counter model and 0.45 watts for the 2-counter model. Overall, the 1-counter model was slightly more accurate than the 2-counter model. We believe that the 2-counter model resulted in over-fitting for the MS loops data and consequently showed larger errors when evaluated with the a different (SPEC CPU2000) test data set.

AOur models have higher average error at higher frequency and throttle levels. We examine the 2000MHz case in more detail to highlight the problems associated with the larger variation when more processor resources are available. Errors for the original 2-counter model built with micro-benchmarks and applied to 2000 MHz SPEC CPU2000 benchmark data resulted in an average error of 0.95 watts Using SPEC CPU2000 data to build and evaluate a 2-counter model at 2000 MHz reduced the average error to 0.67 watts. Removing the train-test differences reduced the average error, approaching the level of measurement-error, but additional sources of error still exist.

We find that a minimal set of performance counters can provide power estimates based on monitored activity rates. The prediction accuracy is limited by several factors, including measurement error, incomplete power information captured by the restricted number of counters, and differences between test and training data sets.

Despite the constraints, the 1-counter and 2-counter static models presented here provide a low-complexity power estimate with small average error. The simple equations, small amount of state to store for coefficients, and small number of performance counters to track make these static models a practical alternative to more complex power estimates in power-management applications. We envision two orthogonal approaches to increase the accuracy of such estimation approaches: (a) increase number of and/or customized event tracking counters for even higher activity-power correlation, and (b) dynamically adapt models to current execution characteristics, using power measurement feedback. We are currently investigating the latter for potential improvements in estimation accuracies.

6. MODEL USES

Here, we briefly present prototypes of two new dynamic power management applications enabled by our models - Performance Maximizer (PM) and Power Saver (PS). The two usage scenarios are applied to the *ammp* benchmark. For both scenarios, one of the models is used to make predictions every 10ms and the system operating point $S(f, t = unthrottled)$ is changed in response to these predictions. While the prototypes in the examples below present only frequency changes, our models can obviously be used to make throttling decisions too.

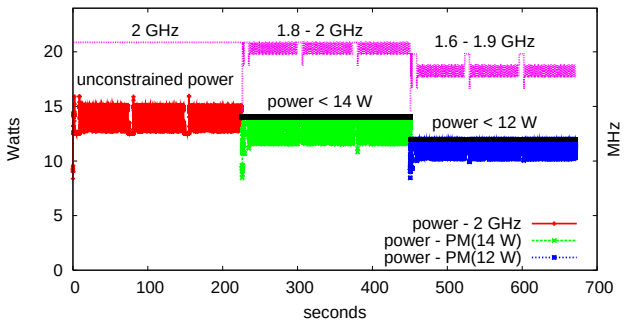


Figure 6: Performance Maximizer with *ammp*: unrestricted, 14-W limit, and 12-W limit

.

6.1 Performance Maximizer

The Performance Maximizer (PM) seeks to maximize performance within a power limit. The PM is implemented as high-priority user-level software. It monitors the DPC and DCU performance counters and uses them in the 2-counter power model to estimate power consumption at multiple frequencies. PM then dynamically chooses the highest frequency for operation that will still allow the processor to operate under the specified power limit. This application is dynamic in adapting to both changes in workload conditions as well as changes in power limits.

Figure 6 shows the PM application managing to power limits for `ammp`. The figure has three sections for different power limits, all showing both power estimates (lower half/left y-axis) and frequency (upper half/right y-axis) with time. The leftmost plot shows `ammp` running at 2 GHz when there is no power limit. The middle section shows `ammp` under a 14W limit - PM adjusts the frequency as the workload changes to always adhere to the limit, and the third plot shows how PM adapts to a still lower limit of 12-W exploiting even lower frequencies. Viewed from right to left the sections show how PM can adaptively increase performance when power limit is increased with execution time decreasing from 256.2s (12W), 234.8s (14W), 226.9s (unconstrained).

PM's ability to continuously choose the maximum frequency that meets power constraints is enabled by monitoring counter events for relevant workload information and using our power models at each interval to make the right decisions.

6.2 Power Saver

The Power Saver (PS) provides the ability to save power for a given constraint on tolerable performance loss. Implemented on the same infrastructure as PM, it monitors performance counters (IPC and DCU) each interval to assess the workload characteristics and performance (Instructions Retired per Second - IPS). The activity models are used to determine peak performance possible and to identify the frequencies that can sustain an IPS within the specified bound from the peak. The lowest frequency that can satisfy that bound is then selected because it would consume the least power.

Figure 7 shows a snapshot from the execution of `ammp`

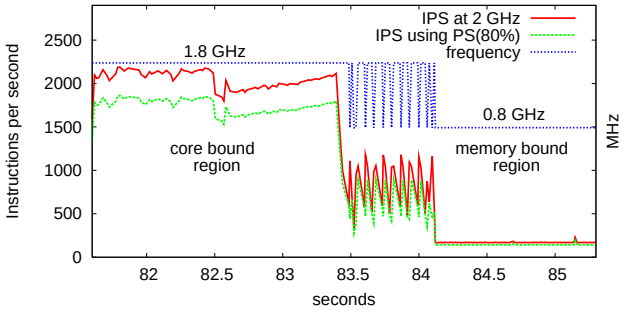


Figure 7: Power Saver with *ammp* at a performance requirement of 80% of maximum

.

when the system is allowed to reduce performance down to 80% of maximum possible performance i.e. 80% of performance at 2 GHz.

PS selects frequency with the lowest power (lowest frequency) that the performance activity models estimate will have an IPS value of at least 80% of the IPS at 2GHz. In the core-bound region shown in the figure, PS chooses 1.8 GHz to meet the performance requirement. In the middle phase, `ammp` swings quickly between compute- and memory-bound behavior, and PS responds by adjusting the frequency between 1.8 and 0.8 GHz. In the consistently memory bound region, where performance is impacted much less by lower frequencies, PM chooses 800MHz as a steady operating point. Considering the full execution, the execution under PS meets the performance constraint finishing in 275.1s compared to an unconstrained execution time of 226.9s (82% of maximum performance) at an average power of 9-W which is lower than the unconstrained execution's average power consumption of 13.5-W.

The performance activity models allow the PS application to tailor the CPU frequency to the current workload and exploit opportunities to save power while preserving performance at the required level.

7. CONCLUSIONS

In this paper, we address the estimation of power and performance across processor execution state changes used by dynamic power management solutions. Our methodology for this incorporates (a) activity-based models for power and performance across different $S(f,t)$, and (b) run time monitoring of required activity rates through widely available processor performance counters. Because of the low-overheads for counter access and prevalence of the mechanism across different processor architectures our approach is widely applicable and easily portable.

We develop models to estimate power and performance from monitored activity rates for this purpose. Our effort is distinguished from previous work in focusing on models suitable for online deployment – we can apply our estimation solution using a single observation of involved activity rates. Further our prediction models work across multiple power-management mechanisms, more specifically the two most prevalent ones, dynamic voltage and frequency scaling and clock throttling.

We prototype our approach with an Intel Pentium M processor-based system and validate it using an experimental approach keyed towards assessing the model effectiveness for real-time power management. We identify the activity rates to be monitored for power and performance estimation on this platform as IPC, DPC, and DCU. Developed using a custom suite of simple micro-benchmarks our models are validated using the SPEC CPU2000 benchmarks suite. We achieve reasonable prediction accuracy for power and performance. Further, we demonstrate how our models enable new adaptive power management solutions presenting prototype implementations of two new dynamic power management applications.

8. REFERENCES

[1] C. Lichtenau, M. Recktenwald, T. Pflueger, R. Hilgendorf, and P. A. Sandon, "Powertune: An energy-efficient high performance multi-processor powerpc system architecture," in *ACEED*, 2003.

[2] D. Bodas, "New server power-management technologies address power and cooling challenges," *Intel Magazine*, Sept. 2003.

[3] C. Poirier, R. McGowen, C. Bostak, and S. Naffzigr, "Power and temperature control on a 90nm itanium-family processor," in *IEEE International Solid-States Circuits Conference 2005*, March-April 2005.

[4] Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, "Advanced configuration and power interface specification (ACPI) revision 2.0b." http://www.acpi.info/DOWNLOADS/ACPIspec-2-0b.pdf, Oct. 2002.

[5] Intel Corp, "Enhanced Intel SpeedStep technology." http://support.intel.com/support/processors/mobile/pm/sb/CS-007981.htm, Jan. 2006.

[6] Advanced Micro Devices, "PowerNow with optimized power management." http://www.amd.com/us-en/0,,3715_12353,00.html, Jan. 2006.

[7] K. Flautner and T. Mudge, "Vertigo: Automatic Performance-Setting for Linux," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 105–116, December 2002.

[8] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: Understanding the runtime effects of frequency scaling," in *Proceedings of the International Conference on Supercomputing (ICS)*, 2002.

[9] B. Brock and K. Rajamani, "Dynamic power management for embedded systems." IEEE International SOC Conference, September 2003.

[10] S. Ghiasi, T. W. Keller, and F. L. Rawson, "Scheduling for heterogeneous processors in server systems," in *Proceedings of the International Conference on Computing Frontiers (CF 2005)*, May 2005.

[11] F. Bellosa, "The benefits of event-driven energy accounting in power-sensitive systems," in *ACM SIGOPS European Workshop*, October 2000.

[12] A. Weissel and F. Bellosa, "Process cruise control: Event-driven clock scaling for dynamic power management," in *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES 2002)*, pp. 238–246, October 2002.

[13] G. Contreras and M. Martonosi, "Power prediction of intel xscale processors using performance monitoring unit events," in *2005 International Symposium on Low Power Electronics and Design*, August 2005.

[14] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunitites," in *International Symposium on Low Power Electronics and Design (ISLPED)*, August 2005.

[15] Radisys Corporation, "Endura LS855 Product Data Sheet." http://www.radisys.com/oem_products/ds-page.cfm?productdatasheetsid=1158, Oct. 10 2004.

[16] A. Petitet, C. Whaley, J. Dongarra, and A. Cleary, "HPL - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," tech. rep., University of Tennessee, Jan. 20 2004.