RC24222 (W0703-100) March 21, 2007
Computer Science

IBM Research Report

Application-Aware Power Management

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

*Also with The University of Texas at Austin

Application-Aware Power Management

Karthick Rajamani*, Heather Hanson†*, Juan Rubio*, Soraya Ghiasi* and Freeman Rawson*
*IBM Austin Research Lab
† The University of Texas at Austin
{karthick,rubioj,sghiasi,frawson}@us.ibm.com, hhanson@cs.utexas.edu

Abstract— **This paper presents our approach for application-aware power management. We combine continuous monitoring of critical workload indicators, online power and performance model usage and timely p-state control to realize application-aware power management. We present two new power management solutions enabled by our methodology: PerformanceMaximizer (PM) finds the best possible performance under specified power constraints and PowerSave (PS) saves energy while keeping performance above specified requirements. We evaluate both using the SPEC-CPU2000 suite on a Pentium M platform discussing implications of workload characteristics and benefits of being workload-aware.**

I. 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.

Techniques like dynamic voltage and frequency scaling (DVFS) provide a way of regulating the power consumption of microprocessors and other circuits at the cost of altering system performance. Once proposed for battery-operated environments, power management is now in wide use, including PowerTune in Apple's IBM PowerPC 970-based systems [1], AMD's PowerNow [2], Intel's DBS (demand-based switching) for server power management [3], and Linux's `cpufreq`-based drivers and Windows' power management which exploit ACPI-defined processor p-states. However, their usage is primarily limited to saving energy, reducing power consumption when systems are under-utilized.

Power-efficient techniques such as clock gating lead to variable power consumption characteristics across workloads, even at fixed frequencies. Figure 1 presents measured power samples through time for the SPEC CPU2000 benchmark suite executing on a Pentium M processor at 2GHz, with a constant system-perceived load of 100%. It shows the significant differences in measured power consumption – the range spans over 35% of the chip's peak operating power. Figure 2 shows the converse picture – the performance impact of using different DVFS-based ACPI-defined *p-states* for the Pentium M. The memory-bound `swim` benchmark shows minimal performance impact from p-state changes between 1600, 1800, and 2000 MHz processor frequencies. At the other end, compute-bound `sixtrack's` performance scales linearly with frequency. `gap` illustrates a workload in-between the extremes. Thus, power-management solutions should not be workload-agnostic; rather, they should exploit the causes of power variation for more performance-aware, advanced power management.

Our perspective is that understanding application characteristics is important for designing and enabling intelligent power management techniques. Further, a wider set of power management capabilities are enabled by the ability to continuously monitor workload behavior and adapt power management decisions to workload characteristics. To enable this, power management needs the ability to assess, in real-time, the impact on both performance and power of any action it can employ at a given moment. Knowledge of the quantitative impact of these actions can best guide which actuation and to what extent the actuation should be engaged to address the given situation.

To do this, we developed an application-aware power management methodology incorporating processor performance counters for monitoring critical workload characteristics, event-counter-based power and performance projection models, and low-overhead DVFS-based p-state change mechanisms. Guiding principles for our methodology were ease of incorporation in software solutions at the kernel- or user-level and low run-time overheads.

We introduce two new power-management solutions enabled by our methodology, PerformanceMaximizer (**PM**) and PowerSave (**PS**). They capitalize on the information provided by power and performance models to tailor dynamic p-state adaptation to workload behavior and system constraints. PM uses power projections for the currently executing workload to increase performance while operating within explicitly specified power constraints. PS applies performance projections across p-state changes to save energy for the executing workload while operating above explicitly specified performance levels.

This paper makes the following contributions:

- It demonstrates the need for and effectiveness of application-aware power management.
- It introduces our methodology for this which incorporates online power and performance models for multiple p-states.
- It presents two new microprocessor power management solutions enabled by our methodology.
- It evaluates these solutions analyzing their adaptability to application characteristics and the role of application characteristics in the obtained benefits.

Section III presents our methodology and infrastructure for prototype development and evaluation. Section IV presents the

PerformanceMaximizer and PowerSave solutions, their implementation and evaluation, and discusses workload characteristics in the SPEC CPU2000 suite that influence the solutions' effectiveness. Finally, we conclude with Section V.

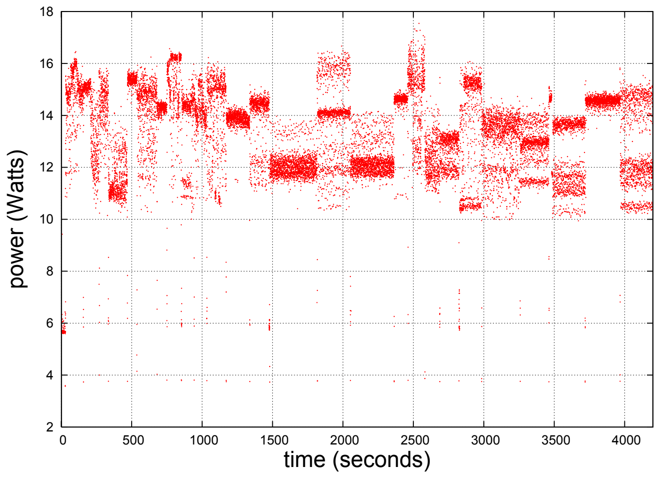

Fig. 1. Power variation for the SPEC CPU2000 benchmark suite executing at 2GHz on a Pentium M.

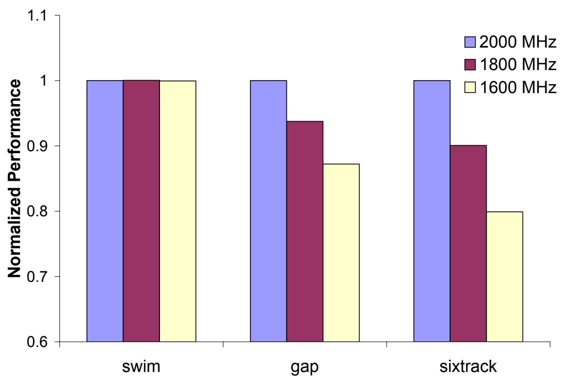

Fig. 2. Workload-specific performance impact across three p-states.

II. Related Work

Industrial standards such as Advanced Power Management(APM) and the Advanced Configuration and Power Interface (ACPI) [4], define both active (*p-states*) and standby power management states for devices and processors. Multiple p-states are available for many commercial processors, such as dynamic voltage and frequency (DVFS) states used in SpeedStep [5] and PowerNow [2]. Several run-time control techniques modulate p-states in response to changes in available power supply and cooling resources or application behavior. Two critical management objectives for high-performance systems are (1) maximizing performance under power constraints and (2) minimizing power while providing acceptable performance.

Intel's Foxton Technology provides sophisticated closed-loop control to select the maximum processor frequency within programmable power and thermal envelopes [6] for a processor chip; at the system level, the Automatic Control of Power Consumption (ACPC) solution prescribes a system power feedback control mechanism to enforce user-defined power limits [3]. Both require new hardware investments for real-time power measurements feedback and closed-loop control unlike our counter-based predictive approach. Another context in which performance maximization under power constraints has been studied is with a power budget shared among multiple components – Felter, *et al.* [7] study this for a processor and memory system with shared supply/cooling and Isci, *et al.* for a multi-core chip with a chip-level power budget [8].

Given the potential non-linear power cost of high performance, meeting quality-of-service agreements and other performance expectations without squandering power resources is of paramount importance to large-scale systems. Kotla, *et al.*, use dynamic program classification and scheduling techniques to adjust the frequency and voltage of a system to minimize the power consumption while maintaining existing performance levels [9]. Demand-Based Switching and many other techniques capitalize on under-utilized components or schedule slack to reduce power [3], [10], [11] without excessive performance degradation. PowerSave (PS) takes this one step further by providing the user the ability to choose an explicit performance trade-off for desired power savings.

Performance-monitoring counters provide run-time visibility into processor and memory system activity, enabling intelligent decisions for choosing p-states to meet management objectives. Lee and Brooks [12] use statistical methods to develop power and performance models from a set of predictors including data captured by performance counters to simulate the effect of a large range of microarchitectural parameters. They found that performance models were more accurate when tailored to a specific application (benchmark); our objective is a model suitable for a wide range of applications, tailored to a specific platform.

Empirically derived counter-based power and energy models have been developed for several platforms. Bellosa, *et al.* pioneered the development of energy models from performance counters for XScale architecture [13], [14]. Contreras, *et al.* develop linear equations for power, including memory power, for the XScale PXA-255 processor for several DVFS p-states [15] and Isci, *et al.* rotate 24 event types through 15 physical counters to build a per-component power model based on observed activity for the Pentium 4 processor [16]. Bircher notes the importance of capturing speculative behavior, not simply events from completed instructions, for accurate power models [17] for the Pentium 4.

Our models are distinct from prior art in the following ways:

- We predict the impact of **p-state changes** for power and performance, unlike much of prior work which predicts only for the current p-state.
- Prior power model evaluations focused on program-average power prediction accuracy, where over- and under-estimates would compensate for better overall accuracy.

We focus on per-sample accuracy for tighter run-time control.

- We use a small number of performance counters, a feasible solution for future systems with counters dedicated to power management (as well as many current systems with a small set of general-purpose counters).
- Counter-based models are inherently platform-specific, and our models are the first of which we are aware for the Pentium M system.

Power and performance estimations across p-state changes allow judicious use of p-state control mechanisms by considering their predicted effect on the system, without the overhead of test-and-set or other exploratory approaches. Anticipating the system's reaction to a p-state change is especially useful to fine-tune p-states to rapidly changing program behavior.

Finally, in terms of overall evaluation approach, the work of Wu, *et al.* [18], is somewhat similar to ours in that they use a similar hardware platform, not simulation, and have a measurement set-up like our own. However, they propose a dynamic compilation technique for scheduling DVFS changes – their approach is applicable for single applications. Our solutions are implemented adapting to monitored behavior at the system level. We see their work as complementary to our approach.

III. Methodology for Application-Aware Power Management

As shown in section I, workloads' individual characteristics lead to varied power consumption and impact from power management decisions. Our philosophy for application-aware power management is that a dynamic, adaptive power management infrastructure is needed to continuously monitor workload characteristics and adapt to them to obtain the best results within the dictated constraints. Consequently, we employ a three-phased process depicted in figure 3:

- Monitor: continuously measure information to understand present workload behavior.
- Estimate/Predict: determine the impact of taking specific power management actions. The monitored characteristics combined with estimation models predict the power consumption and performance impact for each power management action.
- Control: from the projected impact on power and performance, choose the most appropriate p-state to meet the constraints and enable the actuator(s).

The focus of this work is application-aware power management for microprocessors. In this study, we use the ACPI-compliant processor p-states (voltage and frequency pairs) to exercise power management decisions. For monitoring workload characteristics, we opt for a low-overhead approach of monitoring processor performance counters, also known as performance monitoring counters (PMC) and hardware performance counters (HPC). We identify suitable counters and develop models to predict processor power for our prototype platform based on observed processor activity. We also develop performance projection models from counters to estimate performance impact across p-state changes.

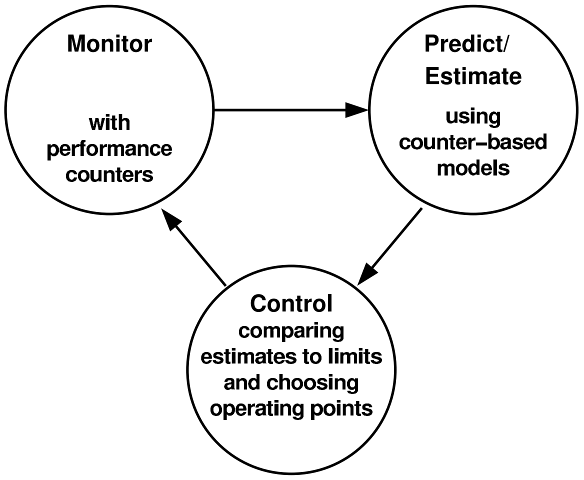
Monitor
with performance counters
Predict/ Estimate
using counter–based models
Control
comparing estimates to limits and choosing operating points

Fig. 3. Integral Elements of Application-aware Power Management Methodology

A. Online Estimation of Power and Performance

We exploit performance counters for online power and performance estimation. Counters have low overhead for frequent monitoring and multiple researchers have shown the feasibility of performance-counter events as suitable proxies for processor power consumption.

Table I summarizes the four loops used for this study. All four perform simple array access operations. The loops are configured for multiple data footprints to intensively exercise each of the memory hierarchy levels (L1 and L2 on-chip caches, and off-chip DRAM main memory). Our training data set consists of 12 data points (4 loops, 3 footprints each) per p-state setting. The advantage of using small, well-defined loops is that their performance and power characteristics are relatively stable during the course of a run and across multiple runs. The microbenchmarks execute at the highest real-time priority mode to minimize external interference during characterization.

1) Power Model: The power model estimates power consumption for multiple p-states based on the `Decoded Instructions` counter per cycle (DPC) which we found to be fairly well-correlated with measured power. As noted in [17], it is important to capture speculative instruction activity in the processor; typical performance rates using completed instructions do not reflect the variation in power consumption due to speculative activity. Tracking decoded instructions allows a more complete picture of power in the processor pipeline. The power estimation model is constructed as a linear fit of measured DPC, minimizing the absolute-value error between the measured power and estimated power. We develop a distinct equation for each p-state because the system operating frequency and supply voltage have a significant impact on power. The dynamic power for a CMOS circuit is given by

$$Power_{dynamic} = \alpha C V^2 f \tag{1}$$

Micro-benchmark	Description
DAXPY	The daxpy function from the Linpack benchmark [19]. 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 I

MICRO-BENCHMARKS USED TO STUDY CHARACTERISTICS AND AS TRAINING SET FOR THE MODELS.

frequency (MHz)	voltage (Volts)	α	β
600	0.998	0.34	2.58
800	1.052	0.54	3.56
1000	1.100	0.77	4.49
1200	1.148	1.06	5.60
1400	1.196	1.42	6.95
1600	1.244	1.82	8.44
1800	1.292	2.36	10.18
2000	1.340	2.93	12.11

TABLE II

DPC-BASED POWER MODEL FOR EACH P-STATE

where C is the capacitance, α is the switching activity factor, V is the supply voltage and f is the frequency, illustrating the non-linear influence of voltage-frequency pairs on dynamic power dissipation. Leakage power, although frequency independent, does depend on supply voltage. Model coefficient values for each p-state are shown in Table II.

$$Power = \alpha \cdot DPC + \beta \tag{2}$$

These coefficients capture representative behavior of the microbenchmark suite. Consequently, there will be errors in the estimates for workloads that exhibit very different characteristics. The results for the PerformanceMaximizer (PM) solution show the effectiveness of this model applied to the SPEC CPU2000 suite.

2) Performance Model: For performance estimation, we track the impact of each p-state on the rate of `Instructions Retired` (IPC) with MS-Loops microbenchmarks. A linear model will not describe all workloads due to the differences in performance response to p-state changes, as shown in Figure 2. To simplify the performance estimation, we group workloads into core-bound and memory-bound classes, each with a distinct IPC-estimation formula as shown in equation 3.

$$IPC' = \begin{cases} IPC, & DCU/IPC < 1.21(core) \\ IPC \cdot (f/f')^{.81}, & DCU/IPC \geq 1.21(memory) \end{cases} \tag{3}$$

In this set of equations, the $'$ indicates a a different p-state. DCU is the per-cycle rate of the `Data Cache Unit Miss Outstanding` counter, which tracks the number of cycles in which the DL1 cache waits for data after a cache miss. The ratio of `DCU/IPC`, the number of cycles stalled in DL1 cache per completed instruction, is an approximate measure of the workload's memory-boundedness, and serves to distinguish core-bound and memory-bound regions of the equation. The threshold (1.21) to classify memory-boundedness and the frequency-dependence exponent (0.81) were obtained by optimizing parameters to minimize errors with the microbenchmark training data set. A detailed discussion of the model development and evaluation for power and performance impact estimation for both DVFS and clock throttling power-management mechanisms is available in [20].

B. Infrastructure

The experimental infrastructure for this study uses a Pentium M 755 (90 nm Dothan) with Enhanced SpeedStep, which supports dynamic voltage and frequency scaling with frequencies and corresponding voltage levels shown in Table II.

A Radisys system board [21] with high-precision sense resistors between the processor and each of two voltage regulators is used to measure the power consumption of the processor. Figure 4 shows a diagram of the configuration. The current, calculated from measured voltage across the sense resistor, and the supply voltage are filtered, amplified, digitized and collected periodically with a National Instruments SCXI-1125 module and National Instruments PCI-6052E DAQ card. The power measurement system is non-intrusive and has a peak sampling capability of 333 K samples/s, more than adequate for the 10 ms sampling intervals in this study.

We developed low-overhead drivers for Linux and Windows XP to monitor performance counters and change fre-

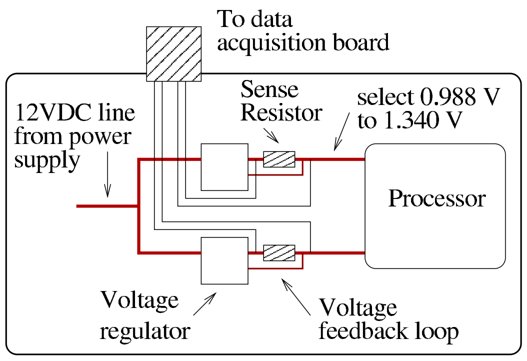
To data acquisition board
12VDC line from power supply
Sense Resistor
select 0.988 V to 1.340 V
Processor
Voltage regulator
Voltage feedback loop

Fig. 4. Experimental platform: system under test with sense resistors and data acquisition probes to measure processor power.

quency and voltage levels. Frequency and voltage changes are done by configuring the machine specific registers that control the internal PLL of the processor and the external voltage identification signals that control the voltage regulator. The Pentium M processor has two performance counters that can be configured to monitor any of 92 different events. The monitoring driver collects the counters every 10 ms with negligible performance impact. Additionally, we use a 3.3 V GPIO signal at appropriate points (start and end of each benchmark run) to the power data collection system to synchronize the application execution with the data collected by the measurement system. The software stack is controlled by a user-level application that accesses the drivers to monitor the behavior of the workload and adjust the frequency and voltage settings according to power and performance requirements.

IV. Application-Aware Power-Management Solutions

We enable two new power management solutions with our application-aware power management methodology – PerformanceMaximizer (PM) and PowerSave (PS). Evaluations for both solutions are performed with SPEC CPU2000 applications on a Pentium M system. To account for the variability in workload execution times, we employ the standard SPEC approach of executing three times and reporting data from the run with the median execution time.

A. PerformanceMaximizer

The PerformanceMaximizer (PM) exploits DVFS levels to maximize performance while ensuring power consumption is within a set limit. Conventional approach to system design required that power supply and cooling solutions be adequately provisioned to meet the 'realistic' worst-case consumption at the maximum operating frequency. With increasing constraints on over-provisioning power and cooling solutions, a processor with *static clocking* would have to adopt a low fixed *nominal* frequency such that the worst-case 'realistic' workload would run safely within power and cooling constraints. For a given power constraint, PM with its *dynamic clocking* approach can provide higher performance for most real workloads by exploiting power slack between the executing workload and the worst-case workload.

PM takes advantage of workload characteristics, boosting frequency when the workload's characteristics permit and lowering frequency before violating power limits. Our solution is useful for a number of situations: (i) controlling multiple components with shared power supply/cooling resources, (ii) providing flexible system design options to meet power-supply restrictions (iii) continuing operation with maximal (but safe) performance in the event of partial supply/cooling failures.

1) Implementation: The PM prototype is implemented as high-priority user-level software and follows the three-phased process of

- **monitoring** DPC (decoded instructions per cycle) every 10 ms.
- **predicting** DPC at other p-states using equation 4, then applying the power model described in Section III to estimate power consumption at all p-states,
- **controlling** power consumption by comparing the estimated power consumption at each p-state to the current power limit, choosing the highest frequency with estimated power consumption below the power limit.

The PM prototype can receive a new power limit at any instant (implemented as a Unix signal SIGUSR1 or SIGUSR2 delivered to the process), effective immediately. Thus, PM can adapt to both changes in workloads and power limits. Based on earlier studies with this model, we add a 0.5 W guardband to the estimated power to accommodate model inaccuracies and system variability. PM enforces power constraints at 100ms intervals using a moving window of ten, 10ms samples. The DVFS controller within the prototype software lowers frequencies immediately when a single 10ms sample indicates a need, but waits for 100ms worth of consecutive samples that indicate frequency may be raised before doing so to minimize power-limit violations during difficult-to-predict periods of workload behavior.

$$DPC(f') = \begin{cases} DPC(f) \cdot (f/f') & f' \leq f \\ DPC(f) & f' > f \end{cases} \tag{4}$$

Figure 5 shows PM controlling the `ammp` benchmark with unconstrained 2 GHz operation and power limits of 14.5 W and 10.5 W. Note that the frequency modulates according to workload demands.

2) Evaluation of PM: We evaluate PM with the SPEC CPU2000 suite and 8 different power limits, from 17.5 W to 10.5 W in 1-W steps, for both adherence to power constraints and for boosting performance over static clocking for a given power limit. We evaluate the effectiveness of PM to meet the power constraint over 100ms windows (moving average of ten 10 ms samples). PM is able to enforce the power limit for every benchmark except `galgel`, which in the worst case, spends approximately 10% of run-time over the power limit (for the 13.5W limit). We observed that `galgel` peaks at 16.6 W for the 100 ms moving-average window, exceeding 18 W in individual 10 ms samples (the highest of all SPEC benchmarks),

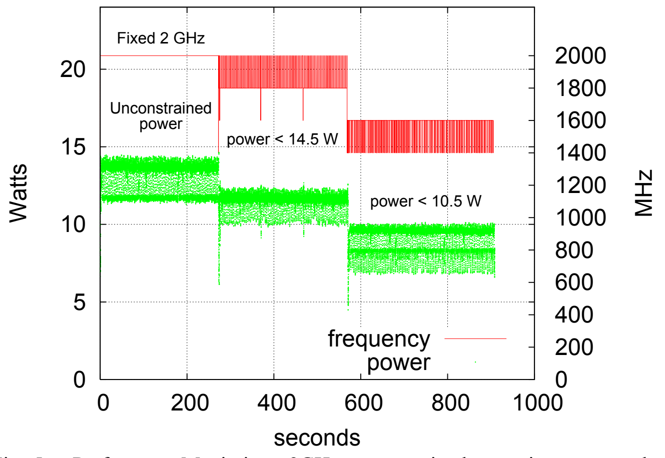

Fig. 5. PerformanceMaximizer: 2GHz unconstrained operation compared to PM with two power limits. `ammp` runs to completion in each case.

Frequency (MHz)	FMA-256KB Measured Power (W)
600	3.86
800	5.21
1000	6.56
1200	8.16
1400	10.16
1600	12.46
1800	15.29
2000	17.78

TABLE III

POWER VS FREQUENCY FOR WORST-CASE WORKLOAD

Power Limit (W)	Static Frequency (MHz)
17.5 W	1800
16.5 W	1800
15.5 W	1800
14.5 W	1600
13.5 W	1600
12.5 W	1600
11.5 W	1400
10.5 W	1400

TABLE IV

POWER-LIMIT DETERMINED STATIC FREQUENCIES

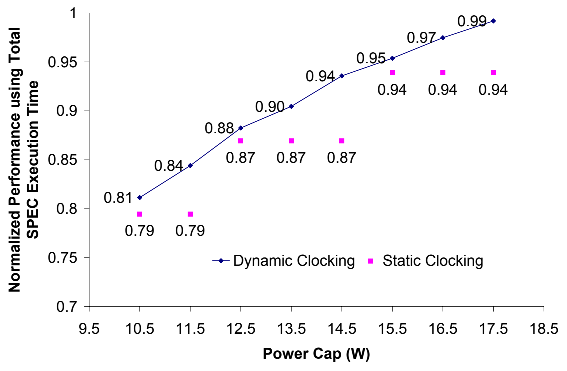

Fig. 6. Performance versus Power Limits for PM's Dynamic Clocking and Static Clocking

with bursty behavior between low and peak power levels. We are currently exploring ways to incorporate measured power feedback into the PM control algorithm – PM could adapt model coefficients on the fly or scale measured power for *p-state* changes – to address workloads like `galgel` that are difficult to predict with the static model.

The key benefit of PM's dynamic clocking approach is improved performance compared to a worst-case based static frequency setting. We evaluate this by comparing the performance for the SPEC CPU2000 suite for different power limits for PM and static clocking. To determine an appropriate frequency setting for static clocking, we use the L2-resident *FMA* workload – the highest power microbenchmark in MS-Loops – as a proxy for the worst-case power. Table III shows the power consumption for FMA-256KB at multiple frequency settings; conversely, Table IV shows the consequent static frequency choices for multiple power limit settings (frequencies represent p-states of voltage-frequency pairs).

Figure 6 shows the performance benefit of the dynamic clocking capability of PM compared to conventional static clocking within a power limit. Normalized performance is computed as the total execution time without power constraints divided by the total execution time with the power constraint. The line shows dynamic clocking performance and the dots show static clocking performance, indicating that static clocking performance approaches dynamic clocking performance only when the power limit is near the peak power consumption of a fixed frequency. Note that the static clocking approach is not practical if the power limit were to change during runtime.

Figure 7 shows the speedup for PM compared to static clocking for SPEC CPU2000 benchmarks for a 17.5 W power limit. At this limit, the static clocking approach uses a fixed frequency of 1800 MHz to enforce the power limit. The maximum possible performance would be continuous operation at 2000 MHz, which would violate the power limit. PM alternates between 1800 and 2000 MHz, adjusting the frequency and voltage based on workload behavior to meet the power limit, reaching 86% of maximum performance based on the total execution time of the full benchmark suite. In the figure, the performance bar for each benchmark is composed of two segments: the speedup of PM compared to static clocking at a 17.5 W limit, and the performance gap between PM at 17.5 W and the maximum performance with unconstrained power, 2000 MHz. Negative-speedup segments indicate cases where increasing frequency beyond 1800 MHz with either PM or unconstrained operation had negligible effect on performance and the natural variation in execution-time dominated the speedup calculations.

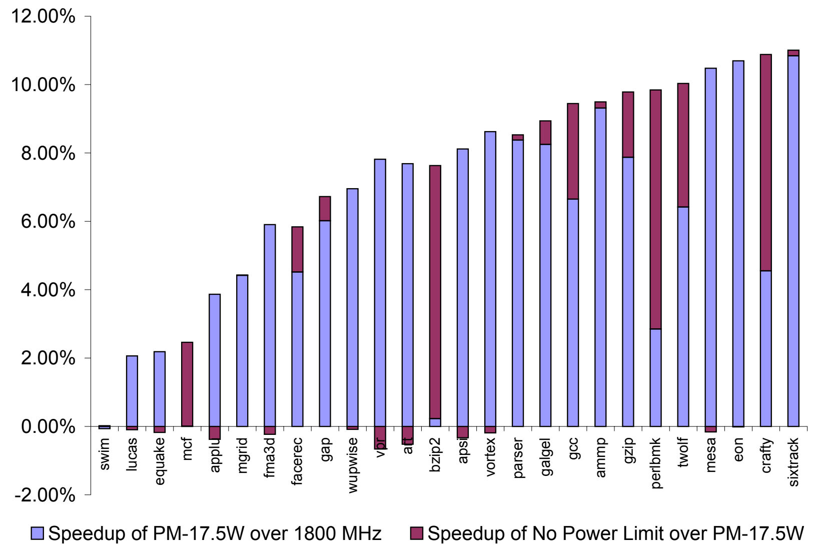

Fig. 7. PM Speedup and Unconstrained Speedup (2000 MHz) over Constrained Static Clocking (fixed 1800 MHz) for 17.5W limit

Workload characteristics determine the extent that increasing frequency from 1800 MHz to 2000 MHz improves performance. Memory access times are independent of processor clock frequencies, for example, so execution times for memory-bound benchmarks do not necessarily improve with higher frequencies. In the graph, workloads are sorted from left to right in the order of increasing speedup of 2000 MHz compared to 1800 MHz execution times.

At the left, `swim` does not noticeably change in execution time despite a 10% frequency difference; at the other extreme, `sixtrack` benefits fully from the increased clock frequency. Workloads throughout the spectrum exhibit a range of frequency-dependent behavior. The reasons for performance differences can be traced to the underlying workload characteristics:

- `swim`, `lucas`, `equake`, `mcf`, `applu`, and `art` exhibit relatively high `DCU Miss Outstanding cycles/cycle` and/or `Resource Stalls/cycle`. Further they exhibit relatively high `Memory Requests/cycle` indicating that their DCU stalls are from waiting for DRAM (and not L2 accesses, which would scale with processor frequency).
- `perlbmk`, `mesa`, `eon`, `crafty`, and `sixtrack` all have low rates of `DCU stalls`, `Resource Stalls` and `Memory Requests`, which indicates that spend less time waiting for memory accesses, and can take more advantage of a faster core frequency.
- `crafty` and `perlbmk` have the highest average power in the SPEC workloads, followed by `galgel`. While their performance could benefit from higher frequencies, their relatively high processor activity (have both high `Instructions Decoded` rates and `L2 Request` rates) cause high power consumption, which requires lower frequencies to meet the power limit. This is also applicable to `bzip2`, which has slightly lower performance improvement and slightly lower power.

The results show that workload-aware PM obtains a significant fraction of the possible performance improvement. The results also indicate that core-bound doesn't equate to power-limited, so the lower-power core-bound workloads reap the maximum benefits from the PM approach.

B. *PowerSave*

The goal of the PowerSave (PS) solution is to provide energy savings while maintaining performance within specified limits. Most popular power savings solutions exploiting DVFS like *Demand-Based Switching* [3] save energy by reducing frequencies during periods of low system load with a goal of minimizing the performance impact. With power and energy considerations playing a larger role in system management, saving energy only during low utilization is insufficient and squanders energy resources on low-priority workloads. PowerSave takes energy-saving to the next logical step – it conserves energy even at full load by relaxing performance as permitted to a minimum *performance floor* or a *cap in the performance reduction*.

1) Implementation: PowerSave (PS) provides a performance-loss bounding capability when exploiting DVFS by tracking the reduction in completed instruction throughput, which is a suitable proxy for performance for most real workloads. The PS prototype is implemented as a high-priority user-level software. It follows the three phases:

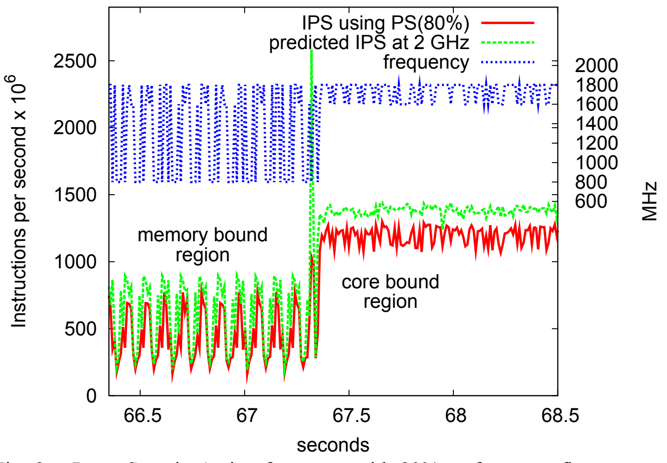

Fig. 8. PowerSave in Action for ammp with 80% performance floor

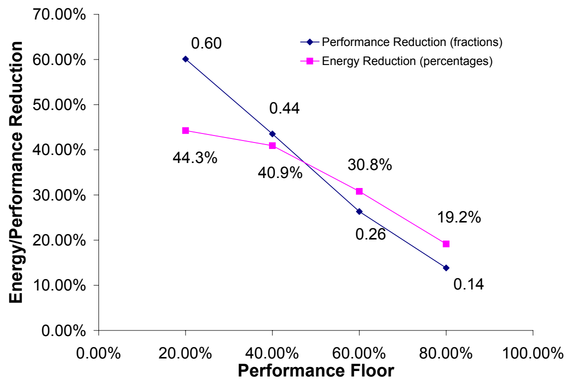

Fig. 9. Performance Reduction and Energy Savings versus PS Performance Floor for SPEC CPU 2000

- **monitor** IPC (Retired Instructions per cycle) and the DCU (Data Cache Unit Miss Outstanding Cycles per cycle) via event counters every 10ms
- **estimate/predict** IPC and DCU for all p-states
- **control** choose the lowest frequency that will meet performance requirements.

Figure 8 shows a segment of the ammp benchmark executing with PowerSave set to 80%. Note that the PS algorithm is able to reduce the frequency during memory-bound regions while supporting the 80% of peak performance requirement.

2) Evaluation of PS: We evaluated PS with the SPEC CPU2000 suite using 4 different *performance floor* constraints – 80%, 60%, 40%, and 20%, where the constraint indicates the minimum acceptable performance compared to peak. An 80% constraint indicates the performance *loss* should be no larger than 20%. We also measured the suite at maximum and minimum frequencies for peak performance and the upper bound on energy savings. Figure 9 shows PS's primary benefit – energy savings within allowed performance trade-off. Performance reduction is computed from the increase in total SPEC CPU2000 execution time compared to running at full-speed, 2000 MHz. Energy is computed by summing energy values computed from each 10ms power sample; energy savings are relative to full-speed execution. First, we observe that PS meets performance-floor requirements, *e.g.*, at the 60% floor, the performance loss is only 30.8%, lower than the allowed 40% reduction. Note that because of discretized p-states it is not possible to exactly reach the performance floor, using the next lower frequency would push the performance below the floor.

Figure 10 shows energy savings for individual workloads sorted by the maximum benefit the workload can obtain with DVFS (at 600 MHz). The ALLBENCH point corresponds to the savings across all the benchmarks separating the workloads with above-average savings on the left and below-average savings on the right. We can now relate the extent of savings for the workloads to their underlying characteristics. At the right extreme are core-bound workloads whose performance greatly benefits from higher frequencies, like eon, sixtrack, crafty, twolf and mesa. PS lowers the frequency the least for these applications. For memory-bound workloads like swim, equake, mcf, lucas and applu, PS can lower the frequency more with only slight performance degradation, for a larger energy savings.

Figure 11 shows the performance reduction by individual workloads, sorted by maximum reduction at 600MHz, the minimum-frequency p-state. The ALLBENCH point separates the workloads with above-average reduction on the right and below-average reductions on the left – nearly duplicating the behavior for energy savings. The memory-bound applications on the left display the least performance reduction and the core-bound applications to the right exhibit the most performance reduction. Examining this data in detail, we find that art and mcf violate the performance constraint significantly. At the 80% setting art has a 42.2% reduction and mcf nearly 27.7% reduction, both exceeding the desired maximum 20% loss. At the 60% setting art has 54.3% reduction that exceeds the desired maximum reduction of 40%.

We determined the cause for these violations to be errors in IPC estimation. The model based on MS loops training set was heavily influenced by both strongly core-bound and strongly memory-bound workloads, with sparse data points between the extremes. Consequently, workloads in the 'in-between' region have greater error. We re-calculated the results with a modified equation, using a value of 0.59 rather than 0.81 (both were local minima for error during the model creation) in equation 3, and found that PS was able to better support the performance requirements. mcf's reduction at 80% became 17.9% (within the desired 20%) and art's became closer, with 26.3% and 48.3% performance loss for the 20% and 40% loss tolerances, respectively.

C. Summary

In prototyping PerformanceMaximizer and PowerSave, we observed the following:

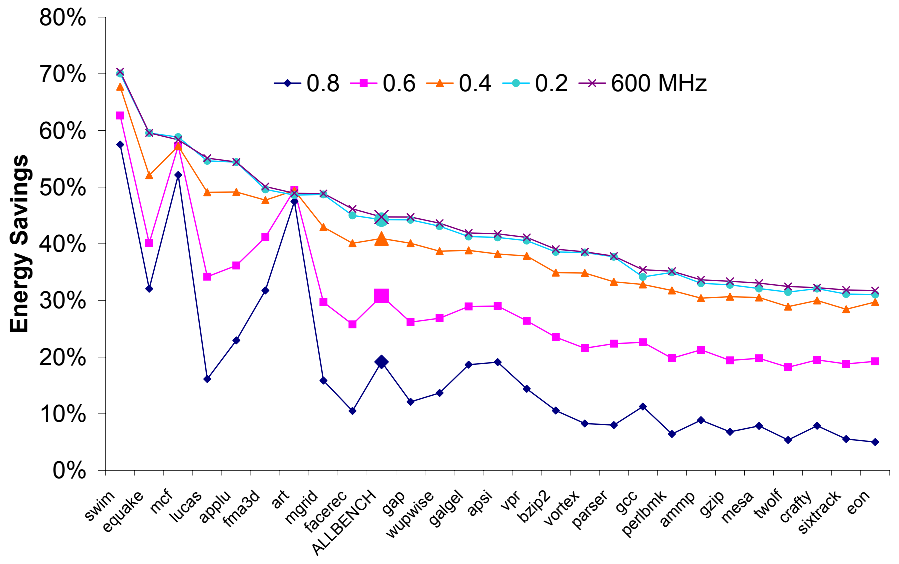

Fig. 10. Energy Savings for each CPU 2000 workload with PS Performance Floor Setting

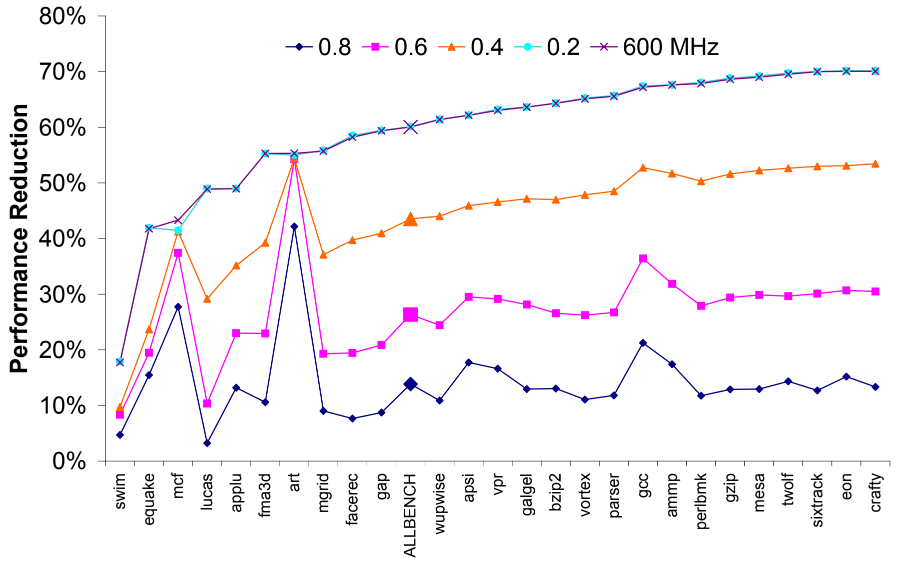

Fig. 11. Performance Reduction for each CPU 2000 workload with PS Performance Floor Setting

- Powerful, new power management capabilities can be enabled even with simple, relatively non-intrusive approaches to application-awareness.
- The extent of benefits are application-dependent, and continuous adaptation to easily monitored workload characteristics allows PM and PS to achieve user-specified

constraints on power and performance.

- While our simple models for power and activity estimation appear to work well in practice, we expect additional refinements could further improve both.

V. Conclusions

Power and energy concerns are now on par with performance concerns for computer systems design and management. As computer system designs evolve towards high-efficiency from purely high-performance, the differences in workload characteristics show up as increasing variations in their power consumption. Further, systems adopt active power management mechanisms like DVFS whose performance impact is determined by workload characteristics. This paper recognizes that the time for application-aware approaches to power management has arrived.

We present a practical methodology for implementing application-aware power management solutions. At its core is a three-phased approach incorporating continual monitoring, prediction/estimation, and control. We build on prior research that have shown the usefulness of performance counters for power estimation developing it for runtime power prediction. To this we add a runtime performance estimation approach giving us the ability to infer, at runtime, both power and performance impact of power management decisions based on monitored workload characteristics.

We present two new power management solutions enabled by our approach. Our PerformanceMaximizer (PM) solution manages the processor power to user-specified power limits, tailoring DVFS settings to workload demands. Our PowerSave (PS) solution provides the ability to trade-off user-specified levels in performance for energy savings, even under fully-loaded conditions.

We prototype our methodology and solutions on an Intel Pentium M processor-based system and evaluate their effectiveness using the SPEC CPU2000 benchmark suite. Our results show in detail the exploitation of workload characteristics for both the power management solutions. Further, they prove that our low-overhead, practical approach to application-aware power management can be very effective. For example, PM obtained 86% of the possible speedup for the full SPEC CPU2000 benchmark while operating within a power budget of 17.5 Watts and PS obtained 19.2% energy savings for 10% reduction in performance which was within the constraint imposed by the 80% *performance floor* specification. More importantly we prove that the next step in power management, becoming application-aware, is both feasible and effective through our three-phased approach incorporating simple, low-overhead mechanisms and intelligent prediction.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCH3039004.

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] Advanced Micro Devices, "PowerNow with Optimized Power Management." http://www.amd.com/us-en/0,,3715_12353,00.html, Jan. 2006.

[3] D. Bodas, "New Server Power-Management Technologies Address Power and Cooling Challenges," *Intel Magazine*, Sept. 2003.

[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] 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.

[7] W. Felter, K. Rajamani, C. Rusu, and T. Keller, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems," in *Proceedings of the 19th ACM International Conference on Supercomputing*, June 2005.

[8] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," in *Proceedings of the 39th International Symposium on Microarchitecture (MICRO-39)*, December 2006.

[9] R. Kotla, S. Ghiasi, T. W. Keller, and F. L. Rawson, "Scheduling Processor Voltage and Frequency in Server and Cluster Systems," in *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, April 2005.

[10] 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.

[11] J. R. Lorch and A. J. Smith, "Operating System Modifications for Task-Based Speed and Voltage Scheduling," in *Proceedings of MobiSys 2003*, 2003.

[12] B. Lee and D. Brooks, "Statistically Rigorous Regression Modeling for the Microprocessor Design Space," in *ISCA-33: Workshop on Modeling, Benchmarking, and Simulation*, June 2006.

[13] F. Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems," in *ACM SIGOPS European Workshop*, October 2000.

[14] 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.

[15] 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.

[16] C. Isci, G. Contreras, and M. Martonosi, "Hardware Performance Counters for Detailed Runtime Power and Thermal Estimations," in *Hardware Performance Monitor Design and Functionality Workshop in conjunction with 11th International Symposium on High-Performance Computer Architecture (HPCA-11)*, February 2005.

[17] 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.

[18] Q. Wu, V. Reddi, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. W. Clark, "A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance," in *Proceedings of the 38th International Symposium on Microarchitecture (MICRO-38)*, November 2005.

[19] 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.

[20] K. Rajamani, H. Hanson, J. Rubio, S. Ghiasi, and F. Rawson, "Online Power and Performance Estimation for Dynamic Power Management," Tech. Rep. RC24007, IBM Research, July 2006.

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