Core-Level Activity Prediction for Multi-Core Power Management

W. Lloyd Bircher[1] and Lizy John[2]

[1]Advanced Micro Devices, Austin Texas
[2]Department of Electrical and Computer Engineering, University of Texas at Austin

Abstract - **Existing power management techniques operate by reducing performance capacity (frequency, voltage, resource size) when performance demand is low, such as at idle or similar low activity phases. In the case of multi-core systems, the performance and power demand is the aggregate demand of all cores in the system. Monitoring aggregate demand makes detection of phase changes difficult (active-to-idle, idle-to-active, etc.) since aggregate phase behavior obscures the underlying phases generated by the workloads on individual cores. This causes sub-optimal power management and over-provisioning of power resources. In this paper, we address these problems through *core-level, activity prediction*.**

The *core-level* view makes detection of phase changes more accurate, yielding more opportunities for efficient power management. Due to the difficulty in anticipating activity level changes, existing operating system power management strategies rely on *reaction* rather than *prediction*. This causes sub-optimal power and performance since changes in performance capacity by the power manager lag changes in performance demand. To address this problem we propose the *Periodic Power Phase Predictor* (PPPP). This activity level predictor decreases SYSMark 2007 client/desktop processor power consumption by 5.4% and increases performance by 3.8% compared to the reactive scheme used in Windows Vista operating system. Applying the predictor to the prediction of processor power, we improve accuracy by 4.8% compared to a reactive scheme.

Index Terms – **Dynamic power management, prediction, multi-core, power modeling.**

I. INTRODUCTION

Dynamic power management provides a significant opportunity for increasing energy efficiency and performance of computing systems. Energy efficiency is increased by reducing performance capacity (frequency, parallelism, speculation, etc.) when the demand for performance is low. Efficiency or a lack thereof may impact the performance of a system. The challenge in applying power management to increase efficiency is in identifying when to adapt performance capacity. The ubiquitous, architected solution implemented in operating systems such as Windows/Linux is to *react* to changes in performance demand. Though this approach is simple, it performs sub-optimally [4][10] for workloads with many distinct and/or short phases. Each time a workload transitions from a phase of low performance demand to a phase of high performance demand, reactive power management increases performance capacity sometime *after* the phase transition. During the time between the change in demand and capacity, performance may be less than optimal. Similarly, power consumption is sub-optimal on transitions from high to low demand. The amount of performance loss is proportional to the number of phase changes in the workload and the lag between demand and capacity.

Identifying when to adapt is complicated by the presence of multiple cores sharing power resources. Consider Fig. 1. Core-level power consumption is shown for a system with multiple simultaneous threads. The program threads are fixed to the cores with thread N on core N, thread N-1 on core N-1, etc. Since power monitoring is typically provided at the system-level [12], existing power control techniques use the erratic fluctuations in the total power for predicting future behavior. This is unfortunate since in this example, the individual threads have a periodic, easily discernable pattern, while the pattern in the aggregate power is less discernable. If power phases can be tracked at the core-level, accurate dynamic power management schemes can be devised.

We propose to improve the effectiveness of power management through the use of predictive, core-level power management. Rather than reacting to changes in performance demand, we use past activity patterns to predict future transitions. Rather than using aggregate power information, we use activity and power measured at the core-level.

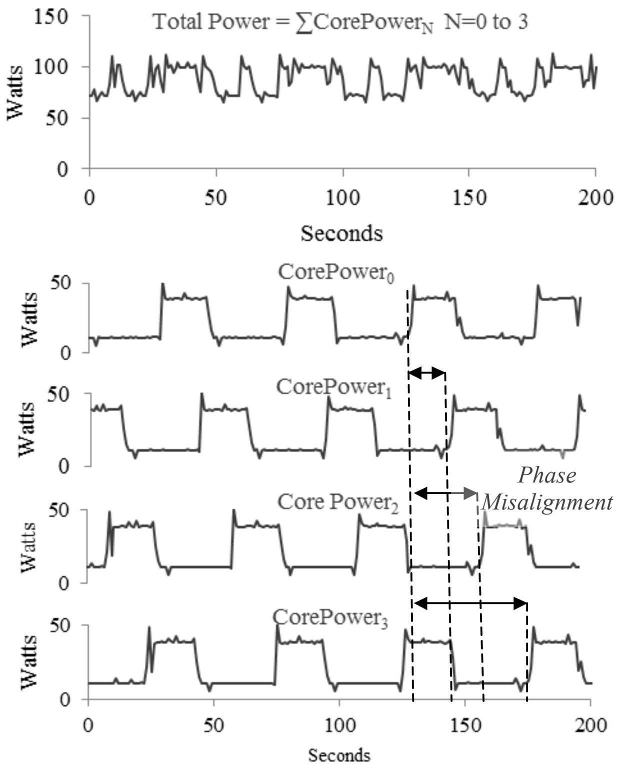

Fig. 1. Thread and Aggregate Power Patterns

To analyze the effectiveness of our predictor we use the SYSMark 2007 benchmark. It contains numerous, desktop/personal computing applications such as Word, Excel, PowerPoint, Photoshop, Illustrator, etc. The benchmark is structured to have a high degree of program phase transitions, including extensive use of processor idle states (c-states) and performance states (p-states) [1]. Rather than being strictly dominated by the characteristics of the workload itself, SYSMark 2007 accounts for the impact of periodic, operating system scheduling and I/O events. Using this observation, we construct a periodic power phase predictor. This simple periodic predictor tracks and predicts periodic phases by their duration. By predicting future demand, aggressive adaptations can be applied. The core-level predictive technique outperforms existing reactive power management schemes by reducing the effective lag between workload phase transitions and power management decisions. By predicting the individual power contribution of each thread rather than predicting the aggregate effect, complex phases can be predicted.

The contributions of this study are summarized as follows.

(1) Concept of core-level power phases in multi-core systems. Core-level power phases unveil more opportunities for power saving adaptations than are possible if only aggregate system level information is used.

(2) A simple, periodic power phase predictor. Though prior research [11][14] demonstrates the effectiveness of power management using predictive schemes on uniprocessors, our research shows its effectiveness when applied to multi-core systems with operating system scheduling interactions. The proposed predictive power management is compared against the reactive algorithm in the Windows Vista operating system. Previous research focused on single-threaded SPEC CPU 2000 benchmarks, while our study uses SYSMark 2007 which includes popular desktop/personal computing applications such as Microsoft Word, Excel, PowerPoint, Adobe Photoshop, Illustrator, etc. These workloads include difficult to predict power management events due to large numbers of interspersed idle phases and frequent thread migrations.

(3) Detailed, power management-aware, CPU power model for an AMD Quad-Core Opteron processor. Since power values are available at system level only, core-level prediction is performed using performance counters as proxies. To realize our core-level phase detection objectives, we develop a detailed power model for this quad-core processor, using core-level performance metrics. We improve upon exiting models based on performance counters [5] to account for leakage, temperature and power management effects.

II. COMMERCIAL DVFS ALGORITHM

Existing, commercial DVFS algorithms in Windows and Linux operating systems select processor clock frequencies by reacting to changes in core activity level [25]. Activity level represents the ratio of architected code execution (active time) to wall clock time (active + idle time). Processors become idle when they exhaust the available work in their run queues. The intent of these algorithms is to apply low frequencies when a processor is mostly idle and high frequencies when mostly active. This provides high performance when programs can benefit and low power when they cannot.

In this study we compare our predictive DVFS algorithm to that currently used in the Windows Vista operating system [25]. This Vista algorithm reactively selects DVFS states (core frequency) in order to maintain a target core activity level of 30%-50%. See Fig. 2. The "predicted" activity level is the last observed activity level, hence this is a reactive scheme.

When the core activity level is greater than 50%, the reactive algorithm selects a higher frequency to increase performance enough to allow the core to be idle more than 50% (i.e. active < 50%) of the time. The new frequency is selected assuming a 100% frequency increase reduces active time by 50%. For example assume a core operates at 1GHz and is 100% active. In order to achieve an activity level of 50%, the algorithm would attempt to double the frequency to 2GHz. Frequency reductions are similar in that activity levels below 30% cause the algorithm to reduce frequency in order to increase activity levels to 30%. Since processors have a finite number of architected DVFS states, the algorithm selects the nearest frequency which meets the target activity level.

$$Activity\ Level = \frac{Time_{Active}}{Time_{Active} + Time_{Idle}}$$

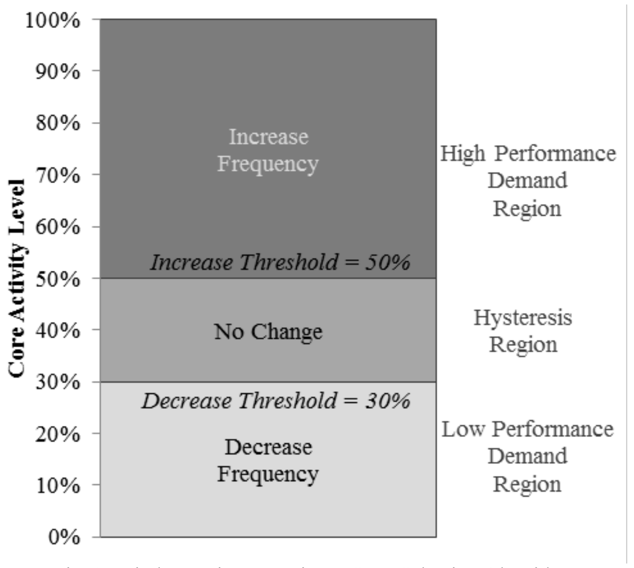

Fig. 2. Windows Vista Reactive P-State Selection Algorithm

III. WORKLOAD CHARACTERIZATION

To analyze the power/performance impact of our predictive power management scheme on real-world workloads, we characterize a system running the desktop/client SYSMark 2007 benchmark. This benchmark represents a wide range of desktop computing applications. The benchmark components are E-Learning, Video Creation, Productivity, and 3D. The individual subtests are listed in Table I. This benchmark is particularly important to the study of dynamic power adaptations since it provides realistic user scenarios that include user interface and I/O delays. These delays cause a large amount of idle-active transitions in the cores. Since current OSs determine dynamic adaption levels using core activity level, the replication of these user interactions in the benchmark is critical.

One of the most critical aspects of this workload is shown in Fig. 3. Unlike traditional scientific and computing benchmarks, core activity level varies greatly over time. The primarily single-thread E-Learning and Productivity subtests are composed of single core active phases interspersed with all cores idle. The frequent transitions between active and idle make power management decisions difficult. Reacting too quickly to idle phases can induce excessive performance loss as code execution is halted to allow transition of clocks, voltage planes or component state. At the other extreme the 3D and Video Creation workloads have large sections of all cores being active. These regions, also interspersed with all-idle and one-active regions are critical for power sharing strategies. As the workload/operating systems adds and removes threads from cores the resultant power level changes drastically. The power difference between cores in the active versus idle state is much greater than differences within the active state.

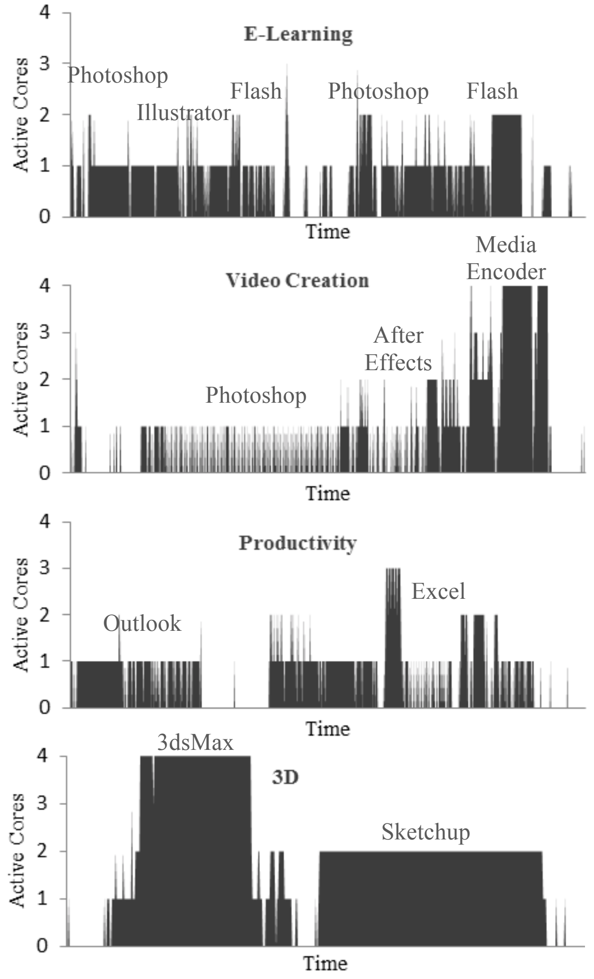

Fig. 3. Utilization of Multiple Cores by SYSMark 2007 Benchmark

The effect of these frequent transitions on power and performance is significant. Bircher et al [4] show that the slow reaction of the Vista DVFS algorithm leads to a performance loss of 6%-10% for SYSMark07. We show that this performance loss is due to a large portion of the benchmark operating at DVFS state with frequencies as low as 1/3 of the maximum frequency. Similarly, power consumption is excessively high due to the DVFS algorithm choosing high-power states for many of the frequent, idle phases in the benchmark.

TABLE I
SYSMARK 2007 COMPONENTS

E-Learning	Video Creation	Productivity	3D
Adobe	Adobe	Microsoft	Autodesk
- Illustrator	- After Effects	- Excel	- 3Ds Max
- Photoshop	- Illustrator	- Outlook	Google
-Flash	- Photoshop	- Word	- SketchUp
Microsoft	Microsoft	- PowerPoint	
- PowerPoint	- Media Encoder	- Project	
	Sony	Corel	
	- Vegas	- WinZip	
Performance Loss for Vista Reactive Power Management [4]			
8.8%	6.2%	9.5%	5.9%

IV. METHODOLOGY

To expose the effect of core-level phase prediction we develop a, performance monitoring counter (PMC)-based, core-level power model. It is necessary to use a power model since it is impossible to measure core-level power consumption in existing multi-core processors due to cores sharing a single power plane [31]. Physical instrumentation thus cannot provide core-level power information, whereas the power model using performance counters can provide power estimates from individual cores. Prior studies show that accurate power models can be built using performance counter measurements [2][15][5][19]. Our model is more accurate due to its ability to account for leakage power, temperature effects and the impact of power management that are present in modern multi-cores. These models are preferred for dynamic power management since there is no need to measure power with out-of-band instrumentation. The details of our power and performance counter tracing methodology are provided below.

A. Power Measurement

To measure power consumption, we instrument an AMD quad-core processor. Processor core power consumption is measured using a hall-effect sensor placed in-line between CPU cores and their power supply (Core VDD). This sensor produces an output voltage that is linearly proportional to current. We also measure voltage levels at the point where the current enters the processor socket. We perform all sampling at a rate of 1 MHz, using a National Instruments NIUSB-6259 [23] data acquisition system. This granularity allows the measurement of most power phases that are sufficiently long enough for power/performance adaptation. Though shorter duration phases exist, current adaptation frameworks are not readily able to exploit them.

B. Performance Counter Measurement

To sample performance monitoring counters we developed a small kernel that provides periodic sampling of the four Opteron performance counters. This kernel uses a device driver to give privileged access to user-mode applications. This approach is preferred over existing user-mode performance counter APIs as it affords more precise control of sampling and lower overhead. In all experiments the sampling overhead for performance counter access averaged less than 1%. Another benefit of using a device driver is that it provides access to others registers besides performance counters. In particular, our approach requires access to model specific registers (MSRs) and PCI configuration registers. These registers allow our application to take control of processor frequency, voltage, power management. They also give access to on-die CPU temperature. This is required to account for static power consumption. Finally, sampling is invoked at a user-specified periodicity using the built-in Windows multimedia timer [30].

V. PERIODIC POWER PHASE PREDICTOR

While power management in operating systems like Windows/Linux is reactive, there have been proposals to use predictive power management [14][11][10]. Isci [14] uses table-based predictors of memory operations/instruction, to direct DVFS (dynamic voltage and frequency scaling) decisions for single-threaded workloads. Duesterwald *et al.* [11] examine table-based predictor techniques to predict performance-related metrics (IPC, cache misses/instruction and branch misprediction rates) of single-thread workloads, but not power. Diao [10] uses machine learning to predict activity patterns. The predictions are used to make policy decisions for entering core idle states. In contrast we propose the periodic power phase predictor (PPPP) which makes use of table-based prediction structures and the repetitive nature of power phases to predict performance demand and/or power consumption. The predictor is shown in Fig. 4. Like traditional table-based predictors, the main components are: a global phase history register (GPHR), pattern history table (PHT) and predicted level. Typically, table-based predictors track sequences of events such as branch outcomes or IPC samples [11][14]. This predictor is distinct in that it tracks run-length-encoded sequences of core active/idle phases. Activity in this case is defined as execution of architected instructions. In APCI[1] terminology that is popularly used for power management, it is known as the C0 state. All other time is considered non-active or idle. Idle time is explicitly defined as "executing" in a processor idle state via the HLT instruction or other idle state entry method [3]. This state is also known as the C_x state where x = 1 to N. These non-C0 states are responsible for the largest power reductions due to the application of clock and power gating. Large power changes or phases can be detected by tracking core activity patterns. For this reason we construct the PPPP to track core activity patterns.

A diagram and functional description of the predictor are provided in Fig.s 4 to 5 and Table II. The main feature of the predictor is its ability to capture frequent, power-relevant events by tracking active/idle patterns. Transitions to active or idle states and the resultant power level can be predicted by tracking previous patterns. One of the most common events is the periodic timer tick event used in many commercial operating systems [28]. This event occurs on a regular interval to provide timing and responsiveness to the operating system scheduler. When a core is otherwise idle, the timer tick produces an easily discernable pattern.

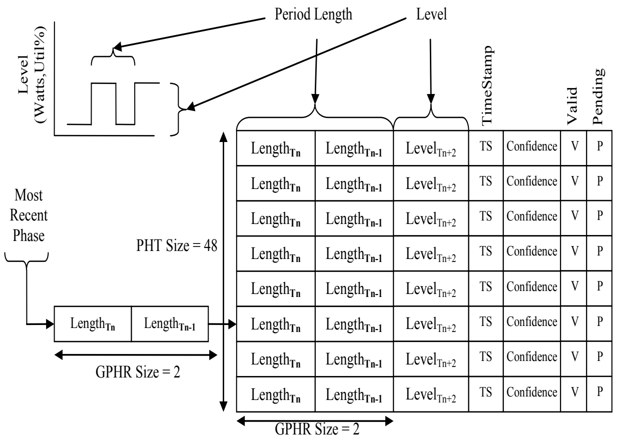

Fig. 4. Periodic Power Phase Predictor

For Windows operating systems, the boot strap processor remains idle for periods of 16 milliseconds interrupted by active periods lasting about 100 microseconds.

The predictor tracks the duration of the idle and active phases in the *length* fields. As a pattern of active and idle period repeats the predictor updates the quality of the prediction using the *confidence* field. The correctness of the prediction is assessed by comparing the predicted time of the next transition (*timestamp* field) to the actual transition time. Also the power level observed at the transition is recorded. Returning to the previous example, when an idle core wakes to respond to the timer tick interrupt, the power consumption is compared to the previously observed power level. If the power levels do not match, the prediction confidence is reduced. The *valid* and *pending* fields are used to determine which predictor entries can be used for predictor matches and which have outstanding predictions respectively.

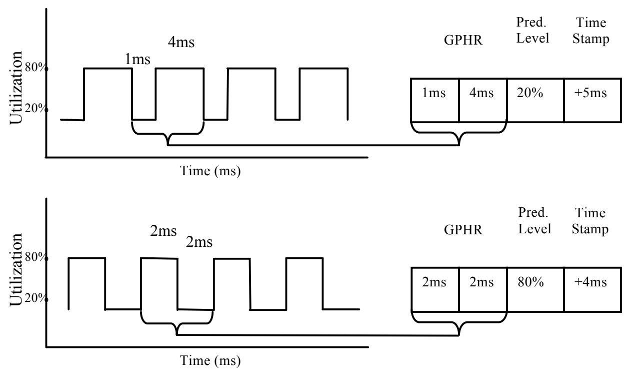

Fig. 5. Example of Program Phase Mapping to Predictor

TABLE II
PERIODIC POWER PHASE PREDICTOR FIELD DESCRIPTIONS

Predictor Field	Description
Length	Duration of phase. This is also the table index. When a periodic phase is detected, it is used to index the prediction table.
Level	Predicted level at next transition. For utilization predictor this is active or idle. For power prediction this is the last power level seen when this phase occurred.
Timestamp	Records timestamp of when predicted phase change is to occur. This is the most critical value produced by the predictor. It is used by the power manager to schedule changes in power/performance capacity of the system. This value allows for optimal selection of performance capacity given the anticipated duration of operation at a particular performance demand.
Confidence	"Quality" of phase as a function of past predictions and duration. The confidence is used by the power manager to determine if a prediction will be used or not. It is also used by the replacement algorithm to determine if the phase will be replaced if the predictor is full. All newly detected phases start with a confidence of 1. If the phase is subsequently mispredicted, the confidence is reduced by a fixed ratio.
Valid	Indicates whether this entry has a valid phase stored with a "true" or "false."
Pending	Indicates if this phase is predicted to occur again. This value is set "true" on the occurrence of the phase and remains true until the phase prediction expires.

VI. PREDICTING CORE ACTIVITY LEVEL

This section provides power and performance results for the core and aggregate-level periodic power phase predictor in comparison to a commercial reactive scheme. We compare prediction accuracy, prediction coverage, power and performance. Also, a characterization of core activity phases is given.

First we consider prediction accuracy. We define accuracy according to the existing, commercial, reactive DVFS algorithm used in the Windows Vista operating system [25]. A correct prediction is one in which the selected DVFS frequency selection keeps the processor within the target range of 30% to 50% activity.

The accuracy of the reactive scheme is determined by analyzing traces of core DVFS and activity levels from a real system. If the selected frequency did not cause the core to have an activity level between 30% and 50%, the selection is considered *wrong*. For the predictive schemes, the activity level trace is played back through our predictor while allowing it to select a frequency to meet the 30%-50% target. Since core activity level changes according to core frequency, the resultant activity level must be scaled accordingly. The amount of scaling is determined experimentally by measuring performance of the SYSMark workload under a range of core frequencies. Performance, and therefore activity level, scale 70% for each 100% change in core frequency.

Using this approach we present results for SYSMark 2007 prediction accuracy in Table III. DVFS hit rate is provided for three predictors. Core-level PPPP represents our predictor applied to each core. Aggregate PPPP represents our predictor driven by the total activity level. All target activity levels remain the same. A single predictor, driven by the aggregate activity level (i.e. average of all cores) is used to select the next core frequency. Core-level reactive represents the Windows Vista DVFS algorithm.

TABLE III
SYSMark 2007 DVFS Hit Rate

Predictor	E-Learning	Productivity	Video Creation	3D
Core-Level PPPP	82.6%	73.8%	76.4%	72.7%
Aggregate PPPP	26.8%	26.3%	40.2%	30.7%
Core-Level Reactive (Vista)	66.4%	65.2%	63.5%	59.7%

The limitations of reactive DVFS selection are evident. Due to frequent transitions between high and low activity levels, the reactive scheme is only able to achieve the correct frequency about 2/3 of the time. PPPP applied at the aggregate level is much worse with an average of 31% accuracy. The best case is achieved with the core-level PPPP which averages 76%. The differences in the success of these predictors are a result of prediction coverage and accuracy of the predicted phases. See Table IV. Coverage is defined as percentage of the workload in which a prediction is available. A prediction could be unavailable if the last observed activity pattern has not been seen before or has caused too many mispredictions. The reactive scheme does not have coverage since it does not predict. In contrast PPPP has much lower prediction coverage, especially for the aggregate predictor. The aliasing of multiple core phases obscures predictable behavior to less than 3% for E-Learning and Productivity. Video Creation and 3D are slightly better at 16% and 8% respectively. One possible reason is that these workloads have larger portions of multi-threaded execution. The aggregate activity level is likely more representative of core-level activity compared to the single-threaded E-Learning and Productivity. Core-level PPPP achieves the highest accuracy by having a large workload coverage of 43% and accuracy over 95% in the covered portion. Outside of the covered portions the predictor selects frequency according to the reactive algorithm.

TABLE IV
SYSMark 2007 Prediction Coverage

Predictor	E-Learning	Productivity	Video Creation	3D
Core-Level PPPP	57.0%	33.5%	43.0%	37.9%
Aggregate PPPP	1.3%	2.3%	16.3%	8.0%
Core-Level Reactive (Vista)	N/A	N/A	N/A	N/A

To quantify the improved predictability of core-level versus aggregate PPPP, Table V presents a characterization of core active and idle durations for SYSMark 2007. Durations group into the following ranges: < 10 milliseconds, 10-100 milliseconds, 100-1000 milliseconds and > 1000 milliseconds. One of the major distinctions between core-level and Aggregate is the high concentration of short phases, less than 10ms for CoreTotal. Just as in the example shown in Fig. 1, these short phases are largely a result of misalignment of the core-level activity. In particular, the most common phases are in the 10-100ms range. This is caused by the timer tick, scheduling and power adaptation intervals for the Windows operating systems.The timer tick normally occurs on 16ms boundaries. Thread creation and migration events also occur on these boundaries. Power adaptations (DVFS) occur on 100ms boundaries. Therefore, idle phases are very frequently interrupted by these events. Similarly, active phases are often terminated by threads being migrated to other cores on these same boundaries. Any misalignment of these events between cores causes the effective activity durations to be shorter and less predictable.

Fig. 6 provides the frequency distribution of active and idle phases across SYSMark 2007. Active and idle phase are considered as a group since both are relevant for prediction. Idle phases must be predicted in order to anticipate how long a power surplus will be available. Similarly, active phase must be predicted to anticipate durations of power deficits. In both cases the predicted durations is needed in order to weigh the power and performance cost of transitioning to low power states or changing the DVFS operating point. Several local maximums are present due to the periodic nature of the interaction between power management, OSs and system hardware. By removing or varying the intensity of these various events and observing the change in frequency distribution, we are able to relate period length to its source. Note the prevalence of phases in the 10-15ms range that corresponds to the OS scheduling interval. Also, consider the spikes at 100ms, which corresponds to the DVFS scheduling interval. Additional, longer-duration maximums occur in the 200ms and higher range. These correspond to GUI interaction and I/O delays occurring in the SYSMark benchmark.

TABLE V
Core Phase Residency by Length

	E-Learning		Video Creation	
Phase Length	Core	Aggregate	Core	Aggregate
< 10 ms	11%	93%	44%	82%
10 - 100 ms	49%	7%	27%	2%
100 - 1000 ms	10%	0%	14%	9%
> 1000 ms	30%	0%	16%	7%

	Productivity		3D	
Phase Length	Core	Aggregate	Core	Aggregate
< 10 ms	55%	97%	55%	97%
10 - 100 ms	30%	3%	30%	3%
100 - 1000 ms	5%	0%	5%	0%
> 1000 ms	11%	0%	11%	0%

Next we consider the resultant power and performance impact of the core-level PPPP versus reactive DVFS selection. Aggregate PPPP is not considered due its very poor prediction accuracy. Table VI presents power and performance results for the two schemes. Power and performance are estimated using the measured and predicted DFVS, active and idle states shown in Table VII. On average, power is reduced by 5.4% while achieving a speedup of 3.8%. This improvement is caused by PPPP more frequently selecting high frequencies for active phases and low frequencies for performance-insensitive idle phases. This shift can be seen in the active residencies of all subtests. The 2.4GHz – Active state increases by 0.6 to 2.5 percentage points. Similarly, the active time in lower frequencies is reduced an average of 0.76 percentage points. The performance impact of selecting a low frequency for an active phase can be large. For example, selecting 800MHz rather than 2.4GHz yields a performance loss of 47% ((1-0.8GHz/2.4GHz) x 70%). Therefore, it takes only a small change in residency to drastically impact performance. Also, the impact on performance is larger due to active time representing only an average of 17% total time. This magnifies the performance impact by about 6x (1/0.17). The net effect on active frequency is an increase of 144 MHz from 1.55GHz to 1.69GHz. Note that though frequency increases by 9.3%, performance increases only 3.8% due to limited frequency scaling of the workload (70%) and reduced total time in the active state.

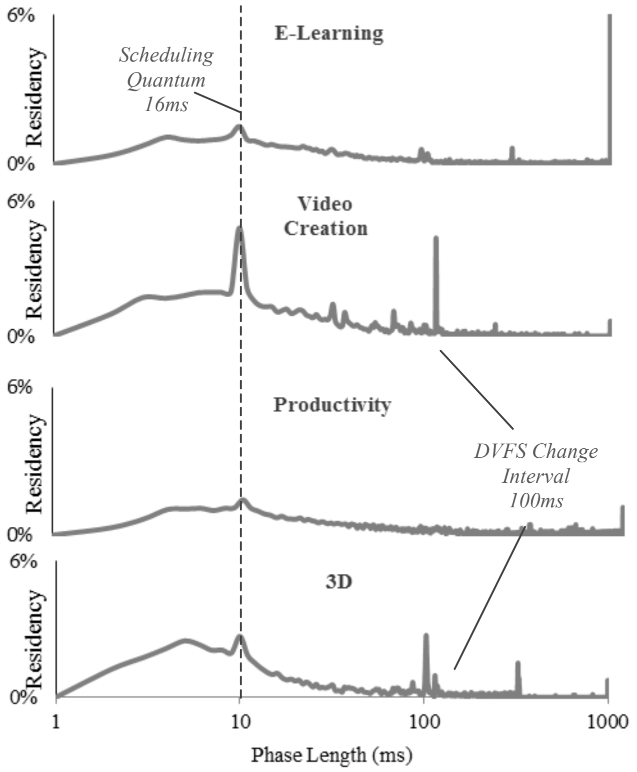

Fig. 6. Core-Level Phase Length Probability Distributions

TABLE VI
SYSMark 2007 Power and Performance Impact of PPPP

	E-Learning		Productivity	
	Predictive (PPPP)	Reactive (Vista)	Predictive (PPPP)	Reactive (Vista)
Power (W)	16.6	18.2	14.3	15.1
Power Savings	8.3%		5.3%	
Delay (sec)	924	963	585	607
Speedup	4.2%		3.7%	
Energy (KJ)	15.4	17.5	8.4	9.2
Energy Savings	12.3%		8.7%	

	Video Creation		3D	
	Predictive (PPPP)	Reactive (Vista)	Predictive (PPPP)	Reactive (Vista)
Power (W)	18.6	19.5	18.6	19.5
Power Savings	4.7%		2.9%	
Delay (sec)	1129	1172	1129	1172
Speedup	3.8%		3.6%	
Energy (KJ)	20.9	22.8	20.9	22.8
Energy Savings	8.2%		6.3%	

Next we consider power savings. Though it is possible to bias a reactive DVFS algorithm to achieve performance comparable to a predictive algorithm, it is not possible to do so without increasing power consumption drastically.

Prediction allows DVFS selection to select the "correct" frequency for both performance and power savings. In this case our predictor achieves a 3.8% performance increase while reducing power consumption by 5.4%. The primary cause is a shift in idle frequency selections away from the high-performance, high-leakage states. Residency in the most inefficient state, 2.4GHz – Idle, was reduced by an average of 7.8 percentage points. Residency in other idle states above the minimum frequency also decreased, but by a smaller 3.1 percentage points. This increases idle residency in the minimum frequency idle state of 800MHz by an average of 15%. Average idle frequency decreases by 200MHz from 1.2GHz to 1.0GHz.

TABLE VII
SYSMark 2007 P-State and C-State Residency of PPPP versus Reactive

	E-Learning		Productivity	
	Predictive (PPPP)	Reactive (Vista)	Predictive (PPPP)	Reactive (Vista)
2.4GHz - Active	5.4%	4.6%	2.9%	2.4%
2.4GHz - Idle	7.1%	17.4%	4.4%	9.6%
1.6GHz - Active	1.2%	1.4%	0.8%	0.8%
1.6GHz - Idle	5.5%	9.4%	3.8%	6.2%
1.2GHz - Active	1.1%	1.2%	1.2%	1.2%
1.2GHz - Idle	6.9%	9.8%	6.6%	9.7%
0.8GHz - Active	3.2%	4.5%	3.8%	4.7%
0.8GHz - Idle	69.5%	51.8%	76.5%	65.3%
Active Frequency	1.72 GHz	1.56 GHz	1.47 GHz	1.34 GHz
Idle Frequency	1.01 GHz	1.24 GHz	0.94 GHz	1.07 GHz

	Video Creation		3D	
	Predictive (PPPP)	Reactive (Vista)	Predictive (PPPP)	Reactive (Vista)
2.4GHz - Active	6.8%	5.3%	6.8%	5.3%
2.4GHz - Idle	5.7%	12.4%	5.7%	12.4%
1.6GHz - Active	2.7%	3.2%	2.7%	3.2%
1.6GHz - Idle	5.6%	9.6%	5.6%	9.6%
1.2GHz - Active	3.8%	4.8%	3.8%	4.8%
1.2GHz - Idle	9.2%	13.8%	9.2%	13.8%
0.8GHz - Active	3.7%	4.8%	4.7%	6.9%
0.8GHz - Idle	62.3%	46.1%	51.6%	36.7%
Active Frequency	1.65 GHz	1.51 GHz	1.92 GHz	1.77 GHz
Idle Frequency	1.01 GHz	1.20 GHz	1.07 GHz	1.32 GHz

VII. PREDICTING POWER LEVELS

The second application of periodic power phase prediction is for predicting core power consumption. Predicting power levels provides opportunities for increased performance and efficiency. Existing power control systems such as power capping[12] and turbo boost[9] apply power and performance limits statically based on user-specified or instantaneous power consumption. Knowing power levels a priori could increase performance by avoiding adaptations for short duration phases. For example, a core that encounters a short, high-power phase of execution may cause the power controller to reduce its or other processors' frequency. If the controller could know that the phase would be too short to cause a power or temperature violation, the reduction in performance could be avoided.

To this end we apply PPPP to prediction of core-level and aggregate power consumption. We compare results to a last value predictor also at the core and aggregate level. Core-level power is measured using our PMC-based power model. The model allows fine-grain, power management and temperature-aware estimation of core power. Additional details of the models are provided in the Appendix[6].
Rather than using core activity level to predict *core activity level*, we use it to cross predict *power level*. The predicted activity-level in the predictor is replaced by the modeled core power level. The prediction table index remains as sequences of core activity levels. This approach provides better pattern matching as variations in temperature and application of DVFS tends to hide otherwise discernable patterns.

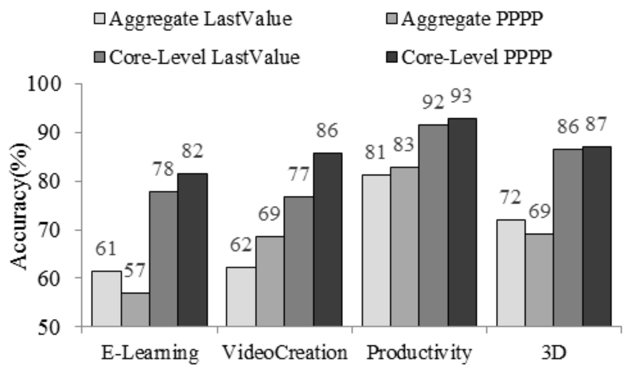

Fig. 7. Prediction Accuracy of Core Power for Various Predictors

Fig. 7 shows the weighted average percent accuracy of our periodic power phase predictor compared to a last-value predictor. Weighted average is chosen since SYSmark 2007 power consumption contains many idle, low-power phases. In these phases, a small error in absolute terms yields a very large percentage error. Therefore, we scale error values by the magnitude of the measured power sample compared to the maximum observed. For example, a 10% error on a 5W sample has half the impact of a 10% error on a 10W sample. For all subtests, the core-level versions of the predictors outperformed the aggregate versions. The best overall performance was 86% accuracy for the periodic core-level predictor compared to 83% for the core-level version of the last-value predictor. The benefit of core-level prediction of power is less pronounced than for prediction of activity level. This is due to the smaller dynamic range of power consumption compared to activity level. Though activity levels regularly vary from

0% to 100%, power levels remain in a much smaller range of approximately 25% to 75%.

VIII. CONCLUSION

This paper presents the concept of core-level phase prediction and its application to dynamic power management. By observing changes in performance demand and power consumption at the core-level, it is possible to perceive predictable phase behavior. Prediction of phases allows power management to avoid over or under provisioning resources in response to workload changes. Using this concept we develop the PPPP, a simple, table-based prediction scheme for directing DVFS selection. We apply the predictor to the SYSMark2007 benchmark suite and attain simultaneous performance and power improvements. Compared to the reactive DVFS algorithm used by Windows Vista, performance is increased by 5.4% and while power consumption is reduced by 3.8%. We also present a power and temperature-aware core-level model for processor power consumption. Using this model, we show processor power can be predicted by PPPP with accuracy 4.8% better than a last-value predictor.

APPENDIX - CPU POWER MODEL

To provide core-level power measurement we develop a performance counter based power model for the AMD Opteron processor. The model is similar to other models [2][15][5][19] in that it is composed of a small set of performance events that are highly correlated to processor power consumption. The primary distinction is that our model accounts for temperature and voltage effects. This allows isolation of microarchitecture-independent power consumption such as leakage current and DVFS states.

Using a real system instrumented for power measurement we develop polynomial, regression models for power consumption. The details of the model are given in Tables VIII and IX. The model improves on existing on-line models by accounting for power management and temperature variation. All model coefficients are tuned empirically using a real system instrumented for power measurement. Like existing models ours contains a workload dependent portion that is dominated by the number of instructions completed per second. In this case we use the number of fetched operations per second in lieu of instructions completed. The fetched ops metric for is preferred as it also accounts for speculative execution that does not update the architected state.

TABLE VIII
AMD Quad-Core Opteron Power Model

Power Models	Equation
Total Power	$\left(\sum_{N=0}^{3} WorkloadDependent_N + Ungateable_N + Gateable_N\right)$
Workload Dependent Power	$(FetchOps_N/Sec)\cdot Coeff_F+(FloatPointOps_N/Sec)\cdot Coeff_{FP}+(DCAccess_N/Sec)\cdot Coeff_{DC}$
Gateable Power	$(\%Halted_N)\cdot Coeff_{Gateable}\cdot(Voltage)^2\cdot Frequency_N$
Ungateable Power	$(\%NonHalted_N)\cdot Coeff_{Ungateable}\cdot(Voltage)^2\cdot Frequency_N$
Static Power	$(Temp2\cdot Coeff_{T2}+Temp\cdot Coeff_{T1}+\cdot Coeff_{T2})_{Voltage}$
	Average Error = 0.89%

TABLE IX
Model Parameter Descriptions

Quantity	Description
N	Core Number.
FetchOps	Micro operations fetched. Includes speculative operations.
FloatPointOps	Floating point operations retired. Accounts for difference in power between INT and FP.
DCAccess	Data cache access. Accounts for power consumed in caches.
%Halted	% of cycles in which the core was halted.
%Not Halted	% of cycles in which the core was not halted.
Voltage	Maximum requested voltage for all cores. Due to shared voltage plane.
Frequency	Current core frequency. This can be read via AMD model specific register.
Temperature	Current processor temperature. This can be read via AMD model specific register.
$Coeff_x$	Model coefficient. The values are determined empirically using measurement/regression.

The primary distinction of our model is that is contains a temperature dependent component. Using workloads with constant utilization, we vary processor temperature and voltage to observe the impact on static leakage power. Temperature is controlled by adjusting the speed of the processor's fan. Temperature is observed with 0.125 degree Celsius resolution using an on-die temperature sensor [3]. This sensor can be accessed by the system under test through a built-in, on-chip register. Voltage is controlled using the P-State control register. This allows selection of one of five available voltage/frequency combinations. Voltage is observed externally using our power instrumentation. Like the workload dependent model, we tune the coefficients of the polynomial model using regression techniques. Note that the static power model is highly process dependent. Processors manufactured with different semiconductor process parameters require the model to be re-tuned.

The dominant power management effects (voltage/frequency scaling, clock gating) are further accounted for using the Gateable and Ungateable power models. Gateable power is found by measuring the effect of enabling/disabling idle core clock gating (Cache Flush on Halt). Ungateable represents the portion of power which cannot be gated. These components are also found experimentally. The resultant, average error in the model was 0.89%. The error distribution for SPEC CPU2006 and SYSMark 2007 is provided in Fig. 8.

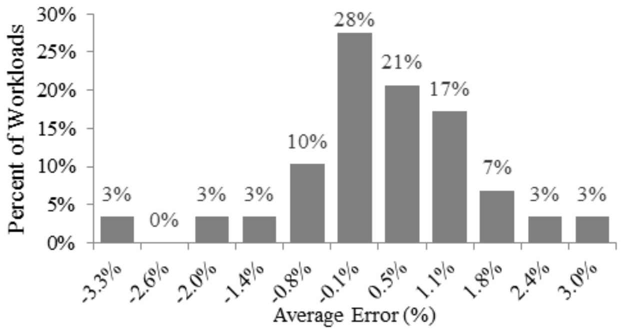

Fig. 8. Model Error Analysis – SPEC CPU 2006 and SYSMark 2007

REFERENCES

[1] Advanced Configuration & Power Interface. http://www.acpi.info (November 2007).

[2] Bellosa, F. The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems. In *Proceedings of 9th ACM SIGOPS European Workshop* (September 2000), 37-42.

[3] BIOS and Kernel Developer's Guide for AMD Family 10h Processor. http://www.amd.com. (November 2007).

[4] Bircher, W. L. and John, L. Analysis of Dynamic Power Management on Multi-Core Processors. In *Proceedings of the 22nd Annual International Conference on Supercomputing* (Kos, Greece, June 2008), 327-338.

[5] Bircher, W. L and John, L. Complete System Power Estimation: A Trickle-Down Approach based on Performance Events. In *Proceedings of the 7th Annual International Symposium on Performance Analysis of Systems and Software* (April 2007), 158-168.

[6] Bircher, W. L and John, L. Complete System Power Estimation using Processor Performance Events. IEEE Transactions on Computers, accepted for publication December 2010.

[7] Meisner, D., Gold, B. and Wenisch, T. PowerNap: Eliminating Server Idle Power. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems* (Washington, DC, March 2009), 205-216.

[8] Blade Server Technology Overview. http://www.blade.org/techover.cfm (March 2007).

[9] Charles, J., Jassi, P., Narayan, A., Sadat, A. and Fedorova, A. Evaluation of the Intel® Core™ i7 Turbo Boost Feature. In *Proceedings of the IEEE International Symposium on Workload Characterization* (October 2009).

[10] Diao, Q., Song, J. Prediction of CPU Idle-Busy Activity Pattern. In *Proceedings of the International Symposium on High-Performance Computer Architecture* (February 2008).

[11] Duesterwald, E., Cascaval, C., and Dwarkadas, S. Characterizing and Predicting Program Behavior and its Variability. In *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Technique* (September 2003), 220-231.

[12] Dynamic Power Capping TCO and Best Practices White Paper. http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA2-3107ENW.pdf (May 2010).

[13] Fan, X., Weber, W., and Barroso, L. A. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture* (San Diego, California, June 2007), 13-23.

[14] Isci, C., Contreras, G., and Martonosi, M. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture* (December 2006), 359-370.

[15] Isci, C. and Martonosi, M. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In *Proceedings of the 36th Annual ACM/IEEE International Symposium on Microarchitecture* (December 2003), 93.

[16] Kotla, R., Devgan, A., Ghiasi, S., Keller, T., and Rawson, F. Characterizing the Impact of Different Memory-Intensity Levels. In *Proceedings of the 7th Annual IEEE Workshop on Workload Characterization* (Austin, Texas, October 2004).

[17] Lefurgy, C., Wang, X. and Ware, M. Server-level power control. In *Proceedings of the 4th IEEE International Conference on Autonomic Computing* (Jacksonville, Florida, June 2007).

[18] Li, J. and Martinez, J. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture* (Austin, Texas, February 2006).

[19] Li, T. and John, L. Run-Time Modeling and Estimation of Operating System Power Consumption. In *Proceedings of ACM/SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (San Diego, California, June 2003), 160-171.

[20] Li, Y., Brooks, D., Hu, Z., and Skadron, K. Performance, Energy, and Thermal Considerations for SMT and CMP Architectures. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture* (San Francisco, California, February 2005), 71-82.

[21] McGowen, R., Poirier, C., Bostak, C., Ignowski, J., Millican, M., Parks, W., and Naffziger, S. Temperature Control on a 90-nm Itanium Family Processor. *IEEE Journal of Solid State Circuits*, 41, 1 (January 2006).

[22] Minerick, R. J., Freeh, V. W., and Kogge, P. M. Dynamic Power Management Using Feedback. In *Proceedings of the 3rd Workshop on Compilers and Operating Systems for Low Power (COLP'02)* (Charlottesville, Virginia, September 2002).

[23] National Instruments Data Acquisition Hardware. http://www.ni.com/dataacquisition/ (April 2008).

[24] Pallipadi,V. and Starikovskiy, A. The On Demand Governor: Past, Present and Future. In *Proceedings of the Linux Symposium* (Ottawa, Canada, July 2006).

[25] Processor Power Management in Windows Vista and Windows Server 2008. http://www.microsoft.com (November 2007).

[26] Rajamani, K., Hanson, H., Rubio, J., Ghiasi, S., and Rawson, F. Application-Aware Power Management. In *Proceedings of the 2006 IEEE International Symposium on Workload Characterization* (San Jose, California, October 2006), 39-48.

[27] Ranganathan, P., Leech, P., Irwin, Y. and Chase, J. Ensemble-Level Power Management for Dense Blade Servers. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture* (Boston, Massachusetts, June 2006), 66-77.

[28] Siddah, S., Pallipadi, V., and Van de Ven, A. Getting Maximum Mileage Out of Tickless. In *Proceedings of the Linux Symposium* (Ottawa, Canada, June 2007).

[29] Wang, X. and Chen, M. Cluster Level Feedback Power Control for Performance Optimization. In *Proceedings of the 14th International Symposium on High-Performance Computer Architecture* (Salt Lake City, Utah, February 2008).

[30] Windows Multimedia: timeEndPeriod(). http://msdn.microsoft.com/en-us/library/ms713415(VS.85).aspx (November 2008).

[31] Wu, Q., Juang, P., Martonosi, M., Peh, L., and Clark, D. Formal control techniques for power-performance management. *IEEE Micro*, 25, 5 (September/October 2005).