Analysis of Dynamic Power Management on Multi-Core Processors

W. Lloyd Bircher and Lizy K. John
Laboratory for Computer Architecture
Department of Electrical and Computer Engineering
The University of Texas at Austin
{bircher, ljohn}@ece.utexas.edu

ABSTRACT
Power management of multi-core processors is extremely important because it allows power/energy savings when all cores are not used. OS directed power management according to ACPI (Advanced Power and Configurations Interface) specifications is the common approach that industry has adopted for this purpose. While operating systems are capable of such power management, heuristics for effectively managing the power are still evolving. The granularity at which the cores are slowed down/turned off should be designed considering the phase behavior of the workloads. Using 3-D, video creation, office and e-learning applications from the SYSmark benchmark suite, we study the challenges in power management of a multi-core processor such as the AMD Quad-Core Opteron™ and Phenom™. We unveil effects of the idle core frequency on the performance and power of the active cores. We adjust the idle core frequency to have the least detrimental effect on the active core performance. We present optimized hardware and operating system configurations that reduce average active power by 30% while reducing performance by an average of less than 3%. We also present complete system measurements and power breakdown between the various systems components using the SYSmark and SPEC CPU workloads. It is observed that the processor core and the disk consume the most power, with core having the highest variability.

Categories and Subject Descriptors
C.0 [Computer Systems Organization]: General

General Terms
Design, Measurement and Performance.

Keywords
power management, performance, operating system, ACPI, multi-core

1. INTRODUCTION
The recent shift to multi-threaded and multi-core processors has created a new set of challenges for dynamic power management. Compared to single-threaded processors, adapting power and performance for multiple threads is more complex. The difficulty centers around two issues: program phase behavior and resource dependencies between threads. Program phase behavior is made more complex by the aggregate phases created by the combination of multiple threads. Phase behavior is used to control the application of power adaptations, making the decision criteria more complex. The decision criteria for adapting must primarily consider the performance cost of the adaption and the likelihood of encountering a particular performance demand. For example, consider a case in which voltage and frequency scaling is used to reduce power consumption during a phase of low performance demand. For each voltage change the processor must briefly suspend execution while the voltage source stabilizes at the new operating point. This has a performance cost that is proportional to the number of program phase changes. For a sporadic workload this cost can outweigh the benefit of the power adaptation. The concept also applies to other adaptations such as resource resizing/power down. Reducing the active portion of a cache causes a performance loss when the resource is reactivated due to the need for warm-up. Disabling a pipeline has a similar effect, as instructions do not complete until the newly active pipeline refills with instructions.

In the multi-threaded case, the decision criteria are more complex because the adaptations may affect the performance of other threads. The cause is shared resources in a multi-threaded system. Since the degree of resource sharing varies among processor types, the performance dependence also varies. For example, a typical multi-core processor shares the top-level cache among all cores on the chip and provides an independent level one (L1) cache. Any power adaptation that affects the performance of this shared cache affects the performance of all cores. In contrast, adapting performance of the L1 cache has little effect on the other cores. The resultant increase in complexity of power adaptations is due to the presence of multiple *independent* threads which have *dependent* performance due to shared resources.

In this paper we seek to improve the effectiveness of power adaptations through a study of program phase behavior and how those phases affect performance in a multi-core processor. We show that the performance impact of power adaptations in Quad-Core AMD Opteron™ and AMD Phenom™ processors is dominated by four characteristics: cache snoop activity, idle core frequency, program phase behavior, and operating system control of power adaptations. Workloads such as equake from SPEC CPU 2000 and 3D workloads from SYSmark® 2007 have a strong performance dependence on cache snoop latency. This latency is shown to be dependent on the frequency of idle cores. The amount of time a core spends in the idle or active state is

ICS'08, June 7–12, 2008, Island of Kos, Aegean Sea, Greece.

dictated by program phase characteristics and the operating system (OS) power adaptation policy. We study these items in the framework of the Advanced Configuration and Power Interface (ACPI). This interface specification was developed to establish industry common interfaces enabling robust OS-directed power management of both devices and entire systems. ACPI is the key element in OS-directed configuration and power management. From a power management perspective, ACPI promotes the concept that systems should conserve energy by transitioning unused devices into lower power states including placing the entire system in a low-power state (sleeping state) when possible. The interfaces and concepts defined within the ACPI specification are suitable to all classes of computers including (but not limited to) desktop, mobile, workstation, and server machines. We also show that compared to benchmarks such as SPEC CPU 2000, recent benchmark suites such as SPEC CPU 2006 shift power consumption significantly from the processor to the memory subsystem due to increased working set sizes. Using these findings, we propose a power management configuration/policy which has an average power reduction of 30 percent with less than 3 percent performance loss.

2. BACKGROUND

In this study we consider issues surrounding the use of dynamic power adaptations on a real system. The objective is to make optimal decisions regarding the tradeoff between performance and power savings. For this purpose we consider areas such as: program power/phase behavior, power saving techniques, and adaptation control policies.

In the area of program phase behavior, studies which characterize typical program phases with respect to power are most relevant. Studies by Boher, Mahesri, and Feng [5][15][7] present power characterizations of programs running on hardware ranging from mobile to clustered servers. Our study differs in that the presented power characterization includes phase duration. This information is needed since power adaptations must be applied with consideration for performance costs associated with transitioning hardware to various levels of power adaption. Two studies which do consider phase duration are presented by Bircher [3][4]. Our study differs in that we consider desktop workloads. The inclusion of desktop workloads is a critical difference, as it allows the analysis of workloads that contain many more power phase transitions. The reason is that desktop workloads, such as the ones included here, contain user input and think time events. These events introduce a large number of power phase transitions. As for our phase classification technique, we make use of phase classification metrics as described by Lau [12]. Our study differs in that we make use of these techniques for exploring power phase characteristics of programs running on an actual system. Their study instead considers a range of classification techniques, but does not characterize workloads.

To quantify the effect of power adaptations we present performance and power consumption results for a range of adaptation levels. Studies such as [18] [8][9] consider the performance and power impact of applying power adaptations. Our study differs in that we study power adaption and policies in the framework of a multi-core processor. While these studies consider adaptations and policies which optimize efficiency by accounting for architecture-dependent characteristics such as memory-boundedness, we examine policies which may only consider program slack time in performing adaptations. To meet the goal of increasing energy efficiency within this constraint we analyze the inherent characteristics of the hardware power adaptations and identify optimal configurations. Through this approach we are able to increase performance and reduce power consumption without runtime knowledge of program characteristics.

3. POWER MANAGEMENT

3.1. Active and Idle Power Management

An effective power management strategy must take advantage of program and architecture characteristics. Designers can save energy while maintaining performance by optimizing for the common execution characteristics. The two major power management components are active and idle power management. Each of these components use adaptations that are best suited to their specific program and architecture characteristics. Active power management seeks to select an optimal operating point based on the performance demand of the program. This entails reducing performance capacity during performance-insensitive phases of programs. A common example would be reducing the clock speed or issue width of a processor during memory-bound program phases. Idle power management reduces power consumption during idle program phases. However, the application of idle adaptations is sensitive to program phases in a slightly different manner. Rather than identifying the optimal performance capacity given current demand, a tradeoff is made between power savings and responsiveness. In this case the optimization is based on the length and frequency of a program phase (idle phases) rather than the characteristics of the phase (memory-boundedness, IPC, cache miss rate). In the remainder of this paper we will make reference to active power adaptations called *p-states* and idle power adaptations called *c-states*. These terms represent adaption operating points as defined in the ACPI specification. ACPI [1] "…is an open industry specification co-developed by Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. ACPI establishes industry-standard interfaces enabling OS-directed configuration, power management, and thermal management of mobile, desktop, and server platforms."

3.1.1. Active Power Management: P-states

A p-state (performance state) defines an operating point for the processor. States are named numerically starting from P0 to PN, with P0 representing the maximum performance level. As the p-state number increases, the performance and power consumption of the processor decrease. Table 1 shows p-state definitions for a typical processor. The state definitions are made by the processor designer and represent a range of performance levels which match expected performance demand of actual workloads. P-states are simply an implementation of dynamic voltage and frequency scaling (DVFS). The resultant power savings obtained using these states is largely dependent on the amount of voltage reduction attained in the lower frequency states.

Table 1. Example P-states Definition

	Frequency (MHz)	VDD (Volts)
P0	Fmax 100%	Vmax 100%
P1	Fmax 85%	Vmax 96%
P2	Fmax 75%	Vmax 90%
P3	Fmax 65%	Vmax 85%
P4	Fmax 50%	Vmax 80%

Table 2. Example C-states Definition

	Response Latency(us)
C0	0
C1	10
C2	100
C3	1000
C4	10000

3.1.2. Idle Power Management: C-states

A c-state (CPU idle state) defines an idle operating point for the processor. States are named numerically starting from C0 to CN, with C0 representing the active state. As the c-state number increases, the performance and power consumption of the processor decrease. Table 2 shows c-state definitions for a typical processor. Actual implementation of the c-state is determined by the designer. Techniques could include low latency techniques, clock and fetch gating, or more aggressive high latency techniques such as voltage scaling or power gating.

3.2. Quad-Core AMD Processors and System Description

The Quad-Core AMD Opteron™ and AMD Phenom™ processors used in this study are 1.6GHz-2.4GHz, 3-way superscalar, four-core processors implemented on a 65 nm process. The processor provides an interesting vehicle for the study of dynamic power adaptations due to its ability to operate each of its cores at an independent frequency. This ability provides better opportunity for power savings, but increases the complexity of configuration due to the performance dependence introduced by the independent operating frequencies. Two platform types were used, server and desktop. The server system utilizes 8GB of DDR2-667 configured for dual channel operation. The desktop system uses 1GB of DD2-800 also configured for dual channel.

3.2.1. Quad-Core AMD Processor P-state Implementation

Each core may operate at a distinct p-state. However, a voltage dependency exists between cores in a single package. All cores in a package must operate at the same voltage. The actual voltage applied to all cores is the maximum required of all. Therefore, the best power savings occurs when all cores are operating in the same p-state.

3.2.2. Quad-Core AMD Processor C-state Implementation

Two architecturally visible c-states are provided: C0 and C1. In C0, the active state, fine-grain clock gating throughout the processor provides the power savings. This gating is automatically applied by hardware and has a negligible effect on performance. The other available state, C1, is applied during idle phases by execution of the HALT instruction. This state effectively reduces frequency by a programmable power of 2. For example, the C1 state may reduce frequency by a factor of 2, 4, 8, 16, 128 or 512. Though the responsiveness of cores in the C1 state is not greatly affected by the frequency reduction, the performance of active cores is. This dependency is introduced through shared cache resources. When an active core makes a request for a cache block, a cache probe (snoop) is made to the idle cores. Since the idle core is operating at a reduced frequency, the time to service the probe is increased. Designers can mitigate this effect through the use of adaptations such as increasing idle core frequency in response to probe requests ("CPU Direct Probe Mode"). This approach must be applied carefully since it can greatly reduce idle power savings. In order to balance probe responsiveness with power savings, Quad-Core AMD processors provide a tuning parameter to control how long the idle processor remains at an increased frequency in response to a probe. The result is a hysteresis function. This approach is effective due to the bursty nature of cache probe traffic.

In addition to the architecturally visible C0 and C1, an additional state *C1e* (enhanced C1) is provided. C1e is applied automatically by the hardware in response to idle phases in which all cores are idle. This mode provides larger power savings since there is no need to service cache coherence traffic when all cores are idle. Additional power is saved in the on-chip memory controller and through more aggressive power settings in the cores. These settings are reasonable since the likelihood of waking any one core is less when all cores are idle.

3.2.3. Quad-Core AMD Processor Power Savings Potential

The power saving states described in this section provide a significant range of power and performance settings for optimizing efficiency, limiting peak power consumption, or both. However, other parameters greatly influence the effective power consumption. Temperature, workload phase behavior, and power management policies are the dominant characteristics. Temperature has the greatest effect on static leakage power. This can be seen in Figure 1 which shows power consumption of a synthetic workload at various combinations of temperature and frequency. Note that ambient temperature is 20°C and "idle" temperature is 35°C. As expected, a linear change in frequency yields a linear change in power consumption. However, linear changes in temperature yield exponential changes in power consumption. Note that static power is identified by the Y-intercept in the chart. This is a critical observation since static power consumption represents a large portion of total power at high temperatures. Therefore, an effective power management scheme must also scale voltage to reduce the significant leakage component. To see the effect of voltage scaling consider Figure 2.

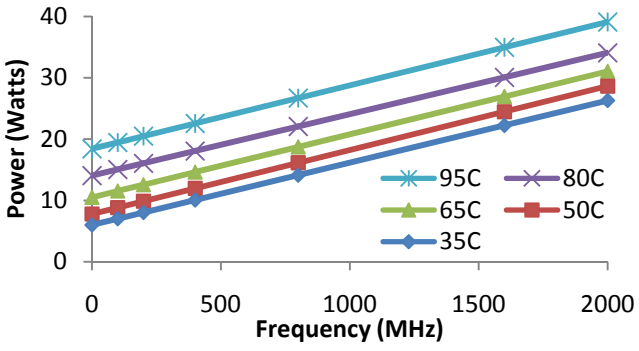
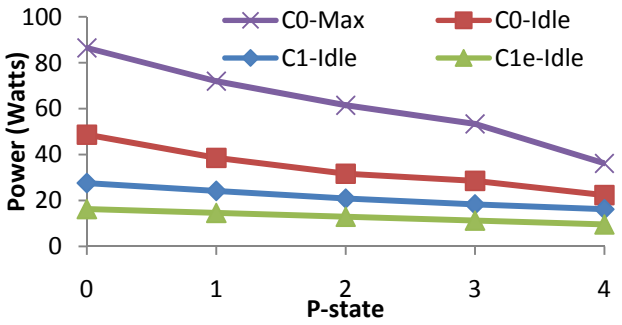


Figure 1. Temperature Sensitivity of Power Reduction through Frequency Scaling



C0-Max	All Cores Active IPC ≈ 3
C0-Idle	All Cores Active IPC ≈ 0
C1- Idle	At Least One Active Core, Core ≈ 0 MHz
C1e-Idle	All Idle, Core ≈ 0 MHz, MemCntrl ≈ 0 MHz

Figure 2. Power by C-state/P-state Combination

Figure 2 shows the cumulative effect of p-states and c-states. Combinations of five p-states (x-axis) and four operating modes are shown. The lowest power case, C1e-Idle, represents all cores being idle for long enough that the processor remains in the C1e state more than 90 percent of the time. The actual amount of time spent in this state is heavily influenced by the rate of input/output (I/O) and OS interrupts. This state also provides nearly all of the static power savings of the low-voltage p-states even when in the P0 state. Second, the C1-Idle case shows the power consumption assuming at least one core remained active and prevented the processor from entering the C1e state. This represents an extreme case in which the system would be virtually idle, but frequent interrupt traffic prevents all cores from being idle. This observation is important as it suggests system and OS design can have a significant impact on power consumption. The remaining two cases, C0-Idle and C0-Max, show the impact of workload characteristics on power. C0-Idle attains power savings though fine-grain clock gating. The difference between C0-Idle and C0-Max is determined by the amount of power spent in switching transistors, which would otherwise be clock-gated, combined with worst-case switching due to data dependencies. C0-Max can be thought of as a pathological workload in which all functional units on all cores are 100 percent utilized and the datapath constantly switches between 0 and 1. All active phases of real workloads exist somewhere between these two curves. High-IPC compute-bound workloads are closer to C0-Max while low-IPC memory-bound workloads are near C0-Idle.

3.3. Costs of Adaptation

The p-state and c-state adaptations described above define the bounds of power consumption possible. In this section we consider what effect these adaptations have on performance and efficiency. The actual power/performance obtained can be quite different due to the physical limitations of how the adaptations are implemented, phase characteristics of workloads, and power management policies.

3.3.1. Transition Costs

Due to physical limitations, transitioning between adaptation states may impose some cost. The cost may be in the form of lost performance or increased energy consumption. In the case of DVFS, frequency increases require execution to halt while voltage supplies ramp up to their new values. This delay is typically proportional to the amount of voltage change (seconds/volt). Frequency decreases typically do not incur this penalty as most digital circuits will operate correctly at higher than required voltages. Depending on implementation, frequency changes may incur delays. If the change requires modifying the frequency of clock generation circuits (phase locked loops), then execution is halted until the circuit locks on to its new frequency. This delay may be avoided if frequency reductions are implemented using methods which maintain a constant frequency in the clock generator. This is the approach used in Quad-Core AMD processor c-state implementation. Delay may also be introduced to limit current transients. If a large number of circuits all transition to a new frequency, then excessive current draw may result. This has a significant effect on reliability. Delays to limit transients are proportional to the amount of frequency change (seconds/MHz). Other architecture-specific adaptations may have variable costs per transition. For example, powering down a cache requires modified contents to be flushed to the next higher level of memory. This reduces performance and may increase power consumption due to the additional bus traffic. When a predictive component is powered down it no longer records program behavior. For example, if a branch predictor is powered down during a phase in which poor predictability is expected, then branch behavior is not recorded. If the phase actually contains predictable behavior, then performance may be lost and efficiency may be lost. If a unit is powered on and off in excess of the actual program demand, then power and performance may be significantly affected by the flush and warm-up cycles of the components. In this study we focus on fixed cost per transition effects such as those required for voltage and frequency changes.

3.3.2. Workload Phase and Policy Costs

In the ideal case the transition costs described above do not impact performance and save maximum power. The reality is that performance of dynamic adaption is greatly affected by the nature of workload phases and the power manager's policies. Adaptations provide power savings by setting performance to the minimum level required by the workload. If the performance demand of a workload were known in advance, then setting performance levels would be trivial. Since they are not known, the policy manager must estimate future demand based on the past. Existing power managers, such as those used in this study (Windows Vista and SLES Linux), act in a reactive mode. They can be considered as predictors which always predict the next phase to be the same as the last. This approach works well if the possible transition frequency up the adaptation is greater than the phase transition frequency of workload. Also, the cost of each transition must be low considering the frequency of transitions. In real systems, these requirements cannot currently be met. Therefore, the use of power adaptations does reduce performance to varying degrees depending on workload. The cost of mispredicting performance demand is summarized below.

- **Underestimate**: Setting performance capacity lower than the optimal value causes reduced performance. Setting performance capacity lower than the optimal value may cause increased energy consumption due to increased runtime. It is most pronounced when the processing element has effective idle power reduction.
- **Overestimate:** Setting performance capacity higher than the optimal value reduces efficiency as execution time is not reduced yet power consumption is increased. This case is common in memory-bound workloads.

- **Optimization Points:** The optimal configuration may be different depending on which characteristic is being optimized. For example, Energy·Delay may have a different optimal point compared to Energy·Delay^2.

3.4. Workloads

To represent typical user programs, we performed all experiments using SPEC CPU 2006, CPU 2000 and SYSmark® 2007. SPEC workloads include the complete suite of scientific and computing integer and floating point codes. The CPU 2006 version is included to give representative results for current applications. The CPU 2000 version is included due to its wide familiarity. The most significant difference between the two benchmark suites is working set size. Therefore, results obtained with CPU 2000 tend to be compute-bound while CPU 2006 results are more communication-bound. This difference is made clear in our experiments. Additionally, we present data from the SYSmark 2007 benchmark suite. This suite represents a wide range of desktop computing applications. The major categories are: e-learning, video creation, productivity, and 3D. The individual subtests are listed below. This suite is particularly important to the study of dynamic power adaptations since it provides realistic user scenarios which include user input and think time. Since current operating systems determine dynamic adaption levels using thread idle time, these user interactions must be replicated in the benchmark.

Table 3. SYSmark 2007

E-Learning	3D
Adobe® Illustrator®	Autodesk® 3Ds Max
Adobe Photoshop®	Google™ SketchUp
Microsoft PowerPoint®	
Adobe Flash®	
Productivity	Video Creation
Microsoft Excel®	Adobe After Effects®
Microsoft Outlook®	Adobe Illustrator
Microsoft Word®	Adobe Photoshop
Microsoft PowerPoint	Microsoft Media Encoder
Microsoft Project®	Sony Vegas
Winzip®	

3.5. Measurement Environment

To measure power consumption, we instrumented a system at a fine-grain level. For each subsystem we inserted a precision series resistor to measure current flow. We also measured voltage levels at the point of delivery. Using these quantities, it is possible to measure power consumption of a particular subsystem. We considered all major power subsystems, including: CPU core, memory controller, DRAM, PCIe, video, I/O bus, and disk. We performed all sampling at a rate of 1 KHz, using a National Instruments NIUSB-6259 [17]. This granularity allowed the measurement of most power phases which were sufficiently long to perform adaptations. Though shorter duration phases exist, current adaptation frameworks are not able to readily exploit them.

3.6. Phase Classification

To understand the effect of dynamic power adaptations on power and performance it is necessary to understand the phase behavior of workloads. Depending on the number of phase transitions a program contains, the performance cost to apply adaptations may vary. Phase transitions are inherent in programs, but are also introduced artificially through the operating system control of scheduling. A common example is context switching. Consider a single-processor system in which multiple software threads run simultaneously via multiplexing. Each thread runs until its allotted time expires. The operating system then saves the current system state and replaces the current thread with a waiting thread. Since the current phase of the various threads are not necessarily the same, the effective phase observed on the processor changes with each context switch. This presents a challenge since power adaptations are applied based on the hardware's perspective of the current program phase. In this paper we quantify program phase behavior by measuring phase characteristics of a wide range of workloads. We measure phases in terms of power consumption since adaptations are applied in order to control power. Also, this data is used to motivate the use of predictive power adaptations in a power-constrained environment. Therefore, it is necessary to know the duration and of intensity power allocation overshoot and undershoot.

In this study we defined a program phase as consecutive time events in which the power level of the subsystem is constant. The boundaries of a phase are specified by a change in the power level. The method we use for phase classification is similar to that used by Lau [12], in which a phase candidate is measured using the coefficient of variation (CoV = StandardDeviation/Average). We selected a CoV threshold using qualitative assessment and an error analysis. If the candidate phase has a CoV less than the threshold, then it is considered to be a phase. To find all possible phase lengths, we searched the data for the longest phases. Once we identified a portion of the data as being a phase, we removed that portion and no longer considered it in the search. The search continued with decreasing phase size until we classified all data. In our study we considered phase durations in the range of 1 ms to 1000 ms, as these represent cases useful for dynamic adaptation.

3.7. OS P-state Transition Latency

With the increasing availability and aggressiveness of power adaptations, it is becoming increasingly important to provide a mechanism for controlling the manner in which the adaptations are applied. In the case of Microsoft Windows® Vista® [16] , a wide range of controlling parameters is made available to users with a built-in utility. The major behaviors adjusted are frequency or p-state transitions, time thresholds for promotion/demotion, utilization thresholds for promotion/demotion, and p-state selection policy. These parameters may be changed at runtime in order to bias p-state selection for power savings, performance, or any intermediate level. Means are also provided for controlling c-state transitions, though these will not be discussed in the paper. A summary of critical parameters follows:

Timecheck: P-state change interval
Increase/Decrease Time: How long a thread must be in excess of the transition threshold before a transition is requested
Increase/Decrease percent: Transition threshold. A thread must exceed this threshold in order to be eligible for a transition.
Increase/Decrease Policy: P-state transition method. Three methods are available: Ideal, single, and rocket.

- Ideal: OS calculates ideal frequency based on current utilization.
- Single: new frequency is one step from current frequency.
- Rocket: go directly to maximum or minimum frequency.

4. RESULTS

4.1. Performance Effects

P-states and C-states impact performance in two ways: **Indirect** and **Direct**. Indirect performance effects are due to the interaction between active and idle cores. In the case of Quad-Core AMD processors, this is the dominant effect. When an active core performs a cache probe of an idle core, latency is increased compared to probing an active core. The performance loss can be significant for memory-bound (cache probe-intensive) workloads. Direct performance effects are due to the current operating frequency of an active core. The effect tends to be less compared to indirect, since operating systems are reasonably effective at matching current operating frequency to performance demand. These effects are illustrated in Figure 3.

Two extremes of workloads are presented: the compute-bound crafty and the memory-bound equake. For each workload, two cases are presented: fixed and normal scheduling. Fixed scheduling isolates indirect performance loss by eliminating the effect of OS frequency scheduling and thread migration. This is accomplished by forcing the software thread to a particular core for the duration of the experiment. In this case, the thread runs always run at the maximum frequency. The idle cores always run at the minimum frequency. As a result, crafty achieves 100 percent of the performance of processor that does not use dynamic power management. In contrast, the memory-bound equake shows significant performance loss due to the reduced performance of idle cores. We see direct performance loss in the green dashed and red dotted lines, which utilize OS scheduling of frequency and threads. Because direct performance losses are caused by suboptimal frequency in active cores, the compute-bound crafty shows a significant performance loss. The memory-bound equake actually shows a performance improvement for very low idle core frequencies. This is caused by idle cores remaining at a high frequency following a transition from active to idle.

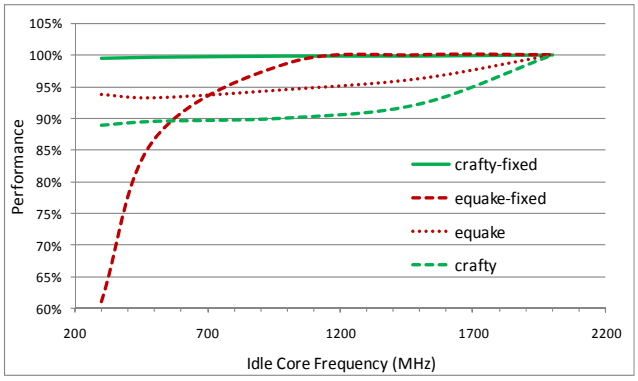


Figure 3. Direct and Indirect Performance Impact

4.1.1. Indirect Performance Effects

The amount of indirect performance loss is mostly dependent on the following three factors: Idle core frequency, OS p-state transition characteristics, and OS scheduling characteristics. The probe latency (time to respond to probe) is largely independent of idle core frequency above the "breakover" frequency ($Freq_B$). Below $Freq_B$ the performance drops rapidly at an approximately linear rate. This can be seen in Figure 3 as the dashed red line. The value of $Freq_B$ is primarily dependent on the inherent probe latency of the processor and the number of active and idle cores. Increasing the active core frequency increases the demand for probes and therefore increases $Freq_B$. Increasing the number of cores has the same effect. Therefore, multi-socket systems tend to have a higher $Freq_B$. Assuming at least one idle core, the performance loss increases as the ratio of active-to-idle cores increases. For an N-core processor, the worst-case is N-1 active cores with 1 idle core. To reduce indirect performance loss, the system should be configured to guarantee than the minimum frequency of idle cores is greater than or equal to $Freq_B$. Since the recommended configuration for Quad-Core AMD processors is "K8-style" probe response (CpuPrbEn=0) [2], the minimum idle core frequency is determined by the minimum p-state frequency. An explanation of these settings is provided later, in section 4.2.2. For the majority of workloads, these recommended settings yield less than 10 percent performance loss due to idle core probe latency.

The other factors in indirect performance loss are due to the operating system interaction with power management. These factors, which include OS p-state transition and scheduling characteristics, tend to mask the indirect performance loss. Ideally, the OS selects a high frequency p-state for active cores and a low frequency for idle cores. However, erratic workloads (many phase transitions) tend to cause high error rates in the selection of optimal frequency. Scheduling characteristics that favor load-balancing over processor affinity worsen the problem. Each time the OS moves a process from one core to another, a new phase transition has effectively been introduced. We give more details of OS p-state transitions and scheduling characteristics in the next section on direct performance effects.

4.1.2. Direct Performance Effects

Since the OS specifies the operating frequency of all cores (p-states), the performance loss is dependent on how the OS selects a frequency. To match performance capacity (frequency) to workload performance demand, the OS approximates demand by counting the amount of slack time a process has. For example, if a process runs for only 5ms of its 10 ms time allocation it is said to be 50 percent idle. In addition to the performance demand information, the OS p-state algorithm uses a form of low-pass filtering, hysteresis, and performance estimation/bias to select an appropriate frequency. These characteristics are intended to prevent excessive p-state transitions. This has been important historically since transitions tended to cause a large performance loss (PLL settling time, VDD stabilization). However, in the case of Quad-Core AMD processors and other recent designs, the p-state transition times have been reduced significantly. As a result, this approach may actually reduce performance for some workloads and configurations. See the red dotted equake and solid green crafty lines in Figure 3. These two cases demonstrate the performance impact of the OS p-state transition hysteresis.

As an example, consider a workload with short compute-bound phases interspersed with similarly short idle phases. Due to the low-pass filter characteristic, the OS does not respond to the short duration phases by changing frequency. Instead, the cores run at reduced frequency with significant performance loss. In the pathologically bad case, the OS switches the frequency just after the completion of each active/idle phase. The cores run at high frequency during idle phases and low frequency in active phases.

Power is increased while performance is decreased. OS scheduling characteristics exacerbate this problem. Unless the user makes use of explicit process affinity or an affinity library, some operating systems will attempt to balance the workloads across all cores. This causes a process to spend less contiguous time on a particular core. At each migration from one core to another there is a lag from when the core goes active to when the active core has its frequency increased. The aggressiveness of the p-state setting amplifies the performance loss/power increase due to this phenomenon. Fortunately, recent operating systems such as Microsoft Windows Vista provide means for OEMs and end users to adjust the settings to match their workloads/hardware (see powercfg.exe).

4.2. Workload Power Characterization

4.2.1. Subsystem Power Breakdown

In this section we consider average power consumption levels across a range of workloads. We draw two major conclusions for desktop workloads: the core is largest power consumer, and contains the most variability across workloads. Though other subsystems, such as memory controller and DIMM, have significant variability within workloads, only the core demonstrates significant variability in average power across desktop workloads. Consider Figure 4: while average core power varies by as much as 57 percent, the next most variable subsystem, DIMM, varies by only 17 percent. Note, this conclusion does not hold for server systems and workloads in which much larger installations of memory modules cause greater variability in power consumption. The cause of this core power variation can be attributed to a combination of variable levels of thread-level parallelism and core-level power adaptations. In the case of 3D, the workload is able to consistently utilize multiple cores.

At the other extreme, the productivity workload rarely utilizes more than a single core. Since Quad-Core AMD processor power adaptations may be applied at the core level, frequency reduction achieves significant power savings on the three idle cores. As a result, the productivity workload consumes much less power than the 3D workload. The remaining workloads offer intermediate levels of thread-level parallelism and therefore have intermediate levels of power consumption. Also note that this level of power reduction is due only to frequency scaling. With the addition of core-level voltage scaling, the variation/power savings is expected to increase considerably.

We draw a slightly different conclusion for server workloads and systems. Due to the presence of large memory subsystems, DIMM power is a much larger component. Also, larger working sets such as those found in SPEC CPU2006 compared to SPEC CPU2000 shift power consumption from the cores to the DIMMs. Consider CPU2000 in Figure 5 and CPU20006 in Figure 6. Due to comparatively small working sets, CPU2000 workloads are able to achieve high core power levels. The reason is that, since the working set fits easily within the cache, the processor is able to maintain very high levels of utilization. This is made more evident by the power increases seen as the number of simultaneous threads is increased from 1 to 4. Since there is less performance dependence on the memory interface, utilization and power therefore continue to increase as threads are added. Result is different for CPU2006 workloads. Due to the increased working set size of these workloads, the memory subsystem limits performance. Therefore, core power is reduced significantly for the four-thread case. Differences for the single-thread case are much less due to a reduced dependency on the memory subsystem. The shift in utilization from the core to the memory subsystem can be seen clearly in Figure 7. For the most compute-bound workloads, core power is five times larger than DIMM power. However, as the workloads become more memory-bound, the power levels converge to the point where DIMM power slightly exceeds core power.

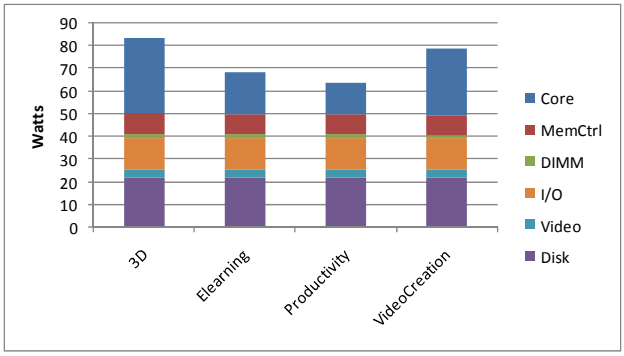


Figure 4. Desktop Subsystem Power Breakdown

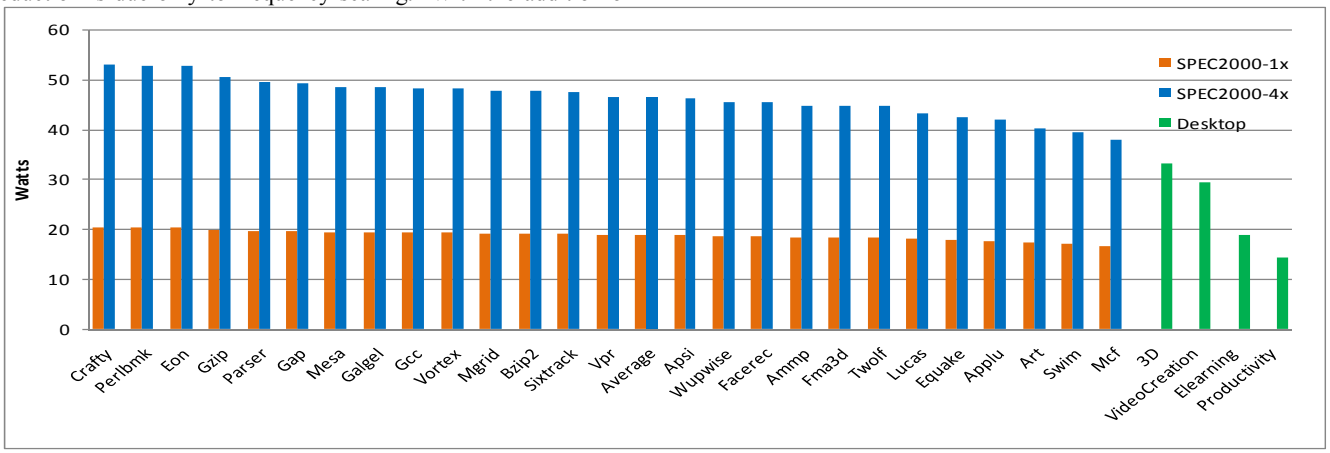


Figure 5. CPU2000 Average Core Power

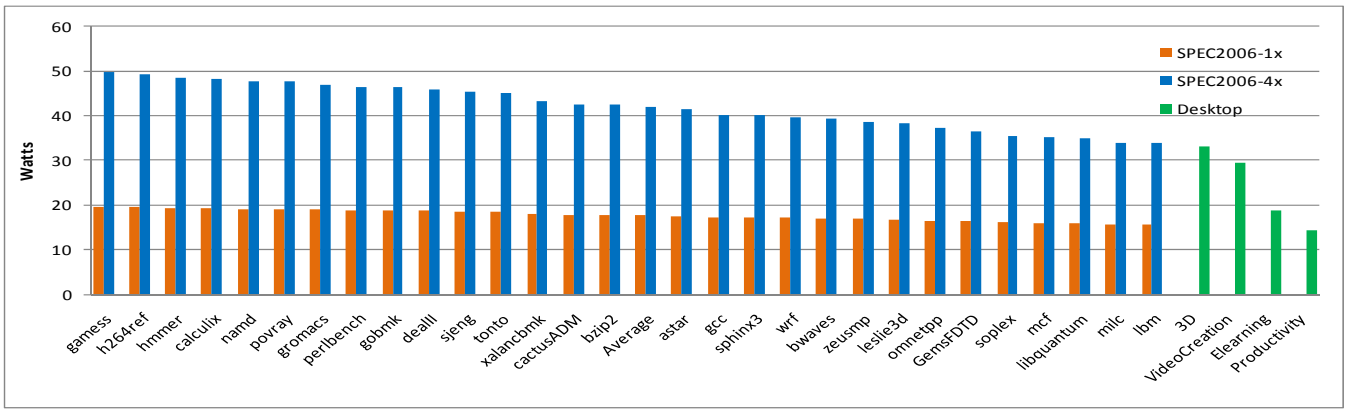


Figure 6. CPU2006 Average Core Power

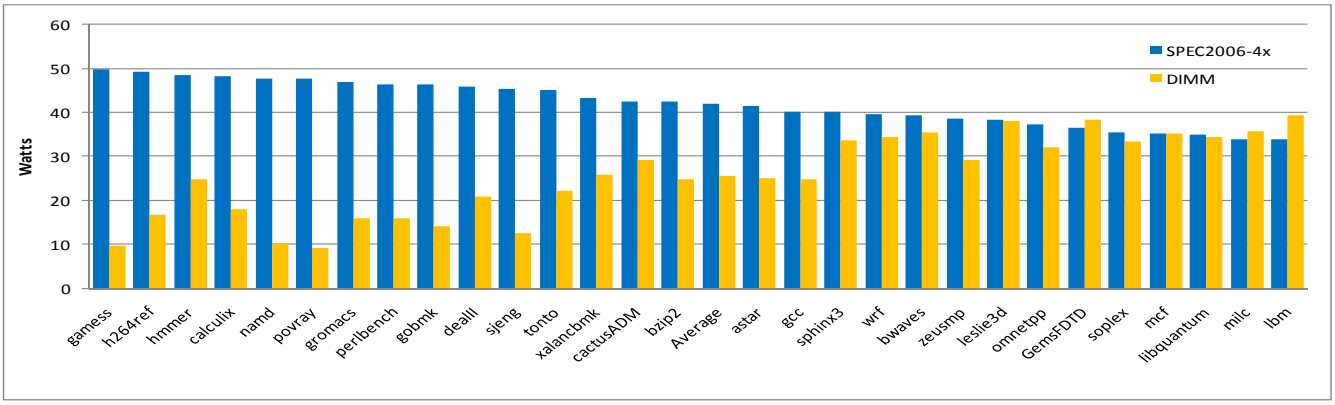


Figure 7. CPU2006 Average Core vs. DIMM Power

4.2.2. Core Power Phase Characteristics

The previous section demonstrates the core as having the most variable average power consumption across the various subsystems. In this section we present the intra-workload phase characteristics which contribute to the variation. These results are attributable to the three dominant components of power adaptation: hardware adaptation, workload characteristics, and OS control of adaptations. In Figure 8 we present a distribution of the phase length of power consumption for desktop workloads. We draw two major conclusions: the operating system has a significant effect on phase length and interactive workloads tend to have longer phases.

First, the two spikes at 10 ms and 100 ms show the effect of the operating system. These can be attributed to the period timer tick of the scheduler and p-state transitions requested by the operating system. In the case of Microsoft Windows Vista, the periodic timer tick arrives every 10 ms. This affects the observed power level since power consumed in the interrupt service routine is distinct from "normal" power levels. In the case of high-IPC threads, power is reduced while servicing the interrupt, which typically has a relatively low-IPC due to cold-start misses in the cache and branch predictor. In the case of low-power or idle threads, power is increased since the core must be brought out of one or more power saving states in order to service the interrupt. This is a significant problem for power adaptations since the timer tick is not workload dependent. Therefore, even a completely idle system must "wake up" every 10 ms to service an interrupt, even though no useful work is being completed. Also, 10 ms phase transitions are artificially introduced due to thread migration. Since thread scheduling is performed on timer tick intervals, context switches, active-to-idle, and idle-to-active transitions occur on 10 ms intervals. The 100 ms phases can be explained by the OS's application of p-state transitions. Experimentally, it can be shown that the minimum rate at which the operating system will request a transition from one p-state to another is 100 ms. When p-state transitions are eliminated, the spike at the 100 ms range of Figure 8 is eliminated.

The second conclusion from Figure 8 is that interactive workloads have longer phase durations. In the case of 3D and video creation workloads, a significant portion of time is spent in compute-intensive loops. Within these loops, little or no user interaction occurs. In contrast, the productivity and e-learning workloads spend a greater percentage of the time receiving and waiting for user input. This translates into relatively long idle phases which are evident in the lack of short duration phases in Figure 8.

This is further supported by Figures 9 through 12, which group the most common phases by combinations of amplitude and duration. Note that all phases less than 10 ms are considered to be 10 ms. This simplifies presentation of results and is reasonable since the OS does not apply adaptation changes any faster than 10 ms. These figures show that the highest power phases only endure for a short time. These phases, which are present only in 3D and – to a much lesser degree – in video creation, are only

possible when multiple cores are active. We attribute the lack of long duration high power phases to two causes: low percent of multithreaded phases and higher IPC dependence during multithreaded phases. The impact of few multithreaded phases is expected and has been demonstrated in Figures 5 and 6. The dependence on IPC for phase length increases as the number of active cores increases. Figure 2 from section 3.2.2 shows that power increases significantly as IPC increases from 0 to 3. Assuming active cores running in the P0 (highest frequency) state, IPC has the largest effect on power consumption since IPC varies much more quickly (nanoseconds) than transitions between power states (10's of milliseconds). Consistent power consumption levels are less likely as the number of active cores increases.

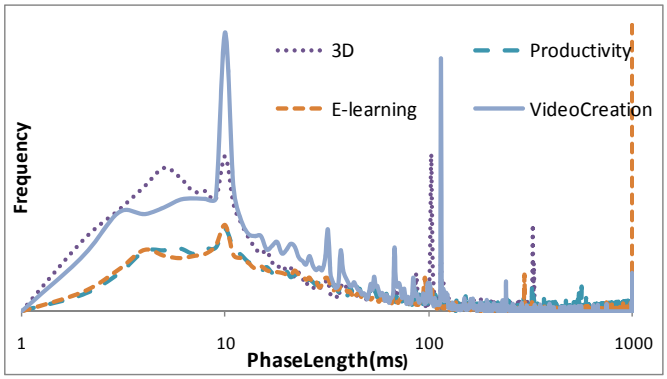


Figure 8. Core Power Phase Duration

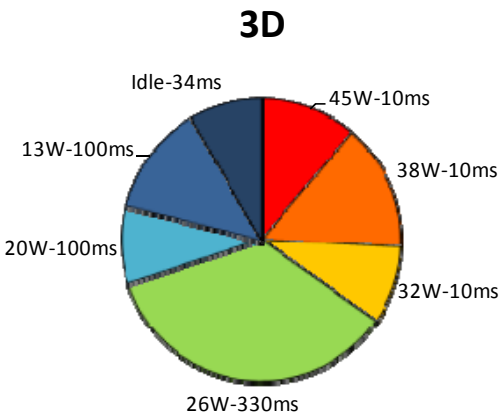


Figure 9. Core Power Phases – 3D

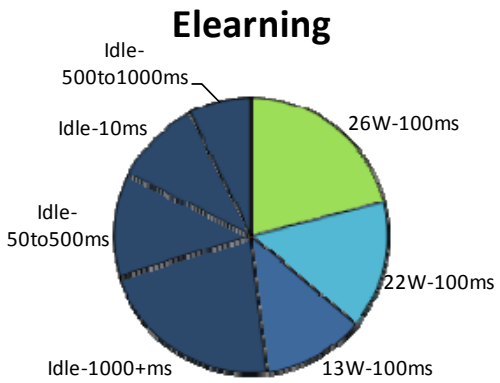


Figure 10. Core Power Phases – E-learning

4.3. Identifying Optimal Adaption Settings

In this section, we present results to show the effect that dynamic adaptations ultimately have on performance and power consumption. We obtained all results on a real system, instrumented for power measurement. The two major areas presented are probe sensitivity (indirect) and operating system effects (direct).

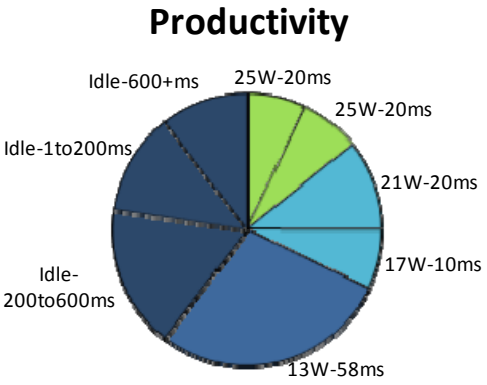


Figure 11. Core Power Phases – Productivity

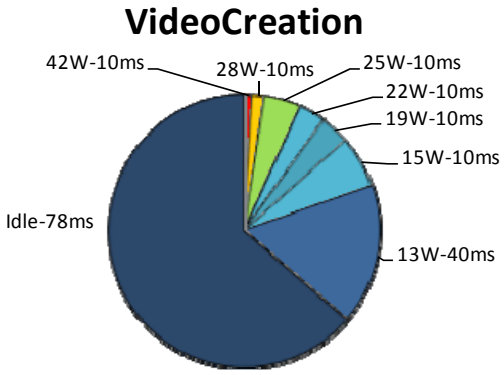


Figure 12. Core Power Phases – Video Creation

First we consider probe sensitivity of SPEC CPU2006. Table 4 shows performance loss due to the use of p-states. In this experiment the minimum p-state is set below the recommended performance breakover point for probe response. This emphasizes the inherent sensitivity workloads have to probe response. Operating system frequency scheduling is biased towards performance by fixing active cores at the maximum frequency and idle cores at the minimum frequency. These results suggest that floating point workloads tend to be most sensitive to probe latency. However, in the case of SPEC CPU2000

workloads, almost no performance loss is shown. The reason, as shown in section 4.3.1, is that smaller working set size reduces memory traffic and, therefore, the dependence on probe latency. For these workloads only swim, equake, and eon showed a measureable performance loss.

Next we show that by slightly increasing the minimum p-state frequency it is possible to recover almost the entire performance loss. Figure 13 shows an experiment using a synthetic kernel with very high probe sensitivity with locally and remotely allocated memory. The remote case simply shows that the performance penalty of accessing remote memory can obfuscate the performance impact of minimum p-state frequency. The indirect performance effect can be seen clearly by noting that performance increases rapidly as the idle core frequency is increased from 800 MHz to approximately 1.1 GHz. This is a critical observation since the increase in power for going from 800 MHz to 1.1 GHz is much smaller than the increase in performance. The major cause is that static power represents a large portion of total power consumption. Since voltage dependence exists between all cores in a package, power is only saved through the frequency reduction. There is no possibility to reduce static power since voltage is not decreased on the idle cores.

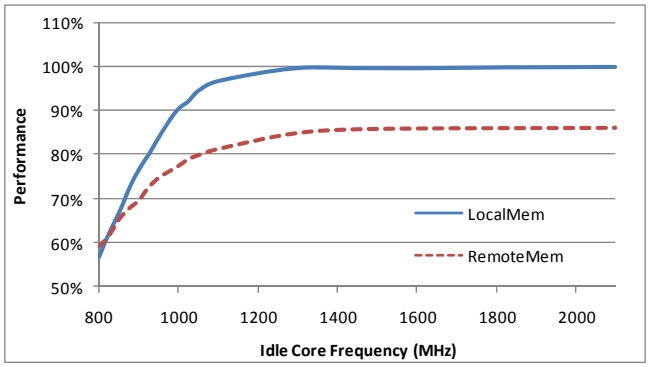


Figure 13. Remote and Local Probe Sensitivity

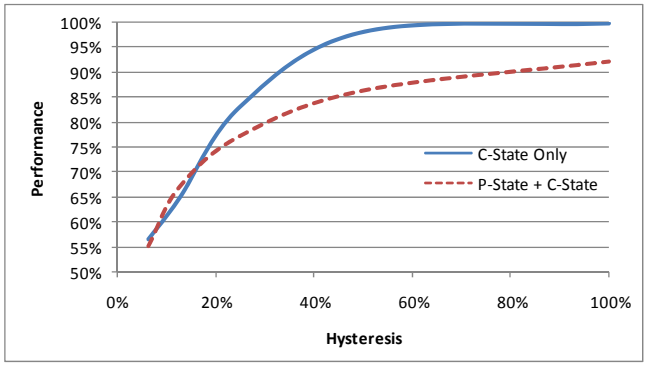


Figure 14. C-state vs. P-state Performance

Using the same synthetic kernel we also isolate the effect of p-states from c-states. Since the p-state experiments show that indirect performance loss is significant below the breakover point, we now consider c-state settings that do not impose the performance loss. To eliminate the effect of this performance loss we make use of K8-mode probe response. In this mode, idle cores increase their frequency before responding to probe requests. To obtain an optimal tradeoff between performance and power settings, this setting mode can be modulated using hysteresis, implemented by adjusting a hysteresis timer. The timer specifies how long the processor remains at the increased frequency before returning to the power saving mode. The results are shown in Figure 14. The blue line represents the performance loss due to slow idle cores caused by the application of c-states only. Like the p-state experiments, performance loss reaches a clear breakpoint. In this case, the breakover point represents 40 percent of the maximum architected delay. Coupling c-states with p-states, the red shows that the breakover point is not as distinct since significant performance loss already occurs. Also, like the p-state experiments, setting the hysteresis timer to a value of the breakover point increases performance significantly while increasing power consumption on slightly.

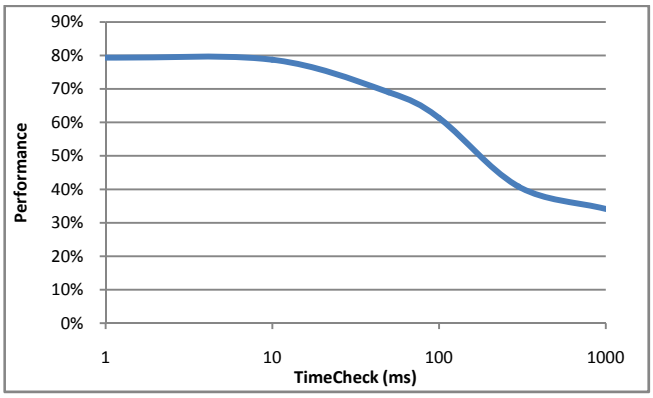


Figure 15. Varying OS P-state Transition Rates

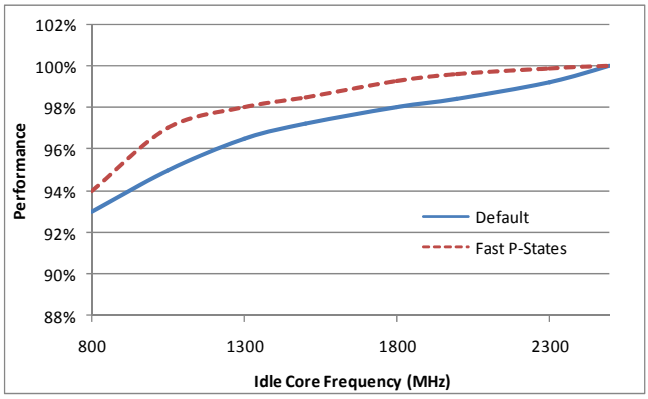


Figure 16. Effect of Increasing P-state Transition Rate

Next we consider the effect of operating system tuning parameters for power adaptation selection. In order to demonstrate the impact of slow p-state selection, we present Figure 15. The effect is shown by varying a single OS parameter while running a phase transition intensive kernel. In this graph, the TimeCheck value is varied from 1 ms to 1000 ms. TimeCheck controls how often the operating system will consider a p-state change. We found two major issues: minimum OS scheduling quanta and increase/decrease filter.

First, performance remains constant when scaling from 1 us to 10 ms (< 1 ms not depicted). We attribute this to the OS implementation of scheduling. For Microsoft Windows Vista, all processes are scheduled on the 10 ms timer interrupt. Setting TimeCheck to values less than 10 ms will have no impact since p-state changes, like all process scheduling, occur only on 10-ms

boundaries. Second, even at the minimum TimeCheck value, performance loss is at 80 percent. The reason is that other settings become dominant below 10 ms. In order for a p-state transition to occur the workload must overcome the in-built low-pass filter. This filter is implemented as a combination of two thresholds: increase/decrease percent and increase/decrease time. The percent threshold represents the utilization level that must be crossed in order to consider a p-state change. The threshold must be exceeded for a fixed amount of time specified by increase/decrease time. Since the increase time is much longer than TimeCheck (300 ms vs. 10 ms), significant performance is lost even at the minimum setting.

To reduce the impact of slow p-state transitions we select OS settings that increase transition rates. In a general sense, frequent p-state transitions are not recommended due to the hardware transition costs. However, our experiments have shown that the performance cost for slow OS-directed transitions is much greater than that due to hardware. This can be attributed to the relatively fast hardware transitions possible on Quad-Core AMD processors. Compared to OS transitions which occur at 10 ms intervals, worst-case hardware transitions occur in a matter of 100's of microseconds. Figure 16 shows the effect of optimizing p-state changes to the fastest rate of once every 10 ms. The probe-sensitive equake is shown with and without "fast p-states." This approach yields between 2 percent and 4 percent performance improvement across the range of useful idle core frequencies. As we will see in the next section, this also improves power savings by reducing active-to-idle transition times.

Table 4. Performance Loss Due to Low Idle Core Frequency

SPEC CPU 2006 - INT			
perlbench	-0.8%	sjeng	0.0%
bzip2	-1.0%	libquantum	-7.0%
gcc	-3.6%	h264ref	-0.8%
mcf	-1.8%	omnetpp	-3.7%
gobmk	-0.3%	astar	-0.5%
hmmer	-0.2%		
SPEC CPU 2006 - FP			
bwaves	-5.6%	soplex	-6.7%
games	-0.6%	povray	-0.5%
milc	-7.9%	calculix	-0.6%
zeusmp	-2.1%	GemsFDTD	-5.9%
gromacs	-0.3%	tonto	-0.6%
cactusADM	-2.6%	lbm	-5.6%
leslie3D	-6.0%	wrf	-3.2%
namd	-0.1%	sphinx3	-5.6%
dealII	-1.3%		

4.4. Power and Performance

In this section we present results for p-state and c-state settings which reflect the findings of the previous sections. In this case we study the Microsoft Windows Vista operating system running desktop workloads. This approach gives the highest exposure to the effect the operating system has on dynamic adaptations. By choosing desktop workloads, the number of phase transitions and, therefore, OS interaction is increased. Since these workloads model user input and think times, idle phases are introduced. These idle phases are required for OS study since the OS makes use of idle time for selecting the operating point. Also, Microsoft Windows Vista exposes tuning parameters to scale the built-in adaptation selection algorithms for power savings versus performance. Table 5 shows power and performance results for SYSmark 2007 using a range of settings chosen based on the results of the previous sections. In order to reduce p-state performance loss, the idle core frequency is set to 1250 MHz. To prevent c-state performance loss, K8-mode is used with the hysteresis time set above the breakover point. Also, C1e mode is disabled to prevent obscure idle power savings due to the architected p-states and c-states.

Two important findings are made regarding adaption settings. First, setting power adaptations in consideration of performance bottlenecks reduces performance loss while retaining power savings. Second, reducing OS p-state transition time increases performance and power savings. Table 5 shows the resultant power and performance for a range of hardware and software settings. We show that performance loss can be limited to less than 10 percent for any individual subtest while power savings average 45 percent compared to not using power adaptations. The effect of workload characteristics is evident in the results. E-learning and productivity show the greatest power savings due to their low utilization levels. These workloads frequently use only a single core. At the other extreme, 3D and video creation have less power savings and a greater dependence on adaption levels. This indicates that more parallel workloads have less potential benefit from p-state and c-state settings, since most cores are rarely idle. For those workloads, idle power consumption is more critical. These results also point out the limitation of existing power adaptation algorithms. Since current implementations only consider idle time rather than memory-boundedness, the benefit of p-states is underutilized.

Additionally, we show the effect of adjusting operating system p-state transition parameters. Columns Fast and Fast-perf represent cases in which p-state transitions occur at the fastest rate and bias towards performance respectively. Since existing operating system such as Microsoft Windows XP and Linux bias p-state transitions toward performance, these results can be considered representative for those cases. The default configuration of Microsoft Windows Vista biases toward reducing the number of p-state transitions. Since the normal case, below, uses that configuration, performance and power are impacted accordingly.

5. CONCLUSION

In this paper we have presented a power and performance analysis of dynamic power adaptations in a Quad-Core AMD processor. We have shown that performance and power are greatly affected by direct and indirect characteristics. Direct effects are composed of operating system thread and frequency scheduling. We show that slow transitions by the operating system between idle and active operation cause significant performance loss. The effect is greater for compute-bound workloads which would otherwise be unaffected by power adaptations. Slow active-to-idle transitions also cause reduced power savings. Indirect effects due to shared, power-managed resources such as caches can greatly reduce performance if idle core frequency reductions are not limited sufficiently. These effects are more pronounced in memory-bound workloads since performance is directly related to accessing shared resources between the active and idle cores. Finally, we show that performance loss and power consumption can be minimized through careful selection of hardware adaptation and software control parameters. In the case of Microsoft Windows Vista running desktop workloads,

performance loss using a naïve OS configuration is less than 8 percent on average for all workloads while saving an average of 45 percent power. Using an optimized OS configuration, performance loss drops to less than 2 percent with power savings of 30 percent.

Table 5. Power/Performance Study: SYSmark® 2007

	P-States	PerformanceLoss	PowerSavings
E-Learning	Normal	8.80%	43.10%
VideoCreation	Normal	6.20%	44.70%
Productivity	Normal	9.50%	45.30%
3D	Normal	5.90%	45.90%
E-Learning	Fast	6.40%	45.90%
VideoCreation	Fast	5.20%	46.10%
Productivity	Fast	8.00%	47.80%
3D	Fast	4.60%	48.20%
E-Learning	Fast-perf	1.50%	32.90%
VideoCreation	Fast-perf	1.80%	25.40%
Productivity	Fast-perf	2.50%	27.90%
3D	Fast-perf	1.40%	35.10%

6. ACKNOWLEDGEMENTS

This research was supported in part by Advanced Micro Devices and NSF Award numbers 0429806 and 0702694. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

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

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

[3] Bircher, W. L. Measurement Based Power Phase Analysis of a Commercial Workload. Workshop on Unique Chips and Systems (Austin, Texas, March 2006).

[4] Bircher, W. L. and John, L. Power Phase Availability in a Commercial Server Workload. International Symposium on Low Power Electronics and Design (Tegernsee, Germany, October 2006).

[5] Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. The Case for Power Management in Web Servers. IBM Research, Austin TX 78758, USA. www.research.ibm.com/arl

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

[7] Feng, X., Ge, R., and Cameron, K. W. Power and Energy Profiling of Scientific Applications on Distributed Systems. International Parallel & Distributed Processing Symposium, pages 34-50 (Denver, Colorado, April 2005).

[8] Hanson, H., Keckler, S.W. Power and Performance Optimization: A Case Study with the Pentium M Processor. The Austin Center for Advanced Studies Conference (February 2006).

[9] Hanson, H., Keckler, S.W., Rajamani, K., Ghiasi, S., Rawson, F., and Rubio, J. Power, Performance, and Thermal Management for High-Performance Systems. 3rd Workshop on High-Performance, Power-Aware Computing, held in conjunction with 21st Annual International Parallel & Distributed Processing Symposium (Long Beach, California, March 2007).

[10] Isci, C., Buyuktosunoglu, A., Cher, C., Bose, P., and Martonosi, M. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In Proceedings of the 39th Annual IEEE/ACM international Symposium on Microarchitecture (Orlando, Florida, December 2006).

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

[12] Lau, J., Schoenmackers, S., and Calder, B. Structures for Phase Classification. IEEE International Symposium on Performance Analysis of Systems and Software, pages 57-67 (Austin, Texas, March 2004).

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

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

[15] Mahesri, A. and Vardhan, V. Power Consumption Breakdown on a Modern Laptop. Workshop on Power Aware Computing Systems, 37th International Symposium on Microarchitecture (Portland, Oregon, December 2004).

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

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

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

[19] Inside Barcelona: AMD's Next Generation http://www.realworldtech.com . November 2007.

[20] Siddah, S., Pallipadi, V., and Van de Ven, A. Getting Maximum Mileage Out of Tickless. The Linux Symposium. (Ottawa, Canada, June 2007).