Complete System Power Estimation using Processor Performance Events

W. Lloyd Bircher and Lizy K. John

Abstract— This paper proposes the use of microprocessor performance counters for online measurement of complete system power consumption. The approach takes advantage of the "trickle-down" effect of performance events in microprocessors. While it has been known that CPU power consumption is correlated to processor performance, the use of well-known performance-related events within a microprocessor such as cache misses and DMA transactions to estimate power consumption in memory and disk and other subsystems outside of the microprocessor is new. Using measurement of actual systems running scientific, commercial and productivity workloads, power models for six subsystems (CPU, memory, chipset, I/O, disk and GPU) on two platforms (server and desktop) are developed and validated. These models are shown to have an average error of less than 9% per subsystem across the considered workloads. Through the use of these models and existing on-chip performance event counters, it is possible to estimate system power consumption without the need for power sensing hardware.

1. INTRODUCTION

In order to improve microprocessor performance while limiting power consumption, designers increasingly utilize dynamic hardware adaptations. These adaptations provide an opportunity to extract maximum performance while remaining within temperature and power limits. Two of the most common examples are dynamic voltage/frequency scaling (DVFS) and clock gating. With these adaptations it is possible to reduce power consumption and therefore chip temperature, by reducing the amount of available performance. Due to the thermal inertia in microprocessor packaging, detection of temperature changes may occur significantly later than the power events causing them. Rather than relying on relatively slow temperature sensors for observing power consumption it has been demonstrated [3][16][24][5] that performance counters are effective proxies for power measurement. These counters provide a timely, readily accessible means of observing power consumption in real systems.

This paper extends the concept of using performance events as proxies for power measurement beyond the microprocessor to various computer subsystems. Models are presented for six subsystems: microprocessor, graphics processing unit (GPU), chipset, memory, I/O and disk. Though microprocessors are typically the largest consumers of power, other subsystems constitute 40%-60% of total power. By providing a means for power management policies to consider these additional subsystems it is possible to have a significant effect on power and temperature. In data and computing centers, this can be a valuable tool for keeping the center within temperature and power limits [35]. Further, since this approach utilizes existing microprocessor performance counters, the cost of implementation is small.

This approach is distinct since it uses events local to the processor, eliminating the need for sensors spread across various parts of the system and corresponding interfaces. Lightweight, adaptive systems can easily be built using models of this type. This study shows that microprocessor performance events can accurately estimate total system power. By considering the propagation of power inducing events within the various subsystems, a modest number of performance events for modeling complete system power are identified. Power models for two distinct hardware platforms are presented: a quad-socket server and a multi-core desktop. The resultant models have an average error of less than 9% across a wide range of workloads including SPEC CPU, SPECJbb, DBT-2, SYSMark and 3DMark. Though power models exist for common computer subsystems, these models rely on events *local* to the subsystem for representing power, which are typically measured using sensors/counters within the subsystem. Our emphasis is on creating a model using no additional sensors or counters other than what the performance engineers have already incorporated.

2. COMPLETE SYSTEM POWER MODEL

Trickle-down power modeling [6] provides an accurate representation of complete-system power consumption using a simple methodology. The approach relies on the broad visibility of system-level events to the processor. This allows accurate, performance counter-based models to be created using events local to the processor. These local events can be measured using ubiquitous performance counters found in all modern microprocessors. Local events are preferred since power models can be built using a single interface. There is no need to create interfaces to multiple devices and subsystems that have inconsistent or incomplete performance counter APIs (Application Programming Interface). It is particularly common at the system level since components are often designed by multiple vendors.

Trickle-down modeling also addresses hardware costs in systems implementing direct measurement. Rather than providing sensors and power measurement hardware for multiple subsystems, measurement need only be implemented on a single system during the design stage. The model is created based on measurement from a small number of systems which allows power measurement hardware to be eliminated from the final product. This paper focuses on the study of the subsystems with the largest variation. Areas not accounted for in this study, such as cooling and power supply inefficiency, are well known and easily accountable. For example power

supply losses can be accounted for as a function of total power [42].

While the trickle-down approach simplifies power modeling of complete systems it requires a modest knowledge of subsystem-level interaction. The effectiveness of the model at capturing system-level power is determined by the selection of comprehensive performance events. Some events such as top-level cache or memory accesses are intuitive. A miss in the first level cache will necessarily generate traffic in higher level caches and or the memory subsystem. Other events such as those found in I/O devices are not as obvious. Consider the system diagram in Figure 1.

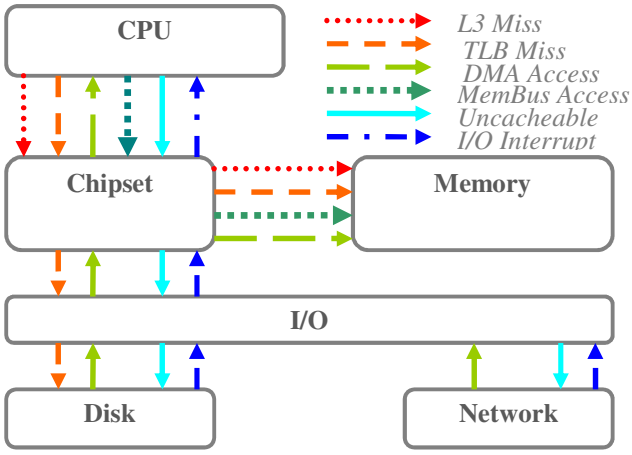

Figure 1. Propagation of Performance Events

The depiction in Figure 1 represents a general server type system. Specific systems will have other components which should appropriately be added to the diagram as needed. The arrows flowing outward from the processor represent events that originate in the processor and trickle-down to other subsystems (L3 Miss, TLB Miss, MemBus Access and Uncacheable Access). Arrows flowing inward such as DMA (Direct Memory Access) or bus master access and I/O interrupts may not be directly generated by the processor, but are nevertheless visible. Since DMA access is typically performed to addresses marked as cacheable by the processor, they can be observed in the standard cache access metrics. To distinguish DMA accesses by a particular device, events should be qualified by address range. Each device typically uses a private range of addresses in system memory for DMA access. Similarly interrupts from multiple devices can be distinguished by interrupt number or address in the case of message signaled interrupts.

In this paper, we present the application of the model to two different systems, a quad-socket Intel server and an AMD dual-core with a graphics processing unit (GPU). Figure 1 adequately represents the quad-socket server, while the GPU has to be added to the depiction in Figure 1 in order to adequately represent the AMD dual-core platform.

Figure 2 illustrates the iterative modeling procedure that we developed to estimate power from performance events. This procedure utilizes linear and polynomial regression techniques to build power models for individual subsystems. The user identifies workloads which target a particular subsystem (cache, system memory, disk) and performs regression modeling using performance events as inputs. The model is then applied to a larger set of workloads to confirm accuracy and the lack of outlier cases. Depending on the outcome, the process is repeated with alternate performance events as inputs. Though an exhaustive search of performance events can be performed, a rapid solution is found when events are selected with high correlation to subsystem activity. The modeling process in Figure 2 involves several steps:

1. Measure subsystem-level power using subset of workloads. Begin with simple, easy-to-run workloads.

2. Confirm that Coefficient of Variation is greater than a threshold α for the chosen workload. The simplest workloads often do not generate sufficient power variation for model tuning. For example consider any of the cache-resident workloads in SPEC CPU 2000 which generate little or no activity in subsystems outside of the processor cores such as memory. Tuning the model based on these low-variation workloads may cause the process to include performance events that do not correlate well with power.

3. Based on basic domain knowledge, choose performance events, measureable by performance counters that are most relevant to the subsystem in question. Choose counters that are expected to "trickle-down" to other subsystems. The pool of candidate performance counters may need to be expanded if sufficient accuracy is not achieved.

4. Using the selected performance counter events as the input variables and subsystem power as the output variable, perform linear regression modeling. For example, in the general linear equation $y = mx + b$, vary the coefficients m and b until the sum-of-squares error is minimized. Multiple linear or polynomial regression may be used in subsequent iterations of algorithm if sufficient accuracy is not obtained using simple linear regression.

5. Using a subset of workloads calculate average error per sample. If less than the desired ρ% error cannot be achieved, a new performance event must be chosen. One should select ρ depending on the required model accuracy and time required for solution. Setting ρ to a low (restrictive) value may extend the time taken to reach a solution. It may also prevent the process from finding a solution.

6. Assess the representativeness of the model by graphically comparing modeled versus measured power. This avoids the case in which statistical assessment cannot detect major errors such as those seen in Anscombe's Quartet [1].

7. Using complete set of workloads calculate average error per sample. If less than the desired δ % error cannot be achieved, choose a new performance event. Like ρ, δ is selected according the accuracy and time-to-solution requirements.

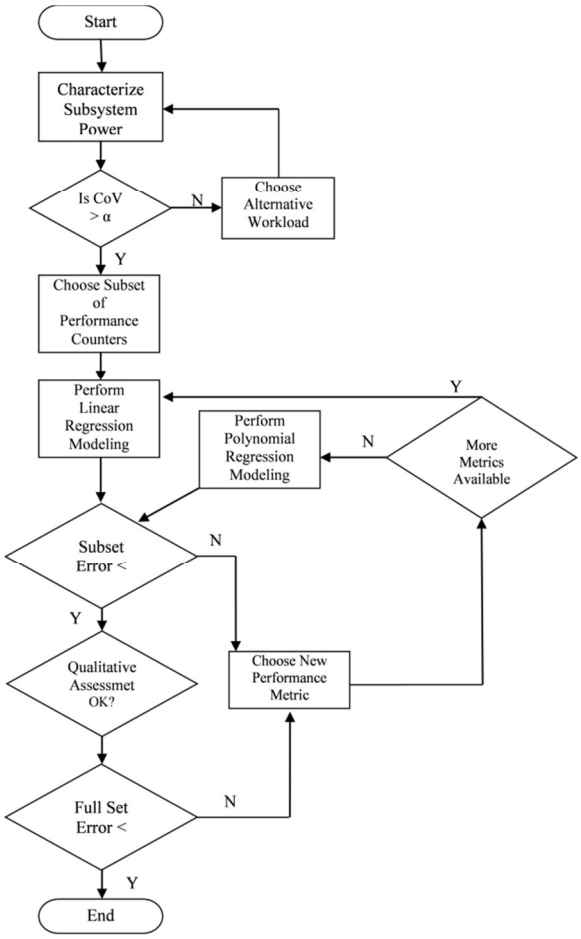

Figure 2. Trickle-Down Modeling Process

This modeling process is applied to two platforms, (i) a server and (ii) a desktop/embedded system. In section 3, the application of the process to an Intel quad-CPU server and illustrate the creation of the model as well as its validation is presented. In section 4, the feasibility of extending the model to a completely different system with different components is presented, illustrating the applicability of the model to other platforms. The second platform we use is a dual-core AMD system with a graphics processing unit (GPU).

3 APPLICATION TO SERVER PLATFORM

In this section, we describe the application of the proposed power modeling process to a quad-CPU system. The target system is composed of a quad-CPU server with two hardware threads per CPU,

3.1 SUBSYSTEM DESCRIPTION

The server system is composed of five major subsystems whose power can be separately measured: CPU, chipset, memory, I/O and disk. The CPU subsystem is composed of four Pentium IV Xeon 2.8 GHz MP processors. Chipset is defined as processor interface chips not included in other subsystems. The memory subsystem includes memory controller and DRAM power. I/O includes PCI buses and all devices attached to them. The disk subsystem is composed of two SCSI disks. The system components are listed in Table 1.

Table 1. Server System Description

Platform Segment	Server
Model	IBM x440
Processor(s)	Quad-socket 130nM 2.8GHz
Memory	8GB DDR-200
Power Management	CPU Clock Gating and DRAM Power Down
Graphics	Rage ProXL
Observable Subsystems	CPU, Chipset, Memory, I/O and Disk

3.2 POWER MEASUREMENT

To measure power in the five subsystems, resistors connected in series with the power source are employed. This allows each subsystem to be measured independently of the others. This is important since the subsystems must be isolated in order to accurately correlate power consumption in the subsystem to performance events in the processor. The voltage drop across the resistor is directly proportional to the power being consumed in the subsystem. This voltage drop is captured using data acquisition hardware in a separate workstation. Ten thousand samples are taken each second and are then averaged for relation to performance counter samples taken at the rate of once per second.

Since the performance counter samples are taken by the target system itself, a synchronization signal is included to match data from the two sources. At each sampling of the target performance counters, a single byte is sent to a USB serial port attached to the target system. The transmit line of the serial port is sampled by the data acquisition hardware along with the other power data. The single byte of data acts as a synchronization pulse signature. Then using the synchronization information, the data is analyzed offline using software tools.

3.3 PERFORMANCE MEASUREMENT

To gather a record of performance events in the processor, the Pentium IV's on-chip performance monitoring counters are periodically sampled. Sampling is performed on each processor at a rate of once per second. The total count of various events is recorded and the counters are cleared. Software access to the performance counters is provided by the Linux perfctr [25] device driver. As described in the power measurement section, a synchronization signal is asserted at each performance counter sample.

3.4 WORKLOAD SELECTION

The selection of workloads is driven by two major factors: the workload's effectiveness at utilizing particular subsystems and a diverse set of behaviors across all workloads. The first requirement is important for development and tuning of the power models. The second is required to validate the models.

In order to meet the requirement of subsystem utilization, the power measurement system is employed. Workloads are chosen based on their apparent utilization of a subsystem. Then actual power measurement is done to verify the selection. It is found that high subsystem utilization is difficult to achieve using only conventional workloads. As a result, small synthetic workloads are created that are able to sufficiently utilize the subsystems. Additionally, multiple instances of single-threaded workloads such as SPEC CPU 2000 are

combined to produce high utilization. Since the target system is composed of a quad-socket SMP with two hardware threads per processor, it is found that most workloads saturate (no increased subsystem power consumption) with eight threads.

In addition to utilizing a particular subsystem, it is necessary to have sufficient variation within the workload for training of the models. In the case of the 8-thread workloads, the start of each thread is staggered by a fixed time, approximately 30 seconds, to generate a range of activity levels (i.e 1 thread running at certain times, 2 threads at other times, etc). This broad range of utilization ensures that the models are not only valid within a narrow range of utilization. Also, this ensures a proper relationship between power and the observed metric. Without a sufficiently large range of samples, complex power relationships may appear to be simple linear ones.

3.5 MODEL VALIDATION

Eleven workload are used for validation: eight from the SPEC CPU 2000 benchmark suite [38], two commercial server-type (SPECjbb and DBT-2) and a synthetic disk workload. The SPEC workloads are computationally intensive scientific applications intended to stress the CPU and memory subsystems. The only access to other subsystems by these workloads occurs at program initialization and completion. In this study only single or multiple instances of identical workloads are considered. Two commercial workloads DBT-2 [31] and SPECjbb [39] are used to create system-level activity. DBT-2 is intended to approximate the TPC-C transaction processing benchmark. This workload does not require network clients, but does use actual hard disk access through the PostgreSQL [32] database. Unfortunately, the target system does not have a sufficient number of hard disks to fully utilize the four Pentium IV processors. Therefore, the SPECjbb server-side java benchmark is included. This benchmark is able to more fully utilize the processor and memory subsystems without a large number of hard disks.

To further validate the I/O and disk models, a synthetic workload is developed to generate high disk utilization. Each instance of this workload creates a large file (1GB). Then the contents of the file are overwritten. After approximately 100K pages have been modified, the sync() operating system call is issued to force the modified pages to disk. The synthetic workload is used because none of the application based benchmarks are able to create sufficient level of disk activity.

For all subsystems, the power models are trained using a single workload trace that offers high utilization and variation. The validation is then performed using the entire set of workloads.

3.6 PERFORMANCE EVENT SELCTION

With over forty [40] detectable performance events, the Pentium IV provides a challenge in selecting events that are most representative of subsystem power. In this approach the interconnection of the various subsystems pictured in Figure 1 are considered. By noting the "trickle-down" effect of events in the processor, a subset of the performance events can be selected to model subsystem power consumption. As a simple example, consider the effect of cache misses in the processor. For the target server processor, the highest level of cache is the L3. Transactions that cannot be satisfied (cache miss) by the L3 cause a cache line (block) sized access to the main memory. Since the number of main memory accesses is directly proportional to the number of L3 misses, it is possible to approximate the number of accesses using only L3 misses. Since these memory accesses must go off-chip, power is consumed proportionally in the memory controller and DRAM. In reality the relation is not that simple, but there is still a strong causal relationship between L2 misses and main memory accesses.

Though the initial selection of performance events for modeling is dictated by an understanding of subsystem interactions (as in the previous example), the final selection of which event type(s) to use is determined by the average error rate during regression analysis and a comparison of the measured and modeled power traces. If wrong performance events are chosen in the beginning, the process described in Figure 2 will eventually identify the relevant performance events. The system knowledge simply helps to reduce the time to arrive at an appropriate solution. The dominant, power-related performance events identified for the server system are described below.

Cycles – *Execution time in terms of CPU clock cycles*. The cycles metric is combined with most other metrics to create per cycle metrics. This corrects for slight differences in sampling rate. Though sampling is periodic, the actual sampling rate varies slightly due to cache effects and interrupt latency.

Halted Cycles – *Cycles in which clock gating is active*. When the Pentium IV processor is idle, it saves power by gating the clock signal to portions of itself. Idle phases of execution are "detected" by the processor through the use of the HLT (halt) instruction. When the operating system process scheduler has available slack time, it halts the processor with this instruction. The processor remains in the halted state until receiving an interrupt. Though the interrupt can be an I/O device, it is typically the periodic OS timer that is used for process scheduling/preemption. This has a significant effect on power consumption by reducing processor idle power from ~36W to 9W. Because this significant effect is not reflected in the typical performance metrics, it is accounted for explicitly in the halted cycles counter.

Fetched µops – *Micro-operations fetched.* The micro-operations (µops) metric is used rather than an instruction metric to improve accuracy. Since in the P6 architecture instructions are composed of a varying number of µops, some instruction mixes give a skewed representation of the amount of computation being done. Using µops normalizes the metric to give representative counts independent of instruction mix. Also, by considering fetched rather than retired µops, the metric is more directly related to power consumption. Looking only at retired µops would neglect work done in execution of incorrect branch paths and pipeline flushes.

L3 Cache Misses – *Loads/stores that missed in the Level 3 cache.* Most system main memory accesses can be attributed to misses in the highest level cache, in this case L3. Cache

misses can also be caused by DMA access to cacheable main memory by I/O devices. The miss occurs because the DMA must be checked for coherency in the processor cache.

TLB Misses – *Loads/stores that missed in the instruction or data Translation Lookaside Buffer*. TLB misses are distinct from cache misses in that they typically cause trickle-down events farther away from the microprocessor. Unlike cache misses, which usually cause a cache line to be transferred from/to memory, TLB misses often cause the transfer of a page of data (4KB or larger). Due to the large size of pages, they are often stored on disk. Therefore, power is consumed on the entire path from the CPU to the hard disk.

DMA Accesses – *Transaction that originated in an I/O device whose destination is system main memory*. Though DMA transactions do not originate in the processor, they are fortunately visible to the processor. As demonstrated in the L3 Miss metric description, these accesses to the processor (by an I/O device) are required to maintain memory coherency. Being able to observe DMA traffic is critical since it causes power consumption in the memory subsystem. An important thing to consider in the use of the Pentium IV's DMA counting feature is that it cannot distinguish between DMA and processor coherency traffic. All memory bus accesses that do not originate within a processor are combined into a single metric (DMA/Other). For the uniprocessor case this is not a problem. However, when using this metric in an SMP environment such as this, care must be taken to attribute accesses to the correct source. Fortunately, the workloads considered here have little processor-processor coherency traffic. This ambiguity is a limitation of the Pentium IV performance counters and is not specific to this technique.

Processor Memory Bus Transactions – *Reads or writes on processor's external memory bus*. All transactions that enter/exit the processor must pass through this bus. Intel calls this the Front Side Bus (FSB). As mentioned in the section on DMA, there is a limitation of being able to distinguish between externally generated (other processors) and DMA transactions.

Uncacheable Accesses – *Load/Store to a range of memory defined as uncacheable*. These transactions are typically representative of activity in the I/O subsystem. Since the I/O buses are not cached by the processor, downbound (processor to I/O) transactions and configuration transactions are uncacheable. Since all other address space is cacheable, it is possible to directly identify downbound transactions. Also, since configuration accesses typically precede large upbound (I/O to processor) transactions, it is possible to indirectly observe these.

Interrupts – *Interrupts serviced by CPU*. Like DMA transactions, most interrupts do not originate within the processor. In order to identify the source of interrupts, the interrupt controller sends a unique ID (interrupt vector number) to the processor. This is particularly valuable since I/O interrupts are typically generated by I/O devices to indicate the completion of large data transfers. Therefore, it is possible to attribute I/O bus power to the appropriate device. Though, the interrupt vector information is available in the processor, it is not available as a performance event. Therefore, the presence of interrupt information in the processor is simulated by obtaining it from the operating system. Since the operating system maintains the actual interrupt service routines, interrupt source accounting can be easily performed. In this case the "/proc/interrupts" file available in Linux operating systems is used.

3.7 MODEL FORMAT

The form of the subsystem power models is dictated by two requirements: low computational cost and high accuracy. Since these power models are intended to be used for runtime power estimation, it is preferred that they have low computational overhead. For this reason initial attempts at regression curve fitting use single or multiple input linear models. If it is not possible to obtain high accuracy with a linear model, single or multiple input quadratics are chosen.

3.8 RESULTS

3.8.1 AVERAGE WORKLOAD POWER

In this section a power characterization of eleven workloads is presented. The average power in Watts for the considered workloads are given in Table 2. Also, workload variation is presented in Table 3 as the standard deviation of the power values in Watts.

With a maximum sustained total power of just over 305 Watts, the system consumes 46% of the maximum power at idle. This is lower than the typical value of 60% suggested for IA32 systems by Rajamani et al.[33]. The largest contributor to the reduced power at idle is the clock gating feature implemented in the microprocessor. Without this feature, idle power would be approximately 80% of peak. Due to the lack of a power management implementation, the other subsystems consume a large percentage of their peak power at idle. The chipset and disk subsystems have nearly constant power consumption over the entire range of workloads.

For the SPEC CPU 2000 workloads, there is the expected result of high microprocessor power. For all eight workloads, greater than 53% of system power goes to the microprocessors. The next largest consumer is the memory subsystem at 12%-18%. All of the top consumers are floating point workloads. This is expected due to the high level of memory boundedness of these workloads. I/O and disk consume almost the same power as the idle case since there is no access to network or storage during the workloads.

Table 2. Subsystem Average Power (Watts)

Workload	CPU	Chipset	Memory	I/O	Disk	Total
idle	38.4	19.9	28.1	32.9	21.6	141
gcc	162	20.0	34.2	32.9	21.8	271
mcf	167	20.0	39.6	32.9	21.9	281
vortex	175	17.3	35.0	32.9	21.9	282
art	159	18.7	35.8	33.5	21.9	269
lucas	135	19.5	46.4	33.5	22.1	257
mesa	165	16.8	33.9	33.0	21.8	271
mgrid	146	19.0	45.1	32.9	22.1	265
wupwise	167	18.8	45.2	33.5	22.1	287
DBT-2	48.3	19.8	29.0	33.2	21.6	152
SPECjbb	112	18.7	37.8	32.9	21.9	223

DiskLoad	123	19.9	42.5	35.2	22.2	243

The commercial workloads exhibited quite different power behavior compared to the scientific workloads. In DBT-2 the limitation of sufficient disk resources is evident in the low microprocessor utilization. Memory and I/O power are marginally higher than the idle case. Disk power is almost identical to the idle case also due to the mismatch in storage size compared to processing and main memory capacity. Because the working set fits easily within the main memory, few accesses to the I/O and disk subsystem are needed. The SPECjbb workload gives a better estimate of processor and memory power consumption in a balanced server workload with sustained power consumption of 61% and 84% of maximum for microprocessor and memory.

Table 3. Subsystem Power Standard Deviation

Workload	CPU	Chipset	Memory	I/O	Disk
idle	0.340	0.0918	0.0328	0.13	0.027
gcc	8.37	0.226	2.36	0.13	0.053
mcf	5.62	0.171	1.43	0.13	0.033
vortex	1.22	0.0711	0.719	0.14	0.017
art	0.393	0.0686	0.190	0.14	0.0055
lucas	1.64	0.123	0.266	0.13	0.0072
mesa	1.00	0.0587	0.299	0.13	0.0084
mgrid	0.525	0.0469	0.151	0.13	0.0052
wupwise	2.60	0.131	0.427	0.14	0.011
DBT-2	8.23	0.133	0.688	0.15	0.035
SPECjbb	26.2	0.327	2.88	0.06	0.073
DiskLoad	18.6	0.0948	3.80	0.15	0.075

Finally, a synthetic workload intended to better utilize the disk and I/O subsystems is considered. The DiskLoad workload generates the highest sustained power in the memory, I/O and disk subsystems. Surprisingly, the disk subsystem consumed only 2.8% more power than the idle case. The largest contribution to this result is a lack of power saving modes in the SCSI disks. According to Zedlewski [41], the power required for rotation of the disk platters is 80% of the peak amount, which occurs during disk write events. Since, the hard disks used in this study lack the ability to halt rotation during idle phases, at most a 20% increase in power compared to the idle state. There is the possibility that the difference for these disks is even less than the 20% predicted for Zedlewski's [41] mobile hard disk. Unfortunately, this cannot be verified since the hard disk manufacturer does not provide power specifications for the various hard disk events (seek, rotate, read/write and standby). The large increase in the I/O subsystem is directly related to the number of hard disk data transfers required for the workload. No other significant I/O traffic is present in this workload. The large increase in memory power consumption is due to the implementation of the synthetic workload and the presence of a software hard disk cache provided by the operating system. In order to generate a large variation in disk and I/O power consumption, the workload modifies a portion of a file approximately the size of the operating system disk cache. Then using the operating system's sync() call, the contents of the cache, which is located in main memory, are flushed to the disk. Since the memory is constantly accessed during the file modification phase (writes) and the disk flush phase (reads), high memory utilization results.

3.8.2 SUBSYSTEM POWER MODELS

This section describes the details of the subsystem power models. Issues encountered during the selection of appropriate input metrics are described. For each subsystem a comparison of modeled and measured power under a high variation workload is provided.

3.8.2.1 CPU

The CPU power model improves an existing model [5] to account for idle clock cycles. Since it is possible to measure the percent of time spent in the idle or halted state, the greatly reduced power consumption due to clock gating can be accounted for. This addition is not a new contribution, since a similar accounting is made in the model by Isci [16]. The largest distinction is that this implementation accounts for clock gating while retaining a lightweight model composed of only three input metrics: idle cycles, Fetched µops and total cycles. In contrast, Isci's model requires 22 events to attain a similar accuracy level.

Given that the Pentium IV can fetch three instructions/cycle, the model predicts range of power consumption from 9.25 Watts to 48.6 Watts. The form of the model is given in Equation 1.

$$\sum_{i=1}^{NumCPUs} 9.25 + (35.7 - 9.25) \times PercentActive_i + 4.31 \times \frac{FetchedUops_i}{Cycle}$$

Equation 1. Processor Power Model

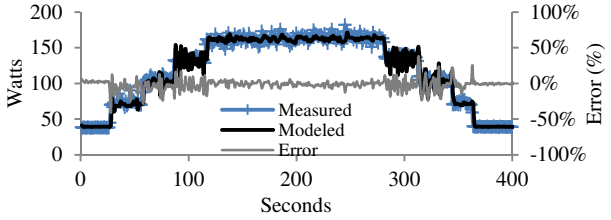

Figure 3. Processor Power Model – gcc

A trace of the total measured and modeled power for the four processors is given in Figure 3. The workload used in the trace is eight threads of gcc, started at 30s intervals. Average error is found to be 3.1%. Note that unlike the memory bound workloads that saturate at eight threads, the cpu-bound gcc saturates after only four simultaneous threads.

3.8.2.2 MEMORY

This section considers models for memory power consumption based on cache misses and processor bus transactions.

The first attempt at modeling memory power made use of cache misses. A model based on only the number of cache misses/cycle is an attractive prospect as it is a well understood metric and is readily available in performance monitoring counters. The principle behind using cache misses as proxy for power is that loads not serviced by the highest level cache, must be serviced by the memory subsystem. As demonstrated in [12], power consumption in DRAM modules is highest when the module is in the active state. This occurs when either read or write transactions are serviced by the DRAM module.

Therefore, the effect of high-power events such as DRAM read/writes can be estimated.

In this study, the number of L3 Cache load misses per cycle is used. Since the Pentium IV utilizes a write-back cache policy, write misses do not necessarily cause an immediate memory transaction. If the miss is due to a cold start, no memory transaction occurs. For conflict and capacity misses, the evicted cache block will cause a memory transaction as it updates memory.

The initial findings show that L3 cache misses are strong predictors of memory power consumption (Figure 4). The first workload considered is the integer workload mesa from the SPEC CPU 2000 suite. Since a single instance of this workload cannot sufficiently utilize the memory subsystem, multiple instances are used to increase utilization. For mesa, memory utilization increases noticeably with each instance of the workload. Utilization appears to taper off once the number of instances approaches the number of available hardware threads in the system. In this case the limit is 8 (4 physical processors with 2 threads per processor). The resultant quadratic power model is given in Equation 2.

$$\sum_{i=1}^{NumCPUs} 28 + \frac{L3LoadMisses_i}{Cycle} \times 3.43 + \frac{L3LoadMisses_i^2}{Cycle} \times 7.66$$

Equation 2. Cache Miss Memory Power Model

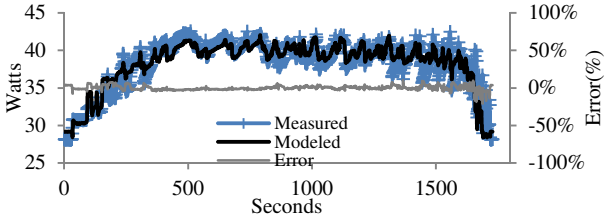

Figure 4. Memory Power Model (L3 Misses) – mesa

The average error under the mesa workload is low at only 1%. However, the model fails under extreme cases. Unfortunately, the L3 miss event does not perform well for the power model under all workloads. In cases of extremely high memory utilization, L3 misses tend to underestimate power consumption. It is found that when using multiple instances of the mcf workload, memory power consumption continues to increase, while L3 misses are slightly decreasing.

One of the possible causes is hardware-directed prefetches that are not accounted for in the number of cache misses. However, Figure 5 shows that though prefetch traffic does increase after the model failure, the total number of bus transactions does not. Since the number of bus transactions generated by each processor does not sufficiently predicting memory power, an outside (non-CPU) agent must be accessing the memory bus. For the target system the only other agent on the memory bus is the memory controller itself, performing DMA transactions on behalf of I/O devices.

Changing the model to include memory accesses generated by the microprocessors and DMA events resulted in a model that remains valid for all observed bus utilization rates.

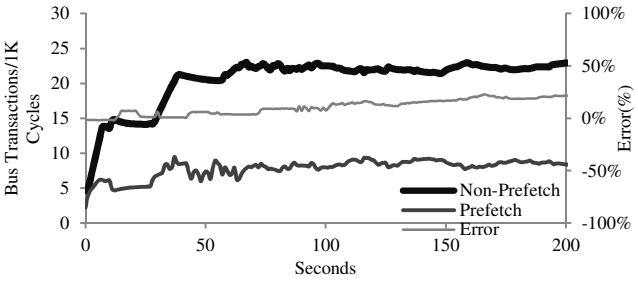

Figure 5. Prefetch & Non-Prefetch Bus Transactions – mcf

It should be noted that using only the number of read/write accesses to the DRAM does not directly account for power consumed when the DRAM is in the precharge state. DRAM in the precharge state consumes more power than in idle/disabled state, but less than in the active state. During the precharge state, data held in the sense amplifiers is committed to the DRAM array. Since the initiation of a precharge event is not directly controlled by read/write accesses, precharge power cannot be directly attributed to read/write events. However, in practice it is found that read/write accesses are reasonable predictors. Over the long term (thousands of accesses) the number of precharge events should be related to the number of access events. The resultant model is given in Equation 3.

$$\sum_{i=1}^{NumCPUs} 29.2 - \frac{BusTransactions_i}{MCycle} \times 50.1 \times 10^{-4} + \frac{BusTransactions_i^2}{MCycle} \times 813 \times 10^{-8}$$

Equation 3. Memory Bus Transaction Memory Power Model

A trace of the model is shown in Figure 6 for the mcf workload that could not be modeled using cache misses. The model yields an average error rate of 2.2%.

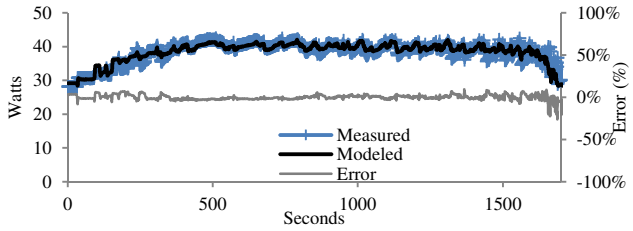

Figure 6. Memory Power Model (Memory Bus Transactions)- mcf

3.8.2.3 DISK

The modeling of disk power at the level of the microprocessor presents two major challenges: large distance from CPU to disk and little variation in disk power consumption. Of all the subsystems considered in this study, the disk subsystem is at the greatest distance and delay from the microprocessor. Therefore, there are challenges in getting timely information from the processor's perspective. The various hardware and software structures that are intended to reduce the average access time to the distant disk by the processor make power modeling difficult. The primary structures that cause difficulty are: microprocessor cache, operating system disk cache, I/O queues and I/O and disk caches. The structures offer the benefit of decoupling high-speed processor events from the

low-speed disk events. Since the power modeling techniques relies on the close relationship between the subsystems, this is a problem.

This is evidenced by the poor performance of the initial models that were created. Initially, two events are considered: DMA accesses and uncacheable accesses. Since the majority of disk transfers are handled through DMA by the disk controller, this appeared to be a strong predictor. Also, the number of uncacheable accesses by the processor are considered. Unlike the majority of application memory, memory mapped I/O (I/O address mapped to system address space) is not typically cached. Generally, I/O devices use memory mapped I/O for configuration and handshaking. Therefore, it should be possible to detect accesses to the I/O devices through uncacheable accesses. In practice it is found that both of these metrics do not fully capture the fine-grain power behavior. Since such little variation exists in the disk power consumption it is critical to accurately capture the variation that does exist.

To address this limitation the manner in which DMA transactions are performed is utilized. Coarsely stated, DMA transactions are initiated by the processor by first configuring the I/O device. The transfer size, source and destination are specified through the memory mapped I/O space. The disk controller performs the transfer without further intervention from the microprocessor. Upon completion or incremental completion (buffer full/empty) the I/O device interrupts the microprocessor. The microprocessor is then able to use the requested data or discard local copies of data that is sent. Our approach is to use the number of interrupts originating from the disk controller. This approach has the advantage over the other metrics in that the events are specific to the subsystem of interest. This approach is able to represent fine-grain variation with low error. In the case of the synthetic disk workload, the number of disk interrupts/cycle is used to achieve an average error of 1.75% is achieved. The model is provided in Equation 4.

$$\sum_{i=1}^{NumCPUs} 21.6 + \frac{Interrupts_i}{Cycle} \times 10.6 \times 10^7 - \frac{Interrupt_i^2}{Cycle} \times 11.1 \times 10^{15} + \frac{DMAAccess_i}{Cycle} \times 9.18 - \frac{DMAAccess_i^2}{Cycle} \times 45.4$$

Equation 4. DMA+Interrupt Disk Power Model

An application of the model to the memory-intensive benchmark program mcf is shown in Figure 7. Note that this error rate accounts for the large DC offset within the disk power consumption. This error is calculated by first subtracting the 21.6W of idle (DC) disk power consumption. The remaining quantity is used for the error calculation.

3.8.2.4 I/O

Since the majority of I/O transactions are DMA transactions from the various I/O controllers, an I/O power model must be sensitive to these events. Three events are considered for observing DMA traffic: DMA accesses on memory bus, uncacheable accesses and interrupts. Of the three, interrupts/cycle is the most representative. DMA accesses to main memory seemed to be the logical best choice since there is such a close relation to the number of DMA accesses and the switching factor in the I/O chips. For example, a transfer of cache line aligned 16 dwords (4 bytes/dword), maps to a single cache line transfer on the processor memory bus. However, in the case of smaller, non-aligned transfers the linear relationship does not hold. A cache line access measured as a single DMA event from the microprocessor perspective may contain only a single byte. This would grossly overestimate the power being consumed in the I/O subsystem. Further complicating the situation is the presence of performance enhancements in the I/O chips.

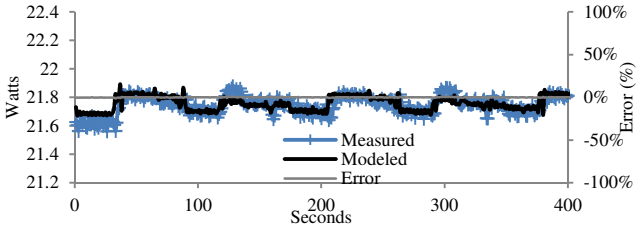

Figure 7. Disk Power Model (DMA+Interrupt) – Synthetic Disk Workload

One of the common enhancements is the use of write-combing memory. In write-combining, the processor or I/O chip in this case combines several adjacent memory transactions into a single transaction further removing the one-to-one mapping of I/O traffic to DMA accesses on the processor memory bus. As a result interrupt events are found to be better predictors of I/O power consumption. DMA events failed to capture the fine-grain power variations. DMA events tend to have few rapid changes, almost as if the DMA events have a low-pass filter applied to them. The details of the model can be seen in Equation 5.

$$\sum_{i=1}^{NumCPUs} 32.7 + \frac{Interrupt_i}{Cycle} \times 108 \times 10^6 - \frac{Interrupt_i^2}{Cycle} \times 1.12 \times 10^9$$

Equation 5. Interrupt I/O Power Model

Accounting for the large DC offset increases error significantly to 32%. Another consideration with the model is the I/O configuration used. The model has a significant idle power which is related to the presence to two I/O chips capable of providing six 133MHz PCI-X buses. While typical in servers, this is not common for smaller scale desktop/mobile systems that usually contain 2-3 I/O buses and a single I/O chip. Further, the server only utilizes a small number of the I/O buses present. It is expected that with a heavily populated, system with fewer I/O buses, the DC term would become less prominent. This assumes a reasonable amount of power management within the installed I/O devices.

3.8.2.5 CHIPSET

The chipset power model proposed here is the simplest of all subsystems as a constant is all that is required. There are two reasons for this. First, the chipset subsystem exhibits little variation in power consumption. Therefore, a constant power model is an obvious choice. Further, it is difficult to identify the effect performance events have on power consumption compared to induced electrical noise in the sensors. The

second, and more critical reason, is a limitation in the power sampling environment. Since the chipset subsystem uses power from more than one power domain, the total power cannot be measured directly. Instead it is derived by finding the average measured difference in power between multiple domains. The average chipset power is 19.9W.

3.8.3 MODEL VALIDATION

Tables 3 and 4 present summaries of average errors for the five models applied to twelve workloads. Errors are determined by comparing modeled and measured error at each sample. A sample corresponds to one second of program execution or approximately 1.5 billion instructions per processor. For performance counter sampling, the total number of events during the previous one second is used. For power consumption, the average of all samples in the previous second (ten thousand) is used. The average for each combination of workload and subsystem model is calculated using Equation 6. The results quantify the error in total power. They do not cover only the dynamic range from idle to maximum power. This should be considered for subsystems that have a small dynamic range such as the I/O and disk subsystems.

$$AverageError = \frac{\sum_{i=1}^{NumSamples} \frac{|Modeled_i - Measured_i|}{Measured_i}}{NumSamples} \times 100\%$$

Equation 6. Average Error Calculation

The I/O and disk models performed well under all workloads. The low error rates are partly due to low power variation / high idle power consumption. The CPU and memory subsystems have larger errors, but also larger workload variation. The worst case errors for CPU occurred in mcf workload which is memory-bound. Due to a high CPI (cycles/instruction) of greater than ten cycles, the fetch-based power model consistently underestimates CPU power. This is because while running mcf the processor only fetches one instruction every 10 cycles even though it is continuously searching for (and not finding) ready instructions in the instruction window. For mcf this speculative behavior has a high power cost that is equivalent to executing an additional 1-2 instructions/cycle.

The memory model averaged about 9% error across all workloads. Surprisingly the memory model faired better under integer workloads. The error rate for floating point workloads tended to be highest for workloads with the highest sustained power consumption. For these cases our model tends to underestimate power. Since the rate of bus transactions is similar for high and low error rate workloads the underestimation is likely caused by access pattern. In particular our model does not account for differences in the power for read versus write access. Also, the number of active banks within the DRAM is not accounted for directly. Accounting for the mix of reads versus writes would be a simple addition to the model. However, accounting for active banks will likely require some form of locality metric.

Idle power error is low for all cases indicating a good match for the DC term in the models. Chipset error is high considering the small amount of variation. This is due to the limitation of the constant model assumed for chipset power.

Table 4. Integer Average Model Error %

Workload	CPU	Chipset	Memory	I/O	Disk
idle	1.74#	0.59	3.80	0.36	0.17
gcc	4.23	10.9	10.7	0.41	0.20
mcf	12.3	7.7	2.2*	0.33	0.15
vortex	6.53	13.0	15.6	0.30	0.33
DBT-2	9.67	0.56	2.17	5.62	0.18
SPECjbb	9.00	7.45	6.14	0.39	0.14
DiskLoad	5.93	3.06	2.93	0.71*	0.16*
Integer Average	7.00	6.18	6.22	1.16	0.19
	±3.50	±4.92	±5.12	±1.97	±0.07
All Workload Average	6.67	5.97	8.80	0.82	0.39
	±3.42	±4.23	±5.54	±1.52	±0.49

Table 5. Floating Point Average Model Error %

Workload	CPU	Chipset	Memory	I/O	Disk
art	9.65	5.87	8.92	0.24	1.90
lucas	7.69	1.46	17.5	0.25	0.31
mesa	5.59	11.3	8.31	0.33	0.17
mgrid	0.36	4.51	11.4	0.37	0.55
wupwise	7.34	5.21	15.9	0.59	0.42
FP Average	6.13	5.67	12.41	0.35	0.67
	±3.53	±3.57	±4.13	±0.14	±0.70
All Workload Average	6.67	5.97	8.80	0.82	0.39
	±3.42	±4.23	±5.54	±1.52	±0.49

4 APPLICATION TO DESKTOP PLATFORM

Section 3 presented a complete system power model using performance events, using a quad-core server as the target platform. In order to prove the generality of the proposed approach, we apply the scheme to a very different system, a desktop platform. The platform is an AMD dual-core system with a graphics Processing Unit (GPU). This platform differs from the previous server in terms of process technology, system architecture, manufacturer and workload among others. They also differ in their power management implementations and subsystem components. A comparison of the two systems used in this study (server and desktop) is provided in Table 7. Of particular importance are two major differences: subsystem level power management and workload characteristics. Power management increases the complexity and utility of the power model as power consumption varies greatly with the application of power management. In contrast, in the server system, power remains near a constant level due to subsystems not reducing performance capacity, and therefore power consumption, during periods of low utilization. Increased power variation is also attributable to desktop-specific workloads. While server workloads tend to always operate at full speed (eq. SPEC CPU) desktop workloads such as SYSMark and 3DMark contain large portions of low utilization. This exposes the impact of power management and the need to model it.

In this section, power modeling of the AMD system with GPU using the approach presented in section 2 is presented. It is shown that though this platform is significantly different than

the server, the trickle-down modeling approach still accurately models power.

4.1 SYSTEM DESCRIPTION

. The target desktop system considered here is optimized for power efficiency rather than performance. This leads to greater variation in power consumption compared to a server since power management features reduce power greatly during low utilization. Server systems tend to employ less aggressive power savings. Therefore, power at low utilization is greater and overall variation is less. This difference is evident in the analysis of average subsystem-level power in Tables 1-2 and 7-8. The power management implementation in the desktop system also requires the use of more extensive power models. Rather than only needing to consider CPU clock gating and DRAM power down modes, the desktop system model must consider DVFS, chipset link power management, disk and GPU power management. The wider range of power consumption also leads to greater temperature sensitivity.

Another major difference is the ability to measure subsystem power at a finer granularity. The desktop platform allows direct measurement of memory controller and GPU in addition to all the subsystems that are measureable in the server system. One exception is the server I/O subsystem which contains numerous PCI-X busses and bridges. The desktop system does not contain comparable I/O subsystem. Therefore, it is not included in the study.

Table 6. System Comparison

Platform Segment	Server	Desktop
Manufacturer	Intel	AMD
Processor(s)	Quad-socket 130nM 2.8GHz	Dual-core 45nM 2.0GHz
Memory	8GB DDR-200	4GB DDR3-1066
Power Management	CPU Clock Gating DRAM Power Down	CPU Clock Gating and DVFS DRAM Pwr Down and Self Ref. Chipset Link Disconnect Harddrive Spin Down and ATA modes GPU Clock Gating
Graphics	Rage ProXL	RS780
Observable Subsystems	CPU Chipset Memory I/O Disk	CPU Chipset Memory Memory Controller GPU Disk

4.2 WORKLOADS

Due to the distinctions between server and desktop systems several desktop or client-appropriate workloads are added. In typical server or desktop benchmarks the GPU subsystem is almost entirely idle. Therefore, to exercise the GPU subsystem the 3DMark06 benchmark is included. 3DMark06 contains six subtests covering CPU and GPU intensive workloads. Four of the subtests (gt1, gt2, hdr1, hdr2) target the GPU's 3D processing engine. The other two tests (cpu1 and cpu2) heavily utilize CPU cores but have almost no GPU utilization. Targeting the 3D engine generates the largest power variation since the 3D engine is by far the largest power consumer in the GPU. An interesting side effect of the desktop GPU is intense system DRAM utilization. To reduce cost and power consumption, desktop systems such as this use a portion of system DRAM in lieu of locally attached, private DRAM. As a result, 3D workloads in desktop systems are effective at generating wide power variation in the memory subsystem.

Overall subsystem level power management is exposed through the addition of the SYSMark07 benchmark. This workload contains Microsoft Office, Adobe Photoshop, Adobe Flash and similar desktop applications. The various programs are classified into four categories called E-Learning, Video Creation, Productivity and 3D. This workload is implemented using simulated user input through the application GUI. The numerous delays required for GUI interaction causes many idle phases across the subsystems. This allows power management to become active. Contrast this to the vast majority of benchmarks which, by design, operate the CPU and other subsystems only at the 100% load level. The DBT-2 database workload is excluded as it is not practical and relevant to run on a desktop platform. For comparison to the server model, SPEC CPU, SPEC jbb and idle workloads are included. The workloads on targeted subsystems are summarized below in Table 7.

Table 7. Desktop Workloads

Workload		Description	Subsystems Targeted
Idle		Only background OS processes	All (power managed)
SPEC CPU 2006	INT	perlbench, bzip2, gcc, mcf, gobmk, hmmer, sjeng, libquantum, h264ref, omnetpp, astar, xalancbmk	CPU Memory Memory Controller
	FP	bwaves, games, milc, zeusmp, gromacs, cactusADM, leslie3d, namd, dealII, soplex, povray, calculix, gemsFDTD, tonto, lbm, wrf, sphinx3	CPU Memory Memory Controller
3DMark06	gt1	Graphics Test 1 and 2	GPU Memory Memory Controller
	gt2		
	cpu1	CPU Test 1 and 2	CPU
	cpu2		
	hdr1	High Dynamic Range Test 1 and 2	GPU Memory Memory Controller
	hdr2		
SYSMark07	EL	E-Learning	CPU Memory Memory Controller Chipset Disk
	VC	Video Creation	
	PR	Productivity	
	3D	3D	
SPECjbb2005		Server-Side Java	CPU Memory Memory Controller

4.3 PERFORMANCE EVENT SELECTION

In this section the various performance monitoring counter events used to construct the trickle-down model for the target desktop system are described. The definition and insight behind selection of the counters is provided.

Fetched µops – *Micro-operations fetched*. Comparable to the Pentium IV fetched micro-operations, this metric is highly correlated to processor power. It accounts for the largest portion of core pipeline activity including speculation. This is largely the result of fine-grain clock gating. Clocks are gated to small portions of the pipelines when they are not being used.

FP µops Retired – *Floating point micro-operations retired*. This metric is used to account for the difference in power consumption between floating point and integer instructions. Assuming equal throughput, floating point instructions have significantly higher average power. Ideally, the number of fetched FPU µops would be useful. Unfortunately, this metric is not available as a performance counter. This is not a major problem though since the fetched µops metric contains all fetched µops, integer and floating point.

DC Accesses – *Level 1 Data Cache Accesses*. This is a proxy for overall cache instruction and data accesses including Level 1,2, and 3. Considering the majority of workloads, level 1 data cache access rate dominates cache-dependent power consumption. No other single cache access metric correlates as well to processor core power (including caches).

%Halted/%Not-Halted – *Percent time processor is in halted state*. This represents power saved due to explicit clock gating. The processor saves power using fine-grain and coarse-grain gating of clock. Fine-grain clock gating saves power in unutilized portions of the processor while instructions are in-flight. Coarse-grain clock gating can save more power than fine-grain yet it requires the processor to be completely idle. The processor applies this type of gating only when the processor is guaranteed to not have any instructions in-flight. This condition by definition occurs following execution of the HLT instruction. Halt residency is controlled by the operating system and interrupts scheduling work on processors.

CPU Clock Frequency – *Core clocks per second*. Due to the use of DVFS, it is necessary to track the instantaneous frequency of the processor. Though some metrics such as µops fetched or retired implicitly track the power consumed in many components due to clock frequency, they do not track workload-independent power consumers such as clock grids. Using clock frequency in conjunction with %Halt it is possible to account for power consumed in these units.

CPU Voltage – *CPU Voltage Rail*. Due to the application of DVFS the processor may operate at a range of discrete voltages in order to save power. Changes in voltage have a significant impact on power consumption due to the exponential relationship between voltage and dynamic power (~V2) and the cubic relationship between voltage and leakage power (~V3). Due to a single, shared voltage plane, the actual voltage applied is the maximum requested of all cores in a socket. The requested voltage can be read using the P-State Status register [4].

Temperature – *CPU Temperature*. At the high voltages required for multi-GHz operation, leakage power becomes a major component of power consumption. Also, at idle when dynamic power is nearly eliminated due to clock gating leakage power can be the dominant contributor. Since temperature has a strong relation to leakage power (T2) it is necessary to account for this effect by measuring temperature. Temperature can be approximated using a series of on-die thermal sensors. The output of these sensors can be obtained using a configuration-space register [4].

GPU Non-Gated Clocks – *Number of GPU clocks per second*. Similar to CPU power, GPU power is greatly impacted by the amount of clock gating and DVFS. In this study DVFS usage is restricted to frequency changes only. Therefore, nearly all GPU power variation can be accounted for by this single metric.

DCT Accesses – $\sum_{N=0-1} DCT_N PageHits + DCT_N PageMisses + DCT_N PageConflicts$. DCT (DRAM Controller) Access accounts for all memory traffic flowing out of the two on-die memory controllers, destined for system DRAM. These events include cpu-generated and DMA traffic.

Link Active% – *Percent time Hypertransport links connected*. To save power in the I/O interconnection during idle periods, the Hypertransport links are disconnected. During periods of disconnect, cache snoop traffic and interrupts are blocked. This allows power to be saved in the CPU I/O interconnect and I/O subsystem. Also, the DRAM may be placed in self-refresh mode since DRAM access is blocked. If a cache snoop or interrupt event occurs, the links are reconnected.

Spindle Active % – *Percent time hard disk spindle is spinning*. In traditional mechanical hard drives, the spindle motor represent the largest single consumer of power in the drive. To save energy the spindle motor can be powered down. Due to the high latency (and energy consumption) for starting/stopping the spindle this can only be done when the drive is expected to be idle for a long time (minutes or more). In practice, typical workloads prevent the spindle from ever powering down. This includes all benchmarks used in this study, except idle. Therefore, spindle activity can be sufficiently accounted for by only distinguishing between idle and all other workloads.

CPUTo IOTransactions – *Non-cacheable access to memory-mapped I/O devices* . I/O device activity can be approximated using a measure of how many memory transactions generated by the CPUs are targeted at non-cacheable address space. Typically, I/O devices contain a DMA controller which performs access to cacheable space in system memory. The configuration and control of these transactions is performed by the CPU through small blocks of addresses mapped in non-cacheable space to each I/O device.

DRAMActive% – *Percent time DRAM channel is active.* Power savings in the DRAM and memory controller is controlled by the memory controller. When a memory channel has not issued a memory transaction for at least fixed period of time, the memory controller sends the channel to one of the precharge power down modes [4]. This primarily saves power in the DRAM chips, but also provides a slight savings in the memory controller.

4.4 RESULTS

4.4.1 AVERAGE WORKLOAD POWER

To understand subsystem-level power consumption average and standard deviation results are presented. Table 8 displays average power of each subsystem measured in Watts. To give an indication of the variation in power consumption Table 9 displays the standard deviation of subsystem power. Two major differences are apparent comparing desktop to server power consumption: power in each subsystem is much less while relative variability is much greater. In both cases, power management plays a large role. Effective power management through DVFS, clock gating and link management reduce average power during idle and low utilization phases. This leads to a greater difference in sample-to-sample power. Additionally, semiconductor process improvements have a major effect.

First, the CPU subsystem is considered. Not surprisingly, the desktop processor's average power is an order of magnitude less than the server processor. This is largely influenced by process (130nM in server vs. 45nM in desktop), DVFS (desktop-only) and idle power management. While the server idle power represents at least 24% of average power, desktop idle power is no more than 4%. These large power savings require the CPU model to include additional metrics such as frequency, voltage and temperature. It is not sufficient to consider metrics associated only with the instruction stream (IPC, cache accesses).

Like CPU, the Chipset also exhibits much greater power variation. Unlike the server chipset which is pragmatically modeled as a constant, the desktop chipset has much greater variation with standard deviation representing as much as 10% of average power. The difference illustrates the impact of link (Hypertransport) power management. Average power values are also much less due to the omission of an L3 cache in the desktop processor. In both platforms the top-level cache is contained in the chipset power rail. To reduce power consumption and cost the desktop designer omitted the L3 cache.

Yet another subsystem with order-of-magnitude power reduction is the DRAM memory. Despite higher operating frequency (533Mhz vs 100MHz) average DRAM power is reduced by almost a factor of 10. The reason is reduced memory voltage (2.8V vs 1.5V), reduced capacity (8GB vs 4GB) and more aggressive memory power management. Note that the desktop system differentiates between DRAM, "Memory" subsystem and the Memory Controller. The server system includes both in the memory subsystem. The desktop memory controller has a similar level of power variation as the DRAMs. This is due to the memory controller management power savings for both subsystems. This also allows implementation of simple trickle-down models in multiple subsystems that are driven by the same performance metrics.

A new subsystem, not present in the server analysis is the RS780 graphics processing unit (GPU). This subsystem has a unique bi-modal power consumption. In all cases GPU power is either near the maximum or minimum levels. For workloads with little or no GPU activity power ranges from 0.8W to 1.3W with little variation. The graphics-centric workloads of 3DMark06 have much greater variation as the workload alternates between approximately 1W and 4W. This gives the GPU one of the largest power variations with a standard deviation covering over 25% of the maximum power. This unique behavior leads to the creation of a simple power model. The bimodal power consumption is caused by aggressive idle power management and low active power variation. Therefore, it is sufficient to create a power model using only the ratio of time spent with clocks gated. Low error rates can be obtained without any instruction or data-dependent metrics.

Lastly, desktop hard drive power is considered. Three factors affect the large average power reduction and relative standard deviation increase: spindle speed, platter size and link power management. Since spindle power is such a large component of drive power consumption, reducing from 7200rpm to 5400rpm has a large impact. To conform to a smaller form factor, disk platter diameter is reduced nearly 28% (3.5" to 2.5"). This reduces spindle and drive head power. Less power is required to rotate a smaller mass and move the drive head a shorter distance. Also, SATA link power management reduces power consumption in the control electronics and links during idle phases. These changes yield a drastic increase in variability with standard deviation representing 32% of average power in the most intense workload (video creation).

Table 8. Subsystem Average Power (Watts)

Workload	CPU	Chipset	Memory	Memory Controller	GPU	Disk	Total
idle	0.63	0.54	0.24	0.52	0.856	0.827	3.62
CPU06 INT	14.1	2.74	2.97	2.35	0.863	0.945	24.0
CPU06 FP	14.8	2.87	3.35	2.48	0.882	0.866	25.3
gt1	10.2	3.25	3.29	2.41	3.79	1.35	21.9
gt2	10.2	3.29	3.52	2.48	4.00	1.35	22.4
cpu1	14.3	2.86	1.61	1.97	1.28	1.34	21.4
cpu2	14.2	2.86	1.62	1.98	1.28	1.30	21.3
hdr1	10.5	3.24	2.90	2.32	3.70	1.38	21.7
hdr2	10.6	3.26	3.03	2.37	3.75	1.35	22.0
EL	10.6	2.61	1.40	1.98	1.08	1.42	17.1
VC	11.0	2.79	1.12	1.89	1.12	1.74	17.8
PR	10.3	2.76	1.16	1.90	1.11	1.58	16.9
3D	11.9	2.62	1.25	1.91	1.06	1.35	18.2
SPECjbb	11.1	2.90	1.71	2.03	1.09	1.27	18.1

Table 9. Subsystem Power Standard Deviation (Watts)

Workload	CPU	Chipset	Memory	Memory Controller	GPU	Disk	Total
Idle	0.03	0.01	0.01	0.01	0.002	0.12	0.13
CPU06 INT	2.59	0.26	1.52	0.45	0.17	0.36	2.69
CPU06 FP	2.33	0.25	1.97	0.50	0.14	0.24	2.26
gt1	0.74	0.09	0.81	0.22	0.90	0.49	1.57
gt2	0.82	0.11	0.91	0.24	1.09	0.49	2.05
cpu1	1.99	0.26	0.71	0.17	0.36	0.47	2.02
cpu2	2.04	0.26	0.71	0.17	0.36	0.42	2.23
hdr1	0.76	0.13	1.04	0.29	1.14	0.53	1.84
hdr2	0.83	0.15	1.13	0.33	1.10	0.50	2.16

EL	0.70	0.16	0.98	0.28	0.05	0.37	1.74
VC	1.77	0.25	0.59	0.11	0.07	0.57	2.54
PR	0.68	0.25	0.81	0.16	0.09	0.44	1.51
3D	1.16	0.17	0.59	0.11	0.03	0.32	1.70
SPECjbb	1.23	0.30	1.10	0.24	0.03	0.23	2.77

4.4.2 SUBSYSTEM POWER MODELS

The impact of effective power management can be seen in the form of the power models of this section. In all cases it is necessary to explicitly account for power management to obtain accurate models. This causes all models to take a similar form. Previous models [5][6] are dominated by terms that are directly proportional to workload activity factors (IPC, cache accesses). While those workload-dependent terms are also used here, Idle Power Management and Irreducible power are also quantified. The idle power management term estimates power saved when instructions or operations are not actively proceeding through the subsystem. For CPUs this primarily occurs when executing the idle loop. The CPU detects one of the idle instructions (HLT, mwait) and takes actions such as clock or power gating. Other subsystems such as memory or I/O links similarly detect the absence of transactions and save power through various degrees of clock gating. Irreducible power contains the "baseline" power which is consumed at all times. This baseline power is largely composed of leakage and ungateable components.

4.4.2.1 CPU

The model presented here improves on existing on-line models [3][16][6] by accounting for power management and temperature variation. Like existing models this one contains a workload dependent portion which is dominated by the number of instructions completed per second. In this case the number of fetched operations per second is used in lieu of instructions completed. The fetched ops metric is preferred as it also accounts for speculative execution. The distinction of our model is that it contains a temperature dependent portion. Using workloads with constant utilization, processor temperature and voltage are varied 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 [4]. 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 power instrumentation. Like the workload dependent model, the coefficients of the static power model are tuned using regression techniques. Note that the static power model is highly process dependent. Processors 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. Ungateable represents the portion of power which cannot be gated. These components are also found experimentally. The resultant, average error in the model is 0.89%. The error distribution for SPEC CPU2006 and SYSmark2007 has the first standard deviation with less than 1% error. Worst-case error is 3.3%. The composition of the CPU model for the AMD processor is summarized in Table 10.

Table 10. AMD Dual-Core Power Model

Power Models	Equation
Total Power	$\sum_{N=0}^{NumCPU} (WorkloadDependent_N + Ungateable_N + Gateable_N + Static_{Volt,Temp}$
Workload Dependent Power	$((FetchOps_N/Sec)\times Coeff_F + (FloatPointOps_N/Sec)\times Coeff_{FP} + (DCAccess_N/Sec)\times Coeff_{DC})\times Voltage^2$
Idle Power Management Power	$(\%Halted_N)\times Coeff_{Gateable}\times Voltage^2\times Frequency_N$
Irreducible Power	$(\%NonHalted_N)\times Coeff_{Ungateable}\times Voltage^2\times Frequency_N$
Irreducible Power	$(Temp^2\times Coeff_T^2 + Temp^1\times Coeff_T^1 + \times Coeff_T^0)_{VoltageN}$

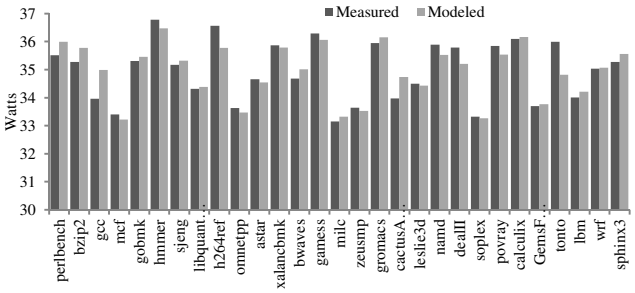

Figure 9. CPU Power Model – SPEC CPU 2006 Power and Average Error

4.4.2.2 GPU

To estimate GPU power consumption, a technique similar to that typically used for CPUs is employed: count the number of ungated clocks. In CPUs this is done by subtracting the number of halted clocks from all clocks [6]. In the case of the RS780 graphics processing unit, the ungated clocks can be measured directly. The GPU cycle counter is not physically located in the CPU, but is directly accessible using a configuration space access register. The latency is approximately 15us per access. Fortunately, there is only a single counter, so the effective overhead is low. Equation 7 presents the GPU power in terms of the non-gated clocks.

0.0068 x (Non-Gated Clocks /sec)/ 106 + 0.8467

Equation 7. GPU Power Model

This approach only accounts directly for power saved due to clock gating. Power reductions due to DVFS are not explicitly represented. Despite this, high accuracy of less than 1.7% error is obtained due to the implementation of DVFS. Unlike CPU DVFS which allows the operating system to reduce voltage and frequency during active phases, GPU DVFS reduces voltage only when clock gating is applied (idle). Therefore, increased power due to operating at the higher voltage is included in the non-gated clock metric. This bi-modal behavior can be seen in Figure 10. The mostly-idle,

clock-gated portion of the HDR1 workload draws about 1.5W. The fully active phase increases voltage and eliminates clock gating. Power increases drastically to over 4W.

An alternative metric for GPU power is also considered: % GUI Active. This metric represents the portion of time in which the GPU is updated the display. The main limitation of this approach is that it does not account for the intensity of work being performed by the underlying GPU hardware. Low-power 2D workloads, such as low-bit rate video playback, appear to have the same GPU utilization as more intense high-resolution video decoding. An example of modeled versus measured GPU power for 3DMark06-HDR1 is provided in Figure 10.

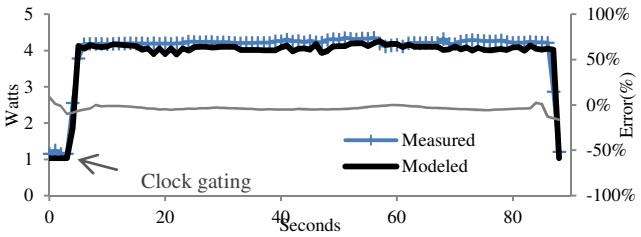

Figure 10. GPU Power Model (Non-Gated Clocks) – 3DMark06-HDR1

4.4.2.3 MEMORY

Memory or DRAM power consumption is one of the more variable subsystems. Similar to the CPU, the application of various power management features yields a wide range of power consumption. For example consider the standard deviation of power consumption in SPECJbb of 1.096W compared to its average of 1.71W. This variation is caused by the three modes of operation: self-refresh, precharge power down and active. Self-refresh represents the lowest power state in which DRAM contents are maintained by on-chip refresh logic. This mode has a high entry/exit latency and is only entered when the system is expected to be idle for a long period. The memory controller selects this mode as part of its hardware-controlled C1e idle [4] state. Since this state is entered in conjunction with the hypertransport link disconnect, the power savings can be represented using the LinkActive% metric. Precharge power down is a higher-power, lower-latency alternative which provides power savings for short idle phases. This allows precharge power savings to be considered with normal DRAM activity power. Light activity yields higher precharge residency. DRAM activity is estimated using the DCTAccess metric. The sum of all DCT accesses on both channels (hit, miss and conflict) correlates positively to active DRAM power and negatively to precharge power savings. Equation 8 presents memory power in terms of the DCTAccess and LinkActive% metrics.

4 x 10-8 x DCTAccess/sec + 0.7434 x LinkActive% + 0.24

Equation 8. Memory Power Model

In most workloads this approach gives error of less than 10%. The two outliers are the CPU subtests of 3DMark06. Due to many of the memory transactions being spaced at intervals just slightly shorter than the precharge power down entry time, the model underestimates power by a larger margin. Higher accuracy would require a direct measurement of precharge power down residency or temporal locality of memory transactions. An example of model versus measured Memory power for SYSMark2007-3D is provided in Figure 11.

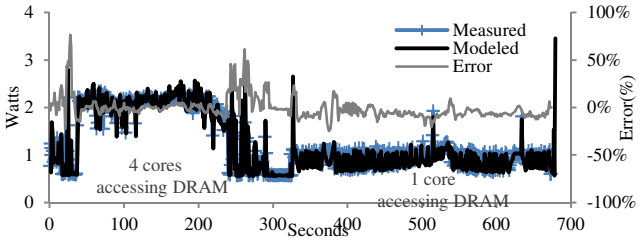

Figure 11. DRAM Power Model (∑DCT Access, LinkActive) – SYSMark2007-3D

4.4.2.4 MEMORY CONTROLLLER

Since the memory controller is responsible to entry and exit of power saving modes for itself and memory, the memory metrics can also be used to estimate memory controller power. Equation 9 presents memory power in terms of the DCTAccess and LinkActive% metrics.

9x10-9 x DCTAccess/sec + 0.798 x LinkActive% + 1.05

Equation 9. Memory Controller Power Model

Though both memory power model and memory controller power model use LinkActive% and DCTAccess metrics, the relative weights are different. Memory power has a large sensitivity to the transaction rate, 4 x 10-8 W/(transaction/sec). In comparison, the memory controller model coefficient is more than four times smaller at 9 x 10-9 W/(transaction/sec). Similarly, transaction-independent portion is much higher for the memory controller at 1.9W compared to 0.98W for memory. This reflects the unmanaged power consumers in the memory controller. The same 3DMark06 error outliers exist here. An example of model versus measured Memory Controller power for 3DMark06-HDR1 is provided in Figure 12.

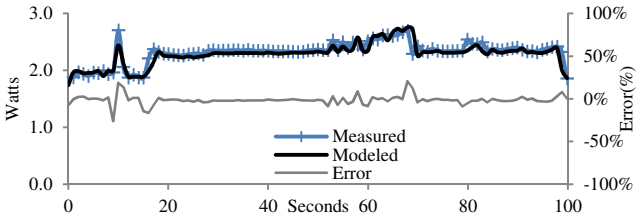

Figure 12. Memory Controller Power (∑DCT Access, LinkActive) – HDR1

4.4.2.5 CHIPSET

The Chipset power model represents power consumed in the Hypertransport controller. Like the memory and memory controller subsystems, power consumption is dominated by the link disconnect state and memory controller accesses. Overall and worst-case are less than the others due to the workload independent contributions being relatively the largest. Equation 10 presents the model for the chip set power.

-10-16 x (DCTAcc/sec)2 + 2x10-8 x (DCTAcc/sec)

+ 1.24 x LinkAct% + 1.34

Equation 10. Chipset Power Model

3.4.2.6 DISK

The improvements in power management for hard disks between the server-class used in the server study [6] and the more recent desktop/mobile disk used here is evident in the power model in Equation 11.

3 x 10-5 x (CPUToIOTrans/sec)

+ 0.629 x SpindleActive + 0.498

Equation 11. Disk Power Model

Rather than the negligible power variation previously observed (<1%), the variable portion (one standard deviation) is on average 30%. This provides more power savings, but a more difficult modeling challenge. As a result average error is higher at 6.6%.

4.4.3 MODEL VALIDATION

Table 11 summarizes the average error results for the six subsystem power models. The results quantify the error in total power. They do not cover only the dynamic range from idle to maximum power. This is less of a concern for the desktop system as most subsystems have large dynamic ranges, unlike the server system. The CPU subsystem has the second lowest error at 1.64% largely due to the comprehensive power model that is used. In comparison, the server power model averaged over 6% error using only three inputs. More importantly, this low error rate suggests that performance counter power models are effective across multiple microprocessor architecture generations, platforms, and manufacturers (Intel and AMD).

The desktop chipset power model is also improved compared to the server chipset model with average error of 3.3%. Like the server model, the desktop model contains a large workload-independent component: although in this case it contributes less than half the total chipset power rather than the 100% seen in the server model.

The memory and memory controller power models have the highest average error with 5.3% and 6.0% respectively. The high error is largely due to the CPU portion of the 3DMark06 workload. This workload is found to generate memory transactions at an interval that prevented effective utilization of precharge power down modes. Therefore, the model tends to underestimate memory power consumption. To resolve this error, a metric of typical memory bus idle duration or power down residency would be needed.

Table 11. Average Error %

Workload	CPU	Chipset	Memory	Memory Controller	GPU	Disk
Idle	0.3	1.8	1.2	0.4	1.7	2.5
CPU06 INT	1.3*	4.0	2.3	3.4	0.2	5.3
CPU06 FP	1.1*	2.6	7.2	2.9	0.5	4.4
gt1	0.8	5.3	3.3	1.3	0.9	7.1
gt2	1.2	5.8	3.3	4.4	0.4	8.7
cpu1	1.6	1.8	12.5	14.1	1.0	6.9
cpu2	1.9	1.9	11.5	13.4	1.3	9.2
hdr1	2.2	2.7	0.8*	0.7*	1.0*	0.9
hdr2	1.9	4.7	1.6	8.6	0.7	2.7
EL	2.7	9.3	8.5	7.7	0.0	1.8*
VC	1.0	1.8	8.4	11.7	1.6	10.6
PR	2.5	1.1*	5.7	5.6	0.9	12.1
3D	2.8	3.5	5.5	4.7	0.0	10.7
SPECjbb	1.5	0.4	2.0	4.1	0.8	9.8
Average	1.63	3.34	5.27	5.93	0.79	6.62

The GPU power model has the lowest error rate at slightly less than 1%. This illustrates the effectiveness of the non-gated GPU clocks as a proxy for GPU power. In most workloads the GPU power has a clear bimodal characteristic. Active regions have a power level that is consistent. Idle regions also have a consistent power level due to the presence of idle clock gating. It is expected that as finer grain power management is applied to the GPU core logic, larger active power variation will occur. This will necessitate a comprehensive power model such as that used in the CPU.

Finally, the disk subsystem is the one subsystem which has a higher error rate compared the server power model. In this case the error can be attributed to the effectiveness of on-disk and link power management. In the case of the server model, no active power management is provided. This allows for an accurate model as the workload independent portion dominates. In contrast the more recent desktop hard drive has a workload dependent portion which contributes as much as 1/3 of total power. This causes modeling error to have a larger impact. Note that the subtests with the highest errors are also those with the highest disk utilization.

5 RELATED WORK

5.1 PERFORMANCE COUNTERS MODELS

The use of performance counters for modeling power is not a new concept. However, unlike past studies [3][16][5][24][23][30] we go beyond modeling power consumed only in a microprocessor to modeling power consumed by an entire system. One of the earliest studies by Bellosa et al. [3] demonstrates strong correlations between performance events (instructions/cycle, memory references, cache references) and power consumption in the Pentium II. Isci [16] develops a detailed power model for the Pentium IV using activity factors and functional unit area, similar to Wattch [8]. Bircher [5] presents a simple linear model for the Pentium IV based on the number of instructions fetched/cycle. Lee [23] extends the use of performance counters for power modeling to temperature.

5.2 SUBSYSTEM POWER MODELS

5.2.1 LOCAL EVENT MODELS

Existing studies [12][41][20][13] into modeling of subsystem power have relied on the use of local events to represent power. In this section existing power modeling studies that make use of local events are considered.

Memory: It is possible to estimate power consumption in DRAM modules by using the number of read/write cycles and percent of time within the precharge, active and idle states

[12]. Since these events are not directly visible to the microprocessor, we estimate them using the count of memory bus accesses by the processor and other events that can be measured at the CPU. We also show that it is not necessary to account for the difference between read and write power in order to obtain accurate models. We use a similar approach as Contreras [11]. His model makes use of instruction cache misses and data dependency delay cycles in the Intel Xscale processor to estimate power. We show that for I/O intensive servers, it is also necessary to account for memory utilization caused by agents other than the microprocessor, namely I/O devices performing DMA accesses. Kadayif [19] uses as similar approach to modeling memory power using cache miss counters available on the UltraSPARC platform.

Disk: A study by Zedlewski et al. [41] shows that hard disk power consumption can be modeled by knowing how much time the disk spends in the following modes of operation: seeking, rotation, reading/writing, and standby. Rather than measuring these events directly from the disk, we estimate the dynamic events, seeking, reading and writing, through processor events such as interrupts and DMA accesses. Kim et al. [18-20] find that disk power and temperature can be accurately modeled using the amount of time spent moving the disk read/write head and the speed of rotation.

I/O and Chipset: Our objective is to estimate power using processor counters without having access to specific disk or memory system metrics. I/O and chipset subsystems are composed of rather homogeneous structures and we estimate their power through traditional CMOS power models. These models divide power consumption into static and dynamic. Static power represents current leakage, while dynamic accounts for switching current of CMOS transistors. Since static power does not vary in our system, due to a relatively constant voltage and temperature, we estimate dynamic power in the I/O and chipset subsystems through the number of interrupts, DMA and uncacheable accesses.

GPU: Ma et al. [27] employ a statistical GPU power model based on six “workload variables”. These variables are similar to performance counter events in that they are directly proportional to functional unit utilization. Specifically, they track utilization of five units: vertex shader, pixel shader, texture unit, geometry shader and render output unit. Our approach achieves similar accuracy while using only one counter: %Non-Gated Clocks. By measuring a wide range of CPU and GPU workloads we find that high accuracy can be obtained by tracking the one event (clock gating) which reduces power by the greatest amount.

5.2.2 OPERATING SYSTEM EVENT MODELS

Rather than using events local to the subsystem, Heath [14] [15] uses software counters in the operating system to model dynamic power of CPU, disk and network subsystems. Our approach differs by making use of hardware performance counters. This reduces the performance loss due to sampling of the software counters. Reading hardware performance counters requires only a small number of fast CPU register accesses. Reading software operating system-counters require relatively slow access using system service routines (file open/close etc.). The difference in access time between hardware performance counters and operating system-counters is greater than an order of magnitude. For example, the interrupt tracking operating system file: “/proc/interrupts”, induces greater than 1% overhead if sampled more frequently than once per 200 milliseconds. In contrast, CPU hardware performance counter sampling is limited by the overhead of the interrupt and context switch required for reading the counters on a particular core. The overhead of these actions is less than 1% for sample intervals as low as 10 milliseconds.

The effective difference in access time can be less due to limitations relating to hardware performance counter implementation. This study was performed using a driver with privileged access to the counters. If application-level access is required the overhead for access could increase. Also, it is assumed that only one entity is accessing the counters. Any sharing of counters among other processes or operating system routines would require virtualized counters that could also reduce the difference in access latency.

Lewis [26] constructs a similar linear regression model using a combination of operating system counters, hardware performance counters and physical sensors. Rivoire [36] compares various system power model types including operating system counter-driven and hardware performance counter-driven.

5.3 DYNAMIC ADAPTATION

Several researchers have demonstrated the effectiveness of techniques for adapting performance/power using DVFS. Kotla et al. [21] use instruction throttling and a utilization-based power model to show the effect of DVFS in a server cluster. At runtime they determine the minimum amount of required processor performance (frequency) and adjust the microprocessors accordingly. Due to the significant variation in web server workloads, Rajamani et al. [33] show that 30%-50% energy savings can be obtained through powering down idle compute nodes (severs). Using simulation Chen [9] applies DVFS and node power down in a dense compute center environment. However, unlike previous studies that only seek to minimize energy consumption while maintaining performance, Chen also considers the reliability impact of powering servers on and off. From the perspective of managing thermal, all of these dynamic adaptation schemes can benefit from the use of power modeling by being able to implement additional power management policies that maintain safe operating conditions.

Recently, it has become common for microprocessors [10][28] to apply DVFS to maximize performance within fixed power and thermal budgets. Our trickle-down approach is distinct in that it provides deterministic performance and allows adaptations to consider complete system power.

5.4 PHASE DETECTION

Researchers have developed numerous techniques for detecting program phases [11][22][17]. Dhodapkar and Smith [11] consider the effectiveness of instruction working sets, basic block vectors (BBV) and conditional branch counts for the

detection of program phases. They find that BBVs offer the highest sensitivity and phase stability. Lau [22] compares program structures such as basic blocks, loop branches, procedures, opcodes, register usage, and memory address information to identify phases. Using variation in CPI, compared to that in the observed structures, they show that loop frequency and register usage provide better accuracy than the traditional basic block vector approach. For the purpose of detecting power phases, Isci [17] compares the use of a traditional control flow metric (BBV) to on-chip performance counters. He finds that performance counter metrics have a lower error rate since they account for microarchitectural characteristics such as data locality or operand values. These techniques for phase detection are valuable for direct dynamic adaptations that increase efficiency of the microprocessor. Our complete-system power model allows power and phase analysis to be extended to several additional subsystems.

5.5 SUBSYSTEM POWER STUDIES

In order to motivate the use of microprocessor performance counters in modeling subsystem power, we demonstrate the significant contribution of the various subsystems to total power consumption. Unlike previous studies focusing on workstation [7] and mobile [29] power consumption, we show that the I/O subsystem makes up a larger part of total power in servers. Bohrer's [7] study of workstation power consumption considers three subsystems: CPU, hard disk, and combined memory and I/O. Our study provides finer granularity in that memory, I/O and chipset power are measured separately. Mahesri's study [29] presents fine grain measurement (ten subsystems), but uses a different hardware (laptop) and software (productivity workloads) configuration. Neither of the previous works present models based on their subsystem power characterizations.

Our study is the first to analyze chipset components discretely. Namely, the memory controller, DRAM and I/O chips are measured separately unlike the previous studies which group one or more of them together. This provides a better understanding of how workloads affect power consumption in the various subsystems.

6 CONCLUSIONS

In this paper feasibility of predicting complete system power consumption using processor performance events is demonstrated. The models take advantage of the trickle-down effect of these events. These events which are visible in the processing unit, are highly correlated to power consumption in subsystems including memory, chipset, I/O, disk and microprocessor. Subsystems farther away from the microprocessor require events more directly related to the subsystem, such as I/O device interrupts or clock gating status. Memory models must take into account activity that does not originate in the microprocessor. In this case, DMA events are shown to have a significant relation to memory power. It is shown that complete system power can be estimated with an average error of less than 9% for each subsystem using performance events that trickle down from the processing unit.

7 ACKNOWLEDGEMENTS

This research is partially supported by the National Science Foundation under grant number 0429806, and by IBM and AMD. We would also like to thank the ISPASS [6] reviewers for their useful suggestions.

8 REFERENCES

[1] F. J. Anscombe. Graphs in Statistical Analysis. American Statistician, pages 17-21, February 1973.

[2] N. Bansal, K. Lahiri, A. Raghunathan, S. T. Chakradhar. Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models, Proceedings of the 18th International Conference on VLSI Design (January 2005), 579-585.

[3] F. Bellosa. The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems. In Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System (Kolding, Denmark, September 2000), 37-42.

[4] BIOS and Kernel Developer's Guide for AMD Family 10h Processor. www.amd.com.

[5] W. L. Bircher, M. Valluri, J. Law, and L. John. Runtime identification of microprocessor energy saving opportunities. In Proceedings of the 2005 International Symposium on Low Power Electronics and Design (San Diego, California, August 2005), 275-280.

[6] W. L. Bircher and L. John. Complete System Power Estimation: A Trickle-Down Approach based on Performance Events. In IEEE International Symposium on Performance Analysis of Systems and Software (San Jose, California, April 2007), 158-168.

[7] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case For Power Management in Web Servers. IBM Research, Austin TX, 2002.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, In Proceedings of the 27th Annual International Symposium on Computer Architecture, (Vancouver, British Columbia, Canada, June 2000), 83-94.

[9] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, Managing Server Energy and Operational Costs in Hosting Centers. In Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (Banff, Alberta, Canada, June 2005), 303-314.

[10] J. Charles, P. Jassi, N. Ananth, A. Sadat, and A. Fedorova. Evaluation of the Intel® Core™ i7 Turbo Boost feature. In Proceedings of the 2009 IEEE International Symposium on Workload Characterization (Austin, Texas, October 2009), 188-197.

[11] G. Contreras and M. Martonosi. Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events. In Proceedings of the 2005 International Symposium on Low Power Electronics and Design (San Diego, California, August 2005), 221-226. [11] Ashutosh Dhodapkar and James Smith. Comparing program phase detection techniques. International Symposium. on Microarchitecture, pp 217-228, December 2003.

[12] Jeff Janzen. Calculating Memory System Power for DDR SDRAM. Micro Designline, Volume 10, Issue 2, 2001.

[13] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John. Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach. In Proceedings of the 8th

International Symposium on High-Performance Computer Architecture (Boston, Massachusetts, February 2002), 141-150.

[14] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and Freon: Temperature Emulation and Management in Server Systems. In Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (San Jose, California, October 2006), 106-116.

[15] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr., and R. Bianchini. "Energy Conservation in Heterogeneous Server Clusters". Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), June 2005.

[16] C. Isci and M. Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In 36th International Symposium on Microarchitecture (San Diego, California, December 2003), 93-104.

[17] C. Isci and M. Martonosi. Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques. In Proceedings of the Twelfth International Symposium on High-Performance Computer Architecture (Austin, Texas, February 2006), 122-133.

[18] R. Joseph and M. Martonosi. Runtime Power Estimation in HighPerformance Microprocessors. In Proceedings of the 2001 International Symposium on Low Power Electronics and Design (Huntington Beach, California, 2001), 135-140.

[19] I. Kadayif , T. Chinoda , M. Kandemir , N. Vijaykirsnan , M. J. Irwin , A. Sivasubramaniam, vEC: virtual energy counters, Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, (Snowbird, Utah, June 2001), 28-31.

[20] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam. Understanding the performance-temperature interactions in disk I/O of server workloads. In Proceedings of the Twelfth International Symposium on High-Performance Computer Architecture (Austin, Texas, February 2006), 176- 186.

[21] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson. Scheduling Processor Voltage and Frequency in Server and Cluster Systems. In Proceedings of the First Workshop on High-Performance Power-Aware Computing in conjunction with International Parallel and Distributed Processing Symposium (Denver, Colorado, April 2005).

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

[23] K. Lee and K. Skadron. Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. In Proceedings of the First Workshop on High-Performance Power-Aware Computing in conjunction with International Parallel and Distributed Processing Symposium (Denver, Colorado, April 2005).

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

[25] Linux Perfctr Kernel Patch Version 2.6, user.it.uu.se/~mikpe/linux/perfctr, October 2006.

[26] A. Lewis, S. Ghosh, and N.-F. Tzeng. Run-time Energy Consumption Estimation Based on Workload in Server Systems. Workshop on Power Aware Computing and Systems, December 2008.

[27] X. Ma, M. Dong, L. Zhong, and Z. Deng. Statistical Power Consumption Analysis and Modeling for GPU-based Computing. In Proceedings of the ACM SOSP Workshop on Power Aware Computing and Systems (HotPower) 2009, (Big Sky, Montana, October 2009).

[28] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks and S. Naffziger. Temperature Control on a 90-nm Itanium Family Processor. IEEE Journal of Solid State Circuits, Vol. 41, No. 1, January 2006.

[29] A. Mahesri and V. Vardhan. Power Consumption Breakdown on a Modern Laptop, In Proceedings of the 4th International Workshop on Power-Aware Computing Systems (Portland, Oregon, December 2004), 165-180.

[30] A. Merkel and F. Bellosa. Balancing Power Consumption in Multiprocessor Systems. Proceedings of the 2006 ACM EuroSys Conference, April 2006.

[31] Open Source Development Lab, Database Test 2, www.osdl.org, February 2006.

[32] PostgreSQL, www.postgresql.org, October 2006.

[33] K. Rajamani and C. Lefurgy. On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters. In Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software (Austin, Texas, March 2003), 111-122.

[34] K. Ramani, A. Ibrahim, and D. Shimizu. PowerRed: Flexible Modeling Framework for Power Efficiency Exploration in GPUs. In Proceedings of the First Worskshop on General Purpose Processing on Graphics Processing Units (GPGPU) (Boston, Massachusetts, October 2007).

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

[36] S. Rivoire, P. Ranganathan and C. Kozyrakis. A Comparison of High-Level Full-System Power Models. Workshop on Power Aware Computing and Systems, December 2008.

[37] D. Snowdon. OS-Level Power Management. PhD Thesis, School of Computer Science and Engineering, University of New South Wales, 2010.

[38] SPEC CPU 2000 Version 1.3, www.spec.org/cpu2000, October 2006.

[39] SPECjbb 2005 Version 1.07, www.spec.org/jbb2005, October 2006.

[40] B. Sprunt. Pentium 4 Performance Monitoring Features,Micro, July-August, pp 72-82, 2002.

[41] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthyy, R. Wang. Modeling Hard-Disk Power Consumption. File and Storage Technologies 2003.

[42] K. Scoones. Power Management Technology for Portable Devices. Austin Conference on Energy-Efficient Design, March 2007.