# Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events

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

## Abstract

This paper proposes the use of microprocessor performance counters for online measurement of complete system power consumption. While past studies have demonstrated the use of performance counters for microprocessor power, to the best of our knowledge, we are the first to create power models for the entire system based on processor performance events. Our approach takes advantage of the "trickle-down" effect of performance events in a microprocessor. We show how well known performance-related events within a microprocessor such as cache misses and DMA transactions are highly correlated to power consumption *outside* of the microprocessor. Using measurement of an actual system running scientific and commercial workloads we develop and validate power models for five subsystems: memory, chipset, I/O, disk and microprocessor. 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 additional power sensing hardware.

## 1 Introduction

In order to improve microprocessor performance while limiting power consumption, designers are increasingly utilizing dynamic hardware adaptations. These adaptations provide an opportunity for extracting maximum performance while remaining within temperature and power limits. These adaptations are valuable tools for reducing power consumption and temperature. Two of the most common examples are dynamic voltage and frequency scaling (DVFS) and clock gating. With these adaptations it is possible 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 which caused them. Rather than relying on relatively slow temperature sensors for observing power consumption it has been demonstrated [1][2][3] that performance counters can be used as a proxy for power measurement. These counters provide a timely, readily accessible means of observing power variation in real systems. In this paper we extend this valuable tool beyond the microprocessor to various computer subsystems. We present models for five subsystems: microprocessor, chipset, memory, I/O and disk. Though microprocessors are the largest consumers of power, the remaining subsystems make up 40%-60% of total power depending on the workload. 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 [4]. Further, since the tool utilizes existing microprocessor performance counters, the cost of implementation is small. 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 at the subsystem. Our approach is distinct since it uses events at the processor, eliminating the need for sensors spread out in various parts of the system and corresponding interfaces. Lightweight adaptive systems can easily be built using models of this type.

In this study we show that microprocessor performance events can accurately estimate total system power. By considering the propagation of power inducing events within the various subsystems, we identify six performance events for modeling the entire system power. The resultant models predict power with an average error of less than 9%.

## 2 Related Work

### 2.1 Performance Counter Power Models

The use of performance counters for modeling power is not a new concept. However, unlike past studies [1][2][3][5][6] 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 [1] demonstrates strong correlations between performance events (instructions/cycle, memory references, cache references) and power consumption in the Pentium II. Isci develops a detailed power model for the Pentium IV using activity factors and functional unit area, similar to Wattch [7]. Bircher [3] presents a simple linear model for the Pentium IV based on the number of instructions fetched/cycle. Lee [6] extends the use of performance counters for power modeling to temperature.

## 2.2 Subsystem Power Models

### *2.2.1 Local Event Models*

Existing studies [8][9][10][11] into modeling of subsystem power have relied on the use of *local* events to represent power. In this section we consider existing power modeling studies that make use of local events.

**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 [8]. 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 [12]. 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.

**Disk:** A study by Zedlewski et al [9] 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 [10] 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 vcc and temperature, we estimate dynamic power in the I/O and chipset subsystems through the number of interrupts, DMA and uncacheable accesses.

### *2.2.2 Operating System Event Models*

Rather than using events local to the subsystem, Heath [13] uses operating system level event counters to model dynamic power of CPU, disk and network subsystems. Our approach differs by making use of on-chip processor performance counters. This reduces the performance loss due to sampling of the counters. Reading On-chip performance counters requires only a small number of fast CPU register accesses. Reading operating system counters requires relatively slow access using system service routines .(file open/close etc.).

## 2.3 Dynamic adaptation

Dynamic adaptation of hardware promises to extend performance gains so common in the era of microprocessor frequency scaling, in spite of the critical limitation of power consumption. By dynamically reconfiguring hardware to match the demands of software, it is possible to obtain high performance and low power consumption. Also, designers are able to develop hardware that conforms to *average* power consumption rather than *peak*. This has a great impact since most modern computing systems spend a majority of the time underutilized [4].

Techniques which adapt in response to temperature changes are at a disadvantage compared to performance counter techniques [6]. Due to the thermal inertia of components, temperature sensors are less able to allow preemptive reaction to impending thermal emergencies. By using performance counters as a proxy for power consumption, it is possible to see the cause of thermal emergencies in a timelier manner.

Current dynamic adaptation implementations are primarily limited to process scheduling, level adaptation or thermal emergency management. However, current microprocessors and systems have the potential for significantly more robust and effective adaptation. Several researchers have demonstrated the effectiveness of techniques for adapting performance/power using DVFS. Kotla et al [15] 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 webserver workloads, Rajamani et al [16] show that 30%-50% energy savings can be obtained through powering down idle compute nodes (severs). Using simulation Chen [17] applies DVFS and node power down in a dense compute center environment. However, unlike previous studies they 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.

## 2.4 Phase Detection

While thermally-directed adaptation has clear indicators (temperature > limit) for when to apply adaptations, performance-directed adaptation thresholds may not be as obvious. Since performance must not be compromised, performance insensitive phases of program execution must be identified. Researchers have developed numerous techniques for detecting program phases [18][19][20]. Dhodapkar and Smith [18] 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 [19] 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 the traditional basic block vector approach. For the purpose of detecting power phases, Isci [20] 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. For the study of phases within a complete system it is also necessary to have power information for additional subsystems.

## 2.5 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 [21] and mobile [22] power consumption, we show that the I/O subsystem makes up a larger part of total power in servers. Bohrer's [21] 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 [22] presents fine grain measurement (ten subsystems), but uses a very different hardware (laptop) and software (productivity workloads) configuration. Finally, neither of the previous works present models based on their subsystem power characterizations.

## *3 Methodology*

In this section we describe our measurement environment, workloads and performance event selection.

## 3.1 Power & Performance Measurement

### *3.1.1 Subsystem Description*

The division of the subsystems in our target server is dictated by the system designer. Fortunately, the design portioned the subsystems in a configuration that is quite useful for study. In particular, we were able to separately measure five major subsystems: CPU, chipset, memory, I/O and disk. The *CPU* subsystem is composed of four Pentium IV Xeon processors. Ideally, we would have been able to measure power for each processor. We are only able to measure the sum of processor power. Fortunately, existing uniprocessor models [2][3] allow observation of individual processor power. We defined *chipset* as processor interface chips not included in other subsystems. The *memory* subsystem includes memory controller and DRAM power. *I/O* included PCI buses and all devices attached to them. The *disk* subsystem was composed of two SCSI disks.

### *3.1.2 Power Measurement*

To measure power in the five subsystems, we employed resistors connected in series with the power source. Since the power source is provides as a regulated voltage, 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 were taken each second and were then averaged for relation to performance counter samples taken at the much slower rate of one per second.

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

### *3.1.3 Performance Measurement*

To gather a record of performance events in the processor, we periodically sample the Pentium IV's on-chip performance monitoring counters. 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 [23] device driver. As described in the power measurement section, a synchronization signal was introduced at each performance counter sampling.

## 3.2 Workloads

### *3.2.1 Selection*

Our selection of workloads was 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, we employ our power measurement system. Workloads are chosen based on their apparent utilization of a subsystem. Then actual power measurement is done to verify the selection. We find that high subsystem utilization is difficult to achieve using only conventional workloads. As a result, we create small synthetic workloads that are able to sufficiently utilize the subsystems. Additionally, we combine multiple instances of single-threaded workloads such as SPEC CPU 2000 to produce very high utilization. Since our target system is composed of a 4-way SMP with two hardware threads per processor, we find that most workloads saturate (no increased subsystem utilization) 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, we stagger the start of each thread by a fixed time, usually 30s-60s. 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 quadratic relationships may appear to be linear.

### *3.2.2 Model Validation*

For the validation of the models we use eleven workloads: eight from the SPEC CPU 2000 benchmark suite [24], two commercial server type and a synthetic disk type. 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 during the loading of the data set at program initialization. In this study we only consider homogeneous combinations of the workloads. For commercial workloads we use dbt-2 [25] and SPECjbb [26]. 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 [27] database. Unfortunately, our target system did not have a sufficient number of hard disks to fully utilize the four Pentium IV processors. Therefore, we included the SPECjbb server-side java benchmark. This benchmark is able to more fully utilize the processor and memory subsystems without a large number of hard disks.

To further validate our I/O and disk models, we developed a synthetic workload to generate very high disk utilization. Each instance of this workload creates a very large file (1GB). Then the contents of the file are overwritten. After about 100K pages have been modified, the sync() operating system call is made to force the modified pages to disk.

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.

## 3.3 Performance Event Selection

With over forty [28] detectable performance events, the Pentium IV provides a challenge in selecting which is most representative of subsystem power. In our approach we considered the interconnection of the various subsystems pictured in Figure 1. By noting the "trickle-down" effect of events in the processor, we were able to successfully select a subset of the performance events to model subsystem power consumption. A simple example would be the effect of cache misses in the processor. For a typical microprocessor the highest level of cache is the L2. Transactions that cannot be satisfied (cache miss) by the L2 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 L2 misses, it is possible to approximate the number of accesses using only L2 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 and a qualitative comparison of the measured and modeled power traces.

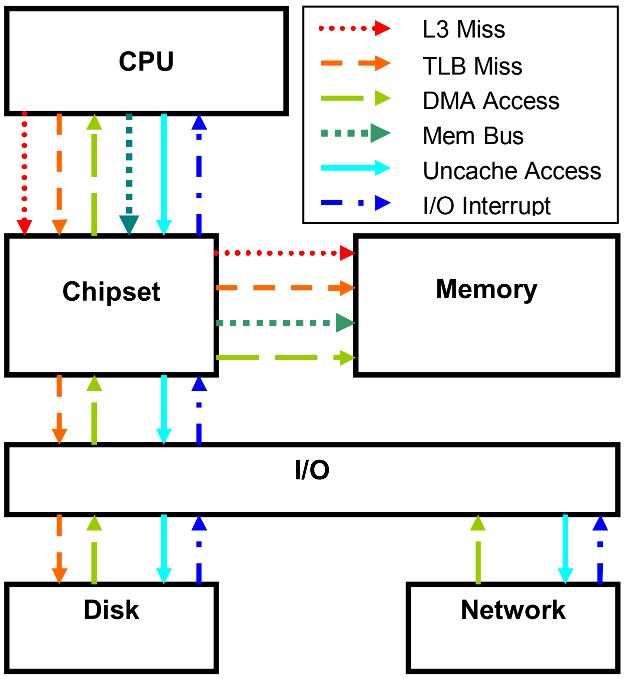

**Figure 1 Propagation of Performance Events**

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

**Level 3 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 level 3. 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 ours, care must be taken to attribute accesses to the correct source. Fortunately, the workloads we considered have very little processor-processor coherency traffic. This ambiguity is a limitation of the Pentium IV performance counters and is not specific to our technique.

**Processor Memory Bus Transactions** – 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, we simulate the presence of interrupt information in the processor 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 our case we made use of the /proc/interrupts file available in Linux operating systems.

### 3.3.1 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 that reason we initially attempt regression curve fitting using linear models. If it is not possible to obtain high accuracy with a linear model, we select single or multiple input quadratics.

## 4 Results

## 4.1 Average Workload Power

In this section we present a power characterization of eleven workloads. Averages in terms of Watts are given in Table 1. Also, workload variation is presented in terms of Watts of standard deviation in Table 2. We will now consider the average idle power. 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 [16]. 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.

Table 1 Subsystem Average Power (Watts)

| | 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 |

For the SPEC CPU 2000 workloads, there is the expected result of very high microprocessor power. For all eight, 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 were floating point workloads. This is expected due to the high level of memory boundedness of these workloads. I/O and disk consumed almost the same power as the idle case since there is no access to network or storage during the workloads.

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 2 Subsystem Power Standard Deviation (Watts)

| | CPU | Chipset | Memory | I/O | Disk |
|---|---|---|---|---|---|
| idle | 0.340 | 0.0918 | 0.0328 | 0.127 | 0.0271 |
| gcc | 8.37 | 0.226 | 2.36 | 0.133 | 0.0532 |
| mcf | 5.62 | 0.171 | 1.43 | 0.125 | 0.0328 |
| vortex | 1.22 | 0.0711 | 0.719 | 0.135 | 0.0171 |
| art | 0.393 | 0.0686 | 0.190 | 0.135 | 0.00550 |
| lucas | 1.64 | 0.123 | 0.266 | 0.133 | 0.00719 |
| mesa | 1.00 | 0.0587 | 0.299 | 0.127 | 0.00839 |
| mgrid | 0.525 | 0.0469 | 0.151 | 0.132 | 0.00523 |
| wupwise | 2.60 | 0.131 | 0.427 | 0.135 | 0.0110 |
| dbt-2 | 8.23 | 0.133 | 0.688 | 0.145 | 0.0349 |
| SPECjbb | 26.2 | 0.327 | 2.88 | 0.0558 | 0.0734 |
| DiskLoad | 18.6 | 0.0948 | 3.80 | 0.153 | 0.0746 |

Finally, we consider a synthetic workload intended to better utilize the disk and I/O subsystems. 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 [9], the power required for rotation of the disk platters is 80% of the peak amount, which occurs during disk write events. Since, our hard disks lack the ability to halt rotation during idle phases, the largest we could expect to see is a 20% increase in power compared to the idle state. There is the possibility that the difference for our disks is even less than the 20% predicted for Zedlewski's mobile hard disk. Unfortunately, we were unable to verify this 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), very high memory utilization results.

## 4.2 Subsystem Power Models

This section describes the details of our subsystem power models. We describe issues encountered during the selection of appropriate input metrics. For each subsystem we provide a comparison of modeled and measured power under a high variation workload.

### 4.2.1 CPU Power

Our CPU power model improves an existing model by [3] to account for halted clock cycles. Since it is possible to measure the percent of time spent in the halt state, we are able to account for the greatly reduced power consumption due to clock gating. This addition is not a new contribution, since a similar accounting was made in the model by [2]. The distinction is that this model is the first application of a performance-based power model in an SMP environment. The ability to attribute power consumption to a single physical processor within an SMP environment is critical for shared computing environments. In the near future it is expected that billing of compute time in these environments will take account of power consumed by each process [14]. This is particularly challenging in virtual machine environments in which multiple customers could be simultaneously running applications on a single physical processor. For this reason, process-level power accounting is essential.

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. A trace of the total measured and modeled power for our four SMP processors is given in Figure 2. The workload used in the trace is eight threads of gcc, started at 30s intervals. Average error was found to be 3.1%. Note that unlike the memory bound workloads that saturate at eight threads, the cpu-bound gcc saturates after only 4 simultaneous threads.

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

**Equation 1 – SMP Processor Power Model**

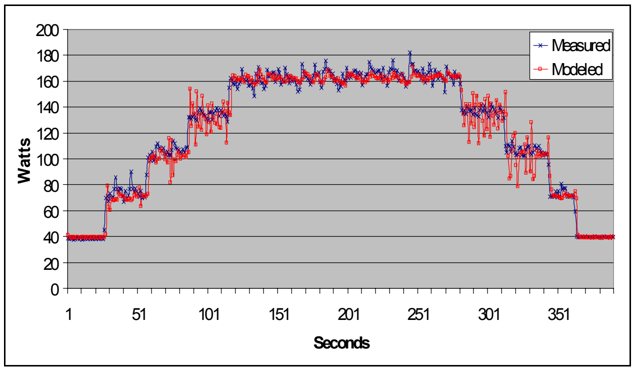

**Figure 2 Four CPU Power Model - gcc**

### *4.2.2 Memory Power*

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

Our 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 [8], 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, we use the number of Level 3 Cache load misses per cycle. Since the Pentium 4 utilizes a write-back cache policy, write misses do not necessarily cause an immediate memory transaction. If the miss was 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.

Our initial findings showed that level 3 cache misses were strong predictors of memory power consumption (Figure 3). The first workload we considered was the integer workload mesa from the SPEC CPU 2000 suite. Since a single instance of this workload could not sufficiently utilize the memory subsystem, we used multiple instances 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 x 2 threads/processor). The resultant quadratic power model is given in Equation 2. The average error under the mesa workload is low at only 1%. However, the model fails under extreme cases.

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

**Equation 2 – Cache Miss Memory Power Model**

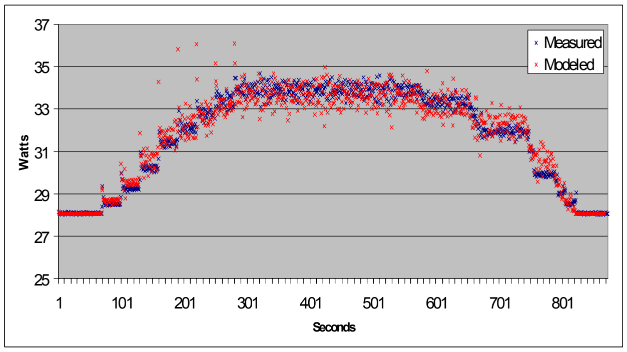

**Figure 3 Memory Power Model (L3 Misses) – mesa**

Unfortunately, L3 misses do not perform well under all workloads. In cases of extremely high memory utilization, L3 misses tend to underestimate power consumption. We found that when using multiple instances of the mcf workload, memory power consumption continues to increase, while Level 3 misses are slightly decreasing.

We determined that one of the possible causes was hardware-directed prefetches that were not being accounted for in the number of cache misses. However, Figure 4 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 was not sufficiently predicting memory power, we concluded that an outside (non-CPU) agent was accessing the memory bus. For our target system the only other agent on the memory bus is the memory controller itself, performing DMA transactions on behalf of I/O devices.

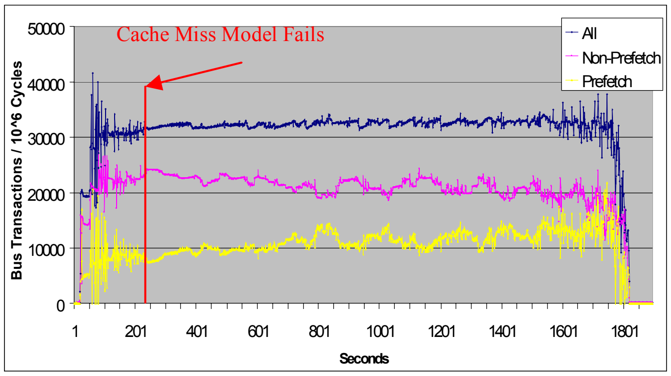

**Figure 4 Prefetch and Non-Prefetch Bus Transactions - mcf**

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.

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 we have found read/write accesses to be 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. A trace of the model is shown in Figure 5 for the mcf workload that could not be modeled using cache misses. The model yields an average error rate of 2.2%.

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

**Equation 3 – Memory Bus Transaction Memory Power Model**

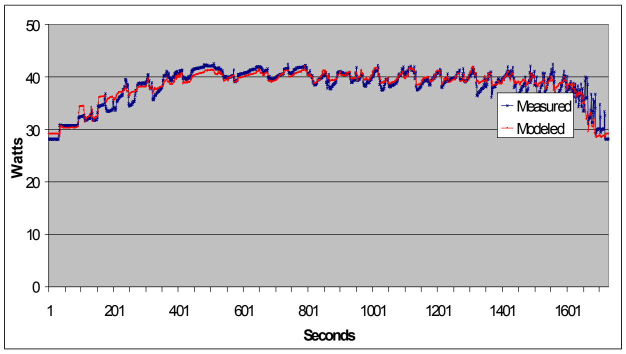

**Figure 5 Memory Power Model (Memory Bus Transactions)- mcf**

### *4.2.3 Disk Power*

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 the farthest away 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 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 our power modeling techniques rely on the close relationship between the subsystems, this is a problem.

This is evidenced in the poor performance of our first attempts. Initially we considered two events: 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. We also considered the number of uncacheable accesses by the processor. 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 we found that both of these metrics did 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 we take advantage of the manner in which DMA transactions are performed. 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 was 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 very low error. In the case of our synthetic disk workload, by using the number of disk interrupts/cycle we achieve an average error rate of 1.75%. The model is provided in Equation 4. An application of the model to the memory-intensive mcf is shown in Figure 6. Note that this error rate accounts for the very 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.

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

**Equation 4 DMA+Interrupt Disk Power Model**

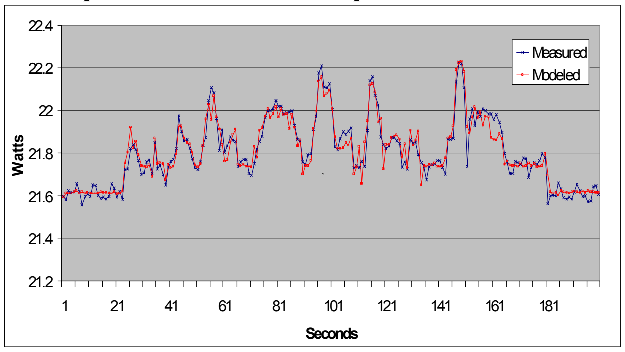

**Figure 6 Disk Power Model (DMA+Interrupt) – Synthetic Disk Workload**

### 4.2.4 I/O Power

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. We considered three events to observe DMA traffic: DMA accesses on memory bus, uncacheable accesses and interrupts. Of the three, interrupts/cycle was 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/each), 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. 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 we found interrupt events to be better predictors of I/O power consumption. DMA events failed to capture the fine-grain power variations. DMA events tended to have few rapid changes, almost as if the DMA events had a low-pass filter applied to them. The interrupt model is pictured in Figure 7 has less than 1% error on average. The details of the model can be seen in Eq.5. 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 with 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, our 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.

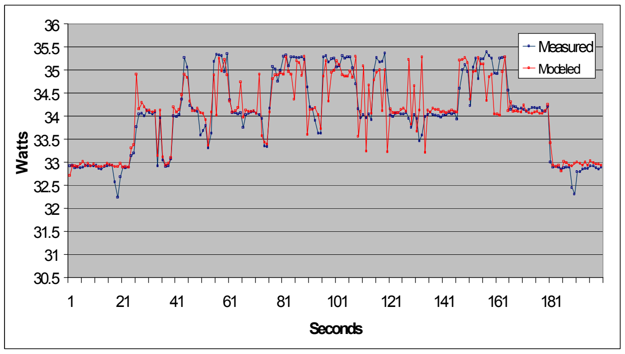

**Figure 7 I/O Power Model (Interrupt) – Synthetic Disk Workload**

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

**Equation 5 – Interrupt I/O Power Model**

### 4.2.5 Chipset Power

The chipset power model we propose is the simplest of all subsystems as we suggest that a constant is all that is required. There are two reasons for this. First, the chipset subsystem exhibits, very 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 our sensors. The second, and more critical reason, is a limitation in our power sampling environment. Since the chipset subsystem uses power from more than one power domain, we cannot measure the total power directly. Instead we derive it from multiple domains. Unfortunately, since a non-deterministic relationship exists between some of the domains, it is not possible to predict chipset power with high accuracy. Therefore, we assume chipset power to be a constant 19.9 Watts.

## 4.3 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.

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

**Equation 6 – Average Error Calculation**

The I/O and disk models performed well under all workloads. The very low error rates are partly due to low power variation / high idle power consumption. The CPU and memory subsystems had larger errors, but also larger workload variation. The worst case errors for CPU occurred in the memory-bound workload: mcf. Due to a very high CPI (cycles/instruction) of greater than 10 cycles, our fetch-based power model consistently underestimates CPU power. This is because under 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 we suspect the cause of underestimation to be access pattern. In particular our model does not account for differences in the power for read versus write access. Also, we do not directly account for the number of active banks within the DRAM. 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 was low for all cases indicating a good match for the DC term in the models. Chipset error was very high considering the small amount of variation. This is due to the limitation of the constant model we assumed for chipset power.

Table 3 Integer Average Model Error

| | CPU | Chipset | Memory | I/O | Disk |
|---|---|---|---|---|---|
| idle | 1.74% | 0.586% | 3.80% | 0.356% | 0.172% |
| gcc | 4.23% | 10.9% | 10.7% | 0.411% | 0.201% |
| mcf | 12.3% | 7.7% | 2.2% | 0.332% | 0.154% |
| vortex | 6.53% | 13.0% | 15.6% | 0.295% | 0.332% |
| dbt-2 | 9.67% | 0.561% | 2.17% | 5.62% | 0.176% |
| SPECjbb | 9.00% | 7.45% | 6.14% | 0.393% | 0.144% |
| DiskLoad | 5.93% | 3.06% | 2.93% | 0.706% | 0.161% |
| Integer Average | 7.06 ±3.50% | 6.18% ±4.92% | 6.22% ±5.12% | 1.16% ±1.97% | 0.191% ±0.065% |
| All Workload Average | 6.67 % ±3.42% | 5.97% ±4.23% | 8.80% ±5.54% | 0.824% ±1.52% | 0.390% ±0.492% |

Table 4 Floating-Point Average Model Error

| | CPU | Chipset | Memory | I/O | Disk |
|---|---|---|---|---|---|
| art | 9.65% | 5.87% | 8.92% | 0.240% | 1.90% |
| lucas | 7.69% | 1.46% | 17.51 % | 0.245% | 0.307% |
| mesa | 5.59% | 11.3% | 8.31% | 0.334% | 0.168% |
| mgrid | 0.360% | 4.51% | 11.4% | 0.365% | 0.546% |
| wupwise | 7.34% | 5.21% | 15.9% | 0.588% | 0.420% |
| FP Average | 6.13% ±3.53% | 5.67% ±3.57% | 12.41% ±4.13% | 0.354% ±0.142% | 0.668% ±0.703% |
| All Workload Average | 6.67 % ±3.42% | 5.97% ±4.23% | 8.80% ±5.54% | 0.824% ±1.52% | 0.390% ±0.492% |

## 5 Conclusion

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

## 6 References

[1] Frank Bellosa. The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems. ACM SIGOPS European Workshop, September 2000.

[2] Canturk Isci, and Margaret Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. International Symposium on Microarchitecture, December 2003).

[3] W. Lloyd Bircher, Madavi Valluri, Jason Law, Lizy John. Runtime identification of microprocessor energy

saving opportunities. International Symposium on Low Power Electronics and Design, pp 275-280, August 2005.

[4] Parthasarathy Ranganathan, Phil Leech, David Irwin and Jeffrey Chase. Ensemble-Level Power Management for Dense Blade Servers. International Symposium on Computer Architecture, June 2006

[5] Tao Li and Lizy John. Run-Time Modeling and Estimation of Operating System Power Consumption. Conference on Measurement and Modeling of Computer Systems, June 2003.

[6] Kyeong Lee and Kevin Skadron. Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. High-Performance, Power-Aware Computing, April 2005.

[7] David Brooks, Vivek Tiwari, and Margaret Martonosi, Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, International Symposium on Computer Architecture, June 2000.

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

[9] John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng, Arvind Krishnamurthyy, Randolph Wang. Modeling Hard-Disk Power Consumption. File and Storage Technologies 2003.

[10] Youngjae Kim, Sudhanva Gurumurthi and Anand Sivasubramaniam. Understanding the performance-temperature interactions in disk I/O of server workloads. The Symposium on High-Performance Computer Architecture, pages 176- 186, February 2006.

[11] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, N. Vijaykrishnan, Mahmut Kandemir, Tao Li, Lizy Kurian John. Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach, Proceedings of the 8th International Symposium on High-Performance Computer Architecture, pages 141-150, 2002.

[13] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and Freon: Temperature Emulation and Management in Server Systems. International Conference on Architectural Support for Programming Languages and Operating Systems, pages 106-116, October 2006.

[12] Gilberto Contreras and Margaret Martonosi. Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events. International Symposium on Low Power Electronics and Design, pages 221-226, August 2005.

[14] Conversation with Gregg McKnight, IBM Distinguished Engineer, xSeries Division. September 2004.

[15] Ramakrishna Kotla, Soraya Ghiasi, Tom Keller and Freeman Rawson. Scheduling Processor Voltage and Frequency in Server and Cluster Systems High-Performance Power-Aware Computing April 2005.

[16] Karthick Rajamani and Charles Lefurgy. On Evaluating Request- Distribution Schemes for Saving Energy in Server Clusters. International Symposium on Performance Analysis of Systems and Software, pp 111-122, March 2003.

[17] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang and Natarajan Gautam, Managing Server Energy and Operational Costs in Hosting Centers. ACM SIGMETRICS, pp 303-314, June 2005.

[18] Ashutosh Dhodapkar and James Smith. Comparing program phase detection techniques. International Symposium. on Microarchitecture, pages 217-228, December 2003.

[19] Jeremy Lau, Stefan Schoenmackers and Brad Calder. Structures for Phase Classification. International Symposium on Performance Analysis of Systems and Software, pages 57-67, March 2004.

[20] Canturk Isci and Margaret Martonosi, Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques. International Symposium on High-Performance Computer Architecture, pages 122-133, Feb. 2006.

[21] Pat Bohrer, Elmootazbellah N. Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ram Rajamony, The Case For Power Management in Web Servers. IBM Research, Austin TX 78758, USA. www.research.ibm.com/arl

[22] Aqeel Mahesri and Vibhore Vardhan, Power Consumption Breakdown on a Modern Laptop, Workshop on Power Aware Computing Systems, December 2004.

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

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

[25] Open Source Development Lab, Database Test 2, www.osdl.org/lab_activities/kernel_testing/osdl_database_test_suite/osdl_dbt-2, February 2006.

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

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

[28] Brinkley Sprunt. Pentium 4 Performance Monitoring Features, IEEE Micro, July-August, pages 72-82, 2002.