Myths in PMC-based Power Estimation

Jason Mair, Zhiyi Huang, David Eyers, and Haibo Zhang

Department of Computer Science
University of Otago
Dunedin, New Zealand
{jkmair;hzy;dme;haibo}@cs.otago.ac.nz

Abstract. Many techniques have previously been proposed for using low-level CPU Performance Monitoring Counters in power estimation models. In this paper, we present some common myths of these techniques, and their potential impact. Such myths include: (1) sampling rate can be ignored; (2) thermal effects are neutral; and (3) memory events correlate well with power. We aim to raise the awareness of these interesting issues, which existing power modeling techniques usually do not address. Our discussions provide some guidance to avoid these myths and their effects through detailed specification of software and hardware configurations.

Keywords: power estimation; Performance Monitoring Counter (PMC); power metering model;

1 Introduction

Much previous research has focused on building accurate power estimation models based upon Performance Monitoring Counters (PMCs) [13,8,14,12,11,10]. These techniques allow runtime power estimation, on a per-application basis, without requiring the use of a power meter or special hardware. They are very useful in power metering of virtual machines in cloud computing [15,16]. Even hardware-based power metering in Intel's Sandy Bridge microarchitecture has to use PMC-based estimation to measure the dynamic power of the cores [9]. However, the research into the use of these techniques needs to be coupled with an improvement in the specification of the hardware and software configurations used in the process of the researchers' power modeling. In previous research, these configuration details were normally given a brief mention, but lacked discussion and analysis on their potential impacts on the resulting power model. Our experiments show that many of these configuration details are crucial in order to achieve reproducible results. The omissions that we have encountered can often be attributed to reasons such as assumption of background knowledge, limited space for publication, or indifference to the authors interpretation of the experimental results.

Detailed specification of the hardware and software configurations used in the process of power modeling is also crucial for the avoidance of myths in this research area. It can help prevent the readers from drawing erroneous conclusions.

For example, suppose there are two different power models with an identical mean measuring error of 5%, but for different machines with an equal total power consumption of 200W. Without further information, the natural conclusion that can be drawn by the reader is both power models are equally accurate. However, a different conclusion can be drawn if it was additionally known that machine 'A' has a dynamic power (aka workload dependent power) of 30%, while machine 'B' has a larger dynamic power of 60%. Given a total error of 10W (5% of total 200W) and the constant static power (aka base power), it is now clear that the model based on machine 'A' has a 16.6% error for dynamic power while the model based on machine 'B' has a much lower error of 8.3% for dynamic power. The extra information on the proportion of static power and dynamic power enables a fair comparison to be made between the two, otherwise identical, power estimation models.

In this paper, we discuss the impact of hardware and software configurations and the possible myths in the research area of PMC-based power estimation due to the lack of detailed information. More precisely, we have identified the following three possible myths.

- *Sampling rate can be ignored*—In previous research, the sampling rate for taking PMC-power samples is seldom mentioned. Very often, when a benchmark is run, the PMC counters are recorded for the whole execution period of the benchmark and the average power of the benchmark during the period is used in the sample. However, we will show that the sampling rate can affect the accuracy of the power estimation.
- *Thermal effects are neutral*—Thermal effects are often not discussed in power estimation. Many experimental results were given without mentioning the temperature condition of the CPU chips. We will show that the temperature of the CPU chips has an effect on the accuracy of the power model and its effect can be eliminated with proper treatment.
- *Memory events correlate well with power*—Previous research often assumes a correlation between memory events and power. Intuitively memory events like cache misses should correlate with power consumption due to many memory fetches. We will show that this is not the case on our multicore system possibly due to a different memory architecture. This observation gives us inspiration that detailed specification of software and hardware configurations is very important for fair comparison and deep understanding of different PMC-based power models.

The intention of this paper is not to criticize any of the existing work, or to propose any alternative methodologies. Instead, our objectives are twofold. First, we would like to raise awareness of the potential pitfalls in the research area. Second, we want to stress the importance of detailed specification of software and hardware configurations in the process of power modeling. If experimental results are published with sufficient context and specification, it helps avoid the myths that we will discuss in power modeling.

The remainder of this paper is organized as follows. Section 2 describes our experimental setup and software/hardware configuration. The myths are discussed in Section 3. Related work is in Section 4, with the conclusions in Section 5.

2 System configuration

In this section and throughout this paper, we make a conscious effort to specify all relevant configuration information, allowing experimental results to be placed in the appropriate context. It will be shown through this paper that changes in sampling rate, execution time and configuration of benchmarks impact the resulting model. If such information was not explicitly specified, as is often the case in many published papers, it becomes harder to compare fairly among alternative approaches and power models. A commonly omitted data value in a power estimation model is the amount of static vs. dynamic power within a system. Static power is the constant, workload independent power consumption of components like the Power Supply Unit (PSU). Dynamic power changes according to the workload, with the most obvious example of the CPU.

Care should be taken to ensure that any neglected details do not impact on the resulting power model or create the potential for erroneous conclusions. Often researchers neglect the detailed specification of system parameters due to limited publication space. Unfortunately, a large number of parameters can have a significant effect on the experimental results. The architecture-specific nature of PMC-based power estimation means it is important to document all relevant hardware and software configurations.

It is for these reasons that this section takes the time to describe the overall experimental setup of hardware and software configuration, including the design of our own micro-benchmark. Certain experiment-specific configuration details are left until the corresponding results sections.

2.1 Experimental setup

The experiments are run on a Dell PowerEdge R905 with four quad-core AMD Opteron 8380 processors (CPUs), with each core having its own floating-point unit (FPU). Each processor is located with 4GiB of memory organized in a NUMA (Non-Uniform Memory Accesses) architecture, providing a total of 16GiB RAM. The processor provides four alternate operating frequencies through DVFS (Dynamic Voltage and Frequency Scaling), however we restrict the frequency to the highest (2.5 GHz) for our experiments as it is the most commonly used frequency. All benchmarks are compiled using gcc 4.6.3 with no optimization enabled, ensuring none of the micro-benchmark operations are optimized away. OpenMP 3.0 [5] is used for the NAS Parallel Benchmarks, which are compiled with gcc 4.6.3, using optimization argument -O3. All benchmarks were running on a standard installation of Linux version 2.6.32-25.

The power is measured with the Watts Up? PRO .net power meter, connected via USB to an external monitoring system. The accuracy of the power meter is $\pm 1.5\% + 0.3$ watts [1]. An iSocket (InSnergy Socket)[1] power meter was additionally used to validate power measurements, which has an accuracy of 1%. The measured base/idle power (i.e., static power) for our server is 249W.

[1] Institute for Information Industry, `http://web.iii.org.tw/`, who we thank for providing this measurement equipment.

The monitoring system was additionally configured to remotely monitor the server's system components and temperatures, while recording the power. This was achieved by altering an instance of IPMItool [7] to log specific measurements at the same rate as the power values. Communication is handled by the Intelligent Platform Management Interface (IPMI) over LAN, bypassing the OS, directly communicating with the Baseboard Management Controller (BMC). Thus, this measurement causes no overhead on the test server.

All performance values are collected from the PMCs, which are a set of four, per-core hardware registers [6]. Each register is set to record one of the 120 available performance measures at the start of each execution.

2.2 Benchmark configuration

The myths discussed in this paper come from observations made during our work on deriving a runtime power estimation model based on PMC values. The idea behind such a model is to find the relationships between key PMCs, the workload type they represent and the resulting power use. For example, if the PMC for FPU utilization is high, the current workload is FPU intensive, causing a large power draw. Alternatively, if the number of cache misses is high, there may be many memory accesses, resulting in lower utilization of the processor and a lower power draw from the processor.

To explore these basic principles, we created a micro-benchmark designed to reproduce five key workload types that could be expected within a system, shown in Table 1. The micro-benchmark, as shown in Program 1, consists of a larger outer loop, intended to perform a large number of iterations, and a small `for` loop for short bursts of each workload type. The inner `for` loops execute quickly, ensuring that whenever the PMCs are sampled during execution, a mix of workload types will be represented. Modest variations in this burst time are provided by the pseudo random numbers. This is then multiplied by a ratio, which is designed to allow extra weight to be added to a single workload type during execution.

Table 1. Types of workload

Micro-benchmark	Description
FPU	Floating point multiplication
INT	Integer multiplication and division. Represents most micro operations
memory	Random memory accesses
NOP	idle loop with NOPs
cache	Memory accesses with high cache-hit ratio

By default, all ratios are set to one. However, if we are interested in the impact of a more FPU-oriented workload, we increase the FPU ratio, leaving all other ratios the same. The ratios shown throughout this work are 1, 2, 4, 6 and 8. To avoid any synchronization overheads, 16 independent concurrent instances are run across all 16 cores.

```
for(large_number_of_iterations)
  for(pseudo_random_number x fpu_ratio)
    fpu_micro();
  for(pseudo_random_number x int_ratio)
    int_micro();
  for(pseudo_random_number x memory_ratio)
    memory_micro();
  for(pseudo_random_number x nop_ratio)
    nop_micro();
  for(pseudo_random_number x cache_ratio)
    cache_micro();
```

Program 1: Pseudo-code of the micro-benchmark

3 Myths

Many alternative models have been proposed for PMC-based power estimation [13,8,14,12,11,10]. They commonly used a black-box approach to data processing, where the regression function is directly applied to the data without necessarily being visualized. In contrast, this section describes some of the observations made during our thorough analysis and visualization of the collected experimental data in the process of power estimation modeling. We will also disclose the differences of the resulting model due to the system architecture used and the selection of statistical methods.

Not all of our observations have gone without a note in the literature, but often only warrant a passing mention without much discussion or analysis. This may be due to space limitations. Also it is worth noting that failing to disclose all relevant information would not affect the correctness of the previous work.

In this section, each subsection introduces a myth that could be observed in published literature. Each myth is followed by a series of observations in our experiments and our approach to avoid the myth. Each myth is then concluded with a brief discussion of how this should be taken into account within our modeling process.

3.1 Myth 1: Sampling rate can be ignored

The rate at which PMCs and power samples are taken can have a direct impact on the strength of correlation and the noise within a dataset. This is intuitive, as it will determine the aggregation of results, but is often not documented.

Observation: execution time and sampling rate The first step in deriving the power model will consist of collecting some initial coarse-grained data for a range of PMCs, sampled once when execution begins and again upon completion of each micro-benchmark iteration. The TSC (Time Stamp Counter) is additionally read whenever PMCs are read in order to provide a measure of time. TSC measures the time as the number of cycles since reset, allowing the intensity of the PMC-related event to be calculated during execution.

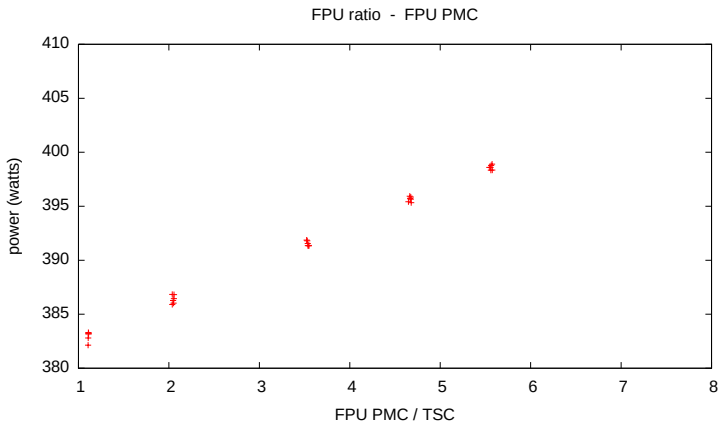

Fig. 1. Intensity of FPU correlated with power measured over entire execution of micro-benchmark running an FPU workload.

For example, Figure 1 shows some intensity values for the FPU activities for multiple iterations of the micro-benchmark, introduced in Section 2.2, in different configurations. The x-axis is the intensity of FPU activities, which is calculated by taking the difference in the two FPU_{PMC} measurements of the execution period and dividing it by the difference in the corresponding TSC values, which provides the intensity of the activities during the execution period. The y-axis is the calculated average power use, measured by the power meter, over the same execution period. In this example, the FPU ratio is adjusted for different iterations, providing the spread of data clusters along the x and y-axis.

In Figure 1, the data points form a series of tight clusters along a linear path, which is what was expected according to previous research.

Knowing this works at the most basic level, the logical next step is to increase the rate at which samples are taken. This was chosen to be once every second, in order to match that of the power meter. Each PMC value and power measurement are logged to a file during execution, with a corresponding timestamp, to allow synchronization and post-processing. This time, the results shown in Figure 2 were not what was expected. The modest horizontal spread of points within each cluster is due to the adjustments in the pseudo random number in the algorithm. However, the vertical stripping was not expected at all. This indicates that something within the system is causing the power values to vary during execution, which was previously obscured by the coarse-grained samples.

Since the micro-benchmark consists of a fairly consistent workload, it was suspected that the stripping was due to an inherent latency in the system responding to starting execution. If such a latency exists, the trend in the data should become more linear with increased execution times. Therefore, each configuration was re-run, increasing the execution time from $\sim$ 3 minutes to $\sim$ 25 minutes. The results in Figure 3 show that when sampled over a longer period with the same sampling rate, data points return to lying on a linear path, though there is a long vertical tail in each cluster, which will be explained in the next

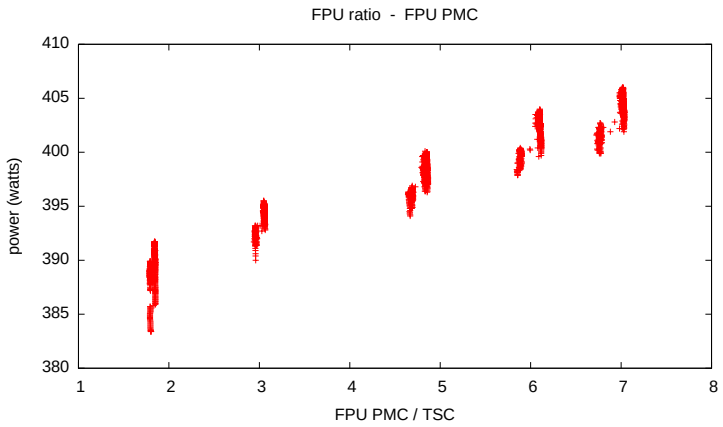

Fig. 2. Intensity of FPU correlated with power, sampled every second for ∼3 minute execution of the micro-benchmark running FPU workload. Vertical stripes are a result of a warm-up effect.

section. The spread of points for each cluster along the x-axis is a result of the increased range of pseudo random times made available by the significantly increased iteration count.

In this section we have seen two ways in which important data may be obscured. First, the extreme case of using a sampling rate which is too coarse-grained for the benchmark being sampled. Second, execution times which are too short will hide longer run trends within a dataset.

Observation: benchmark configuration and execution time The previous observation not only illustrated the importance of choosing an appropriate rate at which to sample a benchmark, but also the importance of ensuring a sufficiently long execution time. However, care must be taken not to introduce more noise when increasing execution time. While the micro-benchmarks experience minimal workload variation during execution, this will not be the case for all other benchmarks. Therefore, increasing execution time in the same way will not have the same effect, contributing more noise than expected.

To test this hypothesis, we ran an OpenMP instance of the Fast Fourier Transform (FFT) benchmark from the NAS Parallel Benchmark (NPB) suite in two different configurations. To ensure each instance to experience the same latency from startup, a 10 minute idle period is run before each instance begins. The first configuration had a problem size of $512 \times 512 \times 512$, completing 250 iterations. The mean power measured was 403.6W with a standard deviation of 56.19W. Looking at the data we found many small periods of low power use during the execution due to the large number of iterations performed, explaining the large deviation in measured power.

For comparison purposes, an alternative configuration was run with a problem size of $1024 \times 512 \times 512$, completing 100 iterations. The mean power of 425.0W was much closer to the peak power, with a smaller standard deviation, 36.2W.

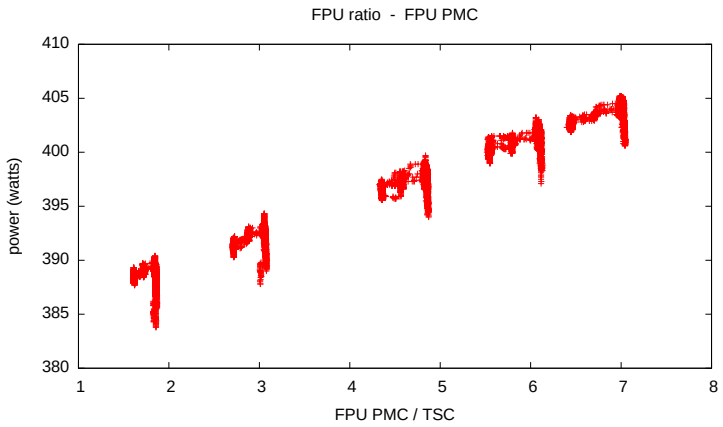

Fig. 3. Intensity of FPU correlated with power, sampled every second for ∼25 minute execution of the micro-benchmark running FPU workload. The linear trend is more clear though vertical stripes still exist due to a warm-up effect.

From this observation, we find it is important to carefully consider how benchmarks are configured to avoid introducing any unexpected noise in measured power.

Discussion When selecting the rate at which to sample the PMC counters, it is important to consider potential smoothing effects if samples are too coarse grained. Not doing so will obscure data trends and characteristics for the sampled benchmark. Similar to this is the importance of execution time, as this additionally impacts the number of samples for a given rate, potentially further obscuring long-run trends.

Documenting both the sampling rate and the execution time help to add context to the commonly reported statistics like the mean power. The significance of such values can be questionable and statistically inaccurate if the data set is believed to be too small.

Also it is good to give the standard deviation of the measured power so that the smoothness of the power changes is known. Mean power value can hide the smoothness of the power trend, as two power trends with the same mean value can have very different standard deviations. Standard deviation can be used to reflect how reliably the mean power value is used to characterize the power feature of the execution period.

3.2 Myth 2: Thermal effects are neutral

Thermal effects are often not discussed in power modeling. For those who are aware of the thermal effects on power consumption, it is commonly perceived that the thermal effects can be negated by locking the fan speed, believing the change of fan speed is the main cause of the variation in power due to changes in thermal load. An alternative technique to locking the fan speed is the use of a CPU warm-up phase before the start of each execution. However, we find neither of these can sufficiently negate the thermal effects.

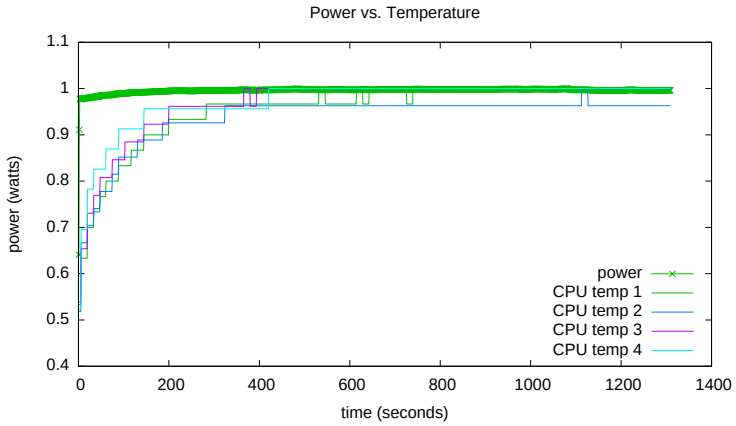

Fig. 4. Normalized CPU temperatures and power meter readings for a long-run execution of the micro-benchmark running FPU workload.

Observation: fan speed and power Section 3.1 mentioned the presence of a long-run latency in power changes corresponding to the beginning of each micro-benchmark execution, which causes the vertical long tail of each cluster in Figure 3. The most likely cause of delayed effect on power within a system is temperature related. To explore this, we used IPMI to monitor CPU temperatures once a second during execution of each micro-benchmark. Figure 4 shows the normalized values for power and temperature of all four CPUs on the y-axis. The x-axis gives the time in seconds. The power curve steadily increases until around 400 seconds where it flattens out. The recorded temperatures follow a similar trend where they continue to increase until about 400 seconds, reaching a stable point with the exception of an occasional temperature spike.

These results illustrate a trend between CPU warm-up and the corresponding power latency. The most surprising aspect of this is the length of time required to reach a stable value, 400 seconds. In many cases this will be longer than the execution time of the benchmarks.

To further confirm this relationship, IPMI was used to monitor the fan speed for each CPU. Despite the temperatures changing, the fans' speed remains constant at 3600rpm. This speed is even maintained under a high thermal load running a CPUburn benchmark [4]. Contrary to the common belief, the power latency is not caused by changes in the speed of the fans when they respond to an increased thermal load. Therefore, policies designed to lock fan speeds through the BIOS are not capable of negating all of the dynamic thermal effects on power consumption.

Observation: warm-up and cool-down In an attempt to remove the effect of the warm-up phase, the micro-benchmark was re-run after a CPU warm-up phase. There is a 15-minute cool-down period before each iteration of the micro-benchmark starts to execute, in order to ensure consistent starting temperature for each iteration. After that, an instance of the CPUburn was run on each core

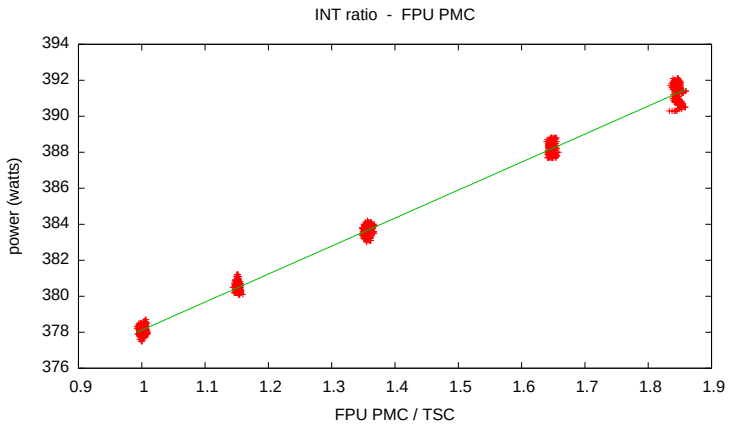

Fig. 5. Intensity of FPU correlated with power, sampled every second for the micro-benchmark running INT workload. Sampling starts after a 60-second period of CPU warm-up.

for different execution times, providing different times of CPU warm-up. The results for two different CPU warm-up lengths of 60 and 90 seconds are shown in Figures 5 and 6, respectively. Same as the previous figures, the x-axis is the intensity of the FPU activities, while the y-axis is the power in Watts. This time, the INT ratio is adjusted between iterations, giving the spread of clusters along the x-axis.

A CPU warm-up phase of 60 seconds, as shown in Figure 5, is enough to eliminate much of the vertical tail caused by the warm-up phase, resulting in a stronger linear correlation. Alternatively, with the CPU warm-up phase of 90 seconds, as shown in Figure 6, it begins to over-warm the CPU, which causes the opposite effect, a vertical stripping above the main linear trend, instead of the vertical tail.

However, different types of workload in the benchmark need different warm-up periods. A 60-second CPU warm-up phase provides the best results for a workload with lots of integer calculations, as shown in Figure 5. However, a workload with lots of floating point calculations requires a 90-second CPU warm-up. These results are illustrated by the Pearson's Correlation Coefficient for the two workload types in Table 2. While differences in correlations are not significant they do illustrate the point that no single CPU warm-up policy is sufficient for all workload types.

Table 2. Pearson's Correlations for different workload type with different warm-up times

CPU warm-up time (seconds)	FPU-type	INT-type
30	0.969611	0.977467
60	0.992794	0.997928
90	0.996615	0.992321
120	0.993053	0.985091

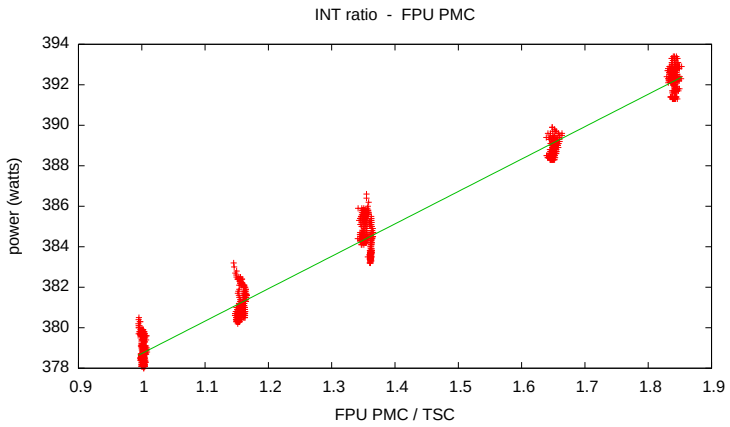

Fig. 6. Intensity of FPU correlated with power, sampled every second for the micro-benchmark running INT workload. Sampling starts after a 90-second period of CPU warm-up.

Discussion Temperature variations within a system have the potential to adversely impact the accuracy of a power estimation model. For example, an instance of the micro-benchmark running a FPU workload initially uses 397W, stabilizing at around 405W. This gives an error of 2% of total power, and more significantly, 5% of dynamic power. Due to the myth of the thermal effects, there is no single solution designed to mitigate all thermal effects.

Unfortunately, it is not likely that a single policy exists to reliably remove all warm-up effects on power consumption. The most likely cause for the warm-up effects is static power leakage from the processor, which is due to the high temperatures. For example, a 12% reduction in CPU (dynamic) power was made in [2] by reducing the operating temperature, while maintaining the same voltage and frequency 4.6GHz.

It might seem that the only way to reliably monitor thermal effects is through embedded temperature sensors. However, since their placement inside the socket is some distance away from the top of CPU, embedded sensors do not provide reliable temperature data [3]. Also such sensors were not designed for high precision temperature reading, as their purpose is to provide an early warning system to prevent hardware damage.

In summary, we have made two key observations. First, benchmarks can experience a large warm-up effect on power consumption during their start-up, which is not due to the changes in fans speed. Second, the length of warm-up phase required varies between different workloads, as illustrated through differences in the warm-up times required by FPU and INT workloads. That means using a fixed period of warm-up for all workloads will not achieve the desired result.

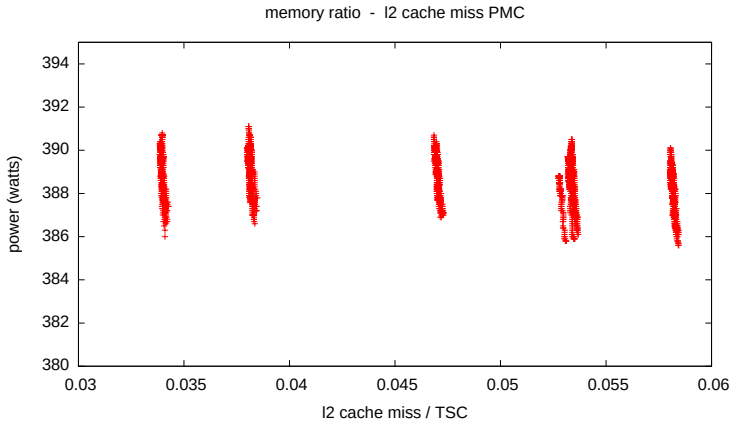

Fig. 7. Intensity of L2-cache misses correlated with power , sampled every second for the micro-benchmark running memory workload.

3.3 Myth 3: Memory events correlate well with power

The correlation between memory-related PMCs like cache miss and memory activities is intuitive and well known. However, there is a myth that memory-related PMCs correlate equally well with power consumption. This proved not to be true for our multicore system.

Observation: neutrality of memory-related PMCs Memory-bound and CPU-bound workloads exhibit quite different power characteristics and in most cases are therefore treated differently. For power estimation, it is also common to use those PMCs with a direct logical connection to memory use. This intuitively makes sense, and is what we expected to be the case too. In modeling the power use of memory workloads, we used PMCs directly related to memory, such as, instruction cache miss, data cache miss, L2-cache miss, L3-cache miss, DTLB miss and DRAM accesses.

Surprisingly, none of these counters provided a strong correlation between memory use and power consumption. To illustrate this, the results for L2-cache misses are plotted in Figure 7. Our micro-benchmark has been configured to execute a large number of memory accesses by increasing the ratio for memory accesses, leaving all other micro-benchmark ratios at the default value of one. The x-axis is the intensity of cache misses, which is calculated taking the difference of two L2-cache-miss PMC values divided by the elapsed TSC value. The y-axis is the measured power. We collect the intensity and power samples at every one second. The warm-up effect, seen by the vertical stripping within each cluster, is present since there is no CPU warm-up phase before data collection.

The most notable observation to be made in Figure 7 is the distinct lack of any vertical offset between clusters. It seems that power is not functionally determined by L2-cache misses. The same results have been found for all other memory-related PMCs mentioned, so we do not repeatedly show the results here.

Discussion A common approach taken in building a power model is to decompose the processor and the expected workload type into several key components, such as FPU, Memory, Stalls, and Instructions Retired [8]. Each component requires a specific, strongly correlated PMC to represent its power consumption. In the case of memory, a PMC like cache misses is expected to correlate well with the activities of the memory subsystem. This approach has worked in other power estimation models [12], but failed to do so on our experimental system due to the neutrality of memory-related PMCs to power consumption.

This difference of results can likely be attributed to the architectural differences, though we are not sure which components have caused the difference, as there are several components which could possibly contribute to such differences. The first component could be the memory architecture. Our system uses NUMA, where, unlike some systems of the previous work, there is no single memory bank shared between all processors. The memory in our system is arranged in 4GiB blocks beside each of the four processors. Given the random memory accesses, extra overhead may be incurred if memory accesses are shifted to a remote processor's memory block.

Also the processors in our system lack the ability to sleep, even at low levels. That means the processor maintains a busy loop or executes some other work while waiting for requests from the memory subsystem. Given the high thermal latencies, temperatures will remain high, despite lower levels of utilization.

4 Related Work

Much of the prior work on PMC-based power estimation has taken the approach of using PMCs to model the underlying architectural components, which are applied in a variety of use cases. Singh et al. [8] proposed a model which used micro-architectural knowledge to decompose the processor into its four main functional units: FP Units, Memory, Stalls, and Instructions Retired. PMC selection is made from initial data collected from the execution of the SPEC benchmark suite. A separate micro-benchmark is designed for each of the four PMCs most strongly correlated with power for each functional unit. The micro-benchmark data is used to form a piece-wise linear function.

Bertran et al. [12] take an even finer grained approach by starting with a set of about 97 micro-benchmarks designed to individually highlight all possible power components. This results in multiple linear equations with an input for each of the seven derived power components. During the runtime period, PMC multiplexing is required, as the micro-architecture does not allow that many counters to be collected simultaneously.

Da Costa et al. [14] present a methodology intended to broaden the range of modeled workloads by supplementing PMC values with process and system level statistics. This means the resulting model, derived through multivariate regression, is not limited to estimating the power of CPU and memory workloads, allowing accurate power estimation of the network and disk synthetic benchmarks.

Such models derive a single, global power estimation function typically used for monitoring system power on a per application basis. Alternatively, Alonso et

al. [11] takes this more targeted approach in proposing a framework for instrumenting source code functions with power metering. The API logs PMC data and power values to derive a specific power model offline, enabling execution traces of power to be used during runtime estimation.

Wang et al. [10] takes the novel approach of using the fewest PMCs possible. The model was built using only CPU operating frequency and IPC, making it universal across microarchitectures. It is built into the SPAN libraries and interfaces to provide source code power estimation.

Dhiman et al. [16] presented a model for accurate power estimation in a virtualized environment. A performance counter manager is run on each host machine, designed to collect and correlate PMC events to each VM. Power estimates are then periodically made for each VM using classification based Gaussian mixture models.

Only a select sample of previous work is presented here to demonstrate some alternate uses for PMC-based power estimation models. A more comprehensive survey on hardware, software and hybrid power estimation techniques can be found in [10]. Given this varied use of PMC-based power models, each myth within this paper was presented and discussed without explicit guidance, ensuring all conclusions remain relevant and universally applicable to different approaches.

5 Conclusions

In this paper we have presented and discussed three myths in PMC-based power estimation models: sampling rate can be ignored; thermal effects are neutral; and memory events correlate well with power. The truth of each myth is revealed through a series of observations made while deriving our own PMC-based estimation model.

Such myths have arisen due to a lack of configuration specifications and accompanying analysis in published literature. This is of particular importance for PMC-based power estimation as the models derived are largely architecture dependent. As we have seen, changes in hardware and software configurations can adversely impact the resulting power models.

While failing to disclose all relevant information would not affect the correctness of the previous work, it can lead to erroneous conclusions being drawn from the results. In raising awareness of the impact of the missing relevant information, we hope researchers in this community more readily document hardware and software configurations in the future. Doing so will prove to be beneficial to the advancement of the research community.

Acknowledgement

This work was partially supported by the COST (European Cooperation in Science and Technology) framework, under Action IC0804.

References

1. Watts up? operators manual., https://www.wattsupmeters.com/secure/downloads/manual_rev_9_corded0812.pdf
2. Anandtech, Overclocking CPU/GPU/Memory Stability Testing Guidelines, http://forums.anandtech.com/showthread.php?p=34255681
3. Overclockers, Reconciling CPU Temperature Measures, http://www.overclockers.com/reconciling-cpu-temperature-measures/
4. Ubuntu Manuals, CPUburn, http://manpages.ubuntu.com/manpages/precise/man1/cpuburn.1.html
5. O.A.R. Board, OpenMP Application Program Interface Version 3.0, May 2008.
6. AMD. BIOS and Kernel Developer's Guide (BKDG) For AMD Family 10h Processors, 2009.
7. IPMItool, http://ipmitool.sourceforge.net/
8. K. Singh, M. Bhadauria, and S. A. McKee. Real time power estimation and thread scheduling via performance counters. SIGARCH Computer Architecture News, 37(2):4655, 2008.
9. Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A., and Weissmann, E.: Power Management Architecture of the 2nd Generation Intel Core microarchitecture, formerly codenamed Sandy Bridge. In Hot Chips: A Symposium on High Performance Chips, Aug. 2011.
10. S. Wang, H. Chen, and W. Shi. SPAN: A software power analyzer for multicore computer systems. Sustainable Computing: Informatics and Systems, 1(1):2334, 2011.
11. P. Alonso, R.M. Badia, J. Labarta, M. Barreda, M.F. Dolz, R. Mayo, E.S. Quintana-Orti, and R. Reyes. Tools for Power and Energy Analysis of Parallel Scientific Applications. Proceedings of International Conference on Parallel Processing (ICPP), Sept. 2012.
12. R. Bertran, M. Gonzlez, X. Martorell, N. Navarro, E. Ayguade. Decomposable and responsive power models for multicore processors using performance counters ICS 10: Proceedings of the 24th ACM International Conference on Supercomputing, ACM, Tsukuba, Ibaraki, Japan (2010), pp. 147158
13. X. Chen , C. Xu , R. Dick and Z. Mao. Performance and power modeling in a multi-programmed multi-core environment, In Proceedings of the 47th Design Automation Conference, ACM, pp. 813-818, 2010.
14. G. Da Costa, and H. Hlavacs. Methodology of measurement for energy consumption of applications. Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on. IEEE, 2010.
15. A. Kansal, F. Zhao, J. Liu, N. Kothari, A.A. Bhattacharya, Virtual machine power metering and provisioning, In Proceedings of the 1st ACM Symposium on Cloud Computing, pp.39-50, 2010.
16. G. Dhiman, K. Mihic, T. Rosing, A system for online power prediction in virtualized environments using Gaussian mixture models, In Proceedings of the 47th ACM IEEE Design Automation Conference, ACM, pp. 807812, 2010.