A Methodology for Developing Simple and Robust Power Models Using Performance Monitoring Events

Kishore Kumar Pusukuri
UC Riverside
kishore@cs.ucr.edu

David Vengerov
Sun Microsystems Laboratories
david.vengerov@sun.com

Alexandra Fedorova
Simon Fraser University
fedorova@cs.sfu.ca

Abstract

In this work we describe a methodology for developing simple and robust power models using performance monitoring events for AMD Quad-core systems running OpenSolaris™. The basic idea is correlating power consumption of a benchmark program with its performance (a measure of performance monitoring events). By using applicable model selection and model assessment techniques, we developed a simple and robust 2-predictor linear power model, which was shown to predict the power consumption of a testing set of benchmarks with better than 95% accuracy on average after being trained on a different set of benchmarks. Unlike previous power models, our model works across multiple CPU frequencies and relies on only two performance events available on most modern CPUs.

1 Introduction

The objective of this paper is to present a methodology for developing simple and robust system power models by correlating system power consumption with performance monitoring events. Knowledge of system power consumption is needed by system administrators to monitor system power statistics, by operating system designers to develop power-aware scheduling algorithms and dynamic power management policies. Unfortunately, hardware sensors for observing power consumption are not available in most modern systems. Therefore, system designers are relying on temperature sensors for observing power consumption. However, because of the thermal inertia in microprocessor packaging, detection of temperature changes may occur significantly later than the power events that caused them [1]. Therefore, it is extremely helpful to use a measure of performance counters as a proxy of power consumption instead of the inaccurate power observations from the temperature sensors.

Although various power models based on performance event counters have been demonstrated by other researchers [1–5], we improve on the previous work by (1) providing a detailed methodology for deriving a performance model for the target hardware, (2) developing a single model that works for all available CPU frequencies in the system (previous work required a different model for each frequency), and (3) developing a model that works well with only two performance events (previous work relied on many more events). Through extensive validation on SPECCPU benchmarks, we show that the models developed based on our methodology can estimate power consumption with better than 95% average accuracy. That is, on average the measured power values are within five percent of the estimated values.

One of the goals for our future work is to develop algorithms for smart Dynamic Voltage and Frequency Scaling (DVFS) based on our power prediction methodology. Therefore, we focused on predicting the power consumption of the CPU/memory subsystem, which is most sensitive to changes in the CPU frequency. We started by selecting 8 CPU performance events that were used in models described in previous work. Then, we derived a simple 2-predictor model (based on CPU unhaulted cycles and last level cache misses) by applying statistical model selection and model assessment techniques. Through extensive experimentation we found that this model has a better prediction accuracy on the standard industry benchmarks (those that stress the CPU/memory subsystem): SPECCPU 2000, SPECCPU 2006, and SPECjbb2005. Also, since fewer predictors are used, the derived model is less prone to over-fitting and is more likely to be robust on the new data. As a result, our methodology has a good portability across modern x86 hardware platforms since all of them are likely to have the two event counters we chose.

This paper is structured as follows: Section 2 gives a brief description of related work, Section 3 explains the methodology and the experimental environment, and also presents our results. Finally, Section 4 presents conclusions and discusses future work.

2 Related Work

The idea of predicting power consumption from performance events is not new. Bellosa et al. [3] developed a CPU power model for thermal management with 8 predictors in a Pentium 4 system. Bircher et al. [1] developed power models for each subsystem (CPU, Memory, IO, Disk) of an Intel XScale System. Gilberto Contreras et al. [2] developed different models for different CPU frequencies in an Intel PXA255 system. Stoess

et al. [4] and Lee et al. [5] developed power models for hypervisor-based virtual machines and for run-time temperature sensing in high-performance processors respectively. Snowdon et al. [6] developed 4-predictor power models for an Intel PXA255 system. Canturk Isci et al. [7] developed techniques to characterize power phases using performance event counters and showed that techniques using performance counters provide better power behavior than code-oriented techniques.

Our work is different from the previous studies in terms of the following contributions:

- It demonstrates the use of a statistical methodology for developing simple (with few predictors) power models across different CPU frequencies and different benchmarks using appropriate model selection and model assessment techniques.
- We show how to develop a single model for all frequencies supported by the CPU. Therefore, it is frequency-agnostic: a feature that is not presented in the previous work.
- Our methodology for model derivation is designed to be portable across most modern x86 hardware platforms. That is, the same performance events are to serve as inputs to the model, regardless of the platform. So far we have demonstrated this portability between AMD Shanghai (server) and Phenom (desktop) systems, and our next goal is to apply it to a wider range of hardware platforms.

3 Methodology

The basic ideas behind the methodology we used for deriving the power prediction models are described below.

Step 1: Develop a multiple linear regression model with all reasonable predictors (performance monitoring events supported by the system) across all CPU frequencies.

Step 2: Assess the model with "leave-one-out cross-validation" (CV) test (whose details are described later in the paper) and also against a separate test data set, and measure prediction accuracy with any reasonable metric.

Step 3: Rank the accuracy of predictors using "Relative Importance Measures" (RIM) (or any other statistical technique designed for this purpose).

Step 4: Perform model selection by choosing the most effective predictors, develop a model, and assess it as described in step 2.

Step 5: Based on the trade-off between prediction accuracy and the number of predictors, either choose the model obtained in step 4 or choose different predictors based on their ranking in step 3.

The "model selection" (steps 3 to 5) significantly reduces the multi-collinearity [9, 10] in the model inputs. Multi-collinearity refers to a situation in which two or more predictors in a multiple regression model are highly correlated. The best regression models are those in which the predictor variables each correlate highly with the dependent (outcome or target) variable but correlate only minimally with each other. Such a model is often called "low noise" and is expected to be statistically robust [10].

Table 1: Performance Events

Event	**Description**
instructions	The number of instructions retired. This counter includes exceptions and interrupts.
unhalted cycles	The number of cycles that the CPU is not in a halted state.
mispredicted branches	The number of branch instructions retired, of any type, that are not correctly predicted.
branch instructions	The number of branch instructions retired.
L2-cache misses	The number of memory access requests that missed in the L2 cache. The AMD processors used in our experiments have a separate L2 cache for each CPU core.
L3-Cache misses	The number of L3 cache misses for accesses from each core. All cores share the L3-cache (last level cache) in the AMD systems.
dispatch stalls	The number of processor cycles when the decoder was stalled for any reason.
micro-ops	The number of micro-ops retired.

Table 2: Eight predictors and their short names

instructions/tick (inst)	unhalted-cycles/tick (unhalted)
mispre-branches/tick (mispred)	retired-branches/tick (retired)
L2 misses/tick (L2)	L3 misses/tick (L3)
dispatch stalls/tick (stalls)	retired micro-ops/tick (uops)

Table 3: AMD Quad-core 10h Family systems

System	CMOS	CPUfrequency (GHz)	L3	Memory
Shanghai	45nm	2.6, 1.9, 1.4, 0.8	6MB	4GB
Phenom	65nm	2.4, 1.2	2MB	2GB

Experimental Setup and Benchmarks: We instrumented two AMD Quad-core 10h family processor based systems (shown in Table 3) equipped with OpenSolaris™for our experiments. We tested our methodology through extensive experimentation on SPECCPU benchmark programs. All together, a total of 31 benchmarks were selected for the experiments: 22 SPECCPU 2000 integer and floating-point benchmark programs, 8 SPECCPU 2006 benchmark programs, and SPECjbb2005. As these benchmarks only exercise CPU and memory, the variation in the power consumed by

CPU/memory subsystem can be measured by measuring the whole-system power (in watts) taken from the power supply unit. We use "benchmark program" and "workload" interchangeably throughout this paper. We chose the eight performance monitoring events shown Table 1 as a compilation of events used by other researchers in the field.

Data Collection: We sampled the performance counters with a 300 millisecond time interval using the *cputrack* command [16], and we used a data acquisition system (equipped with the power meter Extech #380801) to measure the power samples every 300 milliseconds. We tried sampling rates higher than 300 milliseconds but that increased the sampling jitter. For simplicity we settled on sampling every 300 milliseconds.

The *cputrack* command allows one to read only four events at a time as there are only four counters in the AMD processor (which can count any events we specify). We ran each benchmark eight times, first four times for the first four events and the next four times for the remaining four events. The eight runs for each benchmark were then combined into 4 runs, each containing data for all eight performance events. We derived the predictors (inputs for the model) by dividing the total number of events counted with each counter by the execution time of the workload in terms of clock ticks. The clock tick used here is a frequency-independent metric: the duration of the clock tick equals the length of the processor cycle when the processor runs at the highest frequency.

Each run is considered as a data point. It is represented by a 9-tuple containing 8 predictors and the observed power as the target variable. We gathered data for 22 SPECCPU 2000 benchmark programs at four frequency levels (shown in Table 3), giving us 352 (22* 4 frequency levels * 4 runs) total Shanghai data points. Likewise, we gathered data for the Phenom system. Phenom system supports two CPU frequency levels, giving us 176 (22*2 frequency levels*4 runs) total Phenom data points.

Developing the Power Model: We used the R `lm()` [12] method to develop the power models based on the collected data. The R is a programming language for statistical computing and the `lm()` method is used to fit linear regression models (derive the optimal intercept and coefficients that multiply the input variables). Phenom and Shanghai 8-predictor models derived based on the collected SPECCPU 2000 data are shown below. Here 'Sh' and 'Ph' represent the target parameter power of Shanghai and Phenom models respectively, and the predictors are shown as single and double letters of the short names as per Table **??**.

$Sh = 56 + (3.5 * in) + (40 * un) + (-183 * m) + (-4 * r) + (53 * L2) + (705 * L3) + (-13 * s) + (-4 * u)$

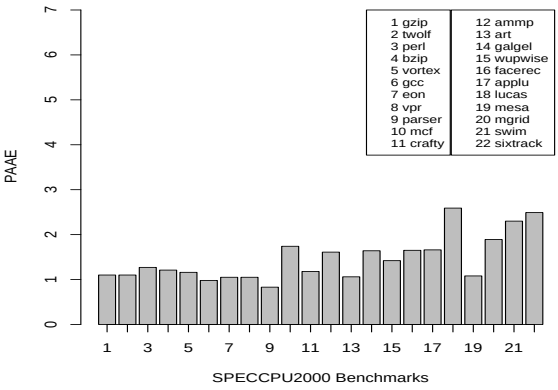

Figure 1: PAAE of the Shanghai 8-predictor model in the CV test on SPECCPU 2000

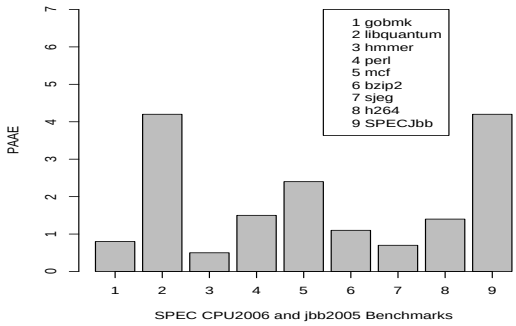

Figure 2: PAAE of the Shanghai 8-predictor model tested on SPECCPU 2006 and SPECjbb2005

$Ph = 65 + (19 * in) + (51 * un) + (-298 * m) + (12.5 * r) + (-298 * L2) + (2089 * L3) + (-18 * s) + (-25 * u)$

Model Assessment: We used the leave-out-one cross-validation (CV) [8] test to assess the prediction quality of the model derived on the SPECCPU 2000 benchmarks. In this test, the function approximator is trained on all the data except for one point and a prediction is made for that point. We also performed model assessment by testing the model against new test data: SPECCPU 2006 and SPECjbb2005 benchmarks. The total system power measured across all the frequencies and the benchmarks varies between 66W and 94W.

The percentage average absolute prediction error (PAAE) [1] was computed for each data point in the test set:

$$PAAE = \frac{\sum_{i=1}^{N} \frac{|M_i - m_i|}{m_i}}{N} \times 100 \quad (1)$$

where $N = 4$ since we collected four data points for the SPECCPU 2000 benchmark, M is the modeled (pre-

dicted) value and m is the measured value.

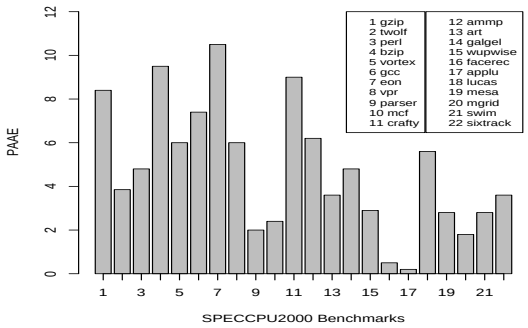

Figure 3: PAAE of the Phenom 8-predictor model in the CV test on SPECCPU 2000

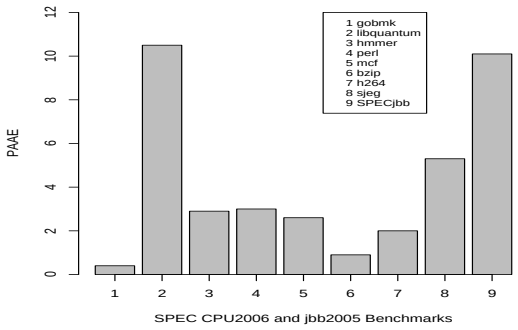

Figure 4: PAAE of the Phenom 8-predictor model tested on SPEC CPU2006 and SPEC jbb2005

As shown in Figures 1 and 2, the Shanghai 8-predictor power model has a better than 96% accuracy for most of the benchmarks in the CV test on SPECCPU 2000 as well as on the new test data. As shown in Figures 3 and 4 shows that the Phenom 8-predictor model has 95% prediction accuracy on average for most of the benchmarks.

Model Selection: Finding the Best Predictors In this section we describe the steps that we followed to select the best model from the 2^8 different models that could be formed using a subset of the 8 initially chosen predictors. The `RIM relaimpo()` [11] method was used to rank the predictors in terms of their effectiveness by computing the R-square value for each predictor (a higher value implies a greater effectiveness). As shown in Figure 5, the `RIM` method shows (in terms of R-squared) that *L3* and *unhalted* are the most important predictors. We also used all-subsets-regression [13] method to find the best model among the 2^8 models, and it showed that the model with unhalted cycles and L3-cache misses (last-level cache misses) is the best model.

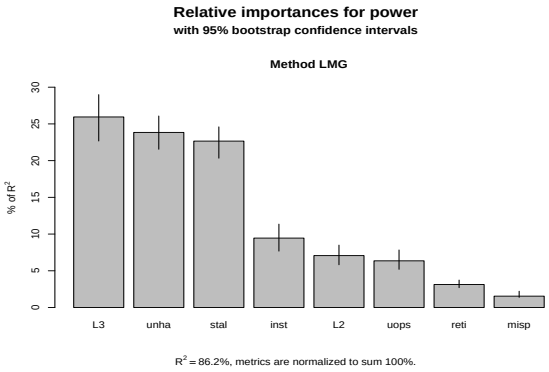

Figure 5: Ranking of predictors for the Phenom power model using the `RIM (lmg)` method

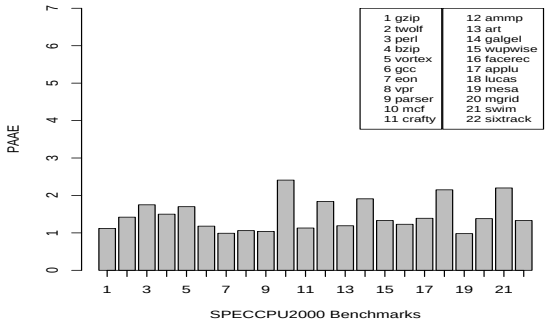

Figure 6: PAAE of the Shanghai 2-predictor model in the CV test on SPECCPU 2000

This makes sense from a computer architecture point of view. Power consumption depends on the CPU clock frequency and on the degree of utilization of the CPU pipeline exhibited by a particular workload. The number of the CPU unhalted cycles correlates with the CPU frequency (the higher the frequency the more unhalted cycles occur per unit of time), while the L3 cache misses correlate with the pipeline utilization (a memory access occurs for each last level cache miss, which idles the CPU pipeline). Therefore, these two events are good heuristics for estimating the power consumption of the CPU/memory subsystem.

Next, we developed models with these two predictors, shown in equations (2) and (3) for the Shanghai and the Phenom platforms:

$$P_{Shanghai} = 56 + (31 * un) + (677 * L3). \quad (2)$$

$$P_{Phenom} = 65 + (33 * un) + (1667 * L3) \quad (3)$$

Figures 6 and 7 show that the Shanghai 2-predictor model provides a greater than 96% prediction accuracy in the CV and the test-data tests for most of the benchmarks, and it also outperforms the 8-predictor model for

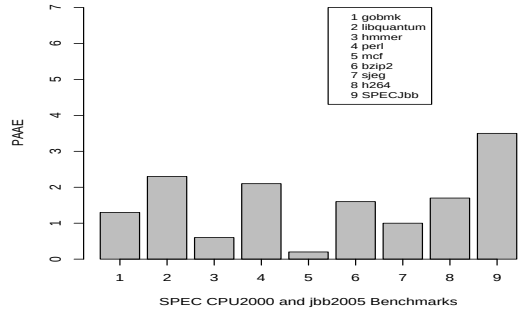

Figure 7: PAAE of the Shanghai 2-predictor model tested on SPECCPU 2006 and SPECjbb2005

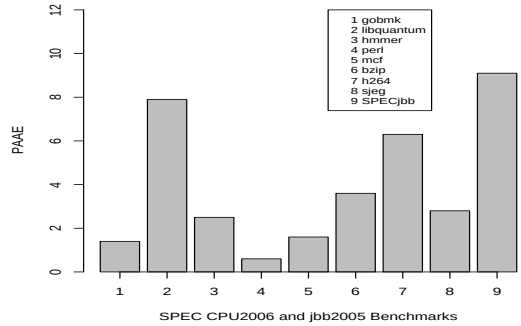

Figure 8: PAAE of the Phenom 2-predictor model tested on SPEC CPU*2006 and SPEC jbb2005*

some of the selected benchmarks. Figure 8 shows that the Phenom 2-predictor model also provides a greater than 95% prediction accuracy on the test data for most of the benchmarks. Its accuracy in the CV tests is similar to that of the 8-predictor model.

The benefit of using the two predictors discovered using the `RIM relaimpo()` method can also be seen using the multicollinearity tests. We used the "Variance Inflation Factor" (VIF) method [9,10] to observe the correlation strength among the predictors. If $\texttt{VIF} > 5$, then the variables are highly correlated. `VIF` test on the 8-predictor model shows that $\texttt{VIF} > 30$ for most of the predictors. However it showed that there is no multi-collinearity problem in the 2-predictor models as $\texttt{VIF} = 1$. From these `VIF` tests, we conclude that the 2-predictor model is likely to be more robust than the 8-predictor model (does not overfit the training data and has a better expected prediction accuracy on the new test data)

Figures 9, 10, 11 and 12 show the visual summary of how the 2-predictor and the 8-predictor models perform on different benchmarks and show that the 2-predictor model actually outperforms the 8-predictor model on

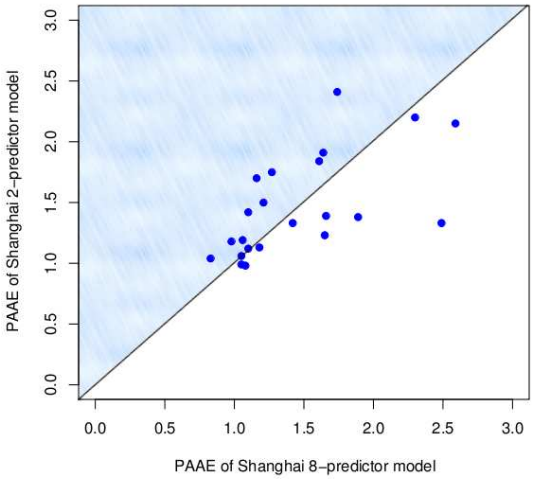

Figure 9: Twenty two benchmarks (of SPECCPU 2000 in CV test), plotted as points with the PAAE of Shanghai 8-predictor model as the x-axis and the PAAE of the Shanghai 2-predictor model as the y-axis.

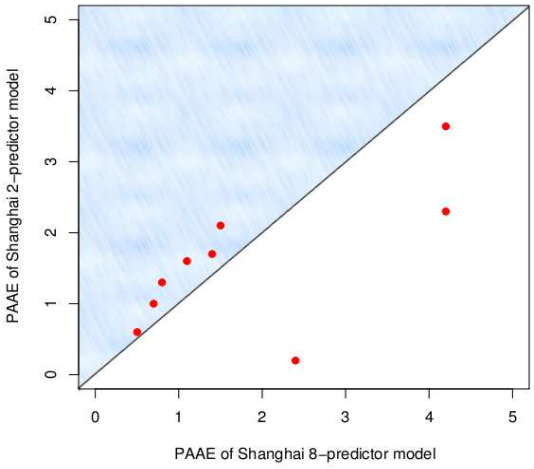

Figure 10: Nine benchmarks (SPECCPU 2006 and SPECjbb2005 as new test data), plotted as points with the PAAE of Shanghai 8-predictor model as the x-axis and the PAAE of the Shanghai 2-predictor model as the y-axis.

some of the benchmarks. The two regions (shown as shaded and non-shaded in the figures) represent the 2- and the 8-predictor models. The x-coordinate of each dot represents PAAE of the 8-predictor model and the y-coordinate represents PAAE of the 2-predictor model when fitted to the same benchmark. If the dots mostly lie

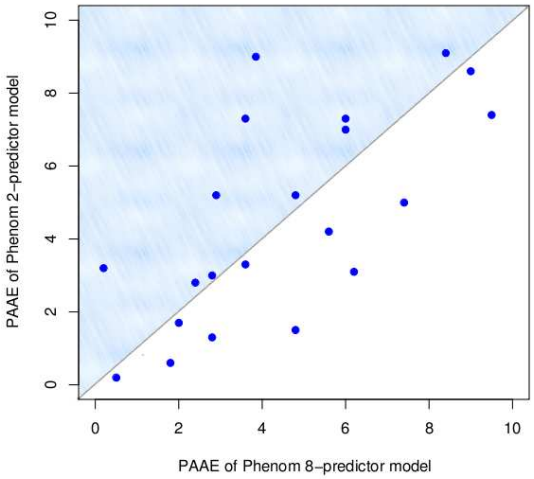

Figure 11: Twenty two benchmarks (of SPECCPU 2000 in CV test), plotted as points with the PAAE of Phenom 8-predictor model as the x-axis and the PAAE of the Phenom 2-predictor model as the y-axis.

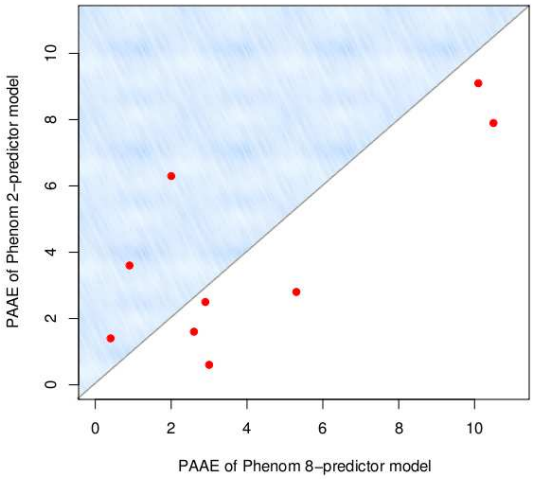

Figure 12: Nine benchmarks (SPECCPU 2006 and SPECjbb2005 as new test data), plotted as points with the PAAE of Phenom 8-predictor model as the x-axis and the PAAE of the Phenom 2-predictor model as the y-axis.

in the 2-predictor model region, then its PAAE is larger than that of the 8-predictor model on this set of benchmarks, and vice versa.

For example, Figures 9 and 11 show that in the CV test, the dots are evenly distributed in both the regions, which means that the 2-predictor model performs approximately the same as the 8-predictor one for both Shanghai and Phenom. Figure 10 shows that the Shanghai 8-predictor model performs better than the 2-predictor model for 6 benchmarks out of 9 on the new test data. On the other hand, Figure 12 shows that the Phenom 2-predictor model performs better than the 8-predictor model for 6 benchmarks out of 9. This demonstrates that the 2-predictor models are as good as the 8-predictor models, portable, and likely to be more robust.

The figures also show that the power models built for the Shanghai are more accurate than those built for the Phenom, which can be explained by the fact that Shanghai has a larger last-level cache than Phenom. As a result, benchmarks running on Shangai will generally experience fewer cache misses, and hence the variation in their power consumption will be smaller, which allows statistical models to fit the power data more closely for Shanghai than for Phenom.

4 Conclusions and Future Research

We presented a methodology for developing simple (with a smaller number of most significant predictors) and robust power models across different CPU frequencies and different benchmarks using applicable model selection and model assessment techniques. This methodology was tested on the Shanghai (server) and Phenom (desktop) AMD platforms, and the models estimated power consumption with 95% accuracy for most of the selected benchmarks.

By deriving a good model with just 2 predictors (unhalted cycles and L3-cache miss events), our methodology showed another strength in terms of portability, as these two events are available on all modern x86 hardware platforms. It is obvious that instrumentation with 8 performance monitoring events is costlier than instrumentation with 2 performance monitoring events. Moreover, since only four performance monitoring registers are available on the AMD Phenon and Shanghai machines, we cannot monitor more than four events at a time on these machines. Therefore, it is clear that building power models with four or less predictors is simpler and more accurate than building models with more than four predictors on this kind of machines.

One important limitation of the models we developed using the two predictors mentioned above is that they are expected to work well only for the workloads that stress CPU and main memory. However, other events that reflect the usage of disk, I/O, chipset, etc. can be added to the original set of predictors. After that, the model selection technique described in Section 3 can be directly applied to find the best model for predicting the power consumption of any particular component or the total system's power consumption.

In the future, we plan to develop power models for

workloads that also use I/O devices (disk and network) and test the portability of this methodology to other platforms (Intel, etc). The Shanghai and Phenom models give large prediction errors (around 9%) for a few particular benchmarks. We will investigate and address this issue in our future work.

Acknowledgements

We thank Lodewijk Bonebakker, Michael Pogue, Keith Hargrove, Yannick Nghiemxuan, Phillip Yelland, Renee Stratulate, Eric Saxe, and Jonathan Chew for their valuable help throughout this work.

References

[1] W. L .Bircher and Lizy K. John, *Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events*, 2007 IEEE International Symposium on Performance Analysis of Systems Software, pp. 158-168.

[2] Gilberto Contreras and Margaret Martonosi, *Power prediction for Intel XScale processors using performance monitoring unit events*. In Proceedings of the ISLPED 2005, San Diego, CA, USA 2005.

[3] F. Bellosa and A. Weissel and M. Waitz and S. Kellne, *Event driven energy accounting for dynamic thermal management*. In Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP03), 2003.

[4] Jan Stoess and Christian Lang and Frank Bellosa, *Energy management for hypervisor-based virtual machines*. In Proceedings of the USENIX Annual Technical Conference, Berkeley, CA, USA, 2007.

[5] Kyeong-Jae Lee and Kevin Skadron, *Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors*. In Proceedings of the (IPDPS'05) - Workshop 11, 2005.

[6] David C. Snowdon and Stefan M. Petters and Gernot Heiser, *Accurate On-line Prediction of Processor and Memory under voltage scaling*. In proceedings of EMSOFT'07, 2007, Salzburg, Austria.

[7] Canturk Isci and Margaret Martonosi, *Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques*. HPCA-12, Princeton University, February, 2006.

[8] T. Hastie and R. Tibshirani and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, 2003, pp. 41-75 and 193-222.

[9] George C. J. Fernandez, *Detection of Model Specification, Outlier, and Multicollinearity in Multiple Linear Regression Models Using Partial Regression/Residual Plots*, SAS Conference Proceedings 22, March 16-19, 1997, San Diego, California.

[10] *Multicollinearity*, `http://en.wikipedia.org/wiki/Multicollinearity`

[11] *Bootstrap Relative Importance Measures*, `http://cran.rproject.org/web/packages/relaimpo/index.html`

[12] *R `lm()` method*, `http://rss.acs.unt.edu/Rdoc/library/stats/html/lm.html`

[13] *All subsets regression*, `http://hosho.ees.hokudai.ac.jp/~kubo/Rdoc/library/leaps/html/leaps.html`

[14] *CPU Power Management in OpenSolaris™*, `http://opensolaris.org/os/project/tesla/Work/CPUPM/`

[15] *AMD performance monitoring events* `http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/31116.pdf`, pp. 339-363.

[16] *Performance Analysis and Monitoring Using Hardware Counters* `http://developers.sun.com/solaris/articles/hardware_counters.html`

[17] *SPECCPU 2000, SPECCPU 2006, SPECjbb2005* `http://www.spec.org/`