H-0191 September 2, 2003
Engineering and Technology

IBM Research Report

Event-Based Power Estimation

Yaakov Yaari
IBM Research Division
Haifa Research Laboratory
Haifa 31905, Israel

Event-based Power Estimation

Yaakov Yaari

April 11, 2003

IBM Research Lab in Haifa, Israel

yaari@il.ibm.com

Abstract

The increasing awareness of power consumption in computer systems focuses attention on the subject of measuring the actual power consumed at any given time. While performance measurement and monitoring is directly supported at the architecture level, and implemented in various software tools, power measurement and monitoring is lagging behind. A software-only power estimation capability could be useful for power management policies, power-aware code optimizations, and other power-related administrative tasks.

The system reported here uses a calibrated power model to get accurate power estimations, without any external devices, including the breakdown of power usage into major power-consuming blocks. Calibration is accomplished using multiple linear regression over a predefined dataset.

The power model allows the breakdown of the estimated power to the consumers associated with each event, thereby providing important insight regarding where power is spent in the application.

The estimation error of this model for a subset of the SPEC CPU2000 benchmark (the full integer suite, plus three of the FP suite) is less then 0.25% for overall energy consumption, and less then 2% for online power consumption.

1. Introduction

The quest for higher performance resulted in increased architecture support for performance measurement and monitoring. This led, in turn, to a rich set of software tools that makes use of these architecture features to optimize and tune performance at the system and application levels. On the other hand, there is very little software-visible architecture support to allow a comparable system-level measurement and monitoring of power in power-aware servers. This is, in part, because power-aware systems were traditionally associated with mobile personal systems, with predefined power management policies.

New high-performance servers use a dense packing of processors to achieve higher throughput. The density of so many processing nodes makes energy consumption a principal concern. Attacking the power challenge can be done at various architecture levels—CPU architecture, system design, OS design, etc. When the focus is on a working system, a running application, or a web service (e.g., servicing a URL request), we may need an accurate estimate of the actual power consumption as the system executes its task. Such a capability could be useful for various power optimization tasks, such as code optimization, and adaptive power management policies. In these cases, one needs an accurate estimate of the actual power consumption at various level of program granularity, in order to select the region to optimize, or to tune the optimization technique.

The motivation for this work is, therefore, the need for continuous reading of the consumed power in power-aware servers, together with the lack of power measurement and monitoring capabilities in the system design.

The events triggered by performance monitoring counters (PMCs), found on most modern architectures, offer an attractive alternative for power estimation . They are always available (as part of the CPU architecture), they provide continuous quantitative estimation of power-related activity, and they impose very minimal load on the system resources.

The system described in this paper proposes this kind of an event-based power estimation methodology. The methodology uses reliable statistical methods that analyze a predefined representative training dataset to determine a set of observation data points, from which the model is to be determined. Using the statistical technique of multiple linear regression [5], these data points are analyzed, and the principal power coefficients are determined with high precision (i.e., minimal RMS error). A side benefit of the resulting linear power estimation model is the insight it provides into the power decomposition of the various contributors (CPU core, multiplier, L1 cache, etc.). Compared to other event-based power estimation methods that have been proposed in the past ([1], [2], [3]), our main contribution is in the methodological, reliable, and repeatable process by which the power model is acquired. The result is a more accurate and practical power model.

The paper is organized as follows: The following section discusses the details and the rationale of the mathematical model used for power estimation. Section 3 describes the system used to collect the energy and performance data. The techniques used to calibrate the power model from this data are discussed in section 4. The system was tested on a subset of the SPEC CPU2000 benchmark [6]. Evaluation and results are given in section 5. The last section summarizes the conclusions and discusses future directions.

2. The Power Model

The power model we use assumes that the measured system consists of a number of functional blocks, each of which makes its own contribution to the total power consumption (register file, multiplier, data cache, etc.). When these blocks are activated they consume a fixed amount of energy. In order to determine the energy consumed in a given time period *t*, E_t, we need to accumulate the contribution of these blocks:

$$E_t = K_0 + \sum_{i=1}^{n} K_i X_{t,i} \qquad (1)$$

Here K_0 is the idle-state energy consumption, $X_{t,\mathrm{i}}$ is the number of activations of block *i* in time period *t*, and K_i is the energy-per-activation, or power coefficient for that block.

The availability of the PMC in modern architectures (e.g., in PowerPC, Alpha, and all IA32 models and their clones) provides the necessary activity information. The idea is then to identify the PMC that corresponds to the principal energy-consuming components, and calibrate their reading using actual energy measurements.

Similar approaches can be found in Bellosa[1], Joseph[2], and Shafi[3]. Joseph et al. use deep knowledge of the underlying CPU architecture to investigate the relationship between energy consumption of a given component and the PMC that measure its activity. However, they do not show a combined power model for the system. Shafi et al. use the model as part of a cycle-accurate system power simulator. The principal difference between our method and that of Bellosa (who like us uses the model for online power estimation) is the selection of PMC and the calculation of the K_i power coefficients. Bellosa selects the PMC a-priori and uses a number of micro-benchmarks to set up a set of linear equations, where the benchmarks are specifically-designed to determine the coefficients. Our method, as opposed to those of the previous works, is based on training, uses representative benchmarks, and employs reliable statistical methods to obtain a more stable power model.

The methodology can be applied to any system using a calibrate-once, use-often mode. That is, a system should be calibrated only if a power-related parameter has changed (different memory (or L2 cache) type or size, different processor speed, etc.). Once that is done, the model that results from the calibration is good for each identically-configured board.

3. Collecting Energy and Performance Data

The measurement system consists of three components: a system-under-test, an energy measurement subsystem, and a data collection subsystem (see Figure 1):

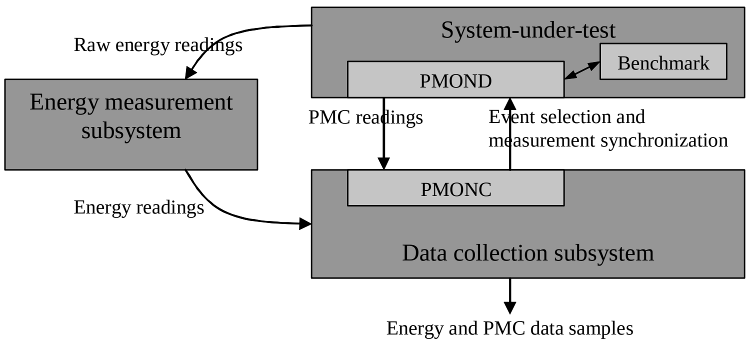

Raw energy readings
System-under-test
Benchmark
PMOND
Energy measurement subsystem
PMC readings
Event selection and measurement synchronization
PMONC
Energy readings
Data collection subsystem
Energy and PMC data samples

Figure 1: Energy and performance measurement system

1. **The system-under-test –** The system runs the PMC data collection daemon, PMOND, which is responsible for configuring the CPU for the requested PMC, collecting and transmitting the PMC data, and synchronizing the benchmark-under-test.
2. **The energy measurement subsystem. –** The subsystem samples energy readings by measuring the voltage drop over calibrated sense resistors inserted into the different supply lines of the system-under-test. The data is pre-processed to give accumulating energy readings, which are transmitted to the data collection subsystem.

3. **The data collection subsystem**. – The subsystem can collect data from a number of sources (e.g., system utilization, network utilization, etc.). For our purposes, the subsystem is configured to collect the energy readings, as well as performance monitoring data from the system-under-test. That last function is performed by PMONC, a module that functions as a client to PMOND.

The data collection subsystem produces a collection of energy and performance readings per second. The specific data items (the type of PMC events and energy components to measure) are configurable. The sample vector can be specified as:

$\{t_i, E_{i,1}, E_{i,2}, ..., E_{il}, X_{i,1}, X_{i,2}, ..., X_{i,n}\}, ...$

Here $\{E_{i,1}, E_{i,2}, ..., E_{i,l}\}$ are the energy components, and $\{X_{i,1}, X_{i,2}, ..., X_{i,n}\}$ are the PMC data for time t_i.

4. Calibrating the Power Model

The PMC, which serves as the basis for the power model, was traditionally designed to analyze a processor's performance and not its power consumption. Thus, there is no a-priori set of PMCs that are best suited for the purpose of power estimation. As a result, there are many possible power models, depending on the selection of the PMCs, and on the subset of observation data points analyzed to create the power model. The selection process is discussed at length in the following sections.

A power model, as described in section 2, is specified by a selection of the PMCs and the set of its energy coefficients. To determine the coefficients $\{K_i\}$ for some energy reading E_j (1), we need to solve the set of linear equations:

$$\begin{bmatrix} 1, X_{1,1}, X_{1,2}, ..., X_{1,n} \\ 1, X_{2,1}, X_{2,2}, ..., X_{2,n} \\ ... \\ 1, X_{m,1}, X_{m,2}, ..., X_{m,n} \end{bmatrix} \begin{bmatrix} K_0 \\ K_1 \\ ... \\ K_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ ... \\ E_m \end{bmatrix} \qquad (2)$$

Here *m* is the number of data samples (equal to number of time intervals in the experiment), and *n* is the number of PMC events read at each time. Because of the indeterminate nature of the problem, we need $m>>n$ to get sufficiently stable coefficients.

In general, each sample could contain more than just one energy component, as shown above. In such cases, Equation 2 should be extended to multiple sets, one for each component, where the left-hand side matrix is the same and the right-hand vector for the *l'th* component contains $\{E_{jl}\}$, with $1 \le j \le m$. For simplicity, the following discussion assumes just one energy component.

In order to determine the power coefficients from the data collection output, the following procedures must be performed:

1. Generating observation data points
2. Selecting the training set
3. Performing multiple linear regression

4. Searching for optimal power model

These steps are discussed in the following subsections

4.1. Generating Observation Data Points

For each stable data point *i*, the *n* PMC values $\{X_{i,1,} X_{i,2}, ..., X_{i,n}\}$, along with the corresponding energy value E_i, are combined to form an observation data point. A stable data point at time *t* is one where $\{x_{t,j}\} = \{\{X_{t,j}\}, E_t\}$ is stable around time *t.*

We need the measured PMC values to be stable during *t* for two reasons. The first is a result of limitations of the processor's architecture. In most architectures, the number of concurrent measurable PMCs is very limited (two in Pentium II/III, 4 in PowerPC). In contrast, the power model needs more counters for greater accuracy. Thus we need to sample PMCs at a higher rate then the output data collection rate, each time measuring a different set of PMCs. The second reason is that there is an inherent delay from activation time until it is reflected in the energy measurement system.

In our case, we require that $\frac{|X_{t+1,i} - X_{t,i}|}{X_{t,i}} \le 0.02$ for all $1 \le i \le n$, where *n* is the number of PMC events measured in each time interval. Data points that do not meet this criterion, i.e., that were not stable during *t*, are not passed to the next (MLR analysis) stage.

Energy fluctuations in real benchmarks make determining stable data points a difficult task. For example, only 54 stable data points (i.e., one-second periods) were found in the *gzip* benchmark for the set of PMCs {*inst_retired, l2_lines_in, data_mem_refs*}. With an execution time of about 1200 seconds, the number of stable data points in this benchmark, used to generate the power model, is less than 5%.

4.2. Selecting the Training Set

Only part of the observation data points are used to determined (or train) the power model. This training dataset should not be too small, neither too large. A set that is too small does not provide the regression engine with sufficient information to determine the power model. On the other hand, if the training set is too large, the result is *over-fitting* -- —the model describes very accurately the power behavior of the trained programs, but fails for most other, *'unseen'* programs.

The process of selecting the training dataset *T* is iterative. Initially, *T* is empty. We then determine the power model based on one benchmark from the suite and find the accuracy of the model for the whole suite (see Section 0 below). We do this for each of the benchmarks and finally find out which benchmark bm_{1i} results in the most accurate model. This benchmark, bm_{1i}, is then added to *T.*

The process is repeated with other benchmarks, bm_{2j}, $j \neq i$, such that $T \cup bm_{2j}$ gives the the highest accuracy, then adding bm_{2j} to *T.*

The process stops when *T* is just big enough to give adequate accuracy. Experience shows that 20% of the total stable data observation points are sufficient to get good accuracy and avoid over-fitting.

In addition to the degrees of freedom involved in selecting the training dataset, we also need also to select the set *P* of PMCs, which are used in the model, from the set *Q* of the total measured events. As was already shown in [1], some accuracy could be achieved with just one or two PMCs. Moreover, improper selection of PMCs can deteriorate the model due to non-linear dependencies between them. Selecting the set *P* could be done in a similar fashion to the method shown above. That is, select the most useful PMC, which gives the best one-PMC model, and add it to *P*. Then find the next one, which together *P* gives the best model, and so on.

4.3. Multiple Linear Regression

Performing multiple linear regression (MLR) analysis ([5]), is the main engine of the training stage. Referring to Equation 2, the process finds the set of coefficients $\{K_i\}$ that minimizes the RMS error for the provided data points. To avoid over-fitting, only a specified subset (10%-50%) of the stable data points for the selected PMC were used to feed the regression analysis engine.

The algorithm used is a Perl implementation by Welch [10] which is an adaptation of the original algorithm of Gentleman [11]. Insertion of a new observation can be done one at a time and only takes a low quadratic time. The storage space requirement is also of quadratic order.

4.4. Searching for Optimal Power Model

As discussed above, determining the power model involves some degrees of freedom. This implies choosing between alternative power models, which require metrics to evaluate the accuracy of a given model.

Two metrics are used to evaluate power estimation accuracy for each benchmark:

1. Total error. The difference (in percentages) between the estimated and measured energy consumed during the full run of the program:

$$TotalError = \frac{(E_{total,estimated} - E_{total,measured})}{E_{total,measured}} * 100 \qquad (3)$$

2. Standard error. The standard statistical error (in percentages) between the estimated and measured energy consumed at each measured interval:

$$StdError = \frac{\sqrt{\sum \left(E_{i,estimated} - E_{i,measured}\right)^2}}{\overline{E}_{measured}} * 100 \qquad (4)$$

To evaluate the combined accuracy over the whole suite, we used the metrics *Stddev*(*TotalError*) and *Average*(*StdError*), measured over all benchmarks.

5. Experimental Results

5.1. Measurement System

The system-under-test we used (see Section 3) was a Linux 2.4/Pentium III/866 MHz with a built-in 256K L2 cache, configured with 256Mb RAM. These parameters have, of course, a major effect on the kinds of PMCs that are available for measurement, and on the resulting coefficients of the power model. Note that the available RAM is determined by the Linux kernel configuration file, and is, in general, a subset of the total available physical memory.

For the energy measurement subsystem, we used a Windows/NT running National Instruments' LabView data acquisition system. The data collection system, which supervises the whole data measurement and collection process, was also a Linux 2.4/Pentium III/866 MHz machine.

All three components are inter-connected via TCP/IP networking. More details on the measurement system is described in detail in [4] .

5.2. Measured Events and Energy Factors

The energy components measured on our system are shown in Table 1**Error! Reference source not found.**. The main contributors are, of course, the CPU board and the memory. The contribution of the disk drive was shown to be relatively low, and practically constant across the experiments.

	Supply line	Powers…
$E_{i,1}$	+3.3V	1GB Memory, video card, and Ethernet card
$E_{i,2}$	+5V	Pentium III CPU board
$E_{i,3}$	+12V	Cooling fan
$E_{i,4}$	+5V	Disk drive controller
$E_{i,5}$	+12V	Disk drive motor and head actuator

Table 1: Measured supply lines

In practice, we used E_i (see Equation 2), which represents the sum of all these readings energy readings, i.e., $E_j = \sum_i E_{ji}$.

Eight PMC events were selected, as shown in Table 2. The PMCs were selected for the experiment in order to provide good coverage for the major power-consuming functional blocks: various function blocks inside the CPUcore, the L1 and L2 caches, and the memory banks.

	Description		Description
cpu_clk_unhalted	# clocks when CPU was not halted	*uops_retired*	# retired uops
inst_retired	# retired instructions	*mul*	# integer & FP multiplies
data_mem_refs	# data memory references	*dcu_lines_in*	# Dcache lines allocated
l2_lines_in	# L2 cache lines allocated	*bus_tran_mem*	# memory bus transactions

Table 2: Measured events

For further details on the Pentium PMC, see [9].

5.3. Benchmarks

The measurement and estimation method was evaluated on a benchmark suite consisting of the following:

1. *Combo*. This is a specially-designed synthetic benchmark that exercises the major power consumer functions, such as integer unit, FP unit, data cache, L2 cache, memory, etc.
2. SPEC CINT2000 [7]. This is the standard benchmark for CPU-intensive integer performance, consisting of 12 integer-oriented programs. On the system-under-test, the benchmarks' elapsed time was between 600 to 1200 seconds.
3. *172.mgrid, 179.art, 188.ammp*. Testing the whole FP suite was not in the scope of our experiment, and SPEC FP programs were not involved in the training stage but rather in the evaluation stage. These three programs of the SPEC CFP2000 [8] suite were arbitrarily selected and added to achieve a more representative suite.

5.4. Selecting the Training Set

We tested the estimation accuracy of using each of the first 13 programs as the basis for training, measuring the accuracy of the resulting model on the entire suite. It turns out that *Combo* gave the best results when used as the basis for training,. This is because the synthetic benchmark better exercises the different power consumption factors. We then added two more benchmarks to the training set, each time by selecting the one that provided the highest accuracy. The final training set was {*combo, bzip2, eon*}.

5.5. Results

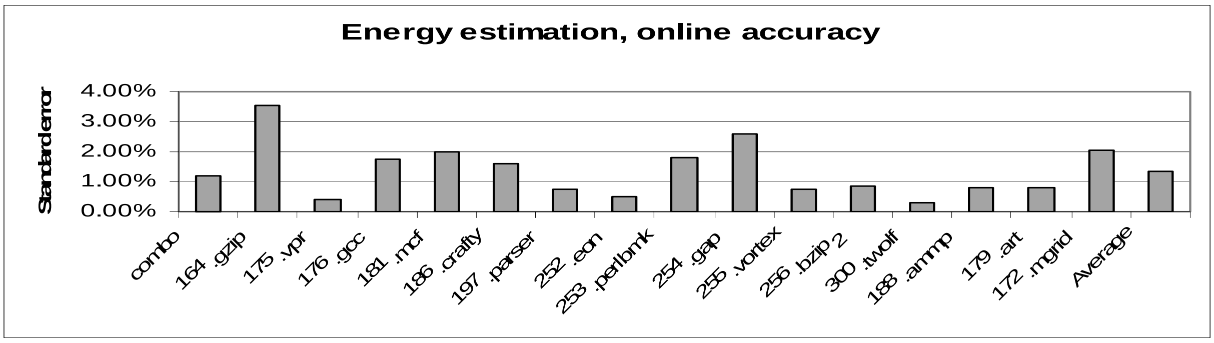
As discussed above, different power models result from different training sets and PMC sets. The training set selected in Section **Error! Reference source not found.**, and the PMCs selected in 5.2, result in the power model shown in Table 3.

	Energy coefficient (μJoules/event)		Energy coefficient (μJoules/event)
cpu_clk_unhalted	0.01305	*uops_retired*	0.00123
inst_retired	0.00349	*mul*	-0.00872
data_mem_refs	0.00702	*dcu_lines_in*	0.10809
l2_lines_in	0.36626	*bus_tran_mem*	0.10045
Idle energy (mWatts)		*46690*	

Table 3: Power model

When examining Table 3, we should remember that the power contribution of the activity related to PMCs is determined at any given time by the product of the PMC coefficient and the PMC value for that interval (see discussion in Section 5.7 below). Thus, the coefficient magnitude by itself has only a limited value.

Note that the energy coefficients can be negative (e.g., for the *mul* event). The rationale for this is that such events are the subset of an event class, and have a lower energy coefficient then the average event in the class. In the case of *mul*, this event is a subset of the *inst_retired* and *uops_retired*.



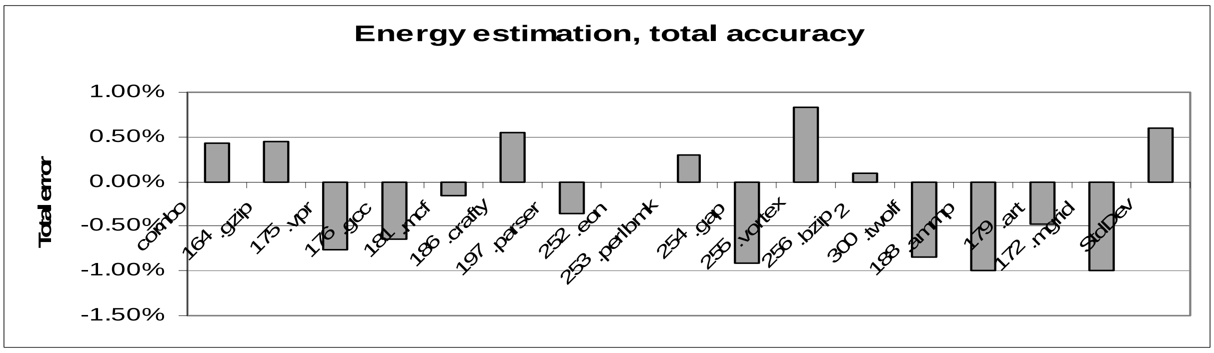


Figure 2: Accuracy of energy estimation model

The accuracy of the energy estimation model, as measured on the 16 benchmarks, is shown in Figure 2.

The top chart shows the online, second-by-second accuracy in terms of standard error, with an average of 1.35% (the rightmost bar). The next rightmost bars show the FP benchmarks. Even though the model was trained on mostly integer code, the accuracy here is not adversely affected.

The bottom chart shows the accumulated error over the full run of the benchmarks, with an average of 0.55% (the rightmost bar). Note the very small accumulated error of benchmarks *eon* (practically zero) and *bzip2*, on which the model was initially trained. In general, the average error is significantly smaller then the second-by-second standard error. Thus, considering 600-1200 measured intervals (seconds) per benchmarks, there is clearly no measurable accumulation of error.

5.6. *Model complexity*

The power model of Table 3 uses eight PMCs. Ideally, accuracy should improve monotonously with the complexity of the model (i.e., the number of PMCs). However, because of the non-linear dependency between the PMCs, and the limited data available, this is not always the case.

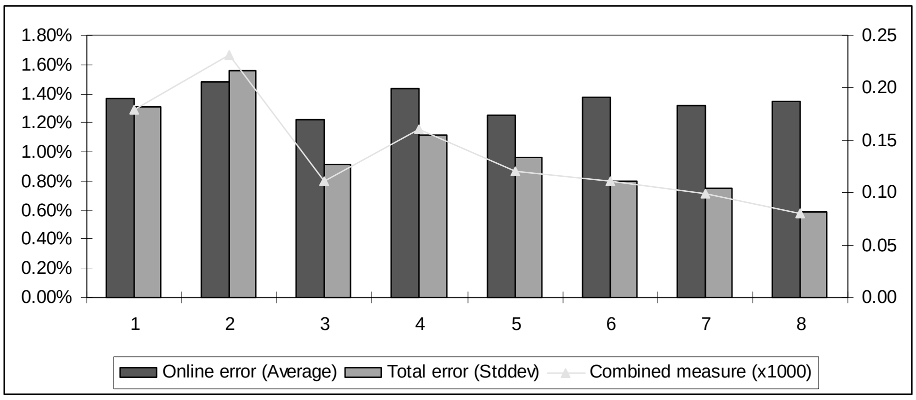
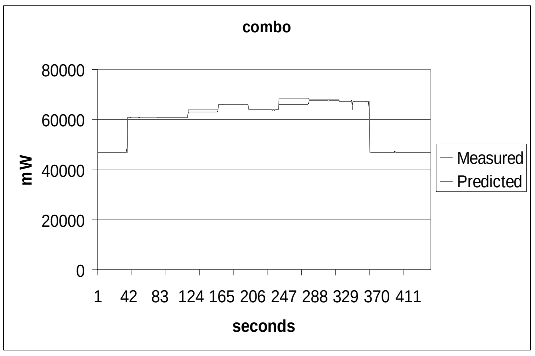


Figure 3: Accuracy vs. model complexity

Figure 3 shows the variation of the two accuracy metrics, as well as the combined metric, as the model complexity increases from 1 to 8 PMCs. We see that the trend is indeed for increasing accuracy. However, depending on the training set, there might be cases where increasing complexity will decrease accuracy, as in the case above when moving from three to four PMCs.

5.7. *Analysis of Power Components*



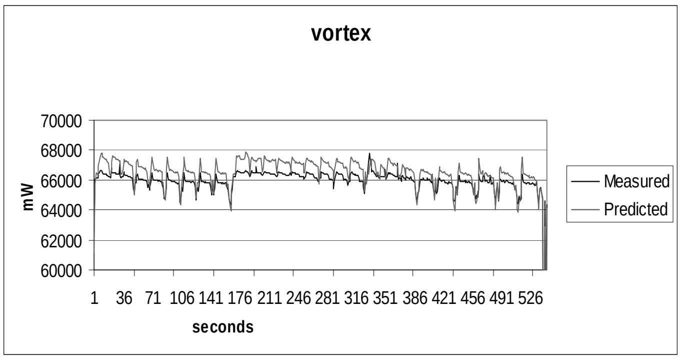
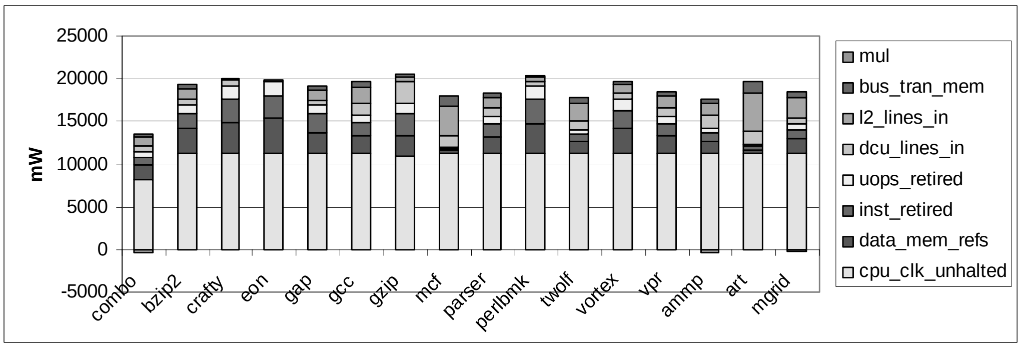


Figure 4: Estimated vs. measured energy

Figure 4 shows the actual energy measurement and estimation over the run of two benchmarks, *combo* (the synthetic benchmark), and *vortex* (a single-user object-oriented database transaction benchmark). The vertical scale of the *vortex* chart (right) was set so that the relatively small variations in energy can be seen more easily. Even after this magnification, we see the predicted power closely following the measured power, in particularly in the last third of the benchmark. Maximum error during the worst case of the benchmark, (its middle third), is about 1.5%.

The exceptional close match between predicted and measured power for the *combo* benchmark (left) is explained, in part, by the fact that this benchmark was one of the three benchmarks used in the training stage. It thus has a larger impact on the resulting model than benchmarks that were not in the training set (like *vortex*, for example).

While the above charts show the total energy consumed on the board, it is interesting to see how the energy consumption is divided among its major consumers, both in the measured and estimated sides.



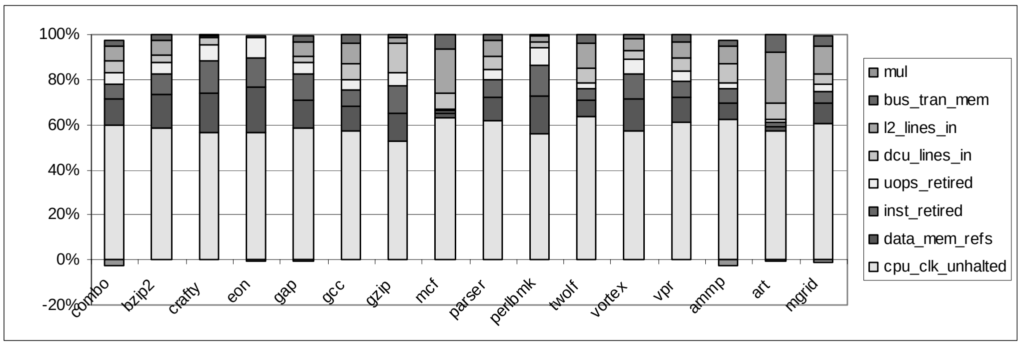


Figure 5: Distribution of estimated power

Figure 5 shows the different contribution of each of the events. The top chart shows average power, while the bottom one shows the relative distribution.

The first observation is the contribution of *cpu_clk_unhalted,* around 60% of the power. The only benchmark that shows a significantly different reading of this event is *combo,* where the program explicitly goes into idle periods. Nevertheless, benchmarks still show significant variability in power consumption—a 16% difference between *gzip* and *amm.* This is very significant because, typically, power utilization of a system is estimated from the relative time the system spends in the idle process,. We see that the accuracy of this simplistic approach for power estimation, therefore, is very limited.

The relative distribution chart shows the different power profiles of the benchmarks. For example, in *mcf* and *art,* significant power is consumed in the memory hierarchy (DCache, L2, and memory). On the other side are benchmarks like *eon* and *crafty,* where power is spent inside the CPU core, with almost none on the memory hierarchy. Similarly, *gzip* spends a relatively much large amount of energy in the data cache.

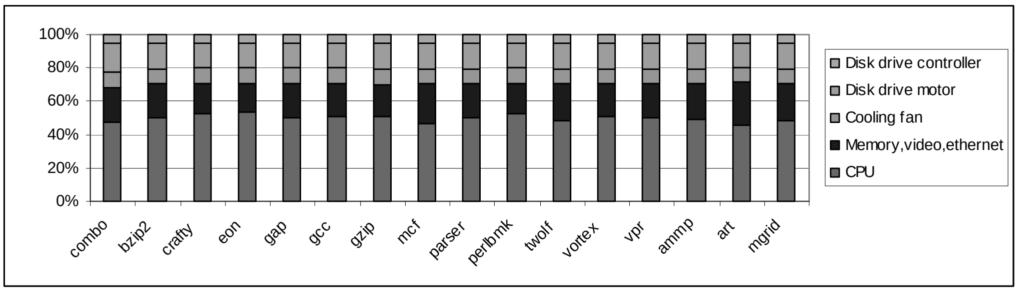


Figure 6: Distribution of measured power

The distribution of the actually measured energy to the different energy components, shown in Figure 6, confirms the above observations. Even though the breakdown here is much coarser, we can still see the higher proportion of the memory subsystem's power consumption in *mcf* and *art*. Similarly, a higher proportion of CPU power consumption is seen in *crafty* and *eon*.

6. Conclusions

Power-aware servers require the availability of continuous accurate power estimation in order to implement advanced power management policies, to allow remote power management and administration, and to enable power-aware code optimizations. Event-based power estimation appears to be an attractive alternative for this, because it is accurate, light-weight (does not burden system resources), and does not require external devices/hardware.

The event-based power estimation system described in this paper exhibits such features. We showed how such a system is calibrated using a predefined training dataset and the statistical method of multiple linear regression. The methodology shown allows the designer to control the resulting accuracy by selecting a proper set of PMCs, which reflect the principal contributors to power consumption in the system.

The resulting power model models represents the underlying system board accurately, better than 98%, on the average, for continuous online power estimation, and better then 99.75%, on the average, for the energy estimates of the full application (measured for 16 benchmarks, including the full SPEC CINT2000).

The linear power decomposition provided by the power model provides a valuable insight into the behavior of different applications. The experiments described in this paper, showed, for example, applications where power consumption of the memory or L2 cache subsystem is much higher then the average. This can lead, in turn, to focusing power-aware code optimizations on the principal contributors.

The overhead for computing the power/energy is negligent — reading a few PMCs (four appears to be sufficient), multiplying each with its coefficient, and summing up. Note that, in practice, the system should be calibrated for different system configurations: processor architecture and speed, and cache and memory configuration.

The basic notion of using PMC events to measure power can be extended in a number of directions: Architecturally, power-meaningful events could be defined in the processor to further improve the estimation accuracy. PMC events can also be defined for other devices which consume power, such as network bridges, and disk drives. With such PMC events, designed to be continuously readable, power estimation can be extended to the whole system, and not just the system board.

If the method is standardized , we can expect the power model to be part of the OS configuration, with an OS service which monitors continuously the system's power consumption. This, in turn, could be used for remote power monitoring, advanced power management schemes, and for power-aware code optimizations.

7. Bibliography

[1] .Bellosa, F. "The case for event-driven energy accounting." Technical Report TR-I4-01-07, University of Erlangen, Department of Computer Science, June 2001.

[2] Joseph, R., D. Brooks, and M. Martonosi. "Live, Runtime Power Measurements as a Foundation for Evaluating Power/Performance Tradeoffs." Workshop on Complexity Effective Design (WCED, held in conjunction with ISCA-28).

[3] Shafi, H., P. Bohrer, J. Phelan, and C. Rusu. "Event-Based System Power Simulation." Austin Conference for Energy-Efficient Design (ACEED) 2002.

[4] Bohrer, P., E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. "The case for power management in web servers," in *Power Aware Computing*, Kluwer Academic Publications, 2002.

[5] Milton J.S. and Jess C. Arnold. *Introduction to Probability and Statistics.* McGraw-Hill, 1995.

[6] SPEC CPU2000 benchmark, http://www.specbench.org/osg/cpu2000/

[7] SPEC CPU2000, integer benchmark, http://www.specbench.org/osg/cpu2000/CINT2000/

[8] SPEC CPU2000, FP benchmark, http://www.specbench.org/osg/cpu2000/CFP2000/

[9] IA-32Intel Architecture Software Developer's Manual, Vol. III: System Programming Guide. *]*

[10] Welch, I. *Regressiion.pm*, CPAN archive, http://search.cpan.org/src/IAWELCH/StatisticsRegression/

[11] Gentleman, W.M. "Basic Description For Large, Sparse Or Weighted Linear Least Squares Problems (Algorithm AS 75),." *Applied Statistics* 23, no.3 *(*1974). Vol 23; No. 3