Predicting Power Consumption of High-Memory-Bandwidth Workloads

Norbert Schmitt
University of Würzburg
norbert.schmitt@uni-wuerzburg.de

Jóakim von Kistowski
University of Würzburg
joakim.kistowski@uni-wuerzburg.de

Samuel Kounev
University of Würzburg
samuel.kounev@uni-wuerzburg.de

ABSTRACT

High performance workloads with high bandwidth memory utilization are among the most power consuming software applications [15]. When writing such applications, developers can directly influence power consumption of the final software through their choice of data size and traversal method, mostly due to caching characteristics. Explicit knowledge on how choices influence power consumption can thus lead to greater overall energy efficiency. In existing work, power prediction for memory accesses and high bandwidth applications requires either detailed measurement information on the system on which the software is executed or it is too generic, not taking significant aspects, such as caching and data size into account. In this paper, we propose a power model that bridges this gap by modeling power consumption based on concrete software properties, while considering hardware characteristics on a more abstract level, characterizing it primarily using publicly available data. The model is designed to enable developers to compare power consumption of implementation alternatives for high memory bandwidth software components. We validate our model by measuring modified versions of the high bandwidth benchmark stream [11]. We show that our model can predict the relative change of power consumption due to implementation changes and the power consumption of a concrete system under test with an average error of 19 percent.

Keywords

Cache, SPEC, Performance Counter, Workloads, Energy Efficiency, CPU, Load level, Utilization

1. INTRODUCTION

Energy efficiency of computing systems has become a significant issue over the past decades. In 2010, the U.S. Environmental Protection Agency (U.S. EPA) estimated that 3% of the entire energy consumption in the U.S. is caused by data center power draw [8]. High performance workloads with high bandwidth memory utilization are among the most power consuming software applications. Especially when comparing high performance CPU workloads, workloads with additional high-bandwidth memory usage draw more power than their less memory intensive counterparts [15].

Software developers can directly influence the power consumption of high-bandwidth software components through their implementation and design decisions. Especially, characteristics such as data type and data traversal methods can have a significant impact both on performance, as well as power consumption. Explicit knowledge on the relationship between power consumption and implementation details can help developers to make conscientious choices regarding energy efficiency during development and may lead to more efficient software.

Existing power models are either too generic or too specific for use in such a context. Simple full-system power models, such as [4, 13] do not consider memory access sufficiently. In stark contrast, models intended for hardware design [12, 10, 9] are too specific, requiring detailed information about target hardware. Similarly, some more general purpose models which consider memory accesses, such as [5] also require very specific hardware information. More generic models that consider memory accesses are mostly intended to be used for system management and thus still require some concrete hardware descriptions. [2] features a processor power model, whereas [3] and [14] focus on management of virtual machines.

In this paper, we propose a power model that bridges this gap of models being either too generic or too specific by modeling power consumption based on concrete software properties. Specifically, model parameters that characterize hardware may be set to default values, in which case the model's power prediction is proportional to actually consumed power on a concrete system. This way, the model enables developers to compare power consumption of implementation alternatives for high memory bandwidth software components. The model may also be used with system-specific hardware parameters, in which case model results absolute accuracy increases.

The major contributions of this paper are as follows:

1. We classify the differences in power consumption that can be achieved by high-memory-bandwidth workloads.
2. We propose a model that allows for comparison of implementation options, based on data type and data traversal method.

We evaluate our model using measurements of nine modified versions of the stream benchmark [11]. We predict power consumption for memory operations using different

ICPE'17, April 22 - 26, 2017, L'Aquila, Italy

ISBN 978-1-4503-4404-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3030207.3030241

data types and strides. We define stride as the step size for each iteration over a contiguous block of memory. We show that our model can accurately predict the relative change in power consumption between the different versions without calibration to the specific hardware. We also show that the model can predict CPU and Memory power consumption with a mean error of 32 and 54 percent, if calibrated to a specific system.

The remainder of this paper is structured as follows: We describe our model and its parameters in Section 2, followed by an evaluation of both relative and absolute prediction accuracy in Section 3. Finally, we conclude the paper in Section 4.

2. POWER PREDICTION MODEL

The goal of the power prediction model is modeling power consumption of *(1)* CPU and *(2)* full-system power. We are also able to estimate memory power. For each of these predictions, we predict either relative changes in power depending on workload characteristics or absolute power, which requires additional information on the hardware.

CPU power ($pwr(CPU)$) and full-system power ($pwr(sys)$) are the results of our workload model, whereas memory power ($pwr(mem)$) can be derived from the former two. To derive memory power, we assume that the high-memory-bandwidth workload exercises a negligible amount of I/O. If so, pwr_{mem} can be derived as follows (Eq. 1):

$$pwr(mem) = pwr(sys) - pwr(CPU) - pwr(idle) \quad (1)$$

pwr_{idle} is a constant that can be read from manufacturer sheets or standard benchmark results, such as SPECpower-ssj2008 [7] or the SPEC SERT [8].

The power models for full-system and CPU power are based on the hypothesis that larger data types and bigger steps in data traversal cause lower CPU power consumption. This behavior is expected due to the way in which execution and power consumption depend on the CPU cache. The general rationale is as follows: Every memory access instruction reads data either from cache (cache hit) or directly from memory (cache miss), each direct memory access (caused by cache miss) causes the CPU to wait, reducing its power consumption, while at the same time, causing memory to perform work, increasing its power consumption. Consequently, the CPU power consumption decreases with the cache miss probability up to the minimum power consumption $pwr_{min}(CPU)$ and $pwr_{min}(sys)$. The memory power consumption increases with the cache miss probability. However, as memory consumes far less power than CPU overall full-system power consumption scales with the CPU power consumption factor p (Eq. 4) to $pwr_{max}(CPU)$ as well as $pwr_{max}(sys)$, as shown in Eq. 2 and Eq. 3.

$$\begin{aligned} pwr(CPU) = {} & pwr_{min}(CPU) \\ & + (pwr_{max}(CPU) - pwr_{min}(CPU)) * p \end{aligned} \quad (2)$$

$$\begin{aligned} pwr(sys) = {} & pwr_{min}(sys) \\ & + (pwr_{max}(sys) - pwr_{min}(sys)) * p \end{aligned} \quad (3)$$

CPU and full-system power values are either system specific or default values. Full-system power values can be read from standard benchmark results, such results from the SPEC SERT [8]. However, the cache miss probability also depends on the workload. We calculate cache hit probability (Eq. 4) based on the data type size, stride, and cache size of the last level cache (LLC). Cache size can also be set using a default value. However, cache sizes of existing processors are public information and available from the manufacturer's data sheets.

$$p = \frac{datasize * stride}{cachesize} * inverseprefetcheraccuracy \quad (4)$$

The $inverseprefetcheraccuracy$ in Eq. 4 is the only hardware parameter that can not be pulled from a manufacturer data sheet or standard benchmark result. Fortunately, it does not affect the ranking of relative results, which means that it can be omitted for relative comparisons. It models the hardware cache prefetcher's ability (or rather inverse ability, as we are modeling cache misses) to pre-fetch data for the cache that would normally not be in the cache according to naive caching algorithms. To keep the model simple, the cache miss probability is the only modeled hardware parameter. This enables the model to compare relative results if neglected as described, and predict the power consumption with reasonable accuracy.

Note that the model's absolute prediction accuracy is expected to strongly depend on the accuracy of the pwr_{max} and pwr_{min} parameters. If these do not correlate for a full-system and a corresponding CPU power model it may be prudent to use different cache miss probabilities for these models, even though this is semantically unintuitive.

The data types are selected which we see as the most commonly used data types in C programming. We do not distinguish between floating point and integer types. Therefore longer strides for integers are not evaluated.

3. EVALUATION

We evaluate the relative and absolute prediction accuracy of our model. To this end, we modify the stream benchmark [11] to iterate over 16 arrays in parallel. We compare the accuracy of comparative predictions (relative accuracy) using default hardware parameters and absolute prediction accuracy using hardware parameters for a specific system.

3.1 Workloads and Measurement

We use modified versions of the stream benchmark [11] to evaluate our model. Stream is a high-memory-bandwidth intensive benchmark that performs sequential memory accesses for scalar multiplication and copy operations on three large arrays. We modify stream to be used according to the SPEC power measurement methodology [1]. To this end, we execute stream in parallel using 16 system processes, each with separate arrays. The array iterations are repeated with the number of iterations per second serving as a throughput metric. Iterations are repeated for the entire measurement duration. In adherence to the SPEC methodology [1], we run the workload for 15 seconds before beginning the measurement phase, which lasts for 120 seconds, collecting throughput, system power (using an external power meter) and CPU performance counters each second. CPU performance counters are monitored using the IntelPCM tool [6]. We measure CPU Power, Memory Power, and memory bytes read and written.

We modify the workload by changing the data type of the stream array (`char`, `int`, `double`, and `long`) and the stride of the array iteration (1, 2, 4, 8, 16, and 24).

To determine the *inverseprefetcheraccuracy*, we select the

	pwr_{min}	pwr_{max}	$inversepre-fetcheraccuracy$
Full system power	$104.7W$	$140.6W$	151,518
CPU power	$63.81W$	$89.37W$	189,552

Table 1: Benchmark results for the machine specific configuration of power consumption

Data Type	Stride	Power Measured	Power Estimated	Ordering
`char`	1	$120.8W$	$140.31W$	8
`int`	1	$109.9W$	$139.43W$	32
`long`	1	$108.7W$	$138.26W$	64
`double`	1	$109.6W$	$138.26W$	64
`double`	2	$110.0W$	$135.26W$	128
`double`	4	$110.4W$	$131.26W$	256
`double`	8	$109.6W$	$121.92W$	512
`double`	16	$107.4W$	$103.23W$	1024
`double`	24	$105.8W$	$84.55W$	1536

Table 2: Measured full system power compared to configured model and predicted ordering

stream workload using its standard data type `double` together with a stride of 16. We select a stride of 16 to accommodate for normal software with a mixture of data with good spatial memory locality, benefiting from the hardware prefetcher, and random access to main memory. The `double` data type with stride 16 is used as a training value and is therefore excluded from the evaluation. The machine onto which the model is trained is a HP DL160 Gen9 with a Xeon E5-2640 v3 (2.6Ghz) processor with 20MiB L3 cache (LLC) and 32GiB (2133Mhz) memory. The machine specific *inverseprefetcheraccuracy* as the mean of both training values, full system power and CPU power, is 170,535. *inverseprefetcheraccuracy* for the trained model is obtained by measuring the L3 cache miss rate with IntelPCM.

As $pwr_{min}(CPU)$ we use the LU benchmark from the SERT and SOR for $pwr_{max}(CPU)$, also from the SERT. They are designed to specifically stress the CPU with a negligible amount of memory access of 0.18KiB/s (LU) and 0.24KiB/s (SOR). The low memory access makes both benchmarks well suited to determine the maximum and minimum CPU power. All benchmarks are executed on the same machine used for the *inverseprefetcheraccuracy* shown in Table 1.

3.2 Prediction Accuracy

We use our model to predict the relative changes in power consumption. To only show the relative changes, we remove all machine specific values from the model. $pwr_{min}(CPU)$ and $pwr_{max}(CPU)$ are set 0 and 1 respectively. The value for *cachesize* and *inverseprefetcheraccuracy* are set to 1, removing all machine specific impacts on our model. The resulting power consumption values therefore only rely on the data type and stride. Results together with the measured full system power as well as the predicted ordering for comparison are shown in Table 2.

As can be seen from our results, the model can keep the ordering in cases the $datasize * stride$ exceeds the cache line size or the size of the data type is small. Comparing the predicted ordering with the measurement results shows that the model can predict the ordering correctly, if the difference

Data Type	Stride	Power Measured	Power Estimated
`char`	1	$69.04W$	$89.14W$
`int`	1	$60.36W$	$88.44W$
`long`	1	$60.30W$	$87.52W$
`double`	1	$60.79W$	$87.52W$
`double`	2	$60.66W$	$85.67W$
`double`	4	$61.04W$	$81.95W$
`double`	8	$60.72W$	$75.58W$
`double`	16	$59.80W$	$59.80W$
`double`	24	$58.24W$	$45.02W$

Table 3: CPU power measured compared to configured model estimation

Data Type	Stride	Power Measured	Power Estimated
`char`	1	$9.13W$	$14.75W$
`int`	1	$9.11W$	$14.50W$
`long`	1	$9.12W$	$14.16W$
`double`	1	$9.14W$	$14.16W$
`double`	2	$9.10W$	$13.49W$
`double`	4	$9.12W$	$12.14W$
`double`	8	$9.09W$	$9.45W$
`double`	16	$9.04W$	$4.070W$
`double`	24	$8.99W$	$-1.32W$

Table 4: Measured and estimated memory power consumption with an idle full system power of $36.4W$

in $datasize * stride$ is large. The model results also show a minor discrepancy between `long` and `double` stemming form different data types. Our model does not take different data types into account that might use different execution units of the CPU resulting in different power consumptions.

To evaluate the prediction accuracy of our model, we measure the CPU power with IntelPCM while executing modified versions of the stream benchmark. The full system power consumption is measured concurrently with an external power meter. Both, CPU and full system power consumption are estimated with the aforementionend *inverseprefetcheraccuracy* and compared to the measurement results.

The results presented in Table 3 shows that the model can predict the CPU power consumption with an average deviation of $21.7W$ / 32.02%. As with the ordering, the best results are achieved if $datasize*stride$ exceeds the cache line size, with a deviation of -15.1%. For the full system power shown in Table 2, an average deviation of $23.5W$ / 19.34% could be achieved, with the lowest deviation using `double` together with a step size of 8.

The measured and estimated memory power are shown in Table 4. As idle power for the configured system, the IDLE workload from the SERT benchmark run is used with a full system power consumption of $36.4W$. The average memory power deviation is $0.36W$ / 3.97% with the lowest deviation at a step size of 8 with `double`. This coincides with the CPU power estimation. With deviations from both, the full system power and CPU power estimate propagating, a higher deviation than for other estimates is expected. The estimated memory power deviates with an average of $4.90W$ or 53.98% from the measurement.

Promising results at the edge cases with a high step size

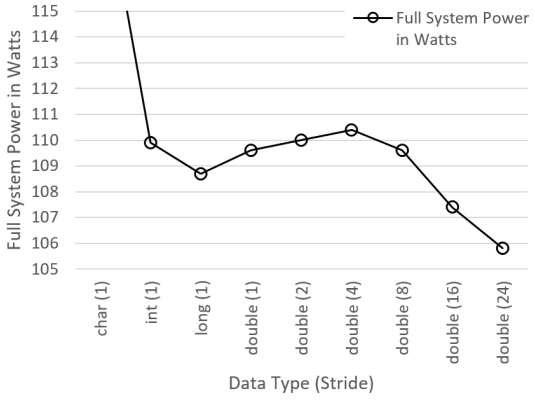

Figure 1: Measured full system power in watts for data types and strides

or low data type size stems from irregularities within our measurement results. While slight divergence from a linear model is expected, power consumption behaves non-linearly for small differences in step size or data type size, shown in Figure 1, which subsequently can lead to the observed misordering and deviations in the estimation.

The figure also shows that further measurements are necessary to improve our model. Also additional research for suitable reference benchmarks, more closely related to high-bandwidth memory workloads, is necessary to achieve better accuracy determining pwr_{min} and pwr_{max} for the CPU and full system power. Future work should also include a guideline for determining the $inverseprefetcheraccuracy$ for predictions. Higher accuracy for the $inverseprefetcheraccuracy$ parameter would also aid in avoiding mispredictions as seen in Table 4 for `double` with a step size of 24.

4. CONCLUSIONS

This paper introduces a model for prediction of power consumption of high-memory-bandwidth workloads. The model bridges the gap between system specific prediction models that require extensive system information, including measurements, and more generic models that do not consider memory specific workloads and their properties.

The model can help developers to compare power consumption of different implementation options by specifying their workload and using publicly available hardware information and/or default values for hardware specification. As a result, the model may lead to implementation choices with less power consumption.

We show that the model is able to predict a system's power consumption accurately with a mean deviation of 19 percent and ranks implementation options correctly if used with abstract default hardware parameters, enabling implementation comparisons.

5. REFERENCES

[1] SPEC Power and Performance Benchmark Methodology. http://spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf.

[2] R. Basmadjian and H. De Meer. Evaluating and modeling power consumption of multi-core processors. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–10, May 2012.

[3] A. Bohra and V. Chaudhary. Vmeter: Power modelling for virtualized clouds. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8, April 2010.

[4] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *The 34th ACM International Symposium on Computer Architecture*, 2007.

[5] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John. Using complete machine simulation for software power estimation: The softwatt approach. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, HPCA '02, pages 141–, Washington, DC, USA, 2002. IEEE Computer Society.

[6] Intel. Intel Performance Counter Monitor. https://software.intel.com/en-us/articles/intel-performance-counter-monitor.

[7] K.-D. Lange. Identifying Shades of Green: The SPECpower Benchmarks. *Computer*, 42(3):95–97, March 2009.

[8] K.-D. Lange and M. G. Tricker. The Design and Development of the Server Efficiency Rating Tool (SERT). In *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering*, ICPE '11, pages 145–150, New York, NY, USA, 2011. ACM.

[9] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, New York, NY, USA, 2009. ACM.

[10] M. Mamidipaka and N. Dutt. eCACTI: An enhanced power estimation model for on-chip caches. *Center for Embedded Computer Systems, Technical Report TR*, pages 04–28, 2004.

[11] J. D. McCalpin. STREAM benchmark. *Link: www. cs. virginia. edu/stream/ref. html# what*, 22, 1995.

[12] G. Reinman and N. P. Jouppi. CACTI 2.0: An integrated cache timing and power model. *Western Research Lab Research Report*, 7, 2000.

[13] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[14] A. Verma, P. Ahuja, and A. Neogi. Power-aware dynamic placement of hpc applications. In *Proceedings of the 22Nd Annual International Conference on Supercomputing*, ICS '08, pages 175–184, New York, NY, USA, 2008. ACM.

[15] J. von Kistowski, H. Block, J. Beckett, K.-D. Lange, J. A. Arnold, and S. Kounev. Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE 2015)*, ICPE '15, New York, NY, USA, February 2015. ACM.