Full-System Power Analysis and Modeling for Server Environments

Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis
Department of Electrical Engineering
Stanford University
{dimeco,rivoire}@stanford.edu, christos@ee.stanford.edu

Partha Ranganathan
Internet Systems and Storage Lab
Hewlett-Packard Labs
partha.ranganathan@hp.com

***Abstract*— The increasing costs of power delivery and cooling, as well as the trend toward higher-density computer systems, have created a growing demand for better power management in server environments. Despite the increasing interest in this issue, little work has been done in quantitatively understanding power consumption trends and developing simple yet accurate models to predict full-system power. We study the component-level power breakdown and variation, as well as temporal workload-specific power consumption of an instrumented power-optimized blade server. Using this analysis, we examine the validity of prior ad-hoc approaches to understanding power breakdown and quantify several interesting trends important for power modeling and management in the future. We also introduce Mantis, a non-intrusive method for modeling full-system power consumption and providing real-time power prediction. Mantis uses a one-time calibration phase to generate a model by correlating AC power measurements with user-level system utilization metrics. We experimentally validate the model on two server systems with drastically different power footprints and characteristics (a low-end blade and high-end compute-optimized server) using a variety of workloads. Mantis provides power estimates with high accuracy for both overall and temporal power consumption, making it a valuable tool for power-aware scheduling and analysis.**

I. Introduction

Power management is becoming an important issue in enterprise environments, both to reduce costs associated with power delivery and cooling as well as to improve compaction, reliability, and compliance with environmental standards. Recent trends toward server consolidation in data centers and adoption of higher-density computer systems such as blades are likely to further exacerbate these problems.

For example, for a 30,000 sq.ft. 10MW data center with 1000 standard computing racks each consuming 10KW, the annual cost of electricity for the computing equipment alone is likely to be close to $8 million [21]. The capital costs for the air conditioning needed to handle such heat dissipation levels is likely to be anywhere between $2-$5 million. Additionally, every watt of power consumption is likely to need another 0.5 to 1W of power to operate the cooling system [21] - adding another $4-$8 million in operational costs.

Power density is also one of the limiting factors preventing greater compaction, particularly with smaller form factors as in blade servers. Future blade servers are estimated to need close to 188K BTU/hr (for 55KW racks) [20]. Such densities may well require liquid cooling in the data center. Furthermore, increases in temperature associated with larger power consumption have been shown to reduce the reliability and efficiency of systems. Every 10C temperature increase over 21C can decrease the reliability of electronics by 50% [24]. Similarly, a 15C rise increases hard disk drive failure rates by a factor of two [10]. Finally, at a more global level, increased enterprise power consumption has also been linked with environmental consequences (e.g., 4 million tons of annual carbon dioxide emissions). This has led to recommendations and incentives from several environmental agencies to reduce enterprise power [25].

Despite the increasing interest in this issue, little work has been done in quantitatively understanding power consumption trends at a system level. Some of the key open questions include: Where is the power spent in enterprise systems? What are the key energy bottlenecks? What are the component-level temporal trends in power variation during application execution? What is the impact of specific workloads on the power usage characteristics of system components? To answer these questions, we instrumented a power-optimized blade server to extract component-level power measurements. We discuss the insights from this experiment and their applicability to developing power models.

Current approaches to power modeling fall into two broad classes: measurements of power at the hardware level [8], [13], [11] or modeling of power at the simulation level [5], [7], [9], [14], [22], [26], [27], [29]. Direct hardware measurements are fast and accurate but are only applicable to existing systems. Power modeling through simulation works for both existing and future systems and can provide detailed analysis and breakdown. Nevertheless, full-system simulators are extremely slow compared to real hardware and cannot be used with long applications and large data-sets. Moreover, simulation cannot be used to guide software monitoring and dynamic optimizations for a deployed system.

This paper presents Mantis, a method for full-system power modeling that is non-intrusive and provides a fast and accurate prediction of power consumption in server systems. Mantis-uses widely available low-overhead OS utilization metrics and performance counters to predict power. It requires a one-time, offline calibration phase to extract basic AC power consumption characteristics and relate them to the system

performance metrics. The calibration phase is run only once for each system. During application runs, Mantis estimates total power consumption using a set of user-level system utilization metrics. While Mantis cannot provide simulation-level accuracy since it is derived from standard user-level metrics, it is still fast, cost-effective, and accurate enough (within 10% for most workloads) to be used for online management of power consumption. It is also flexible and portable enough to be used for power exploration of future system architectures.

The specific contributions of this work are:

- We present component-level power consumption measurements for a blade system and observe trends important to future power research and power modeling.
- We develop Mantis, a novel non-intrusive hybrid hardware-software model for AC power prediction on server systems based on high-level system utilization metrics and hardware performance counters. Our model provides rich functionality, as it can predict peak and average power as well as temporal variation in power.
- We prototype Mantis on two drastically different server systems to demonstrate its portability and validate its predictions using direct AC power measurements. We verify that the model accurately predicts power consumption in both platforms.

The rest of the paper is organized as follows. Section 2 describes our component-level measurements and observations for the blade system. Section 3 presents the design of the Mantis model, including the measurement and calibration methodologies used. Section 4 presents the results of our model validation experiments, while Section 5 presents potential applications of Mantis and provides suggestions for further work. Section 6 presents related work and Section 7 concludes the paper.

II. Power Measurement and Analysis

We study the component-level power consumption of a blade server in order to understand future power consumption trends. The blade is ideal for such a study, since it already incorporates several important power management techniques: voltage and frequency scaling support for the processor, a low-power disk, and a low-power power supply. Using our measurements, we attempt to identify the next set of power bottlenecks and challenges for power modeling.

A. Methodology

While the overall server power consumption can be obtained simply by connecting a power meter between the system and an AC outlet, the component-level power consumption requires measuring the voltage and current drop across different components in the system board.

Our approach leverages our access to the system board schematics and uses board-level modifications to study the power consumed in various portions of the server. Our blade is organized into four power planes:

- A 12V plane whose power budget is dominated (more than 90%) by the processor and memory,
- A 5V plane whose power budget is dominated by the hard disk,
- A 5V auxiliary plane,
- A 3.3V plane that, with the 5V plane, accounts for the power consumed in the network, peripherals, regulators, supplies, and other miscellaneous components of the system.

We developed a power measurement and data acquisition board to measure and log the power consumed in these four planes concurrently. Since the processor and memory are both large components of the total power, we further cut into the 12V power plane (desoldered a component to add an extra sense resistance) to isolate the processor power.

B. Results and Observations

In this section, we discuss some high-level trends common in the results presented above and discuss potential pitfalls and opportunities for future work.

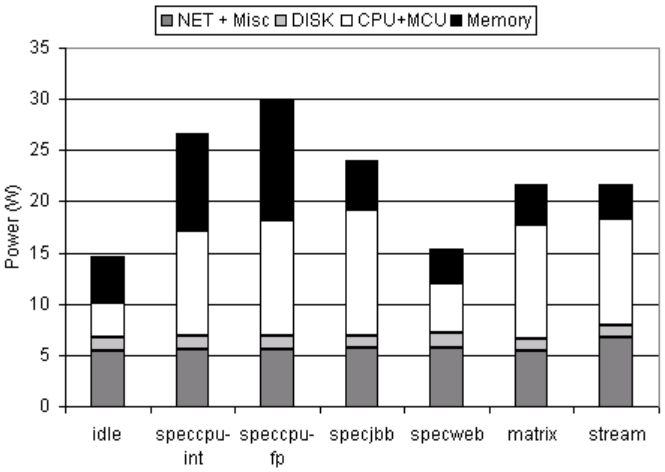

Fig. 1. The component breakdown of the blade's measured power consumption.

Inaccuracies from nameplate ratings. Figure 1 provides information on the absolute power consumption and component-level breakdown for the blade system. Comparison of these numbers with the nameplate power ratings for these machines show significant differences. For the blade system, the nameplate power rating overestimates power by almost 50%, and misestimates the importance of various components. This is particularly important when considered with the fact that it is currently common practice to use nameplate power when provisioning and optimizing the system.

Memory power consumption. Conventional intuition about the energy bottlenecks in the system has identified the processor as the most important component of server power. Our discussion from the previous section indicates that memory power consumption is likely to be equally, if not more, important in the future. Unlike processors that include support for techniques such as voltage and frequency scaling, power optimizations for memory are limited to transitions to lower-power states. While there have been several good studies

evaluating the potential of this approach, it will be important to develop other methods as well. This will be particularly useful in cases when transitioning memory modules to lower-power states can result in reduced bandwidth or increased latency.

The Power consumption in the `misc` component. One of the interesting observations from our power characterization is the large fraction of power spent on the non-processor-and-memory components. For example, more than 30-40% of the power is spent on the disk, the network, the I/O and peripherals, the power supplies, the regulators, and the rest of the glue circuitry in the server. Interestingly, only the disk and the power supply are single large contributors to this collection. At a component level, there are more than 30 other components that contribute to the remaining fraction of power in this category. Approaches to address these therefore will need to think of more holistic solutions to server design. Just as the increasing miniaturization with the "server-on-a-card" approach has led to corresponding consolidation in the chipset and controller space, potential exists for solutions that leverage greater consolidation to more finely-control power. Furthermore, aggressive solutions to turn off components that are not being used will also be beneficial.

III. The Mantis Model

A. Overview

Mantis captures the power characteristics of a system by correlating a few user-level utilization metrics or hardware performance counters with power consumption during a calibration phase. The derived model parameters are then used for predicting power consumption based on the same user-level utilization metrics or hardware performance counters. Hence, Mantis can calculate the overall average and instantaneous power consumption of a system. The update frequency of the utilization metrics and counters are what limit the frequency of the instantaneous power estimates.

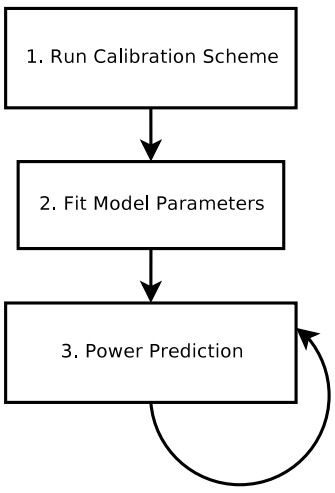

Fig. 2. The stages of Mantis model development and use.

Figure 2 illustrates the process of developing and applying the Mantis model. The first stage is running the calibration process on a system connected to an AC power meter. The calibration process consists of benchmarks that individually stress each major component of the system under test in order to derive the basic correlation between its utilization and power consumption. The second stage is to formulate the model based on the performance metrics and AC power data acquired during the calibration scheme. A linear program is used to fit the model parameters to the data, relating performance metrics to AC power variation for the system. The calibration and model derivation processes need to be run *exactly once* for a specific system type, most likely by its vendor [1]. Precedent exists for this type of additional vendor-supplied power measurement [3]. In that case, end-users of Mantis will proceed to the final stage, power prediction. During this stage, we continuously monitor the utilization metrics through the operating system or hardware counters while running the workload of interest. Based on the metrics and the model parameters, we can derive accurate predictions of overall and component-level power consumption. The power estimates can be fed directly to a scheduler or saved for offline analysis. This final stage is repeated for every workload. It involves no burden to the user such as using an AC power meter. The workload of interest runs at full speed on the computer system under test, with minor overheads associated with extracting utilization metrics and calculating the gradients.

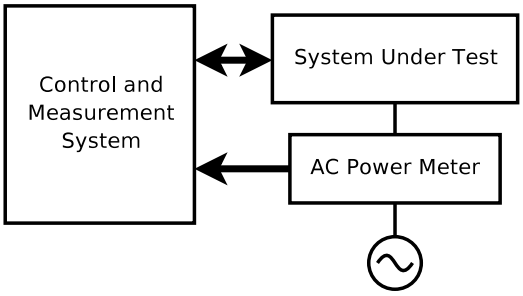

Fig. 3. The control and measurement system initializes the workload execution. Then it records performance metrics for the system under test while receiving power measurements from the AC power meter.

B. Measuring Power and Utilization

For the Mantis calibration stage, we measured the AC power consumed by each server while running a specific workload. Both OS performance metrics and hardware performance counters were used to measure system activity. The OS metrics used were CPU utilization and I/O request rates to the hard disk and network. They were collected on both systems with SAR [1]. Hardware performance counters were used to provide finer-granularity data for the main memory (off-chip misses). We collected performance counter numbers using modules such as *perfctl* and *perfmon* [2].

[1]If a system ships in multiple configurations, the vendor can perform the calibration phase for each likely component. Then the final model is derived at the customer site where the exact configuration is known. Customers will not have to measure AC power for calibration in most cases.

All power and utilization measurements were under the control of the system in Figure 3 to ensure that data were properly synchronized.

C. Calibration

The basic power characteristics of each major contributor to system power consumption are extracted in the calibration phase. Workloads that isolate and stress the system components in a controlled manner are run while utilization measurements are recorded. The data is then run through a linear program to derive the linear relations between power consumption and component utilization.

Gamut [17] emulates applications with varying levels of CPU, memory, hard disk, and network utilization. Accurately modeling components requires that at least one phase of the calibration scheme stresses each component individually. We configure Gamut to isolate one component in each run and vary the component utilization. To accurately model the power consumption of the system when idle, one of the calibration phases must be an idle run. The calibration phase of the Mantis models presented in this paper consists of an idle run and configurations of Gamut stressing the CPU, memory, hard disk, and network.

The linear program relates utilization metrics to power consumption while minimizing the absolute error of the model across all calibration phases. The utilization measurements are compiled into a matrix M with one column for each metric and a row for each time sample. The power measurements are compiled in a vector $\vec{p}_{meas}$. The matrix M is multiplied by the vector of model parameters for each metric (the program solution), $\vec{s}$, to produce $\vec{p}_{pred}$ containing the power prediction for each time sample.

$$\vec{p}_{pred} = M\vec{s}$$

The error of predicted power to measured power is defined as $\vec{\epsilon}$. The values of $\vec{\epsilon}$ are calculated as follows, where i indexes the vector elements:

$$\epsilon_i = \frac{p_{pred,i} - p_{meas,i}}{p_{meas,i}}$$

$\vec{\epsilon}$ is split into n components, each containing the error measurements for one of the N calibration phases. The average error during each calibration phase is defined as $\hat{\epsilon}_n$.

Minimizing absolute error while retaining linearity requires that the objective function of the linear program be defined as the difference between the positive and negative errors of the model. The variables t^+ and t^- are defined to separate the positive and negative errors, with their difference $(t^+ - t^-)$ being the absolute error of the model. The objective function of the linear program is the sum of the absolute average errors of the model during the N calibration phases, described using, t_n^+ and t_n^-, where n indicates the calibration phase:

$$\min \sum_{n=1}^{N} (t_n^+ - t_n^-)$$

$$s.t.$$

System	Blade Server	Itanium Server
CPU	2.2GHz AMD Turion	4x1.5GHz Itanium 2
Memory	512MB SDRAM	1GB DDR
Storage	40GB 2.5” Hard disk	36GB 3.5” Hard disk
Network	10/100MBit Ethernet	10/100MBit Ethernet

Fig. 4. The systems modeled by Mantis in this study.

$$t_n^+ \geq \hat{\epsilon}_n$$
$$t_n^- \leq \hat{\epsilon}_n$$

The objective is minimized while the model parameters ($\vec{s}$) are varied to derive the model parameters with minimum prediction error.

D. Implementation

We implemented Mantis on two very different server systems. The first system is a highly integrated blade server that includes an AMD Turion processor. The blade system has been optimized for power consumption. The second system is a high-end server that contains 4 Itanium2 chips. The Itanium server is optimized for peak performance. For these specific configurations we used, the memory, disk, and I/O systems of the two servers are similar, although in general the Itanium server has more room for additional memory and disks.

The model prediction granularity was limited by the speed at which the utilization metrics would update. For both systems, the prediction granularity was 1 second.

We should point out that Mantis would be implemented and used similarly for any other system. Still, we present its implementation with respect to the two systems studied in order to make the discussion less abstract.

The utilization metrics used in these specific models were:

- u_{cpu}: CPU utilization,
- u_{mem}: Off-chip memory access count,
- u_{disk}: Hard disk I/O rate
- u_{net}: Network I/O rate

The modeled power equations, containing parameters as derived using the linear program, are as follows:

$$P_{blade} = 14.45 + 0.236 * u_{cpu} - (4.47E\text{-}8) * u_{mem} + 0.00281 * u_{disk} + (3.1E\text{-}8) * u_{net}$$

$$P_{itanium} = 635.62 + 0.1108 * u_{cpu} + (4.05E\text{-}7) * u_{mem} + 0.00405 * u_{disk} + 0 * u_{net}$$

The first term in both equations is a constant representing the power consumption of the system when idle. In the following section, we will evaluate how well the Mantis modeling approach works in practice.

IV. Evaluation

To validate the Mantis approach to power modeling, we developed Mantis models for the blade and Itanium servers,

as described in the previous section. We used the models to estimate power consumption for each system running a diverse set of applications. Specifically, we used the SPECcpu2000 integer and floating-point benchmarks, SPECjbb2000, SPECweb2005, the *streams* benchmark, and matrix multiplication. Overall, this represents more than 30 different individual benchmark applications covering different computing domains (workstation, scientific, enterprise) and stressing different subsets of system components. The diversity in the applications used is critical to the validation of the model. If all applications were from a single domain, the model could be missing a critical parameter for a component not exercised by the applications.

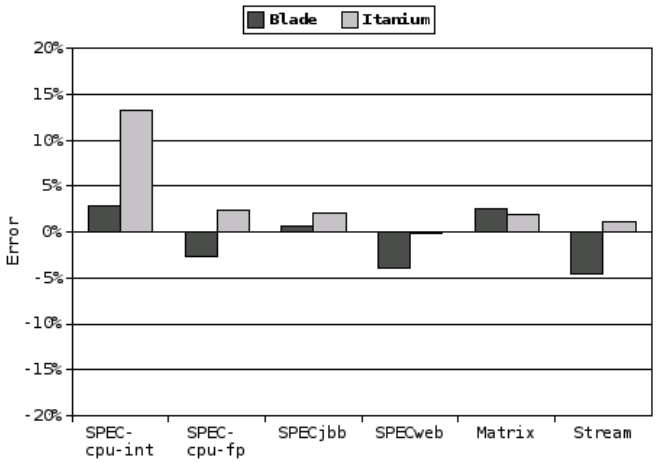

Fig. 5. The average error of the Mantis models during each of the benchmarks.

Figure 5 presents the model prediction accuracy for both systems. Overall, the errors range from 0% to 15%, with the blade model achieving less than 5% errors for all cases.

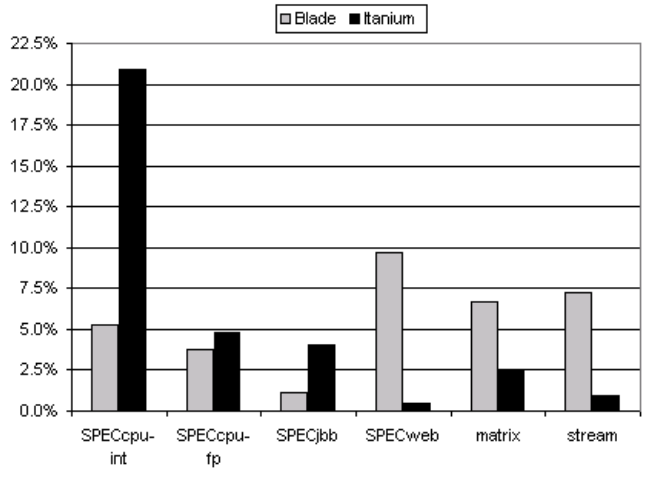

Fig. 6. The 90th percentile error for the models of both systems.

Figure 6 presents the temporal accuracy of the models. At each point in time, the absolute error is computed between the prediction and the measurement, and the 90th percentile of this error over all the samples is reported for each benchmark for both the systems. As can be seen from the figures, the errors are within 10% in all cases except for the Itanium model with SPECcpu2000-int. This is mainly due to the VLIW-esque EPIC architecture of the Itanium processor, for which OS-reported CPU utilization does not factor in the level of ILP of applications. These results indicate that not only is Mantis accurate in predicting the total average power across the benchmark, but it is also accurate in predicting the instantaneous power consumption.

V. Applications of Mantis

Here, we discuss the applications of the Mantis power model.

A. Intra-workload power variation

Workload-specific and server-specific power variation. At an intra-workload level, our results indicate very little power variation for many of the applications. Though there are occasional power ”spikes,” there is very little phase-based power behavior akin to what previous studies have shown for performance. At an inter-workload level, more variation exists. The blade server shows nearly a 50% variation between the SPECcpu-fp and the SPECweb. Also of interest is the power variation between the high-end and low-end system. Most previous intution on power for servers has lumped all servers into one category. Our analysis shows that, in addition to the differences in the absolute power, there are fundamental differences in the nature of the bottlenecks and variation in power trends between different classes of servers. Again, this motivates an online tool like Mantis that can accurately capture the per-system power variation on a run-time basis.

B. Run-time provisioning and control for power and heat

Mantis’s real-time power consumption model can also be used to plug into dynamic control algorithms for power and heat management. Below, we qualitatively discuss some examples.

Online power and heat management. The availability of real-time component-level power breakdown can enable several interesting power management optimizations. For example, PowerShifting [12] attempts to limit the total power budget by dynamically reprovisioning the power budget between the processor and memory components. With a model like Mantis, this approach can be made more accurate and extended to all the other components of the system as well. Mantis can also enable new optimizations. For example, current approaches primarily focus on p-state transitions or voltage scaling to reduce power. Often, changes in CPU utilization can yield similar power savings without the penalties of transitioning between different states, but these savings vary across different systems (often in non-intuitive ways). Mantis can be used to provide run-time calibration of the potential differences in power savings possible with reduced CPU utilization versus changed power states.

Another interesting optimization enabled by Mantis is to dynamically control the fan speed in response to the rest of the system power consumption. Currently, the fan power is constant and invariant across the execution of the total workload. However, the component-level power consumption insights from Mantis can be used to selectively turn on individual fans to better direct cooling resources to areas that need them most. As seen from the Itanium server results, fan power is a growing component of the total power, and an optimization like this can enable significant total server power savings.

TCO-aware resource provisioning in cluster and data center environments. Mantis can be extended beyond a single server to enable optimizations at a broader collection-of-systems level. An interesting application of Mantis is in predicting the ”thermal map” of a data center. The thermal map identifies the temperatures at the inlets of the individual servers in the room and is used to guide optimizations to control the cooling costs at data center level. Current approaches to determining the thermal map involve expensive deployment of *external sensors* to capture the temperature. However, an approach like Mantis that captures the *heat* generated by the server, used in conjunction with on-board per-server temperature sensors that are becoming a standard, can now be used to provide a proxy for the external sensors. In addition to reduced costs, this approach can also provide for faster and synchronized responses to data center level thermal optimizations such as those discussed in [19].

Another benefit of Mantis in enabling resource provisioning at a data center level to reduce the total costs of ownership (TCO). This is particularly important, given recent indications that the electricity costs can outweigh hardware costs in a data center [4]. Mantis can be used to efficiently provide an estimation of electricity costs at per-server, per-rack, per-solution levels. Compared to the conventional approach of deploying an ammeter per rack or at the power-distribution unit (PDU), this approach provides lower costs, as well as finer granularity and better correlation with workload behavior.

Extending this further, as utility-computing environments start provisioning resources based on both the performance guarantees as well as the power and heat implications [19], [18], models like Mantis can now be used to provide ”reverse calculations” on the resources that can be used for a given power budget, and how power-scheduling decisions can influence performance. This enables holistic TCO-aware resource provisioning optimizations that are otherwise not possible.

VI. Related Work

To the best of our knowledge, Mantis is unique in its approach to providing on-the-fly full-system power characterization as a function of OS-level resource utilization and generic performance counter metrics. SimplePower [26], SoftWatt [14], and Mambo [23] provide full-system power estimates but these studies use analytical models tied to low-level architectural events in a simulation system with corresponding drawbacks on speed and portability. These frameworks are difficult to use for online applications of power management.

There has also been significant work on component level power modeling. Wattch [7] is a widely used CPU power model that tries to accurately model the energy consumed by the array structures, wires, and clocking in a microprocessor. There have been other similar models for memory [22], disk [29], and networking [27]. While these models provided a detailed prediction of the power of a single component, they are typically off-line models used along with time-consuming simulationn systems. Hence, they are difficult to use for on-line power management or to analyze large commercial workloads that take too long to simulate.

Using real-time system events can address some of these drawbacks. Bellosa was one of the first to propose the notion of event-driven energy accounting [5]. This work, and other related studies [6], [28], explored the use of performance counters to provide on-the-fly power characterization of real systems. However, using performance monitoring counters alone can be quite inaccurate, as most processors allow for the measurement of only a limited number of concurrent counter readings. Time-multiplexing [15], [16] can address this problem, but at the expense of some loss of coverage. Moreover, processor counters provide no insight into the I/O system (disk and networking). Cignetti et al [9] use system calls that indicate state transitions for different hardware devices to measure power. Our work leverages similar observations to this body of work, but uses the intuition that in most real-world applications, OS-level metrics on *resource utilization* can also provide a good first-order proxy for power consumption, and these, supplemented with a few selected performance counters available on all current processor architectures, can provide good accuracy.

VII. Conclusion

As power consumption is a major limiting factor for current and future computer systems, an increasing amount of research focuses on power optimizations in schedulers or on the development of power-aware system architectures. However, a lot of this work hinges on accurate characterization or measurement of the power consumed by the system at run-time as a function of the workloads being run and the resources being used - an area that has unfortunately not received as much attention.

In this paper, we address a key need in the community, namely the lack of quantitative real-world measurement data showing power breakdown and variation for real-world benchmarks and systems. Leveraging an experimental setup that allowed us to measure the power consumption of the individual power planes, we study the total and component-level power for a number of workloads.

In addition to documenting the power consumption behavior of these benchmarks, our results illustrate several potential pitfalls and opportunities for future work. Specifically, our results show that indiscriminately using nameplate ratings and power calculators can often lead to erroneous conclusions. Our characterization of the variation in component power to

changes in higher-level software parameter indicates promise for a higher-level modeling approach that can provide component level power breakdown with complex hardware support. Though our blade system was already quite well-optimized for power, we show that there are still a lot of opportunities to optimize the power of non-cpu components (e.g., memory). Our data also show that optimizations that focus on improving the efficiency of the average case are likely to achieve further benefits.

We present Mantis - a non-intrusive method for modeling full-system power consumption that can be easily and flexibly used in power research. Mantis relies on component utilization metrics collected through the operating system or standard hardware counters. During an offline one-time calibration phase, components are stressed individually through synthetic workloads and models are created to correlate component utilization metrics to the power measured. These models are then incorporated into Mantis to predict average full-system power consumption at normal use without any direct power measurements.

We discuss the design of the Mantis model focusing on the modeling approach for the main components of the system and their instantiation for two different classes of servers – a low-end blade system and a high-end compute server. To validate the model, we measured the AC power consumption of these systems and compared the results predicted by Mantis. Across a suite of more than 30 benchmarks comprising SPECint, SPECfp, SPECweb, SPECjbb, and other applications like stream and matrix multiply, Mantis comes within 10% of actual measured average power. Additionally, we have good accuracies for fine-grained measurements. As power and heat start becoming one of the key challenges in future system designs, we belive that approaches like Mantis are likely to be a critical component of future power-aware solutions.

In the future, as the challenges with power management in dense servers get harder and simple approaches to improving the power efficiency of the system are exhausted, it will become even more important to consider more radical solutions - crossing hardware and software boundaries, system and rack boundaries, power and cooling boundaries, etc. Studies like this one provide the first step in helping us refine our understanding of the problem, hopefully enabling such solutions to address this emerging critical challenge.

REFERENCES

[1] Information about the Linux/UNIX sar command. http://www.computerhope.com/unix/usar.htm.

[2] Perfmon project. http://www.hpl.hp.com/research/linux/perfmon/.

[3] ASHRAE Handbook. http://resourcecenter.ashrae.org/store/ashrae/newstore.cgi?categoryid=146.

[4] L. Barroso. The price of performance. *ACM Queue*, 3(7), September 2005.

[5] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *Proceedings of the 9th ACM SIGOPS European Workshop*, Kolding, Denmark, Sept. 17–20 2000.

[6] F. Bellosa. The case for event-driven energy accounting. Technical Report TR-I4-01-07, University of Erlangen, Department of Computer Science, June 29 2001.

[7] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th International Symposium on Computer Architecture (ISCA)*, pages 83–94, June 2000.

[8] F. Chang, K. Farkas, and P. Ranganathan. Energy-driven statistical profiling detecting software hotspots. *Workshop on Power-Aware Computer Systems*, 2002.

[9] T. Cignetti, K. Komarov, and C. Ellis. Energy estimation tools for the Palm. In *Proceedings of the ACM Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Aug. 2000.

[10] G. Cole. Estimating drive reliability in desktop computers and consumer electronics. Technology Paper TP-338.1, Seagate Technologies, November 2000.

[11] G. Contreras and M. Martonosi. Power prediction for intel xscale® processors using performance monitoring unit events. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 221–226, New York, NY, USA, 2005. ACM Press.

[12] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *Proceedings of the 19th Annual International Conference on Supercomputing (ICS)*, pages 293–302, 2005.

[13] J. Flinn and M. Satyanarayanan. PowerScope: A tool for profiling the energy usage of mobile applications. In *Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 2–10, Feb. 1999.

[14] 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)*, page 141, Washington, DC, USA, 2002. IEEE Computer Society.

[15] R. Joseph and M. Martonosi. Run-time power estimation in high-performance microprocessors. *International Symposium on Low-Power Electronics and Design (ISLPED)*, pages 135–140, 2001.

[16] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and A. Sivasubramaniam. vEC: Virtual energy counters. In *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE)*, pages 28–31, New York, NY, USA, 2001. ACM Press.

[17] J. Moore. Gamut - Generic Application eMUlaTion, Dec. 2005. http://issg.cs.duke.edu/cod/.

[18] J. Moore, J. Chase, and P. Ranganathan. Weatherman: Automated, online, and predictive thermal mapping and management for data centers. To appear in the Third IEEE Conference on Autonomic Computing, June 2006.

[19] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": Temperature-aware resource assignment in data centers. In *Proceedings of the Usenix Annual Technical Conference*, 2005.

[20] J. Mouton. Enabling the vision: Leading the architecture of the future. Keynote speech, Server Blade Summit 2004.

[21] C. D. Patel, C. E. Bash, R. Sharma, and M. Beitelmal. Smart cooling of data centers. In *IPACK*, July 2003.

[22] F. Rawson. MEMPOWER: A simple memory power analysis tool set, Jan. 2004. IBM Austin Research Laboratory.

[23] H. Shafi, P. J. Bohrer, J. Phelan, C. A. Rusu, and J. L. Peterson. Design and validation of a performance and power simulator for PowerPC systems. *IBM Journal of Research and Development*, 47(5-6):641–651, 2003.

[24] R. Sullivan. Alternating hot and cold aisles provides more reliable cooling for server farms, 2000. Uptime Institute.

[25] United States Environmental Protection Agency. EPA Conference on Enterprise Servers and Data Centers, Jan. 2006. http://www.sun.com/aboutsun/environment/epa.jsp.

[26] N. Vijaykrishnan, M. T. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using SimplePower. In *Proceedings of the 27th International Symposium on Computer Architecture ISCA*, pages 95–106, 2000.

[27] H. Wang, X. Zhu, L. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the 35th Annual International Symposium on Microarchitecture (MICRO)*, pages 294–305, Nov. 2002.

[28] A. Weissel and F. Bellosa. Process cruise control: Event-driven clock scaling for dynamic power management. In *Proceedings of the In-*

ternational Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), October 2002.

[29] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. In *Proceedings of the 2nd Conference on File and Storage Technologies, San Francisco, California*, pages 217–230, Mar. 2003.