Proceedings of the Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)
Krakow, Poland

Jesus Carretero, Javier Garcia Blas
Roman Wyrzykowski, Emmanuel Jeannot.
(Editors)

September 10-11, 2015

Analyzing Power Consumption of I/O Operations in HPC Applications

Pablo Llopis, Manuel F. Dolz,
Javier García-Blas, Florin Isaila,
Jesús Carretero

University Carlos III, Spain
{pllopis,mdolz,fjblas,
florin,jcarrete}@arcos.inf.uc3m.es

Mohammad Reza Heidari,
Michael Kuhn

University of Hamburg, Germany
{heidari,kuhn}@informatik.uni-hamburg.de

Abstract

Data movement is becoming a key issue in terms of performance and energy consumption in high performance computing (HPC) systems, in general, and Exascale systems, in particular. A preliminary step to perform I/O optimization and face the Exascale challenges is to deepen our understanding of energy consumption across the I/O stacks. In this paper, we analyze the power draw of different I/O operations using a new fine-grained internal wattmeter while simultaneously collecting system metrics. Based on correlations between the recorded metrics and the instantaneous internal power consumption, our methodology identifies the significant metrics with respect to power consumption and decides which ones should contribute directly or in a derivative manner. This approach has the advantage of building I/O power models based on a previous set of identified utilization metrics. This technique will be validated using write operations on an Intel Xeon Nehalem server system, as writes exhibit interesting patterns and distinct power regimes.

Keywords HPC, I/O operations, power analysis, system metrics, statistical analysis.

I. Introduction

Modern scientific discoveries have been driven by an insatiable demand for high computing performance. However, as we progress on the road to Exascale systems, energy consumption becomes a primary obstacle in the design and maintenance of HPC facilities. A simple extrapolation shows that an Exascale platform based on the current most energy efficient hardware available in the Green500 [1] would consume 120 MW. The power wall being set to 20 MW [2], this system would still exceed this limit by a factor of five, thus turning it economically unfeasible due to its projected TCO. Indeed, systems will need to reach an energy efficiency of 50 GFLOPS/Watt to face the Exascale challenge. Actually, hardware vendors are already trying to provide more energy-efficient parts and software developers are gradually increasing power-awareness in the current software stack, from applications to operating systems. For example, recent advances in processor technologies have enabled operating systems to leverage new energy efficient mechanisms such as DVFS (Dynamic Voltage and Frequency Scaling) or DCT (Dynamic Concurrency Throttling) to limit power consumption of computing systems.

Data movement has been identified as an extremely important challenge among many others on the way towards the Exascale computing [2]. The low performance of the I/O operations especially in I/O-intensive scientific domains and simulations continues to present a formidable obstacle to reaching Exascale computing in the future large-scale systems. CPU speed and HDD capacity are boosted approximately by factors of 400 and 100 every 10 years, respectively. HDD speed, however, develops at a slower pace and can only be increased by a factor of 20 every 10 years. This issue

triggers a special interest in optimizing storage systems in data centers, and motivates the need for more research to improve the energy efficiency of storage technologies. Therefore, a first step to develop I/O optimizations is to further understand how energy is consumed in the whole I/O stack.

Due to the key role of power constraints, future Exascale systems are expected to work with a limited power budget, and be able to allocate power to different subsystems dynamically. In this scenario, the capability of predicting power consumption based on data movement and I/O operations is a useful resource. In this paper, we take advantage of a new internal wattmeter to deeply analyze the power drawn at every single wire leaving the PSU and feeding the hardware components of a server platform. While some existing works have focused on studying power consumption of system components such as storage devices, CPUs and memory [3], we offer a detailed view of the whole I/O stack power usage across all system components, from operating system mechanisms down to storage devices.

Given the foregoing, this paper makes the following contributions:

- Leveraging our power measurement and system metrics frameworks, we can benefit from data exploration and analysis to provide insights into the relation between power and data movement of I/O operations.
- We present a methodology that identifies key system metrics which are greatly correlated with power usage during data movement and I/O operations.
- Using our methodology, we conclude the most useful metrics in practical terms and narrow them down to a subset that can reflect the power consumption resulting from the data movement across the I/O stack.

The rest of this paper is structured as follows: In the second section, we present some related works about power analysis and modeling. In Section III, we detail our data acquisition framework, which consists of a power measurement and a system data collection framework as well as a detailed description of our new wattmeter. Section IV describes the proposed methodology for analyzing the data acquired from our software and hardware frameworks. Section V presents the results of applying our methodology and provides key insights into how the system consumes power when performing write operations. Finally, the conclusion section summarizes of our contributions and suggests some future works.

II. Related Work

Current approaches for analyzing power usage and estimating energy consumption fall into different categories: power modeling at the hardware level [4, 5], power modeling at the performance counters level [6], and power modeling at the simulation level [7, 8, 9].

Simulation techniques are commonly used for evaluating both performance and energy consumption. Prada et al. [8] describe a novel methodology that aims to build fast simulation models for storage devices. The method uses a workload as a starting point and produces a random variate generator that can be easily integrated into large-scale simulation models. A disk energy simulator, namely Dempsey [10], reads I/O traces and interprets both performance and power consumption of each I/O operation using the DiskSim simulator. Dempsey was only validated on mobile disk drives. This solution predicts energy consumption using the simulated disks characteristics instead of system metrics. Manousakis et al. [5] present FDIO, a feedback-driven controller that improves DVFS for I/O-intensive applications. This solution relies on the node being instrumented for obtaining fine-grained power measurement readings. Their feedback controller detects I/O phases, quickly switches the CPU frequency to all possible states and then selects the optimal setting regarding its power/performance ratio. El-Sayed et al. [11] demonstrated that energy-optimized adaptive policies result in higher quality energy/runtime tradeoffs than the static (constant) policies. While our work also describes the physical instrumentation to obtain fine-grained power readings, we use this instrument to analyze data movement patterns and detect power regimes. Our proposed model does not require having this kind of invasive instrumentation in order to predict energy consumption. Lewis et al. [6] uses

the node temperature to predict energy consumption. The authors discuss the interaction of the different components for their modeling. The authors propose using read and writes per second metric (obtained by iostat) for modeling I/O workloads. Allalouf et al. [4] develop a scalable power modeling method that estimates the power consumption of storage workloads (STAMP). The modeling concept is based on identifying the major workload contributors to the power consumed by the disk arrays. Deng et al. [12] model the flash memory based storage systems constructed as a RAID by leveraging the semantic I/O. In a contribution similar to ours, the authors calculate the cost and energy consumption of storage devices for running a variety of workloads, categorized by their dominant requirements [13]. In contrast, our solution predicts the energy consumed by the complete I/O stack (including CPU and memory consumption) for single/multi core access patterns, in addition to the storage devices. For estimating energy consumption, some works also focus on system performance counters [14, 15, 16]. These works propose linear models that are able to provide run-time power estimations, and are validated with instrumented hardware. Other works concentrate on reducing energy consumption of individual storage devices. Zhu et al. [17] optimize disk energy consumption by tuning cache-replacement strategies that increase idle time and reduce disk spin-ups.

Our work focuses on exploring, analyzing, and modeling the power consumption of the operating system's I/O stack across all components. Like Li et al. [18], we aim to build models that help better understand and reduce the energy consumption of the storage stack. Unlike most works mentioned in this section, we do not provide a generic power model for computation, or limit our analysis to a single system component, but focus on energy consumption caused by data movement patterns across the memory hierarchy and I/O stack.

III. Data Acquisition Framework

In this section, we describe the power performance measurement framework that we use to instrument our platform to perform the I/O analysis in detail. Specifically, we leverage our `pyprocstat` tool for tracing `procfs` and `sysfs` system metrics and the PMLib framework to support both external and internal wattmeter devices. However, we will focus on the data obtained from the internal wattmeter, as this device provides more fine-grained measurements and is able to detect rapid power variations. We use the external wattmeter only for validation purposes. Figure 1 depicts the ontology of our framework, divided in both system and power measurement categories with their corresponding tools.

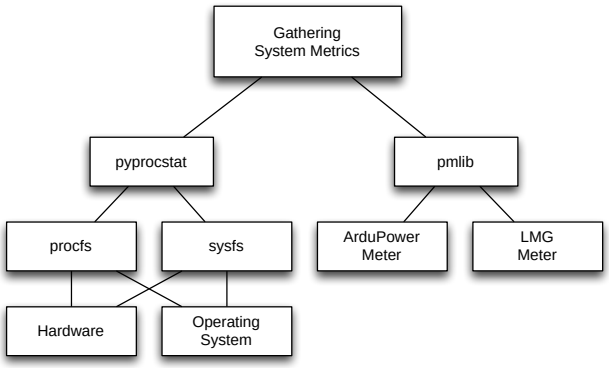


Figure 1: Ontology of system metrics used in our measurement framework.

III.1 System metrics collection framework

We instrument our platform to gather live system metrics in order to analyze workload behaviours and detect the correlation of the system activities with power consumption. These data traces can be easily correlated with power consumption traces for further exploration and analysis, as demonstrated in Section V.

While common UNIX tools such as `top` and `iostat` are able to gather live system information, they are not well-suited for our purposes. Our goal is to obtain traces that are aggregated into a time series consisting of different system metrics. Scripting existing tools in order to generate the resulting data traces is not very practical, since this method ends up launching processes several times and causing unnecessary overheads. Similarly, data stemming from performance counters gathered with the Linux `perf` framework are limited because their main objective is timing function

calls and counting function calls, with limited support for inspecting runtime values. Hence, they are better suited to other tasks such as profiling and performance debugging.

Instead, we develop `pyprocstat` [19], an easy to use, low-overhead, modular and flexible tool specifically built for our purposes. This tool consists of different modules, each of which is in charge of gathering information from different parts of the system. These modules usually collect information directly from kernel-provided interfaces such as procfs and sysfs, providing system data with a low overhead. In this paper, we collect system information from I/O devices, virtual memory, interrupts and per-CPU utilization, adding up to a total of 120 different system metrics.

III.2 Power measurement framework

To measure power consumption, we leverage the PMLib framework, a well-established package for investigating power usage in HPC applications [20]. Its implementation provides a general interface to utilize a wide range of wattmeters, including *i*) external devices, such as commercial PDUs, WattsUp? Pro .Net, ZES Zimmer Lmg450, etc., *ii*) internal wattmeters, directly attached to the power lines leaving from PSU, such ArduPower, *iii*) commercial DAS from National Instruments (NI) and *iv*) integrated power measurement interfaces such as Intel RAPL, NVIDIA NVML, IPMI, etc. The PMLib client side provides a C library with a set of routines in order to measure the application code. The traces obtained can be easily integrated into existing profiling and tracing frameworks such as Extrae+Paraver [21] or VampirTrace+Vampir [22].

III.2.1 ArduPower as a low-cost internal wattmeter

In this section, we describe our new internal ArduPower wattmeter in detail. ArduPower has been conceived as a low-cost DAS to measure the instantaneous DC power consumption of the internal components in computing systems wherever the PSU output power lines are accessible [23]. In general, it offers a spatially fine-grained power measurement by providing 16 channels to monitor the power consumption of several parts, e.g., mainboard, HDD and GPU cards etc., simultaneously, with a sampling rate varying from

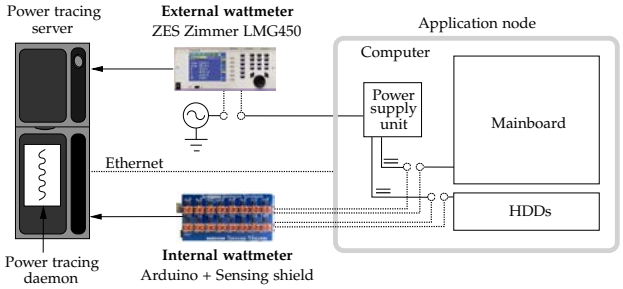


Figure 2: Power measurement setup combining both internal ArduPower *and external* Lmg450 *wattmeters.*

480 to 5,880 Sa/s, depending on the number of selected channels. The total production cost of the wattmeter is approximately 100 €, so it can be easily accommodated in moderate-/large-scale HPC platforms in order to investigate power consumption of scientific applications. As shown in Figure 3, ArduPower wattmeter consists of two basic hardware components: a shield of 16 current sensors and an Arduino Mega 2560 processing board detailed below.

Sensing shield ArduPower comprises a self-designed shield, responsible for sensing the DC currents passing through the wattmeter and providing the outputs to the processor board of the Arduino. It consists of 16 Allegro ACS713 current sensors [24], each of them converting the DC current passing through it to a proportional voltage. The Hall-effect elements provide a highly accurate, low noise output voltage signal proportional to the applied DC current, sensing up to 20 A with a total output error of $\pm 1.5\%$ and a low internal resistance of 1.2 mΩ.

Microcontroller ArduPower also comprises an Arduino Mega 2560 processing board that is connected directly to the power sensing shield. It benefits from one Atmel ATmega2560 [25] as a high-performance low-power 8-bit AVR RISC-based microcontroller running at 16 MHz and combining 256 KiB ISP flash memory, 8 KiB SRAM and 4 KiB EEPROM. The complete board of Arduino Mega 2560 has 16 analog inputs supported by a 10-bit ADC, 4 UARTs (hardware serial ports) and a USB link.

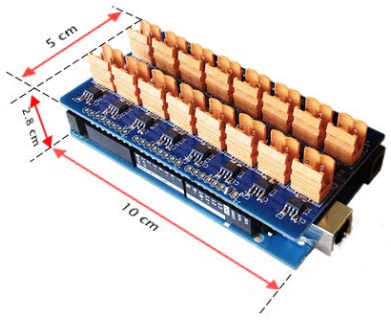


Figure 3: ArduPower *wattmeter, power sensing shield and Arduino Mega 2560 processing board.*

The sensing shield is placed on the top of Arduino Mega 2560 to let an analog to digital converter (ADC) read the outputs of the 16 current sensors and transform them into digital values. The Arduino board communicates with the target tracing server through a serial USB link and sends the measured DC current values to a PMLib server in order to calculate the instantaneous power consumption.

IV. Data Exploration and Methodology

Our goal is to explore the data to understand how the electrical power is consumed in a computing system to perform I/O operations. The analysis should reveal the system metrics which are correlated with the specific I/O operations and are useful for modeling the power usage related to data movement and I/O operations. In order to measure data movement, we perform simple sequential write I/O operations while collecting data as detailed in section III. To carry out this micro benchmark, we leverage `fio` [26], which is a commonly used micro benchmarking tool developed by the lead developer and maintainer of the Linux block IO subsystem.

In spite of the apparent simplicity of these operations, write I/O operations exhibit interesting irregular patterns due to the way the operating system manages data while it moves across the I/O stack, as depicted in Figure 4. However, read I/O operations do not exhibit these patterns and, therefore, this is why this work entirely focuses on write I/O operations.

We identify different power and performance

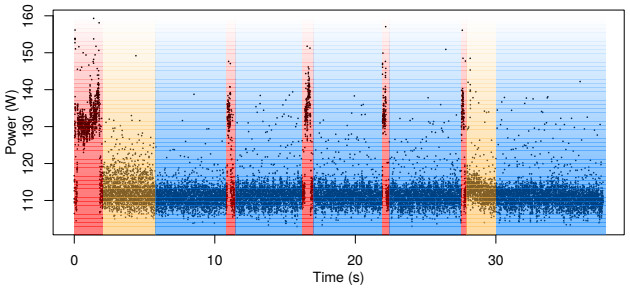


Figure 4: Power regimes during a sequential write of a 4 GiB file.

regimes that correspond to temporal regions in these I/O operations. Figure 4 shows that for a simple write I/O operation, the power consumption can vary significantly. This is due to the fact that the system transitions between different power and performance regimes while data moves from main memory and is written to disk. This shows that a simple straw-man approach to modeling writes using average power and write duration as input would not be sufficient, especially for short write operations. This also motivates the need to have a way for estimating the power consumption of I/O operations.

Using the data collected as a time series, we design a methodology in order to detect highly correlated metrics with regard to power consumption, following a similar approach as in [27]. Identifying these metrics is important for developing new power usage models that are more sophisticated than the aforementioned straw-man approach. This methodology leverages the Pearson's correlation and consists of the following steps:

1. For each collected system metric, we calculate its correlation with power consumption.
2. For each collected system metric, we compute the derivative and calculate its correlation with power consumption.
3. Every correlation whose absolute value is below than an empirically determined threshold t is discarded.
4. The union from both correlations results in a table of system metrics that are relevant for power usage

during data movement of I/O operations.

Note that in the design of the methodology, we have taken several aspects into account. First, some metrics are cumulative values, and therefore monotonically increasing with time, e.g, number of interruptions occurred since boot. Others are defined as a rate or instantaneous value and might vary with time accordingly, e.g., power consumption varies depending on the load of the machine. Therefore, a methodology should be aware of this fact in order to avoid correlations of metrics of different nature, and convert cumulative time series to instantaneous values so all metrics can be compared. The transformation from accumulated to instantaneous values is normally performed by subtracting the previous observation r_{t-1} from the current one r_t at the time t.

Second, since data movement has a direct impact on the power consumption, metrics that are measured in quantities of data should be transformed into the rates of movement by computing their corresponding derivatives. For example, as shown in Figure 5, the dirty memory metric measures the number of bytes in memory that must be written back to the disk at a given time[1]. However, the page dirtying rate (or speed) needs to be derived from the dirty memory metric in order to be compared with the power consumption properly. By calculating the dirtying rate, it is possible to measure the amount of data that any user-space application is writing to main memory. Information closely related to data movement can be obtained from the available system metrics. We believe this is a fundamental step when developing a methodology that identifies most correlated metrics with regard to the power usage.

V. Analysis of System I/O Operations

In this section, we perform an analysis of I/O operations using our measurement framework and the proposed methodology described in Section IV for sequential I/O writes. First, we describe the hardware setup and the software configuration that have been used for this work in detail.

[1]Note that the absolute value of the derivative is computed in order to superimpose positive and negative rates on a single normalized plot.

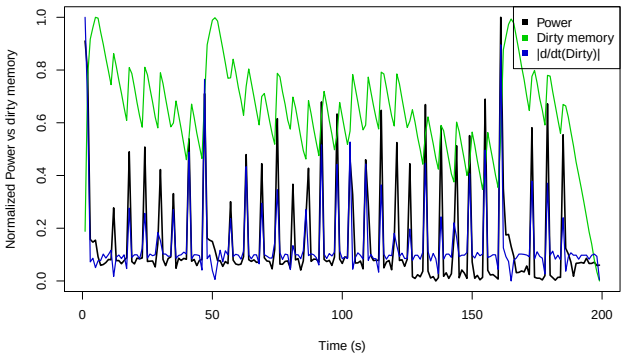


Figure 5: Dirty memory, $|d/dt(\text{dirty memory})|$ and power consumption of a sequential write of a 20 GiB file.

V.1 Configuration setup

Target platform. The analysis and evaluation have been carried out on a server platform, denoted as Nehalem, equipped with 2× Intel Xeon X5560 (total of 8 physical cores) running at 2.80 GHz, 12 GB of RAM memory, and a Seagate Barracuda 500 GB S-ATA II HDD, equipped with a Supermicro PSU 720W 80+ Gold (≈82% energy efficiency).

Software instrumentation framework. We use `pyprocstat` [19] to gather system metrics and obtain time series that describes live system information. We configure this tool to use the following built-in modules: `meminfo` (collects data from `/proc/meminfo`), `stat` (collects CPU utilization data, interrupts, context switches, etc from `/proc/stat`), `vmstat` (collects virtual memory data from `/proc/vmstat`), and `io` (collects I/O data from `sysfs`).

Power measurement framework. As mentioned before, we use the PMLib software package to investigate power usage of HPC applications. Power measurement can be controlled by the applications using a collection of routines that allow the user to query information on the power measurement units, create counters associated with a device where power data is stored, start, continue and terminate power sampling, etc. All this

information is managed by the PMLib tracing server, which is in charge of acquiring data from the devices and sending back the appropriate answers to the invoking client application via the proper PMLib routines (see Figure 2).

Wattmeters. We use the ArduPower wattmeter connected to all the power lines leaving from the PSU and feeding the different components of the server machine. We leverage 16 channels of ArduPower to measure the 3.3 V, 5 V and 12 V lines from a 24 ATX motherboard connector, 2× 4-pin connectors, and a 4-pin Molex connector. We avoid ground and negative voltage lines. On the other hand, we also employ an external ZES Zimmer Lmg450 wattmeter measuring Nehalem in order to verify that the internal measurements are well correlated with the external ones. This wattmeter can measure up to 20 Sa/s.

V.2 Analysis of write operations

Applying our proposed methodology, we obtain one power usage time series and 120 system metric time series for every benchmark run. Figure 6 depicts the correlations of all the 120 system metrics with power consumption during a sequential write of a 4 GiB file on Nehalem. The top plot shows the direct correlations, while the bottom plot takes the derivative of the data before computing the correlation with power. Indeed, the bottom plot clearly shows that only one system metric is highly correlated with power (B4) and the rest have a very low correlation. Table 1 lists the most significant system metrics. As it is shown in the table, those metrics which have values below our empirically obtained threshold of 0.75 have been discarded.

Where `cpu_system` is the system CPU utilization, `Dirty` is the number of dirty memory pages, `softirq` is the number of Linux software IRQs, `procs_running` is the number of running processes, and `cpu_user` is the user mode CPU utilization. Not surprisingly, CPU utilization is highly correlated with power usage since the CPU is the most power intensive component during these operations. Interrupts are also highly correlated with the I/O power usage. However, we argue that the most relevant system metric for write operations

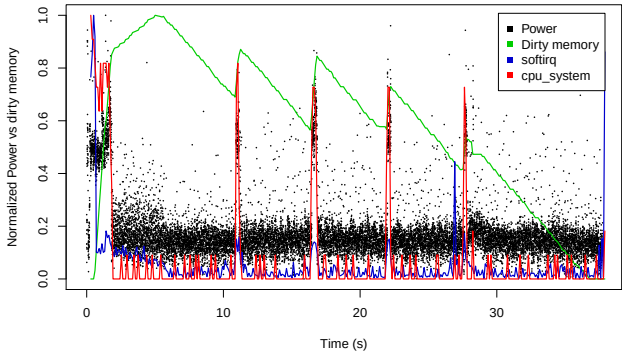


Figure 7: Power consumption vs. dirty memory for the sequential write of a 4 GiB file.

is the derivative of the number of dirty pages. While the system CPU utilization shows a slightly higher correlation, we can hardly use this metric to reflect the level of power consumption used by I/O operations since the system CPU utilization correlates with the power consumption of other workloads running on the machine and, in consequence, this metric is not useful for decoupling I/O from other workloads. Similarly, the number of running processes cannot be considered a good metric due to its nature. Therefore, we conclude that the page dirtying rate, d/dt(`Dirty`), is the best system metric to reflect the I/O-related power usage. Figure 7 clearly shows how well the `Dirty` system metric describes data movement with regard to power consumption.

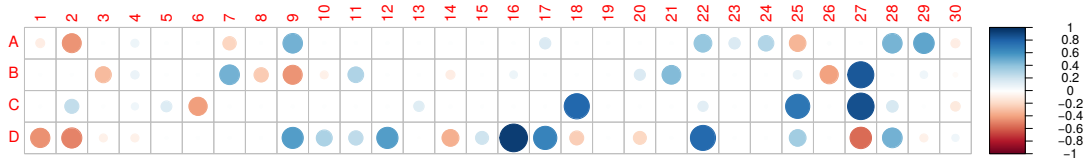
VI. Conclusions

In this paper, we leverage a power and system tracing framework in order to deeply analyze power usage due to data movement across the I/O stack. Among power consumption measurements collected in this framework, we also gather system metrics obtained from the `procfs` and `sysfs` file systems. Next, we present a new methodology to determine which system metrics are highly correlated to power consumption. We validate this technique by performing write operations on an Intel Xeon Nehalem server system instrumented with ArduPower and Lmg450, a fine-grain internal DC wattmeter and one external AC wattmeter, respectively.

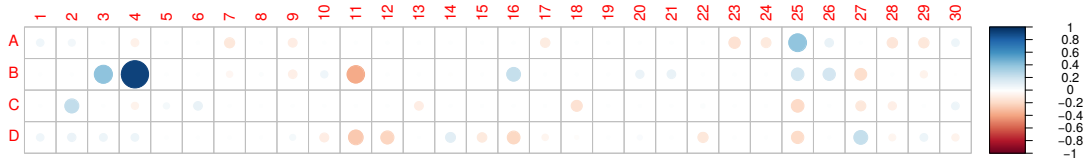
Several aspects are taken into account in designing

Table 1: Correlation of system metrics to power for a sequential write.

Metric	Corr(data)	Corr(d/dt(data))	Corrplot Tile
`cpu_system`	0.94	-0.20	D16
`Dirty`	0.08	0.92	B4
`softirq`	0.87	-0.12	C27
`procs_running`	0.84	-0.17	B27
`cpu_user`	0.77	-0.12	D22



(a) Raw system metrics.



(b) Derivative system metrics w.r.t. time.

Figure 6: Correlation plot between power usage and system metrics for a 4 GiB write.

the methodology. First, some of the metrics measured are reported as accumulated values, e.g., number of interruptions occurred since last read of a counter or the number of dirty pages at a given time. Due to their nature, some metrics do not appear to have direct impact on the power consumption; however, if their derivatives are computed, their instantaneous rates of change can correlate better with power consumption. In other words, the presented methodology can determine if a metric is highly correlated to power in the direct or derivative mode.

The analysis results demonstrate that only a small portion of the metrics, such as the CPU utilization, the rate of change in the number of dirty pages, software interruptions and number of processes running, have direct impacts on the power consumption and, due to their strong correlation, they can eventually be incorporated into I/O power models.

For future works, we plan to refine our methodology in order to analyze the correlations at the PSU wire levels, as obtained from the internal ArduPower wattmeter, and extend our benchmark suite to comprise more I/O operations with different configurations. We also aim to automatically build I/O power models using the methodology presented in this paper.

Acknowledgements

The work presented in this paper has been partially supported by the EU Project FP7 318793 "EXA2GREEN" and partially supported by the EU under the COST Programme Action IC1305, "Network for Sustainable Ultrascale Computing (NESUS)" and by the grant TIN2013-41350-P, *Scalable Data Management Techniques for High-End Computing Systems* from the Spanish Ministry of Economy and Competitiveness.

References

[1] The Green500 Editors. Green500. `http://www.green500.org/`, 6 2015. Last accessed: 2015-8.

[2] US Department of Energy. Top Ten Exascale Research Challenges. Technical report, Department of Computer Science, Michigan State University, February 2014. `http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf`.

[3] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, March 2014.

[4] Miriam Allalouf, Yuriy Arbitman, Michael Factor, Ronen I. Kat, Kalman Meth, and Dalit Naor. Storage modeling for power estimation. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, SYSTOR '09, pages 3:1–3:10, New York, NY, USA, 2009. ACM.

[5] Ioannis Manousakis, Manolis Marazakis, and Angelos Bilas. Fdio: A feedback driven controller for minimizing energy in i/o-intensive applications. In *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'13, pages 16–16, Berkeley, CA, USA, 2013. USENIX Association.

[6] Adam Lewis, Soumik Ghosh, and N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 4–4, Berkeley, CA, USA, 2008. USENIX Association.

[7] Guangyu Sun, Yongsoo Joo, Yibo Chen, Dimin Niu, Yuan Xie, Yiran Chen, and Hai Li. A hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement. In *2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12, Jan 2010.

[8] Laura Prada, Javier Garcia, Alejandro Calderon, J. Daniel Garcia, and Jesus Carretero. A novel black-box simulation model methodology for predicting performance and energy consumption in commodity storage devices. *Simulation Modelling Practice and Theory*, 34(0):48 – 63, 2013.

[9] Timo Minartz, JulianM. Kunkel, and Thomas Ludwig. Simulation of power consumption of energy efficient cluster hardware. volume 25, pages 165–175. Springer-Verlag, 2010.

[10] John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng, Arvind Krishnamurthy, and Randolph Wang. Modeling hard-disk power consumption. In *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, FAST '03, pages 217–230, Berkeley, CA, USA, 2003. USENIX Association.

[11] Nosayba El-Sayed and Bianca Schroeder. To checkpoint or not to checkpoint: Understanding energy-performance-i/o tradeoffs in hpc checkpointing. In *IEEE International Conference on Cluster Computing (CLUSTER)*, pages 93–102. IEEE, 2014.

[12] Yuhui Deng, Lijuan Lu, Qiang Zou, Shuqiang Huang, and Jipeng Zhou. Modeling the aging process of flash storage by leveraging semantic i/o. *Future Generation Computer Systems*, 32(0):338 – 344, 2014.

[13] Yan Li and D.D.E. Long. Which storage device is the greenest? modeling the energy cost of i/o workloads. In *IEEE 22nd International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 100–105, Sept 2014.

[14] Gilberto Contreras and Margaret Martonosi. Power prediction for intel XScale® processors using performance monitoring unit events. In *Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on*, pages 221–226. IEEE, 2005.

[15] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. Full-system

power analysis and modeling for server environments. International Symposium on Computer Architecture-IEEE, 2006.

[16] Tao Li and Lizy Kurian John. Run-time modeling and estimation of operating system power consumption. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '03, pages 160–171, New York, NY, USA, 2003. ACM.

[17] Qingbo Zhu, F.M. David, C.F. Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *In Proceedings of IEE Software*, pages 118–118, Feb 2004.

[18] Jing Li, Anirudh Badam, Ranveer Chandra, Steven Swanson, Bruce Worthington, and Qi Zhang. On the energy overhead of mobile storage systems. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies*, FAST'14, pages 105–118, Berkeley, CA, USA, 2014. USENIX Association.

[19] Pablo Llopis. A powerful and modular tool for gathering live system information as time series. `https://github.com/pllopis/pyprocstat`.

[20] S. Barrachina, M. Barreda, S. Catalán, M.F. Dolz, G. Fabregat, R. Mayo, and E.S. Quintana-Ortí. An integrated framework for power-performance analysis of parallel scientific workloads. In *ENERGY 2013, The 3rd International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 114–119, 2013.

[21] Paraver: the flexible analysis tool. `http://www.cepba.upc.es/paraver`. [Last access: June 2015].

[22] The vampir performance analysis tool-set. `https://www.vampir.eu/`. [Last access: June 2015].

[23] Manuel F. Dolz, Mohammad Reza Heidari, Michael Kuhn, and Germán Fabregat. ArduPower: a low-cost wattmeter to improve energy efficiency of HPC applications. In *5th International Green & Sustainable Computing Conference*, Las Vegas, NV, USA, December 2015. To appear.

[24] LLC Allegro MicroSystems. ACS713: Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor. `http://www.allegromicro.com`, 2015. [Last access: June 2015].

[25] Atmel Corporation. ATmega2560: 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. `http://www.atmel.com/devices/atmega2560.aspx`, 2015. [Last access: June 2015].

[26] Jens Axboe. Flexible i/o tester. `http://freecode.com/projects/fio`.

[27] M. F. Dolz, J. Kunkel, K. Chasapis, and S. Catalán. An analytical methodology to derive power models based on hardware and software metrics. *Computer Science - Research and Development*, 2015.