TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

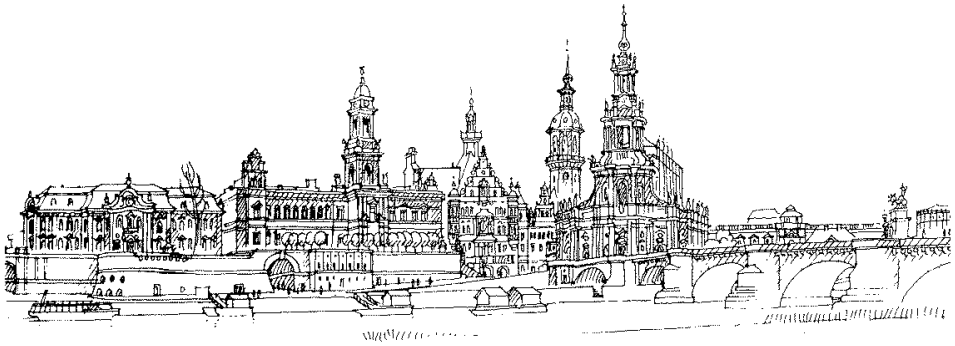
Technische Berichte
Technical Reports
ISSN 1430-211X

TUD-FI-11-06-Sept. 2011

J. Waltsgott, S. Götz, R. Fritzsche, S. Cech, C. Wilke

Institut für Software- und Multimediatechnik

State of the Art: Hardware Energy Management



Technische Universität Dresden
Fakultät Informatik
D−01062 Dresden
Germany
URL: http://www.inf.tu−dresden.de/

State of the Art: Hardware Energy Management

Johannes Waltsgott, Sebastian Götz, Ronny Fritzsche,
Sebastian Cech and Claas Wilke

Technische Universität Dresden, Dresden, Germany

Abstract. The CoolSoftware project focuses on optimizing software's energy consumption due to energy auto-tuning at runtime. Our vision also relies on energy management of hardware components. This report summarizes the results of our literature analysis w.r.t. current hardware energy management technologies. It outlines current approaches of energy management and relates them to CoolSoftware.

1 Introduction

The CoolSoftware[1] project focuses on building an energy auto-tuning runtime environment for software components [11]. Initially, a comprehensive literature analysis covering related work was performed. This report presents a summary of the analysis' results in the field of hardware energy management technologies. Further results w.r.t. energy optimization of storage systems and aspects of software's energy consumption can be found in [10] and [26].

This document is structured as follows: First, an introduction on basic energy management industry standards is given in Section 2. Second, selected technologies and scientific approaches for energy management of computer systems are presented in Section 3. Section 4 discusses the presented approaches w.r.t. their relationship to the CoolSoftware project. Finally, Section 5 summarizes and concludes this report.

2 Industry Standards for Energy Management

First, an overview of hardware energy management technologies is given. These technologies are basic industry standards and manufacturer independent. This section focuses on Advanced Power Management (APM) and Advanced Configuration and Power Interface (ACPI).

2.1 Advanced Power Management (APM)

APM is an energy management standard developed by Intel and Microsoft in the early 1990s. It enables the operating system (OS) to access the BIOS of a Personal Computer via an APM driver to manage its power states. Therefore,

[1] http://www.cool-software.org/

APM defines five system power states (*Full On*, *APM Enabled*, *APM Standby*, *APM Suspend*, *Off*) and some device power states (*On*, *Power Managed*, *Low Power*, *Off*) for APM-aware hardware devices. By calling defined power management functions, an APM driver (part of the OS) can communicate with the BIOS querying current power states or requesting power state transitions. The BIOS, in turn, uses power management events, which are polled by the APM driver, to inform the OS about changes. Figure 1 depicts the APM architecture.

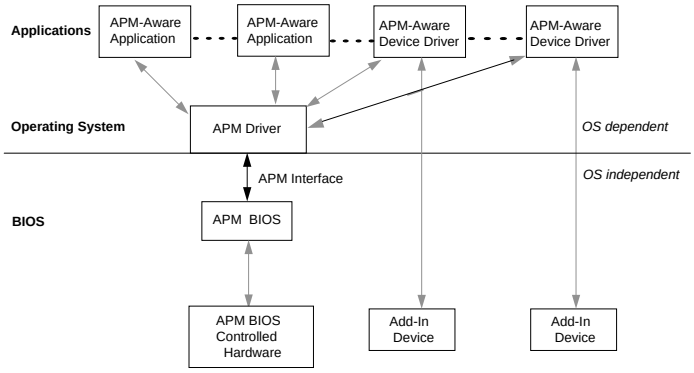


Fig. 1. APM general architecture [18]

Revision 1.2 [18] of the APM specification was released in 1996. Today, APM has been succeeded by ACPI.

2.2 Advanced Configuration and Power Interface (ACPI))

ACPI is an open industry power management specification co-developed by Hewlett-Packard, Intel, Microsoft, Phoenix and Toshiba. The current Revision 4.0a was released in 2010 [13]. ACPI succeeds the former APM power management standard and focuses on OS-based power management. Figure 2 depicts the general ACPI architecture. It consists of three parts: the ACPI registers, the ACPI BIOS and the ACPI tables. The ACPI registers are part of the system's hardware and allow manipulating the hardware properties. The registers' locations are described by the ACPI System Description Tables. The ACPI BIOS is part of the system's firmware. It boots the system and implements common interfaces for operations like *sleep*, *wake*, *restart* etc. Furthermore, it provides the ACPI Description Tables, which contain procedures to manipulate the ACPI registers encoded in the ACPI Machine Language (AML). The OS communicates with the ACPI system via an OS-specific ACPI driver, which contains an AML interpreter to call the AML procedures stored in the ACPI Tables.

For power management, ACPI relies heavily on several states which are defined for single devices or the entire system. The Global System State Definitions

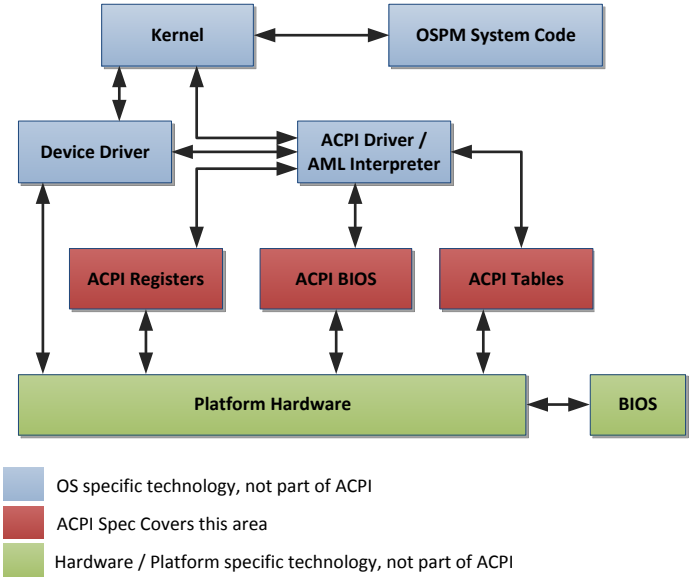


Fig. 2. ACPI overall architecture [13], redrawn

differ between application software running on the system or not, the systems latency to response external events, the power consumption and methods required to return to a working state. There are four global states in ACPI, namely *G0 Working*, *G1 Sleeping*, *G2/S5 Soft Off* and *G3 Mechanical Off*. Similar states are defined for particular devices, that are *D0 Full-On*, *D1*, *D2*, and *D3 Off*. There exists another state *D3 hot*, which is alike to *D3 Off*, but the device remains accessible by software. The specific behavior of the *D1* and *D2* states is defined by each device class.

Within the global sleep state *G1*, ACPI defines several sleeping states (*Sx*). The most common ones are the *S3* state, where the CPU is powered off and starts from its reset vector after wake up while the memory context is preserved by hardware, and the *S5* state, where the whole system is powered down and a memory image is saved on a nonvolatile storage to restore the system's state on the next boot.

For maintaining the CPUs power consumption and thermal management, ACPI defines specific processor power states (*C0* - *C3*) within the global working state *G0*, where *C0* is the full working state. Further states differ regarding the time it takes to return to *C0*. Furthermore, in *C3* the OS is responsible to ensure the processor's cache coherency.

In addition, for processor and device working states (*C0* / *D0*), more performance states (*P0* - *Pn* exist, where *n* is device-specific). *P0* represents the highest performance state with the possibly highest energy consumption. Moreover, the performance and energy consumption are declining for an increasing number

n of states. In summary, the ACPI specification defines an industry-accepted approach to shift responsibility for power management into the operation system and is deployed widely in nearly all contemporary computer systems. At the moment, the ACPI revision 5 is under development [14].

3 Selected Energy Management Approaches in Research and Technology

In this section, we will focus on selected technologies and research to maintain power consumption of computer systems. Starting with energy management at server components level, we will cover industry technologies as well as research focusing on CPU, memory and storage components. Finally, a brief overview of energy management of embedded systems is given.

3.1 Energy Management at Server Components Level

CPU Power Management Technologies Current PC system processors feature flexible power management technologies. The Intel SpeedStep technology [17], introduced with the Intel Mobile Pentium III, enables the system to adjust processor voltage and core frequency dynamically. Firstly introduced in mobile processors, the CPU detected whether the laptop runs on mains or on battery. Therefore, changes to *Px-states* (Section 2.2) provide lower core frequencies, which allows lower core voltage on battery operation. Besides the autonomic mode, the speed-stepping can also be triggered by the OS (and thus, via third-party applications) to react on context conditions regarding the current workload or the user's needs. Today, Intel SpeedStep technology is integrated in nearly all current Intel processors, including mobile and desktop models.

AMD offers technologies similar to Intel SpeedStep: Cool'n'Quiet [1] for desktop processors, PowerNow! [2] for mobile processors and Optimized Power Management for Opteron server processors.

Besides the former mentioned industry techniques, Huang et.al. [15] focus on processor adaptation by cache and register reconfiguration. They propose an new positional approach for adaptation which focuses on the code position (with the granularity of subroutines) rather than the execution time as in common temporal work. Therefore, they introduced three implementations of their adaptation, regarding different workload environments, providing a higher efficiency than temporal schemes. The evaluation showed an average increase in energy savings by 50 %. The concrete savings depend highly on the actual Low Power Techniques of the used CPUs.

Winter et al. focus on thread scheduling algorithms for large heterogeneous many-core CPUs (up to 256 cores) [27]. They performed a study on scheduling and power management algorithms regarding performance, power, sampling, requirements and runtime overhead. Furthermore, they did a formal analysis of the respective computational complexities. Their experimental assessment of coordination between many-core scheduling and power management algorithms

showed that coordination is not necessary and complexity can be reduced. They suggest the *Parallel Hierarchical Hungarian Algorithm* for thread scheduling and the *Steepest Drop* algorithm for global power management which perform up to 150 times faster than former algorithms.

Although, operating systems are capable of CPU power management (cf. ACPI, Section 2.2), heuristics for effective management of CPU power states are still evolving. Bircher and John [6] did an analysis of indirect and direct performance effects of different P-/C-states of AMD quad-core CPUs considering two different workloads (compute-bound / memory-bound), both with fixed scheduling (thread fixed to a specific core) as well as normal (OS) scheduling. As a result of the workload power characterization, the CPU(s), the memory controller and the memory modules turned out as the largest energy consumers having the highest variability. For memory-bound server tasks they highlighted memory power consumption exceeding even CPU power consumption. Furthermore, they exposed slow operating system transitions from idle to active causing performance losses, and slow active to idle transitions reducing energy efficiency. For selected operating system and CPU parameters (timers for P-/C-states) they proved power savings of 45 % while minimizing performance loss to less than 10 %.

Memory Following the CPU-specific approaches, this Section focuses on energy management for memory components in computer systems.

Schmidt and Wehn did a critical analysis of several power management strategies for DRAM [24], since they differ from former SRAM models and strategies. Their main criticism is that current approaches propose an aggressive use of DRAM low power states, but do not model transition overhead in their power models. The results of their analysis are proven by a hardware controlled DRAM unit for ARM[2]-based systems. Finally, they proposed a new model for accurate timing and energy simulation.

Cai and Lu periodically adjusted the size of physical memory and the timeout value to shut down a hard disk for reducing the average power consumption [7]. As a result, they proposed the use of Pareto[3] distributions for modeling the distribution of the idle time, with parameters of distribution adjusted at runtime, achieving power savings up to 60 %.

Li et al. [22] determined the limitations of former control algorithms for memory and disk power management, which are manual tuning of thresholds and the lack of performance guarantees. Their contributions include two new control algorithms with performance guarantees. The *Performance-Directed Dynamic* algorithm features dynamically adjustable thresholds, but is limited to memory management due to complexity. The *Performance-Directed Static* algorithm is a simple threshold-free control algorithm. Furthermore, they introduced a hybrid scheme, combining both algorithms achieving double-digit percent power savings.

[2] Advanced RISC Machines.
[3] A power law probability distribution.

3.2 Energy Management at Server Level

This Section outlines articles focusing on energy management at server level or cross-component and device-independent power management.

Li et al. accomplished the first approach to develop algorithms to jointly control adaptations in multiple interacting hardware components, as for example: CPU and memory [21]. Initially, they performed a separate analysis of CPU and memory optimization algorithms, although both adaptations influence each other. Under the assumption, that a user accepts a certain loss of performance to save costs (such as energy), they set up the requirement for an algorithm, which guarantees performance impact. A corresponding algorithm that assumes a specified target slowdown and seeks to minimize total CPU and memory energy consumption without exceeding this target was introduced and evaluated based on CPU and memory simulation, resulting in average energy savings of 46 % for joint adaptation.

Tolia et al. focus on energy optimization for Blade Server Closures by combining server power management with fan power management to optimize overall energy efficiency [25]. Therefore, they contribute Zephyr, a unified power and cooling management system. It combines distributed system design with heat transfer theory concepts, with the subsystems modeled by mathematical functions. They achieved up to 30 % lesser cooling power consumption and 29 % energy savings for the Blade enclosure.

Bianchini and Rajamony provide basic information on principle power management techniques for servers (and some parts on battery-operated devices), differing in local and cluster-wide techniques [4]. They identified the following future challenges: *power / energy modeling and prediction*, *exploiting service-level information*, *energy conservation for application servers* as well as the need for further research on memory, network interfaces and cluster interconnects.

Chase et al. performed an early analysis on resource management of (internet) hosting centers, with emphasis on energy [8]. The goal was to automattically adapt server resources to the current load by dynamically resizing the active server set. They suggested an economic approach, where servers "bid" for resources as a function of the delivered performance. A switching infrastructure directs incoming requests to servers assigned to the services. Using an experimental setup, they achieved energy reductions of 29 % for typical web workloads.

Lang et al. focus on server cluster and data-intensive workloads [20]. Their approach is to use the fact that some servers are underutilized and could be shut down. The arising problem is that local server data becomes unaccessible, when the server are shut down to save energy. Thus, the use of replication to keep data accessible is suggested. They studied the interaction between power management, load balancing and replication strategies and propose the use of Chained Declustering as replication strategy.

Proving that static power management strategies could lead to poor performance or even unnecessary power consumption, Ren et al. presented a hierarchical scheme for adapative Dynamic Power Management (DPM) under non-stationary service requests [23]. These requests were modeled as Markov-

modulated processes with a set of modes (refering to particular stationary requests). Energy-optimal DPM policies are precalculated offline using standard algorithms for stationary Markov decision processes.

An early analysis of power-optimized system design was performed by Benini and de Micheli [3], accounting for software and hardware components. They identified three types of hardware components: computation units, communication units and storage units. Furthermore, they accounted for three phases of system design: (1) conceptualization and modeling, (2) design and implementation, and (3) runtime management. They reviewed techniques for energy-efficient design of hardware and software for each mentioned phase.

3.3 Embedded Systems

This section gives a brief overview about articles focusing on energy management for embedded systems. Since embedded systems are very specific and slightly out-of-focus in the CoolSoftware project, the outline is skimped.

Bini et al. focused on minimizing energy consumption of embedded systems without violating time requirements [5]. They assumed that processors have discrete modes with defined speed and energy consumption. Further, they modeled tasks with a speed-scale part and a fixed time part (I/O part). This task model and the energy model were implemented as functions. Finally, a function calculates the minimum required frequency for a task and a suitable processor mode is selected.

Further research regarding predictive power management for mobile embedded devices can be found in [16]. Cho et al. focus on power management of embedded devices with application-specific energy cost functions [9].

4 Discussion

The following section discusses the influence of outlined related work on CoolSoftware. First, we will point out our project vision. Later, we will oppose the related work to our approach.

The CoolSoftware project aims for an energy auto-tuning runtime environment for software components in the field of heterogeneous server infrastructures. We focus on three layers, which are hardware (system infrastructure), software (application landscape) and user requirements [11]. Therefore, every layer is represented by its own managers. Energy optimization is going to be achieved by an adaptive distribution of software components upon the system infrastructure and an intelligent management of the hardware resources in use. In addition, user requirements regarding Quality-of-Service of applications are considered. We model the system infrastructure as hierarchical resources, each encapsulated by *Resource Managers* [12]. Resource Managers are responsible for managing and monitoring hardware resources. They propagate and control the energy states provided by these devices. Thereby, resources can be forced into different energy states (e.g., trigger distinctive energy modes, or turn resources

off and on). The energy optimization of the resources within these states is done by the resources themselves. In the field of hardware power management, the presented approaches reveal several points of contact for CoolSoftware.

4.1 Hardware Management at Component Level

Our approach is highly supported by the presented global industry standard for power management ACPI (Section 2.2), which is part of the operating system and controls hardware components according to utilization and efficiency. Furthermore, we highly rely on ACPI management and industry standards such as Intel SpeedStep Technology / AMD Cool'n'Quiet & Power!Now (cf. Section 3.1) for the self-management of the CPUs. Other presented work on CPU energy management will influence our project more generally, since it addresses very specific systems and optimization strategies. For resources such as hard disks, the resource manager can trigger different energy states with, e.g., lower rotation speed (cf. [10]).

4.2 Global Energy Management Strategies

The outlined papers in these field of research highlighted the strong correlation between distinctive power management strategies, such as CPU and memory management or relations between energy and thermal management of servers. However, the majority of those approaches focuses on specific systems (e.g., Blade Closures) or workloads (Internet hosting), which dispute the heterogeneous vision of CoolSoftware, so we will consider them in general.

4.3 Provision for User Utility

Many of the presented approaches emphasize the provision of user utility for power management strategies, since isolated energy management can lead to poor performance. The focus on user requirements is fundamental to CoolSoftware. Therefore, we define energy efficiency as the ratio of utility and the implied energy consumption for a user request. The dependencies between software components and hardware resources to deliver a specific utility to the user are described by our *Energy Contract Language (ECL)* [11] and considered by our runtime environment to select the best system configuration for a given user request within a specific system context (currently performance and utilization).

5 Summary

In this paper the state of the art and the latest research results in the field of hardware energy management were outlined. Many presented approaches and standards are fundamental to CoolSoftware, but most of them focus on power management for specific devices only, which is comprehensible since energy is "consumed" by single devices. Nevertheless, we emphasize on a global energy

optimization at application level as software components run on hardware resources and highly influence their energy consumption. Furthermore, we consider user requirements such as utility for our approach to energy optimization. Finally, the presented work highlights the strong urgency of improved power management for the design of future systems and applications, since energy demand is rising faster [19].

6 Acknowledgments

The project CoolSoftware is part of the Leading-Edge Cluster Cool Silicon, which is sponsored by the Federal Ministry of Education and Research (BMBF) within the scope of its Leading-Edge Cluster Competition. We would like to thank our advisors Prof. Aßmann and Prof. Meißner and the Silicon Saxony e.V.

References

1. Advanced Micro Devices, Inc. AMD Cool'n'Quiet Technology. http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx, 2010.
2. Advanced Micro Devices, Inc. AMD PowerNow! Technology. http://www.amd.com/us/products/technologies/amd-powernow-technology/Pages/amd-powernow-technology.aspx, 2010.
3. Luca Benini and Giovanni de Micheli. System-level power optimization: techniques and tools. *ACM Trans. Des. Autom. Electron. Syst.*, 5(2):115–192, 2000.
4. Ricardo Bianchini and Ram Rajamony. Power and Energy Management for Server Systems. *Computer*, 37(11):68–74, 2004.
5. Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari. Minimizing CPU Energy in Real-Time Systems with Discrete Speed Management. *ACM Trans. Embed. Comput. Syst.*, 8(4):1–23, 2009.
6. W. Lloyd Bircher and Lizy K. John. Analysis of Dynamic Power Management on Multi-Core Processors. In *ICS '08: Proceedings of the 22nd annual international conference on Supercomputing*, pages 327–338, New York, NY, USA, 2008. ACM.
7. Le Cai and Yung-Hsiang Lu. Joint Power Management of Memory and Disk. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 86–91, Washington, DC, USA, 2005. IEEE Computer Society.
8. Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing Energy and Server Resources in Hosting Centers. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 103–116, New York, NY, USA, 2001. ACM.
9. Youngjin Cho, Naehyuck Chang, Chaitali Chakrabarti, and Sarma Vrudhula. High-Level Power Management of Embedded Systems with Application-Specific Energy Cost Functions. In *DAC '06: Proceedings of the 43rd annual Design Automation Conference*, pages 568–573, New York, NY, USA, 2006. ACM.
10. Ronny Fritzsche, Johannes Waltsgott, Sebastian Götz, Claas Wilke, and Sebastian Cech. State of the Art: Optimization of Energy Consumption of Storage Systems. Technical Report, Technische Universität Dresden, 2011.

11. Sebastian Götz, Claas Wilke, Matthias Schmidt, Sebastian Cech, and Uwe Aßmann. Towards Energy Auto Tuning. In *Proceedings of First Annual International Conference on Green Information Technology (GREEN IT)*, 2010.
12. Sebastian Götz, Claas Wilke, Matthias Schmidt, Sebastian Cech, Johannes Waltsgott, and Ronny Fritzsche. THEATRE Resource Manager Interface Specification v. 1.0. Technical Report, Technische Universität Dresden, 2010.
13. Hewlett-Packard, Intel, Microsoft, Phoenix Technologies, and Toshiba. Advanced Configuration and Power Interface Specification, Revision 4.0a. http://www.acpi.info/spec.htm, April 2010.
14. Hewlett-Packard, Intel, Microsoft, Phoenix Technologies, and Toshiba. Advanced Configuration and Power Interface Specification, Revision 5.0. `http://www.acpi.info/Dev_50.htm`, 2010.
15. Michael C. Huang, Jose Renau, and Josep Torrellas. Positional Adaptation of Processors: Application to Energy Reduction. In *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, pages 157–168, New York, NY, USA, 2003. ACM.
16. Young-Si Hwang, Sung-Kwan Ku, Chan-Min Jung, and Ki-Seok Chung. Predictive Power Aware Management for Embedded Mobile Devices. In *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 36–41, Los Alamitos, CA, USA, 2008. IEEE Computer Society Press.
17. Intel Corporation. Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor. Whitepaper. ftp://download.intel.com/design/network/papers/30117401.pdf, March 2004.
18. Intel Corporation, Microsoft Corporation. Advanced Power Management (APM) BIOS Interface Specification. Revision 1.2. `http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/APMV12.rtf`, Feb 1996.
19. International Energy Agency (IEA). Key World Energy Statistics 2010, 2010.
20. Willis Lang, Jignesh M. Patel, and Jeffrey F. Naughton. On Energy Management, Load Balancing and Replication. *SIGMOD Rec.*, 38(4):35–42, 2010.
21. Xiaodong Li, Ritu Gupta, Sarita V. Adve, and Yuanyuan Zhou. Cross-Component Energy Management: Joint Adaptation of Processor and Memory. *ACM Trans. Archit. Code Optim.*, 4(3):14, 2007.
22. Xiaodong Li, Zhenmin Li, Yuanyuan Zhou, and Sarita Adve. Performance Directed Energy Management for Main Memory and Disks. *Trans. Storage*, 1(3):346–380, 2005.
23. Zhiyuan Ren, Bruce H. Krogh, and Radu Marculescu. Hierarchical Adaptive Dynamic Power Management. *IEEE Transactions on Computers*, 54:409–420, 2005.
24. Daniel Schmidt and Norbert Wehn. DRAM Power Management and Energy Consumption: a Critical Assessment. In *SBCCI '09: Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design*, pages 1–5, New York, NY, USA, 2009. ACM.
25. Niraj Tolia, Zhikui Wang, Parthasarathy Ranganathan, Cullen Bash, Manish Marwah, and Xiaoyun Zhu. Optimal Fan Control for Thermal Management of Servers. In *Proceedings of the ASME/Pacific Rim Electronic Packaging Technical Conference and Exhibition (InterPACK '09)*, San Francisco, CA, July 2009.
26. Claas Wilke, Sebastian Götz, Sebastian Cech, Johannes Waltsgott, and Ronny Fritzsche. Aspects of Software's Energy Consumption. Technical Report, Technische Universität Dresden, 2011.

27. Jonathan A. Winter, David H. Albonesi, and Christine A. Shoemaker. Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-Core Architectures. In *PACT '10: Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 29–40, New York, NY, USA, 2010. ACM.