A Recipe for Efficiency? Some Principles of Power-aware Computing

Parthasarathy Ranganathan

HP Laboratories
HPL- 2009-294

Abstract:
Power and energy are emerging to be key design considerations across a spectrum of computing solutions, from supercomputers and datacenters, to handhelds and mobile computers. There has been a large body of work on managing power and improving energy efficiency, all of which can be summarized in two words – "avoid waste!" The challenge, however, is in figuring out where and why the waste happens, and identifying how to avoid the waste. This article addresses this challenge. We argue that at a high level, many inefficiencies (waste) stem from the inherent way in which we currently address complex tradeoffs in the system design process. We discuss the common design practices that lead to power inefficiencies in typical systems and provide an intuitive categorization of high-level approaches to addressing these. Our hope is that this position paper provides practitioners, whether they are in systems, packaging, algorithms, user interfaces, databases, or any other area, a set of tools (a "recipe") to systematically reason about and optimize power in their respective domains.

External Posting Date: September 21, 2009 [Fulltext]
Internal Posting Date: September 21, 2009 [Fulltext]

Approved for External Publication

To be published in Communications of the ACM, Feb 2010.

A Recipe for Efficiency? Some Principles of Power-aware Computing

Parthasarathy Ranganathan
Hewlett Packard Labs
Palo Alto CA

Power and energy are emerging to be key design considerations across a spectrum of computing solutions, from supercomputers and datacenters, to handhelds and mobile computers. There has been a large body of work on managing power and improving energy efficiency, all of which can be summarized in two words – "avoid waste!" The challenge, however, is in figuring out where and why the waste happens, and identifying how to avoid the waste. This article addresses this challenge. We argue that at a high level, many inefficiencies (waste) stem from the inherent way in which we currently address complex tradeoffs in the system design process. We discuss the common design practices that lead to power inefficiencies in typical systems and provide an intuitive categorization of high-level approaches to addressing these. Our hope is that this position paper provides practitioners, whether they are in systems, packaging, algorithms, user interfaces, databases, or any other area, a set of tools (a "recipe") to systematically reason about and optimize power in their respective domains.

1. Why is power management important?

If you are a user of any kind of computing device, chances are that you have a personal anecdote to share about the importance of power management! Power management helps control the electricity (energy) consumed by a computing device. On mobile devices, this directly translates to how long the battery lasts. The battery is often the largest and heaviest component of the system, and improved battery life can also enable smaller and lighter devices. Additionally, with the increasing convergence of functionality on a single mobile device (E.g., phone + mp3 player + camera + web browser), the battery life becomes a key constraint on the utility of a mobile device. Indeed, longer battery life is often the highest-ranked metric in user studies on requirements for future mobile devices, even trumping increased functionality.

Power management is also important for tethered devices (devices that are connected to a power supply). The electricity consumption of computing equipment in a typical household can run to several hundreds of dollars per year. This cost is an even greater problem in enterprises. For example, servers that run Google's data centers have been estimated to consume millions of dollars in electricity costs per year [1]. Last year alone, IDC estimates that the total worldwide spending on power management for enterprises was a staggering $40 billion [1]. Increased power consumption can also lead to increased complexity in the design of the power supplies (and power distribution and backup units in larger systems) which can also add costs.

One of the other challenges from power consumption in systems is the heat generated, and consequently, the term power management is also used to also refer to the heat management in systems. Such heat generated is often an even bigger problem than the amount of electricity spent. To avoid the heat from affecting the user or impacting the electronics in the system, ever more complex thermal packaging and heat extraction solutions are required in systems, adding costs. For large systems like supercomputers or datacenters, these costs can often lead to an additional dollar spent on cooling for every dollar spent on electricity, (captured in a metric that the Green Grid calls

PUE or Power Usage Effectiveness [9]). The heat dissipation in systems can also have additional implications on the compaction and density of computing systems – for example in blade server configurations. Additionally, studies have shown that operating electronics at temperatures higher than their operational range can lead to significant degradation of reliability. For example, Uptime Institute identifies 50% increased chances of server failure for a ten degrees increase over 20C [10]; similar statistics have been shown for hard disk lifetime as well [11,12].

Finally, power management in computing systems has key environmental implications. It has been estimated that the computing equipment in the United States alone consumes 22 million GigaJoules of energy per year, or an equivalent of 4 million tons of carbon-dioxide emissions into the atmosphere [1]. Governmental agencies have identified energy consumption implications on air quality, national security, climate change, and electricity grid reliability, and there are several initiatives worldwide from both governmental agencies (e.g., EPA in the U.S., Intelligent Energy Europe, Market Transformation Program in the U.K., TopRunner in Japan) and industry consortiums (e.g., SPEC, GreenGrid, TPC) on improving energy efficiency (minimizing the amount of energy consumed for a given task).

As we look to the future, the importance of power management is only likely to increase. On mobile devices, the gap between advances in battery capacity and anticipated increases in mobile device functionality is growing. New battery technologies like fuel cells might address this gap, but it is still going to be important to design more power-efficient systems. Energy review data from the Energy Information Administration (EIA) point to steadily increasing electricity costs. Indeed, for datacenters, several reports indicate that costs associated with power and cooling can easily overtake hardware costs [2, 13]. Increased compaction, such as in future predicted blade servers, is estimated to increase power densities by an order of magnitude. Such increased densities start hitting the physical limits of practical air-cooled solutions. There is ongoing research into alternate cooling technologies (e.g., efficient liquid cooling), but it will still be important to be more efficient about generating heat in the first place. All of this requires better power management.

2. What are we doing about this problem?

There has been a lot of prior work looking at power management and energy efficiency. Figure 1 provides a brief overview of some key concepts in the literature categorized across different levels of the solution stack – in process technology and circuits, in architecture and platforms, and in applications and systems design. A detailed discussion of all these specific optimizations is not the intent of this article and indeed, there are several tutorial articles (e.g., [1,3,4] and several conferences devoted purely to power (e.g., ISLPED, HotPower) that provide good overviews of the state-of-the-art in power management. Our goal, with the brief summary picture is to mainly point out that there is a rich body of prior work that has examined power management and energy efficiency. This work can be broadly categorized across different levels of the solution stack (e.g., hardware, software), at different stages of the lifecycle (e.g., design, run-time), at different components of the system (e.g., CPU, cache, memory, display, interconnect, peripherals, distributed systems) across different target domains (e.g., mobile devices, wireless networks, high-end servers) and different metrics (e.g., battery life, worst-case power). It is interesting to note that a lot of prior work has been at the electrical/computer engineering levels, with a relatively smaller fraction of work from the core computer science areas. The prior focus on power and energy challenges at the hardware and systems levels is of course very natural and central, but in the future, significant improvements in power and energy efficiency are likely from rethinking algorithms and applications at higher levels of the solution stack as well. Indeed, recent discussions on the future of power management have increasingly focused on this aspect [14, 15].

In spite of the seemingly rich diversity in the nature of all this prior work on power management, at a high level, there is really one common theme across all the solutions – "Avoid wasted energy". Where the solutions differ, is in the identification and intuition for specific sources of inefficiency, and the specific mechanisms and policies used to target those inefficiencies. This observation raises some interesting questions. Are there any general recurring high-level trends that lead to these inefficiencies at different levels of the system? Are there any common recurring high-level approaches that get customized in the context of specific scenarios? Knowing the answers to these two questions can enable the beginnings of a structure to think about power management in a more systematic manner and potentially help identify opportunities for energy efficiency beyond traditional platform-centric domains to more solution and application-centric domains. The rest of this paper examines the answers to these two questions.

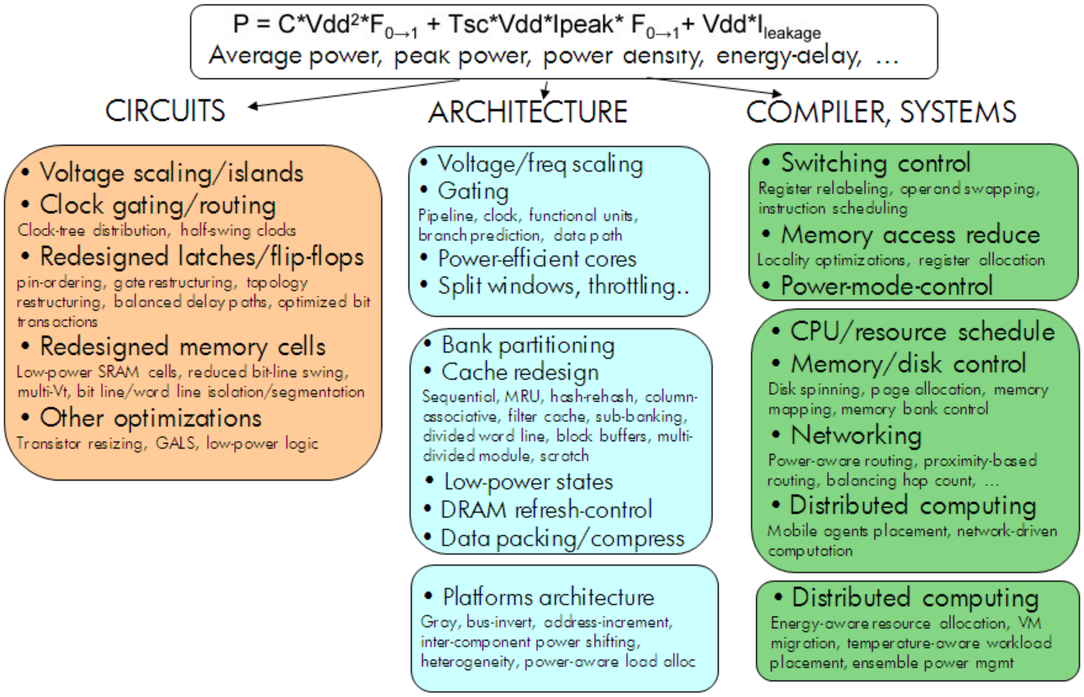
$P = C*Vdd^2*F_{0\rightarrow1} + Tsc*Vdd*Ipeak*F_{0\rightarrow1} + Vdd*I_{leakage}$
Average power, peak power, power density, energy-delay, …
CIRCUITS
ARCHITECTURE
COMPILER, SYSTEMS
• Voltage scaling/islands
• Clock gating/routing
Clock-tree distribution, half-swing clocks
• Redesigned latches/flip-flops
pin-ordering, gate restructuring, topology restructuring, balanced delay paths, optimized bit transactions
• Redesigned memory cells
Low-power SRAM cells, reduced bit-line swing, multi-Vt, bit line/word line isolation/segmentation
• Other optimizations
Transistor resizing, GALS, low-power logic
• Voltage/freq scaling
• Gating
Pipeline, clock, functional units, branch prediction, data path
• Power-efficient cores
• Split windows, throttling..
• Bank partitioning
• Cache redesign
Sequential, MRU, hash-rehash, column-associative, filter cache, sub-banking, divided word line, block buffers, multi-divided module, scratch
• Low-power states
• DRAM refresh-control
• Data packing/compress
• Platforms architecture
Gray, bus-invert, address-increment, inter-component power shifting, heterogeneity, power-aware load alloc
• Switching control
Register relabeling, operand swapping, instruction scheduling
• Memory access reduce
Locality optimizations, register allocation
• Power-mode-control
• CPU/resource schedule
• Memory/disk control
Disk spinning, page allocation, memory mapping, memory bank control
• Networking
Power-aware routing, proximity-based routing, balancing hop count, …
• Distributed computing
Mobile agents placement, network-driven computation
• Distributed computing
Energy-aware resource allocation, VM migration, temperature-aware workload placement, ensemble power mgmt

Figure 1: Overview of some of the previous work on power management

3. Waste not! But where does the waste come from in the first place?

Intuitively, it is easy to imagine that there is a certain amount of minimum electrical energy that is needed to perform a certain task and a corresponding amount of minimum heat that needs to be extracted to avoid thermal problems. For example, Mayo et al [5] performed some simple experiments to measure the energy consumption for some common mobile tasks (e.g., listening to music, making a phone call, email and text messaging, web browsing) implemented on different devices (e.g., cell phones, MP3 players, laptops, PCs) and observed some interesting results. There is a significant difference in energy efficiency, often ten to hundred-fold, across different systems performing the same task. Of course, there are variations in user experiences across difference devices, but even when focusing on duplicating the exact functionality of the best-performing system, these experiments showed that it was impossible to do so at the same energy level on a different worse-performing system.

Why is it that some designs introduce so much more additional inefficiencies over and above the actual energy required for a given task? Our experience has been that these inefficiencies are often introduced when the system design has to reconcile complex tradeoffs that are hard to avoid. For example, systems need to be often designed for the most general-case, most aggressive workload performance, and worst-case risk tolerance. Such designs can lead to a lot of resource over-provisioning to better handle transient peaks and to offer redundancy in the case of failure. Additionally, individual components of a broader system are often designed by different teams (even by different vendors) without consideration of their use with one another. Individual functions of a system are also designed modularly, often without factoring in their interactions with one another, adding further inefficiencies. Further, traditional designs have primarily focused on system performance. This approach has sometimes led to highly resource-wasteful designs to extract small improvements in performance; with the recent emphasis on energy costs, often, these small improvements in performance are overshadowed by the costs of power and heat extraction. On a similar note, additional inefficiencies are introduced when the system design takes a narrow view of performance (versus focusing on actual end-user requirements) or do not adequately address total costs of ownership including design and operational costs.

Below, we list some common design practices that lead to potential sources of inefficiencies in the system and discuss these in greater detail. The section after this discussion addresses how to address the inefficiencies.

General-purpose solutions for different types of applications

General-purpose systems often provide better consumer experiences. For example, most users find it preferable to carry one converged mobile device versus carrying several separate devices (e.g., phone, camera, MP3 player). Additionally, the exigencies of volume economics also motivate vendors to develop general-purpose systems; a product that sells in the millions is usually cheaper to make than, say, an equivalent product that sells in the hundreds.

General-purpose systems, by definition, need to be designed to provide good performance for a multitude of different applications. This requirement results in designers using the “union” of maximum requirements of all the application classes when designing their systems. For example, a laptop that targets a DVD-playback application might incorporate a high-resolution display and a powerful graphics processor. When the same laptop is used for another task such as say, reading email, the high-power characteristics of the display and the graphics processor may not be needed. However, when the laptop is designed for these two workloads, most designs typically ship with a display that has the characteristics of the most aggressive application usage (in this case, a high-resolution display that can play DVD movies well). Without adequate design thought into how energy consumption can be adapted to different kinds of tasks, such a design approach often leads to significant power inefficiencies in the system. Another example is in the datacenter space where optimizing for both mission-critical and non-mission-critical servers in the same facility can lead to significant inefficiencies for cooling costs. Similar situations occur when legacy solutions have to be supported on newer systems.

Planning for peaks and future growth

Most workloads go through different phases when they require different performance levels from the system. For example, previous studies have reported that the average server utilization in several live datacenters can be quite low (10-30%). Mobile systems have also been reported to spend a significant fraction of their time in idle mode or using a small fraction of their resources.

However, most benchmarks on the basis of which systems are designed are typically structured to stress worst-case performance workloads irrespective of how the system is likely to be used in practice. Consequently, many systems are optimized for the peak-performance scenario. In the absence of designs that proportionally scale their energy with resource utilizations, significant inefficiencies can result. For example, many power supplies are optimized for peak conversion efficiency at high loads. When these systems are operated at low loads, the efficiency of conversion can drop pretty dramatically, leading to power inefficiencies.

Similar over-provisioning happens when planning for the future. Most computing systems are designed for 3-5 year depreciation cycles, and in the cases of larger installations like datacenters, even longer lifetimes. Systems need to be designed to ensure that sufficient capacity is designed into the system to meet incremental growth needs. On many systems, such over-provisioning again leads to inefficiencies when the system is not operating at the resource utilization capacities that account for future growth. For example, a datacenter with 1MW of cooling operating, but operating only at 100KW capacity will be more inefficient than an equivalent datacenter operating at 100KW with, say, 150 KW of cooling.

Design process structure

Current system design approaches follow a structured process. System functionality is divided across multiple hardware (CPU, chipset, memory, networking, and disk in a single system, or different individual systems in a cluster) and software components (firmware, virtualization layer, operating systems, applications, etc). Even within a component, there are often multiple layers with well-defined abstractions and interfaces (e.g., the networking stack). Power management is usually implemented within these well-defined blocks, but often without consideration for the interaction across the layers. However such modular independent designs or local optimizations can be sub-optimal for global efficiency without communication across layers. An insidious problem is when each layer of the stack makes worst-case assumptions about other layers of the stack leading to a compounding of inefficiencies.

Information exchange across layers can often enable better power optimizations. For example, a power management optimization at the physical layer of a wireless communication protocol that is aware of higher-level application activity can be more efficient than one that is oblivious to higher-level application activity. Similarly, several a power management solution that optimizes at an ensemble level (e.g., across different components in a system or across different systems in a cluster) can be more efficient.

Similar problems exist at other boundaries of the system architecture. For example, the power management of servers is handled by the IT department while the cooling infrastructure is often handled by a separate facilities department. This organizational structure can lead to inefficiencies as well. For example, a cooling solution that is aware of the non-uniformities in power consumption (and consequent heat generation) can be more efficient than one that is not.

Such inefficiencies due to layering can also be found at other places in the overall solution architecture. For example, in a classic client-server architecture, selectively exchanging information between the clients and servers has been shown to be beneficial for energy optimizations at both levels.

The Tethered-System Hangover of ignoring power for peak performance

A final design practice that leads to inefficiencies is what we refer to as the "tethered-system hangover". These inefficiencies are mainly a reflection of the relentless drive to achieve higher performance, often at the assumption that there is no constraint on power, and are particularly driven by tethered systems with no immediate considerations of battery life, etc. For example, historically, many processor architecture designs have included optimizations that achieved incremental performance improvements inconsistent with the amount of additional power consumed to implement those solutions. Similar tradeoffs can be identified in designs for high availability at the expense of energy (e.g., triple modular redundancy that runs three concurrent executions of the same task to avoid any downtime).

As additional examples, designs with user interfaces that identify the content of interest to the user and expend energy on those areas can be more energy-efficient than designs that simply focus on metrics like refresh rate, etc. Similarly, designs that focus on energy-delay may be significantly more energy efficient, but with only a marginal difference in performance from pure performance-centric designs. In general, several power inefficiencies in current systems stem from a design focus that does not sufficiently address the total costs of ownership and ultimate end-user experience, but instead focuses on one or more narrow metrics disproportionately.

Waste not! How do we reduce the waste?

The previous section discussed how power inefficiencies stem from assumptions inherent in current design practices. Once we identify these inefficiencies, the next step is to identify approaches to reduce them. Below, we discuss the main categories under which such approaches fall.

1. **Use a more power-efficient alternative:** Such approaches include replacing a system component with a more power-efficient alternative that performs the same task with lower energy. For example, a disk drive can be replaced with more energy-efficient non-volatile memory, or optics can be used to replace conventional networking. Sometimes, a more power-efficient alternative might involve adding the right hooks to enable some of the other approaches discussed below. For example, replacing a display with a single backlight with an alternate display that provides more fine-grained control of power can, in turn, enable power optimizations that turn off unused portions of the display. Of course, choosing a power-efficient alternative often comes with other tradeoffs like costs or performance (otherwise, the design

would have used it in the first place!) and additional work may have to be done to address these tradeoffs.

2. **Create "energy proportionality" by scaling down energy for unused resources:** Such approaches involve turning off or dialing-down unused resources proportional to system usage, often referred to as energy proportionality [2] or energy scale-down [5]. This requires algorithms that can respond to the consequences of turning off or turning down a system (e.g., understanding how long it takes to bring the system back on again). Sometimes, if a single component or system does not have the option to be scaled-down, this optimization can be applied at the ensemble level. Examples of such ensemble-level scale-down include changing traffic routing to turn off unused switches or using virtual machine consolidation to coalesce workloads into a smaller subset of systems in a datacenter.

3. **Match work to power-efficient option:** This solution is complementary to the previous approach. Rather than having the resources adapt when they are not fully utilized for a given task, these approaches match tasks to the resources most appropriate to the size of the task. An example of such an approach is the intelligent use of heterogeneity to improve power efficiency (e.g., scheduling for asymmetric or heterogeneous multicore processors). Obviously, this approach implies that there is a choice of different resources for a given task. In cluster or multicore environments, such a choice naturally exists, but other designs can explicitly introduce multiple operation modes with different power-performance tradeoffs.

4. **Piggy back or overlap energy events:** Such approaches seek to combine multiple tasks into one energy event. For example, coalescing multiple reads on a single disk spin can reduce total disk energy. Prefetching data in predictable access streams or using a shared cache across multiple processes are other examples where such an approach provides energy savings. Disaggregating or decomposing system functionality into smaller sub-tasks can help increase the benefits from energy piggy backing by avoiding duplication of energy consumption for similar sub-tasks across different larger tasks.

5. **Clarify and focus on the required functionality:** These approaches design the solutions specific to the actual constraints on the design without trying to be too general-purpose or future-proof. For example, special-purpose solutions such as graphics processors can be more energy efficient for the workloads they are targeted at. Similarly a design that seeks to provide future growth through addition of modular building blocks can be more energy efficient compared to a single monolithic future-proofed design.

6. **Cross layers and broaden the scope of the solution space:** Rather than individual solutions addressing power management at a local level, focusing on the problem holistically at the overall solution level is likely to achieve better efficiencies. Examples where such an approach has been shown to be effective include scheduling across an ensemble of systems or system components, and facilities-aware IT scheduling (e.g., temperature-aware workload placement). Exchanging information across multiple layers of the networking stack is another application of this approach that has been shown to be beneficial for energy efficiency.

7. **Trade off some other metric for energy:** These approaches see to achieve better energy efficiency by

marginally compromising some other aspect of desired functionality. An interesting example includes trading off fidelity in image rendering in DVD playback for improved player batter life. Optimizations that focus on improved energy delay where improvements in energy significantly outweigh degradations in delay also fall in this category.

8. **Trade off uncommon-case efficiency for common-case efficiency:** Some approaches seek to improve overall energy efficiency by explicitly allowing degradation in energy efficiency for the rare cases to improve the energy efficiency for the common cases. For example, a server power supply could be optimized for peak efficiency at normal light loads even if it leads to degraded power efficiency at infrequent peak loads.

9. **Spend somebody else's power:** Several approaches have taken a more local approach to energy efficiency, but at the expense of the energy-efficiency of a different system. For example, a complex computation in a battery-constrained mobile device can be offloaded to a remote server in the "cloud". This improves the energy efficiency of the mobile device. Approaches that scavenge energy, say from excess heat, or from mechanical movement to improve overall energy efficiency, also fall in this category.

10. **Spend power to save power:** A final category of approaches proactively perform tasks that address overall energy efficiency even though these tasks themselves may consume additional energy. Some examples include a garbage collector that periodically reduces the memory footprint to allow memory banks to be put in lower-power states, or a compression algorithm that enables lower energy for communication and storage.

The first five approaches have been well studied and are commonly prevalent in existing power optimizations. The other approaches are less common, but are likely to be increasingly important in the future. Obviously, combinations of these approaches are also possible.

Finally, irrespective of which of the above approaches are used to improve power efficiency, there are three key architectural elements in any solution for energy efficiency:

(1) a rich measurement and monitoring infrastructure

(2) accurate analysis tools and models that predict resource usage and identify trends and causal relationships, and provide prescriptive feedback

(3) control algorithms and policies that leverages the analysis above to meaningfully control power (and heat), ideally coordinated across layers.

Correspondingly, from a design point of view, system support is needed at all levels (hardware, software, application) to facilitate such measurement, analysis, and control and cross-layer information sharing and coordination.

Looking ahead

In spite of the large body of work on power management, we still have a long way to go. By way of illustration, Feynman estimates that, based on the physical limits on the power costs to information transfer [6], a staggering 10^{18} bit-ops/sec can be achieved for 1 watt of power consumption. In terms that are easier to relate to, this implies that we should be able to achieve the computational power of a *billion* Pentium processors in the power consumption of *one* typical handheld device. Obviously, this is a data point on the theoretical physics of energy consumption, but this bound still points to the tremendous potential for improvement of energy efficiency in current systems. Furthermore, when going beyond energy consumption in the operation of computing devices to the energy consumption in the both the supply and demand side of the overall IT ecosystem (from “cradle-to-cradle” [7]), the challenges and opportunities are enormous. We believe that energy efficiency of current systems can be improved by at least an order of magnitude by a systematic examination of current inefficiencies and a rethinking of current designs. In particular, in addition to the large body of work in the electrical and computer engineering community, we see a key role for a new emerging science of power management [14] across the broader computer science community. We hope that the discussions in this article --on the design practices that lead to common inefficiencies and the main solution approaches to address these --provide a starting framework to systematically help think about other new ideas in new domains that will help us achieve these significant improvements.

References

1. Enterprise Power and Cooling: A Chip-to-Data Center Perspective. Chandrakant Patel and Parthasarathy Ranganathan. s.l. : Hot Chips 19, 2007. http://www.hotchips.org/archives/hc19/.

2. The Case for Energy-Proportional Computing. Luiz André Barroso and Urs Hölzle. s.l. : IEEE Computer Society Press, December 2007. IEEE Computer. Vols. 40, Issue 12, pp. 33-37. ISSN:0018-9162 .

3. Power Management for Computer Systems and Datacenters. Karthick Rajamani, Charles Lefurgy, Soraya Ghiasi, Juan C Rubio, Heather Hanson, Tom Keller. 2008. International Symposium on Low Power Electronics and Design (ISLPED). http://www.islped.org/slides.html .

4. Low-power design: From soup to nuts. M. J. Irwin and N. Vijaykrishnan, 2000. Tutorial as part of the International Sympoisum on Computer Architecture.

5. Energy Consumption in Mobile Devices: Why Future Systems Need Requirements-Aware Energy Scale-Down. Robert N. Mayo, Parthasarathy Ranganathan:. 2003. Proceedings of the Workshop on Power-aware Computing Systems. pp. 26-40.

6. Feynman, Richard. Feynman Lectures on Computation. s.l. : Westview Press, 2000. 0738202967.

7. Dematerializing the Ecosystem. Patel, Chandrakant. s.l. : Keynote at 6th USENIX Conference on File and Storage Technologies, 2008.

8. Energy Management for Commercial Servers. Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, Tom W. Keller. 2003. IEEE Computer. Vols. 36, Issue 12, pp. 39-48. ISSN:0018-9162 .

9. The Green Grid, www.thegreengrid.org, The Green Grid Data Center Power Efficiency Metrics: PUE and

DCiE, 2007

10. R. F. Sullivan. Alternating Cold and Hot Aisles Provides More Reliable Cooling for Server Farms. In Uptime Institute, 2000.

11. D. Anderson, J. Dykes, and E. Riedel. More Than an Interface—SCSI vs. ATA. In Proceedings of the 2nd Usenix Conference on File and Storage Technologies (FAST), San Francisco, CA, March 2003.

12. G. Cole. Estimating Drive Reliability in Desktop Computers and Consumer Electronics. In Technology Paper TP-338.1, Seagate Technology, November 2000.

13. "Power could cost more than servers, Google warns", December 9, 2005, CNET news

14. National Science Foundation, Workshop on The Science of Power Management, April 2009

15. Power management from cores to datacenters: Where are we going to get the next 10X? Panel at International Symposium on Low Power Electronic Devices (ISLPED), 2008