Power Optimization – a Reality Check

Stephen Dawson-Haggerty
Andrew Krioukov
David E. Culler

Electrical Engineering and Computer Sciences
University of California at Berkeley

October 19, 2009

Power Optimization – a Reality Check

Stephen Dawson-Haggerty, Andrew Krioukov, David Culler
Computer Science Division
University of California, Berkeley
{stevedh,krioukov,culler}@cs.berkeley.edu

Abstract

A flurry of recent work has examined the interaction between system design and power optimization, with the promise of energy savings. We find that the power consumed by current commodity hardware can be well-approximated by a model with two components: a large always-present constant power and a linear power-performance tradeoff. As a result, many proposed optimizations currently have limited value and the best possible approach is the so-called "race to sleep" model of computation. We survey several recently-released computer systems in support of this conclusion and briefly examine the most important targets for future work.

1 Introduction

Power has become *the* critical resource in computer systems, whether we look at lifetime for portable devices or heat generation and data center efficiency at the high end. Leading microprocessor vendors are making huge investments to improve the power profile of their products, numerous operating system and application software enhancements are being announced and a veritable cottage industry of academic papers on power optimizations has emerged, addressing concerns at every level of the system stack. But all too often the results presented are not in terms of actual power saved or energy consumed, but in terms of proxies for power consumption [6], power consumed by specific subsystems [9, 10], or simulations based on idealized power models [4, 14]. There is good reason for this situation. Reducing the energy and power consumption of whole systems in reality is hard. A form of Amdahl's law is at work; if any portion of the system consumes power without optimization, it will come to dominate as the rest of the system improves. However, a reality check is healthy because it can cause us to focus on approaches and optimizations that really matter.
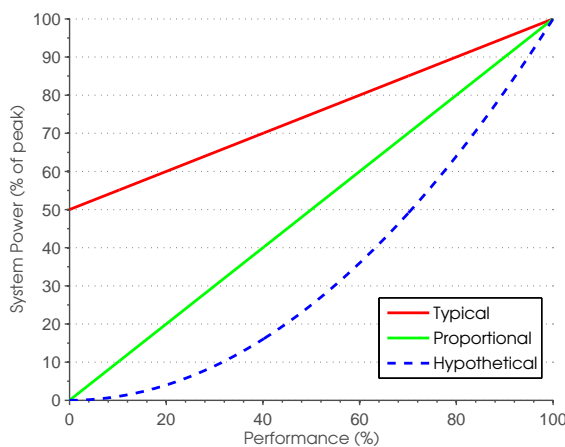

Figure 1: Potential power models

For example, if we assume an ideal scenario where the power consumed by the system is strictly proportional to performance delivered, zero power is consumed when idle and the transition from idle to active is instantaneous (illustrated in Figure 1) then almost none of the sophisticated power optimizations in the literature matter. For a given amount of work performed over a given amount of time, the total energy consumed, the average power consumed, even the total heat generated is the same whether we run as fast as possible and sleep for the rest, run as slow as possible and never sleep, or anything in between. We might as well just optimize for performance, run as fast as possible and then sleep as long as we can. Most power optimization proposals assume a super-idealized scenario where the power-per-performance curve is concave upward. Then there is a sweet spot where we run "just fast enough." In reality the picture is simpler and grimmer. The power-vs-

1

performance curve is nearly linear and it has a huge constant component.

With a constant power penalty at any level of performance, hurry-up-and-sleep is the best available optimization. But unfortunately the time to go to deep sleep and wake up is non-trivial. The whole system is hugely power sub-proportional. The attention paid to the microprocessor power optimization has yielded tremendous improvement in processors, but the power is being spent elsewhere. Until we attend to non-processing parts – the hardware and software "glue", the transition time, and the idle power – few of the HotPower optimizations will matter.

An indication of this reality check for the previous generation of servers that are currently deployed on the field is shown in Figure 3. The figure shows, for a variety of models, the power consumed when the machine was as idle as we could make it versus the power consumed when it is totally pegged. The gap is small: 10-25%. This fact is well understood by industry and substantial improvements have been made. Thus, we illustrate the reality check through study of two highly optimized complete systems recently introduced on the market. At one extreme, optimized for low power while maximizing performance to the extent possible, is the Atom 330 microprocessor and 945CG chipset introduced in June 2008. At the other, optimized for high performance while minimizing power consumption to the extent possible, is the Core i7 introduced in November 2008. Details of these two platforms are shown in Tables 1 and 2. The goal of this study is to understand where the power actually goes in modern, highly optimized but complete solutions and thereby to understand what system level optimizations hold real promise and which are purely academic. We develop a software suite that produces a relatively fine grained empirical power model of the underlying platform by subjecting it to a variety of workloads, observing the power consumed, and then computing a fit of the power model to the observations. This works surprisingly well and gives a clear and intuitive breakdown. The bottom line in the analysis is in fact indicated in Tables 1 and 2: the chipset is a dominant constant term. Until we can apply substantial power management to the glue, all we can do is hurry-up-and-sleep.

2 Hardware

To ground the power-vs-performance model in actual hardware, we examine available power states and

Idle states	C0 (run), C1 (halt)
Cores	2 + HT
Clock speed	1.6GHz
Core voltage	0.9V-1.1625V
Spec Sheet TDP	8W
Chipset TDP	22.2W

Table 1: Properties of the Intel Atom 330 and 945CG chipset

Idle states	C0 – C4
Cores	4 + HT
Clock speed	1.6GHz - 3.2GHz
Core voltage	0.800V-1.225V
Spec Sheet TDP	130W
Chipset TDP	24.1W
Radeon 2400XT TDP	25-35W

Table 2: Properties of the Intel Core i7 Extreme Edition and X58 chipset

conduct detailed experiments on two hardware platforms representing somewhat opposite ends of the spectrum: a low-power Intel Atom netbook platform, and an Intel Core i7 desktop.

2.1 Idle vs. Sleep

Idle states are power-saving modes that a device can enter quickly, without losing state and often automatically when idle. For example, a hard drive will use about about 0.85W when spinning but not performing I/Os, compared to 2.85W when reading sequentially. Similarly, the CPU has "C states" which it enters when halted. These idle states have varying exit latencies and power. However, transition times are very fast, even for the deepest states. On an Intel Core 2 processor the deepest idle can be entered in $17ns$, or approximately 50 clock cycles. Thus, CPU idle states can be used even for interactive applications.

System sleep states, on the other hand, reduce overall power to nearly zero by disabling almost all devices. RAM is usually kept active for fast wake-up times. For instance, the ACPI S3 state has a transition time of $1 - 10$ seconds and typically brings the system power usage to several watts [5].

2.2 Constant Power

Although modern processors have improved efficiency as designers coped with a constant thermal envelope,

processors are not the only component making up a computer; the supporting hardware such as the chipset controller, bus logic, and display drivers are not nearly as visible inside operating systems, yet have a significant impact on power.
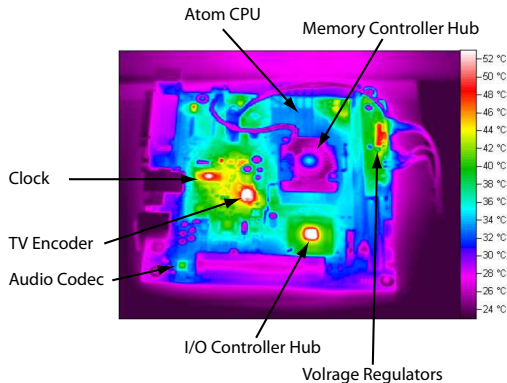

Figure 2: Thermal image of an Intel Atom motherboard while idle

Figure 2 is a thermal image of the Atom platform when idle. The processor itself is actually passively cooled and appears blue, while other components such as the I/O and memory controllers are clearly radiating much more heat then the processor.

A way of explicitly measuring how much power is dissipated regardless of the processing state of the computer is to record periodic wall-power measurements, and compare the power when idle to the observed power when heavily loaded. Figure 3 presents representative data from a previous generation of servers. Needless to say, the amount of dynamic power is very small in all cases, at most 25% of the total dissipated power.

2.3 Dynamic Power

To isolate where the power goes in our two test systems with Core i7 and Atom processors, as well as investigate the predictability of dynamic power, we heavily instrumented the systems by collecting periodic reports of various performance counters available on the system. In addition, we collected synchronized wall-power measurements. We then loaded the system with a series of workloads designed to push the system into different operating regimes. Using the measurements from these experiments, we construct a linear model of power as a function of the observed variables through linear regression. The resulting models have root mean squared errors (RMSE) of
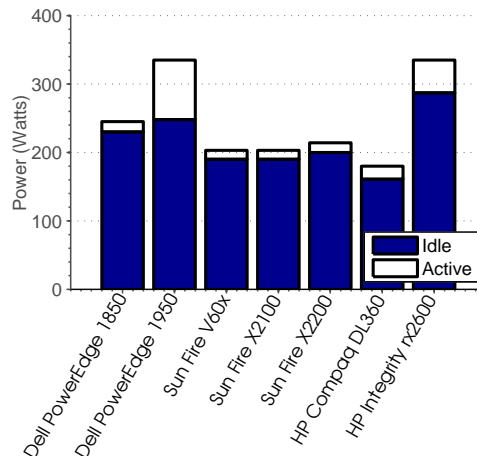

Figure 3: Peak *vs.* idle power consumption for a series of previous generation server hardware

1.16W and 2.17W for the Atom and Core i7 system respectively for our test applications. Once the model is constructed, it can be used to predict the power profile of arbitrary application based only on the counter values obtained on those applications.

Since the model is linear, we can assign power to each subsystem (ie. the disk), by considering only the elements of the measurement vector corresponding to that subsystem. As a result, the model can not only predict the total wall power in use by the system, but also provides a breakdown between the different components.

Results of this technique are displayed in Figure 4(b) and 4(a). Neither displays anything close to zero watts at idle. The high-performance i7 system is more power-proportional then the older servers considered in Figure 3 because the processor is able to substantially reduce its consumption when idle. Nonetheless, the constant factor is roughly equal to the peak processor consumption. Surprisingly, the Atom design is less power proportional because the processor is such a small fraction of the overall power load. In fact, both of their chipsets appear to be drawing nearly the full power indicated by their data sheets; this is in notable contrast to the CPUs. The various resources managed by the operating system, while actually managed well, are incidental in comparison.

To isolate the power-performance tradeoff curve for the processor, we examine the total system power dissipation as the processor frequency is scaled. As seen
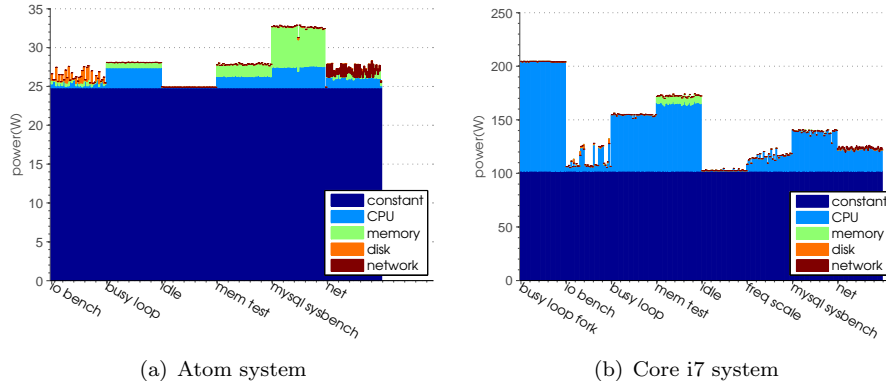
3

(a) Atom system

(b) Core i7 system

Figure 4: Power consumption per subsystem as predicted by a linear model

in Figure 5, the tradeoff is nearly linear which a large constant term. This observation is well-documented for modern processors because of reductions in supply voltage which eliminate the potential quadratic gains in power that were available when dynamic frequency and voltage scaling was proposed [3]. (If the voltage is held constant performance is proportional to frequency and energy per unit computational work is constant.)
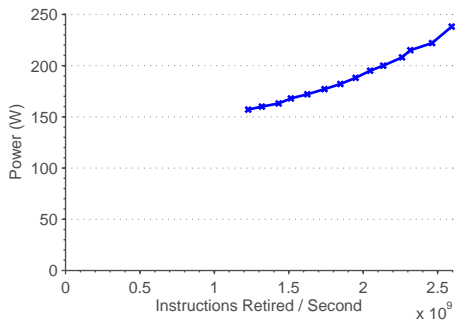

Figure 5: Core i7 processor power draw at different frequencies

We additionally tested our model of the hard disk drive using a digital multimeter while running various workloads. We were able to verify that the linear model predicted the power consumption of the device to within 10%, and that the power consumed was linear with the request rate.

The data in this section strongly supports the "Typical" model presented in Figure 1. The constant power of all systems examined is at least 50% of peak, and furthermore the performance-power tradeoff is linear.

3 Discussion

There are two key observations about the power vs. performance graph of modern systems. First, there is a large constant running power. Second, power is a linear function of performance counters; our results generally agree with those previously reported, such as [12, 1, 7]. We explore the implications of these two observations.

3.1 Race to Sleep

The most immediate implication of these two observations is that they explain anecdotal evidence for a model of computation known as "race to sleep." This idea holds that the most energy-efficient way of scheduling a computation is to put all hardware into the highest-performance state and race to completion as quickly as possible. Once finished, the hardware should drop to very low power modes – potentially ACPI sleep states or even powered off. The explanation for this fact draws on both the constant power and the linear power-performance tradeoff.

Note that this result is not impacted by more efficient idle states – this model applies whenever the constant run power and performance-power linearity holds. Furthermore, adding additional "run" power states which are only linear in power is of little use when designing power-efficient systems.

The additional problem with whole-system power optimization on current machines is that the cost of putting the rest of the system to sleep, i.e., powering off the chipset, is far from zero. It is so large that we consider it only after minutes of idle and it involves major operating system intervention.

4

3.2 Operating Systems

As shown in Figure 5, CPU power can be accurately modeled as a *linear* function of performance. The good fit of the linear regression in Figure 4 further suggests that this linearity holds for other devices, such as network access and disk IO. This simple relationship indicates that elaborate new operating system structures may not be necessary to account for and throttle power consumption. Existing commodity operating systems already contain extensive support for *performance* accounting and throttling such as `ulimit` and `io-throttle`. If power and performance in these devices are truly linear, the difference between the functionality of these tools and tools designed for account for and limit power is merely one of units! Given a system profile, these existing accounting and enforcement mechanisms can be used to both set budgets and limit rates. This model explains to some extent the limited power impact of optimizations like the Tickless kernel [13, 8]. It also says that the focus needs to be on the time to transition to and from sleep.

Furthermore, this means that applications wishing to reduce their energy usage must reduce utilization. This can be done through more efficient algorithms or by helping the OS race to sleep.

4 Limitations

There are two key traits that make current model so simple: linearity and low transition costs. If these traits change with future hardware designs then more complex optimizations and power accounting will be useful.

We have shown that currently the performance-power tradeoff for commodity systems is linear. However, this has not always been the case in CPUs, and it is not the case in all systems today. For instance, in certain systems such as sensor networks it may require significantly less power to send a certain number of bytes over a low-power, slow radio then it would to power on and transmit the same bytes over a high power radio. When the relationship is no longer linear, it becomes possible to reduce energy consumption by running more slowly and exploiting certain scheduling optimizations.

Finally, this result does not diminish the potential for reducing power consumption through treating a number of systems as a single unit for the purposes of power management – indeed, this approach is being successfully explored [2, 11] for exactly the reasons we have noted.

5 Conclusion

In this paper, we have investigated the complexity of the tradeoff space for power design. The result is a negative one: using current commodity hardware, there is a limited opportunity to pursue single-system optimizations which reduce energy usage without reducing performance. To save power in the broadest possible setting, we must examine optimizations which reduce the constant power or reduce the transition latency into extremly low-power sleep states.

References

[1] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner. Event-driven energy accounting for dynamic thermal management. In *COLP*, New Orleans, LA, Sept. 27 2003.

[2] A. M. Caulfield, L. M. Grupp, and S. Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In *ASPLOS*, pages 217–228, New York, NY, USA, 2009. ACM.

[3] A. Chandrakasan, S. Sheng, and R. Brodersen. Low-power cmos digital design. *Solid-State Circuits, IEEE Journal of*, 27(4):473–484, Apr 1992.

[4] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *SIGMETRICS*, pages 303–314, 2005.

[5] I. Corporation. Advanced configuration and power interface specification. 2006.

[6] Intel Open Source Technology Center. *Getting maximum mileage out of tickless*, June 2007.

[7] A. Kansal and F. Zhao. Fine-grained energy profiling for power-aware application design. *SIGMETRICS Perform. Eval. Rev.*, 36(2):26–31, 2008.

[8] M. Larabel. The impact of a tickless kernel. `http://www.phoronix.com/scan.php?page=article&item=651&num=1`.

[9] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In *ASPLOS*, pages 105–116, 2000.

[10] Y.-H. Lu and G. D. Micheli. Adaptive hard disk power management on personal computers. In *Great Lakes Symposium on VLSI*, pages 50–, 1999.

[11] R. Nathuji and K. Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *SOSP*, pages 265–278, New York, NY, USA, 2007. ACM Press.

[12] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In F. Zhao, editor, *HotPower*. USENIX Association, 2008.

[13] S. Siddha, V. Pallipadi, and A. V. D. Ven. Getting maximum mileage out of tickless. In *Proceedings of the Linux Symposium*. Intel Open Source Technology Center, June 2007.

[14] M. Weiser, B. B. Welch, A. J. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *OSDI*, pages 13–23, 1994.