Toward Energy-Efficient Computing

What will it take to make server-side computing more energy efficient?

David J. Brown, Sun Microsystems
Charles Reams, Cambridge University

By now, most everyone is aware of the energy problem at its highest level: our primary sources of energy are running out, while the demand for energy in both commercial and domestic environments is increasing, and the side effects of energy use have important global environmental considerations. The emission of greenhouse gases such as CO₂, now seen by most climatologists to be linked to global warming, is only one issue.

The world's preeminent scientists and thought leaders are perhaps most focused on a strategic solution: the need to develop new sources of clean and renewable energy if we are *ultimately* to overcome the energy problem. Lord Rees, president of the Royal Society, emphasized its urgency in an annual address delivered in 2008.¹³

The practical expectation of new sources of sustainable energy is at least three decades away, however. Steve Chu, who was the director of the Lawrence Berkeley National Laboratory prior to his recent appointment as U.S. Secretary of Energy, placed this in context:³

“A dual strategy is needed to solve the energy problem: (1) maximize energy efficiency and decrease energy use; (2) develop new sources of clean energy. No. 1 will remain the lowest-hanging fruit for the next few decades.”

ENERGY IN THE COMPUTING SPACE

What part does computer equipment play in the demand for energy, and where must we focus to reduce consumption and improve energy efficiency?

In August 2007, the EPA (Environmental Protection Agency) issued a report to Congress on energy efficiency of servers and data centers.⁵ Some key findings from the report include:

- Servers and data centers consumed 61 billion kWh (kilowatt hours) in 2006.
- This was 1.5 percent of total U.S. electricity consumption that year, amounting to \$4.5 billion in electricity costs—equivalent to 5.8 million average U.S. households.
- Electricity use in this sector doubled between 2000 and 2006, a trend that is expected to continue.
- Infrastructure systems necessary to support the operation of IT equipment (e.g., power delivery and cooling systems) also consumed a significant amount of energy, comprising 50 percent of annual IT electricity use.

Excerpts from the EPA report are shown in figure 1 and table 1. There are two particularly notable points in the data. The first is that as much energy is being consumed by site infrastructure as by the computing equipment itself. This infrastructure primarily represents HVAC (heating, ventilation, and air-conditioning) equipment, as well as that used to convert and transmit power and to maintain its continuity (the latter includes transformers and in-building power-switching and transmission equipment, as well as power-conditioning and sustaining equipment such as

FIGURE 1

Electricity Use by End-Use Component, 2000 to 2006

Source: EPA Report to Congress on Server and Data Center Energy Efficiency⁵

uninterruptible power supplies). This factor is of great consequence, but may not be the most obvious domain for computing professionals to address.

Within the computing equipment itself, however, is the second point of interest. Of the five types of IT equipment studied, volume servers alone were responsible for the majority (68 percent) of the electricity used. Assuming that the 17 percent CAGR (combined annual growth rate) of volume servers continues, this suggests that they are the prime targets for energy reduction in the server space. The 20 percent growth rate of storage devices shown here—a rate that more recent data

TABLE 1 Electricity Use by End-Use Component, 2000 to 2006

End Use Component	2000 Electricity Use (billion kWh)	% Total	2006 Electricity Use (billion kWh)	% Total	2000–2006 Electricity Use CAGR
Site infrastructure	14.1	50%	30.7	50%	14%
Network equipment	1.4	5%	3.0	5%	14%
Storage	1.1	4%	3.2	5%	20%
High-end servers	1.1	4%	1.5	2%	5%
Mid-range servers	2.5	9%	2.2	4%	-2%
Volume servers	8.0	29%	20.9	34%	17%
Total	28.2	--	61.4	--	14%

Source: EPA Report to Congress on Server and Data Center Energy Efficiency⁵

suggests is accelerating—indicates another significant trend.

If the exponential growth of data-center computing equipment revealed by this study continues, roughly double the demand for electricity seen in 2006 is expected in data centers by 2011. This poses challenges beyond the obvious economic ones. For example, peak instantaneous demand is expected to rise from 7 gigawatts in 2006 to 12 gigawatts in 2011, and 10 new base-level power plants would be needed to meet such a demand.

Physical limitations on power availability are already a constraint for data centers in some areas; a managing director of IT for Morgan Stanley recently observed that the company is no longer able physically to get the power needed to run a new data center in Manhattan. The situation is serious. Corporations such as eBay, Google, Amazon, Microsoft, and Yahoo have pursued suitable locations where the data centers required to run their contemporary Web applications and services can be constructed.⁹ A number of these companies have already negotiated with certain states in the U.S., as well as internationally, to construct these facilities, along with the power plants necessary to supply them. A few years ago Google touched off what some journalists deemed “a modern-day arms race” when it situated a new facility along the Columbia River in Washington. The combined benefits of lower land cost, lower external ambient temperature, and the availability of running water for cooling and hydroelectric power generation could provide an antidote to both Google’s acute energy-availability problems and its cost.

There is some evidence^a that the amount of energy consumed by mobile and desktop computing equipment is of roughly the same magnitude as that used by servers in data centers, although we do not have a correspondingly comprehensive and authoritative current study to refer to. The EPA data presented here provides some detailed perspective on where the energy goes in the important and growing server segment of the computing landscape. Also, some foundation has already been laid in the mobile and desktop computing space as a result of the earlier focus of the EPA’s EnergyStar program on consumer electronics, which includes computer systems.

POWER AND ITS MANAGEMENT IN COMPUTER SYSTEMS TODAY

Perhaps the key factor to consider with today’s computer systems is that the amount of power they consume does not adjust gracefully according to the amount of work the system is doing. The principal design objective for most general-purpose computer systems to date has been to maximize performance—or, perhaps, performance at a given price point—with very little consideration given to energy use. This is changing rapidly as we near the point where the capital cost to acquire computing equipment will be exceeded by the cost of energy to operate it—even over its relatively short (3- to 5-year) amortization period—unless we pay some attention to energy-conscious system design.

Although the case has been made for energy-proportional computing²—meaning that the amount of power required corresponds directly to a system’s (or component’s) degree of utilization—this is far from the current situation. Many components of computer systems today exhibit particularly poor efficiencies at low levels of utilization, and most systems spend a great proportion of their time operating at relatively low-usage levels. Power supplies have been notorious for their inefficiency, especially when at low load, and fans can waste much energy when operated carelessly. In just four years, however, the efficiency of power supplies has improved,¹ and algorithms that adjust fan speeds more continuously in relation to thermal need, rather than using just a few discreet speed points,

are emerging. The majority of hardware components in today's computer systems must still be managed explicitly, however, and the current widely deployed conceptions and facilities for power management in computer systems remain rudimentary.

POWER MANAGEMENT

There are two basic modalities for power management: a running vs. suspended (not-running) aspect in which a component (or whole system) can be powered off when it is not being used (i.e., once it has become idle), but turned on again when it is needed; and a performance-adjustment aspect (while running) in which the performance level of a component can be lowered or raised, based on either the observed level of its utilization or other needs of the workload.

The running versus not-running choices are often called the component's (or system's) *power states*. While there is a single state to represent running, there may be more than one suspended state. The latter allows power to be removed from progressively more of the hardware associated with the component (or system) if there is some important power-relevant structure to its implementation. CPUs, for example, may have their execution suspended simply by stopping the issuance of instructions or by turning off their clock circuitry. "Deeper" power states, however, might successively remove power from the processor's caches, TLBs (translation lookaside buffers), memory controllers, and so on. While more energy is saved as more of a component's hardware has its power removed, there is then either a greater latency to recommence its operation, or extra energy is required to save and restore the hardware's contents and restart it, or both.

The performance-adjustment choices while running are most naturally called the component's *performance states*. A widely applied technique for adjusting performance is to change the component's operating frequency. When clock speed is slowed, operating voltage levels can also be reduced, and these two factors together—normally called DVFS (dynamic voltage and frequency scaling)—result in a compound power savings. Performance states were first introduced for CPUs, since processors are among the most consequential consumers of power on the hardware platform (something in the range of 35 to 165 watts is typical of a contemporary multicore CPU). Performance states might also be used to control the active cache size, the number and/or operating rates of memory and I/O interconnects, and the like.

ACPI

The most widely implemented architecture for power management is ACPI (Advanced Configuration and Power Interface). It has evolved together with Intel architecture, the hardware platforms based on the most widely available commodity CPUs and related components. Although there are many detailed aspects to the specification, ACPI principally offers the controls needed to implement the two power-management modalities just described. It defines *power states*: seven at the whole-system level, called *S-states* (S0-S6); and four at the per-device level called *D-states* (D0-D3).^b The zero-numbered state (S0 for the system, or D0 for each device) indicates the running (or active) state, while the higher-numbered ones are nonrunning (inactive) states with successively lower power—and correspondingly decreasing levels of availability (run-readiness). ACPI also defines *performance states*, called *P-states* (P0-P15, allowing a maximum of 16 per device), which affect the component's operational performance while running. Both power states and performance states affect power consumption.

ENERGY EFFICIENCY IN COMPUTING

Although ACPI is an important de facto standard with reasonably broad support from manufacturers, it provides only a mechanism by which aspects of the system can be controlled to affect their power consumption. This enables but does not explicitly provide energy *efficiency*. Higher-level aspects of the overall system architecture are needed to exploit this or any similar mechanism.

How does energy-efficient computing differ from power management, and how would you know that you had solved the energy-efficiency problem for a computer system? Here is a simple vision:

“The system consumes the minimum amount of energy^c required to perform any task.”

In other words, energy efficiency is an *optimization* problem. Such a system must adjust the system’s hardware resources dynamically, so that *only what is needed* to perform those tasks (whether to complete them on time, or analogously, to provide the throughput required to maintain a stated service level) is made available, and that the total energy used is minimized as a result.

Traditionally, systems have been designed to achieve *maximum performance* for the workload. On energy-efficient systems, maximum performance for some tasks (or the whole workload) will still be desired in some cases, but the system must now also minimize energy use. It is important to understand that performance and energy efficiency are not mutually exclusive. For example, even when achieving maximum performance, any resources that can be deactivated, or whose individual performance can be reduced without affecting the workload’s best possible completion time or throughput, constitute energy optimization.

Indeed, there are few (if any) situations in which the full capacity of the hardware resources (i.e., all operating at their peak performance levels) on any system is exploited. Systems that strive to achieve maximum performance at all times are notoriously overprovisioned (and correspondingly underutilized). People involved in practical computer system design may note that our science is weak in this area, however. (This area might be called “dynamic capacity planning and dynamic provisioning.”)

Energy optimization is obviously subject to certain constraints. Some examples follow.

REQUIRED PERFORMANCE LEVELS MUST BE MAINTAINED

Tasks with deadlines must be completed on time. In the general case, a deadline is specified for a task or the workload. When any deadline is specified that is less than or equal to the optimum that the system can achieve with any or all of its hardware resources, this implies maximum performance. This is effectively the degenerate case.

Maximum performance for a task or the workload provides an implicit stipulation of the *optimal* deadline (t_o), or “as soon as possible.”^d In this case, energy optimization is restricted to those resources that can be deactivated, or whose individual performance can be reduced, without affecting the workload’s best possible completion time.

If a deadline later than the best achievable deadline is specified, the computation may take any length of time up to this deadline, and the system can seek a more global energy minimum for the task (or workload). Deadlines might be considered “hard,” in which case the system’s energy-optimizing resource allocator must somehow guarantee to meet them (raising difficult implementation issues), or “soft,” in which case a best effort can be tolerated.

Services must operate at required throughput. For online services, the notion of *throughput*, in order to characterize the required performance level, may be more suitable than that of a completion deadline. Since services, in their implementation, can ultimately be decomposed into individual tasks that do complete, we expect there to be a technical analog (although the most suitable means of specifying its performance constraint might be different).

THE SYSTEM MUST BE RESPONSIVE TO CHANGING DEMAND

Real workloads are not static: the amount of work provided and the resources required to achieve a given performance level will vary as they run. Dynamic response is an important practical consideration related to service level.

Throughput (T) must be achievable within latency (L). Specification of the maximum latency within which reserved hardware capacity can be activated or its performance level increased seems a clear requirement, but this must also be related to the performance needs of the task or workload in question.

Throughput is dependent on the type of task. A metric such as TPS (transactions per second) might be relevant for database system operation, triangles per second for the rendering component of an image-generation subsystem, or corresponding measures for a filing service, I/O interconnect, or network interface. Interactive use imposes realtime responsiveness criteria, as does media delivery: computational, storage, and I/O capacity required to meet required audio and video delivery rates. A means by which such diverse throughput requirements might be handled in practice is suggested below.

AVAILABLE POWER MUST NOT BE EXCEEDED

Instantaneous power must never exceed power limit (P). A maximum power limit may be specified to respect practical limits on power *availability* (whether to an individual system or to a data center as a whole). In some cases, exceeding this limit briefly may be permissible.

Combinations of such constraints mean that over-constraint must be expected in some circumstances, and therefore a policy for constraint relaxation will also be required. A strict precedence of the constraints might be chosen or a more complex tradeoff made between them.

APPROACHING A SOLUTION

Given this concept for energy-efficient computing, how might such a system be constructed? How would you expect an energy-efficient system to operate?

A system has three principal aspects that could solve this problem:

- It must be able to construct a *power model* that allows the system to know how and where power is consumed, and how it can manipulate that power (this component is the basis for *enacting* any form of power management).
- The system must have a means for determining the performance requirements of tasks or the workload—whether by observation or by some more explicit means of communication. This is the constraints-determination and performance-assessment component.
- Finally, the system must implement an *energy optimizer*—a means of *deciding* on an energy-efficient configuration of the hardware at all times while operating. That optimization may be relative

(heuristically decided) or absolute (based on analytical techniques). This is the capacity-planning and dynamic-provisioning component.

The first aspect is relatively straightforward to construct. The third is certainly immediately approachable, especially where the optimization technique(s) are based on heuristic methods. The second consideration is the most daunting. It represents an important disruptive consequence of energy-efficient computing and could demand a more formal (programmatic) basis for communicating requirements of the workload to the system. A description of the workload's basic provisioning needs, along with a way to indicate both its performance requirements and its present performance, seem basic to this. A way of indicating *a priori* its expected sensitivity to changes in provisioning of various system resources could also be useful. Fortunately, there are a number of practical approaches to energy efficiency to pursue prior to the refinements enabled by the hoped-for developments in category 2.

POWER MODEL

In order to manage the system's hardware for energy efficiency, the system^e must know the specific power details of the physical devices under its control. Power-manageable components must expose the controls that they offer, such as their power and performance states (D-states and P-states, respectively, in the ACPI architectural model). To allow modeling of power relative to performance and availability (i.e., relative to its activation responsiveness), however, the component interface must also describe at least the following:

- The per-state power consumption (for each inactive state) or power range (for each active state).
- State-transition latency (time required to make each state transition).
- State-change energy (energy expended to change state).

Once the system has such a power model, consisting of all its power-manageable hardware, it has the basic foundation for operating to optimize energy. Importantly, it has the knowledge of those components that consume the most power and those that have the most highly responsive controls that can be used to affect power use.

WORKLOAD CONSTRAINTS AND PERFORMANCE ASSESSMENT

In its desire to limit the amount of active hardware and reduce its performance so as to minimize energy consumption, how is a system to know whether the tasks being run are still achieving enough throughput to maintain appropriate service levels or realize their deadlines?

The assessment of throughput is subject to the task or application in question. The operating system can observe the degree to which its various resources have been and are currently being used, and it might use these observations as its best basis for prediction of future resource needs—thus shrinking or enlarging what is available. This is a relatively weak basis to determine what the workload will need, especially to anticipate its dynamic responsiveness sensitivities. As a result, the system will have to be much more conservative about its reduction of available resources or their performance levels. It seems clear that the best result will be realized if applications assess their own throughput relative to their service-level requirements or completion deadlines, and can convey that information to the operating system through an interface. The system can then use this information to make potentially much more aggressive resource adjustments and realize an improved overall energy-optimization solution accordingly.

Here is the crucial dichotomy: the system is responsible for solving the energy-optimization problem subject to the resources it allocates, while the application is responsible for monitoring its own performance level and informing the system so that appropriate resources can be provided to meet them.

ENERGY OPTIMIZATION BY THE SYSTEM

Once provided with the hardware's power characteristics, and possibly descriptive information from application-level software about its constraints, the operating system must begin the dynamic process of adjusting the hardware's performance and availability levels to control power consumption and improve systemwide energy use. How can the operating system make such decisions?

Heuristic Methods. Provisioning for maximum throughput may, in some cases, optimize energy. This is the conjecture that “[maximum] performance is green,” reflected in the ideas of race-to-idle or race-to-sleep.⁸ Although there is some evidence that this approach has merit in client-side computing when the system becomes idle—especially for embedded and mobile systems where 95 percent of the energy may be saved if the entire system can be put in a suspended state—it is not clear how applicable this may be for server-side computing. A nonlinear increase in the power required to get linear speed-up (throughput) exists in some cases—Intel's Turbo mode on contemporary CPUs is one example—and hence, the energy optimum will not be found at a provisioning and performance point commensurate with maximum throughput in all cases.

A widely used heuristic for energy improvement on active systems is to adjust the hardware's performance level dynamically, based on its current utilization: downward with low utilization or upward with high utilization (utilization below or above some threshold for some duration). This can be an effective technique but is restricted to situations in which both the latency and the energy to make the state change are so low as to be inconsequential.

Constraints-based Optimization as an Approach. In some cases, it may be possible to simplify the problem to such a degree as to provide a complete analytical solution. For example, if we consider only a single task on a single CPU with a well-understood power/performance tradeoff, it is relatively straightforward to specify completely a schedule in which the task will meet its deadline with the minimum total energy; more general formal results are also possible.¹² This relies, however, on a number of assumptions, such as good estimates of the total work required by a process, which frequently do not hold up in practice. Weaker assumptions require online optimization algorithms to perform energy-aware scheduling. There is some existing work in this area but not yet enough to underpin a general-purpose operating system.¹⁷

For an optimization-based approach to be generally applicable, a range of techniques will be necessary. In the simplest cases, autonomous device-level operation is possible; for example, at the hardware level, a GPU can power down unused hardware pipelines aggressively, based solely on instantaneous assessment of their utilization levels, because the latency to bring those pipelines back up as they become necessary is inconsequential. Similar practices appear to be applicable in the use of CPU P-states (CPU performance and energy-cost adjustment based on voltage and frequency scaling), since both the state-transition energy and latency are very low.

Hardware state changes that affect power but exhibit a much greater latency and/or a much greater amount of energy to make the state change require a different treatment. An obvious

example is spinning down a hard disk, considering the long latency to return it to running, but reactivation latency is not the only concern. Semiconductor memory systems in which part of the total physical memory could be powered off if not required, and where power-on latency may be near zero, will still have a consequential transition *energy*, since a great many in-memory transactions may be required to gather the working set into those physical pages that will remain active.^f Resources of this class require greater knowledge of the task or workload behavior, as well as an *anticipatory* treatment of the required hardware resources, to ensure that the activation latency can be tolerated or managed and that the state-change energy will be exceeded by the energy that will be saved while in that state.

Some common optimization techniques may be based on state-change latency, their energy demands, and so on, and a taxonomy of such techniques might arise from this—some formal or analytical, some based on more numerical or heuristic methods.

Although we expect the specific techniques for energy optimization appropriate to different hardware resources or subsystems to be somewhat different, subject to the properties of the hardware resources in question, the hope is that the composition of energy-efficiency optimizers for all such resources will accumulate to form an efficiency scheme for the whole system.^g

GETTING THERE

The vision of systemwide concessions to energy efficiency cannot be accomplished in a single swift step. Today's systems software is not equipped in the ways described, nor are applications written in a way that could exploit that capability. In pragmatic terms, how do we expect this outcome to be achieved, and what steps are already under way?

CONSIDERATIONS FOR THE OPERATING SYSTEM

As a first consideration, systems need to be revised to pay attention to their use of energy; the operating system itself, which is always running, has not yet been optimized in its *own* use of energy. To date, almost all software, including systems software, has been optimized for performance, robustness, and scalability with no consideration of energy. An initial step, therefore, is the redesign and implementation of the operating system so that its operation is energy efficient. This is a significant undertaking, and its full implications are not yet well understood.

It is not clear whether modifying existing operating systems to consider energy as a first-class constraint is feasible, although this would certainly be preferable. Experience with system security shows that attempts to introduce such fundamental considerations after the fact are fraught with complications. We can certainly anticipate fundamental new structures within systems software, and perhaps even that new operating systems will emerge as a result of the energy-efficiency pressure.

At the very least, resource-management facilities within the operating system must be adapted for energy awareness, and then for energy optimization.

Processors. Given the significant fraction of power on contemporary computing platforms attributed to CPUs (and the early introduction of power-management features on them as a result), much progress has already been made with operating-system schedulers/thread dispatchers. Careless activation of hardware when there is no useful work to be done must be eliminated. Polling within the operating system (or within applications) is an obvious example, but the use of a high-frequency

clock-tick interrupt as the basis for timer events, time-keeping, and thread-scheduling can be equally problematic. The objective is to keep hardware quiescent until needed. The “tickless” kernel project¹⁶ in Linux introduced an initial implementation of a *dynamic* tick. By reprogramming the per-CPU periodic timer interrupt to eliminate clock ticks during idle, the average amount of time that a CPU stays in its idle state after each idle state entry can be improved by a factor of 10 or more. Beyond the very good ideas that dynamic ticks and deferrable timers in Linux represent, the Tesla project in OpenSolaris is also considering what the transition to a more broadly event-based scheme for software development within the operating system might imply.

The confluence of features on modern processors—CMT (chip multithreading), CMP (chip multiprocessor), and NUMA (non-uniform memory access) for multiprocessor systems with multiple sockets—invites a great deal of new work to implement optimal-placement thread schedulers.⁶ Given the ability to alter performance levels, energy efficiency and the expected introduction of heterogeneous multicore CPUs^h will only add intrigue to this.^{15,7}

Storage. Compared with CPUs, the power consumed by a disk drive does not seem especially large. A typical 3.5-inch, 7200-RPM commodity disk consumes about 7 to 8 watts—only about 10 percent of what a typical multicore CPU consumes. Although higher-performance 10,000-RPM spindles consume about 14 watts, and 15,000-RPM drives perhaps use around 20 watts, what is the worry? The alarming relative rate of growth in storage, mentioned earlier, could quickly change the percentage of total power that storage devices account for. Performance and reliability factors have already resulted in the common application of multiple spindles, even on desktop systems (to implement a simple RAID solution). In the data center, storage solutions are scaling up much faster. Low-end volume server boxes now routinely house a dozen or more drives, and one example 4U rack-mount storage array product from Sun accommodates 46 3.5-inch drives. A single instance of the latter unit, if it used 10,000- or 15,000-RPM industrial drives, might therefore account for 1,088 watts to 1.6 kilowatts, rather a more significant energy-use picture.

Storage subsystems are now obviously on the radar of the energy attentive. There are at least two immediate steps that can be taken to help improve energy consumption by storage devices. The first is direct attention to energy use in traditional disk-based storage. Some of this work has been started by the disk hardware vendors, who are beginning to introduce disk-drive power states, and some has been started by operating-system developers working on contemporary file systems (such as ZFS) and storage resource management. The second, particularly derived from the recent introduction of large inexpensive flash memory devices, is a more holistic look at the memory/storage hierarchy. Flash memory fills an important performance/capacity gap between main memory devices and disks,^{10,11} but also has tremendous energy-efficiency advantages over rotating mechanical media.

Memory. Main memory, because of its relatively low power requirement (say, 2 watts per DIMM), seems at first glance to be of even less concern than disks. Its average size on contemporary hardware platforms, however, may be poised to grow more rapidly. With hardware system manufacturers’ focus primarily on performance levels (to keep up with the corresponding performance demands of multicore CPUs), maintaining full CPU-to-memory bandwidth is critical. The consequence has been an evolution from single- to dual-channel and now triple-channel DIMMs along with the corresponding DDR, DDR2, and DDR3 SDRAM technologies. Although reductions in the process feature size (DDR3 is now on 50-nanometer technology) have enabled clock frequency to go up and power per DIMM to go down somewhat, the desire for even greater performance via an increase in

DIMMs per memory channel is still increasing the total power consumed by the memory system.

For example, a current four-socket server system (based on the eight-core Sun Niagara2 CPU) with 16 DIMMs per socket using DDR2 dual-channel memory technology has 64 DIMMs total. This would increase to 24 DIMMs per socket (96 total) if its faster successor used DDR3 triple-channel memory instead. A representative DDR2 DIMM consumes 1.65 watts (or 3.3 watts per pair), whereas the lowest-power edition of the current DDR3 DIMMs consume 1.3 watts (or 3.9 watts per trio). The result appears to be an increase of only 20 percent power consumption—from about 100 to 120 watts total in our example.

Given that the next-generation CPU will also have twice as many cores per socket, however, a possible scenario is also to desire twice the number of memory sets per socket (for a possible 192 total DIMMs) to balance overall memory system performance. The result, therefore, could be an increase from 100 watts to 240 watts (a 140 percent increase in power consumption for the whole memory system)! This trend is even being observed on desktop-class machines, admittedly at a much smaller scale, as systems containing quad-core hyperthreaded CPUs (such as Intel's Nehalem) have appeared.

If available physical memory is to be enabled and disabled, and perhaps correspondingly reconfigured as a system's processing capacity is dynamically adjusted, some new functionality will be required of the operating system's memory-management subsystem. The design of a future-looking virtual memory system that is energy aware and able to adjust physical memory resources while running is an open problem.

I/O. Energy aspects of the I/O system on hardware platforms will likely become more important as well. As a simple example, present-day local-area networking interconnect and subsystems have evolved in two important respects: link-aggregation is increasingly used to bolster network bandwidth and reliability; and individual interconnect speed has advanced from 1 GB to 10 GB, with 40 GB on the horizon. A transceiver for a 10-GB network interface card may now require as much as 14 watts when operating at full speed, with a consequential power reduction when its link speed is reduced to 1 GB or lower (about 3 watts at 1 GB, 1 watt at 100 MB). Other high-speed interconnects such as InfiniBand can be expected to have similar energy implications for the overall system. Little attention has been given to the energy implications of communication interconnects in any of their various architectural manifestations, from on-chip to wide area networking.

THE EVOLUTION OF APPLICATION SOFTWARE

The most strategic aspect of energy-efficient computing will be the evolution of application software to facilitate systemwide energy efficiency. Although we can certainly expect new application interfaces to the system software, supporting the development of new energy-efficient applications, the transition of historical and present-day applications represents a long-term evolution. How will we address the problem of greater energy efficiency for the remainder of the installed base in the interim? Obviously, it will not be brought about as the result of a unique epoch in the implementation of all existing applications.

One possibility for addressing the energy agnosticism of existing applications is to perform extrinsic analysis of their runtime behavior. Empirical data can be gathered about the degree to which application performanceⁱ is sensitive to varying levels and types of resource provisioning. For example, one can observe the degree to which performance is increased by the addition of CPU resources, or the allotment of a CPU with higher-performance microarchitecture, and so on.¹⁵ The

application might then be *labeled*, in its binary form, with its measured degree of sensitivity, without requiring the alteration of its existing implementation. The operating system could then use the data to assign resources that pursue a certain specified performance level or to locate an appropriate performance-versus-energy consumption tradeoff.

Inevitably, we expect that a combination of techniques will be needed: both explicit, in which the application itself informs the system of its throughput and resource provisioning needs; and implicit, in which static and dynamic analysis is used to model resource needs relative to performance and energy consumption.

CONCLUSION

We are still at the debut of energy-conscious computing, with a great deal of the industry's attention being given to the introduction and use of power-management mechanisms and controls in individual hardware components rather than to the broader problem of energy efficiency: the minimization of total energy required to run computational workloads on a system. This article suggests an overall approach to energy efficiency in computing systems. It proposes the implementation of energy-optimization mechanisms within systems software, equipped with a power model for the system's hardware and informed by applications that suggest resource-provisioning adjustments so that they can achieve their required throughput levels and/or completion deadlines.

In the near term, a number of heuristic techniques designed to reduce the most obvious energy waste associated with the highest-power components, such as CPUs, are likely to remain practical. In the longer term, and for more effective total energy optimization, we believe that techniques able to model performance relative to the system's hardware configuration (and hence its energy consumption), along with an improved understanding and some predictive knowledge of workloads, will become increasingly important. ◻

NOTES

- a. The U.S. Energy Information Administration (www.eia.doe.gov) showed a figure of 23.1 terawatt hours per year consumed by PCs and printers within U.S. households in 2001.⁴ The figures were similar in 2006.¹⁴
- b. Idiosyncratically, the power states for CPUs are called *C-states* (C0-C3). In any case, the semantics of each nonrunning power state is specific to the device (or device class) in question.
- c. Energy is the time integral of power, so that for constant power, $\text{energy} = \text{power} \times \text{time}$. Power and energy are different concepts and should not be confused.
- d. All values of deadline: $D = t_i$ less than the shortest achievable deadline: t_0 is equivalent to setting $D = t_0$ (that is: $\{\forall t_i \mid t_i < t_0, [D = t_i] \approx [D = t_0]\}$). We can therefore denote maximum performance by $D = 0$.
- e. "The system" here most naturally suggests the *operating system*, although it is clear that this must include the hypervisor for virtualized systems. One can reasonably expect that this concept will need to be broadened to include some aspects of the firmware and even hardware components (on the low end) and important runtimes, such as the Java Virtual Machine, which have responsibility for, and/or particular knowledge of, resource allocation.
- f. It is interesting to consider whether traditional heuristics such as the Five-minute Rule, designed to optimize the memory hierarchy for performance, might have analogs in energy optimization.

- g. We recognize that such reductionism may be overly optimistic if there are interactions between the resources allocated by different subsystems, and that a more holistic approach (e.g., a large dynamic-programming approach) may then be necessary in systems where “every joule counts.”
- h. *Heterogeneous* here means a multicore CPU in which cores of different performance levels (different CPU microarchitectures) are put in the same multicore package, and whose power-consumption consequences are therefore very different.
- i. This assumes one can define some objective external metric of performance, which may be problematic.

REFERENCES

1. 80plus.org. Recent standards for power supply efficiency; <http://www.80plus.org>.
2. Barroso, L., Holzle, U. 2007. The case for energy-proportional computing. *IEEE Computer* (December): 33-37.
3. Chu, S. 2008. The energy problem and Lawrence Berkeley National Laboratory. Talk given to the California Air Resources Board (February).
4. Energy Information Administration, U.S. Department of Energy. 2001. Residential Energy Consumption Surveys; <http://www.eia.doe.gov/emeu/recs/recs2001/enduse2001/enduse2001.html>.
5. Environmental Protection Agency. 2007. EPA report to Congress on server and data center energy efficiency (August); http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.
6. Fedorova, A. 2006. Operating system scheduling for chip multithreaded processors. Ph.D. dissertation, Harvard University (September).
7. Fedorova, A., Saez, J. C., Shelepov, D., Prieto, M. 2009. Maximizing performance per watt with asymmetric multicore systems. *ACM Queue* (November/December); <http://queue.acm.org/detail.cfm?id=1658422>.
8. Garrett, M. 2007. Powering down. *ACM Queue* (November/December).
9. Katz, R. H. 2009. Tech titans building boom. *IEEE Spectrum* (February); <http://www.spectrum.ieee.org/green-tech/buildings/>.
10. Leventhal, A. 2008. Flash storage today. *ACM Queue* 6(4): 25-30 (July/August); <http://queue.acm.org>.
11. Mogul, J., Argollo, E., Shah, M., Faraboschi, P. 2009. Operating system support for NVM+DRAM hybrid main memory. *Proceedings of Usenix HotOS XII* (May).
12. Reams, C. 2010. Energy-conscious computing—formal techniques for energy cost minimization (forthcoming paper).
13. Rees, M. 2008. Anniversary address to the Royal Society, London.
14. Roth, K. W., McKenney, K. 2007. Energy consumption by consumer electronics in U.S. residences. TIAx LLC, Cambridge, MA (January).
15. Shelepov, D., Saez, J. C., Jeffery, S., Fedorova, A., Perez, N., Huang, Z. F., Blagodurov, S., Kumar, V. 2009. HASS: a scheduler for heterogeneous multicore systems. *ACM Operating System Review* 43(2): 66-75, 2009.
16. Siddha, S. 2007. Getting maximum mileage out of tickles. *Proceedings of the Linux Symposium* (June), Ottawa, Ontario: 201-208.

17. Yao, F., Demers, A., Shenker, S. 1995. A scheduling model for reduced CPU energy. *Proceedings of 36th Annual Symposium on Foundations of Computer Science*: 374–382.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

DAVID BROWN is currently working on the Solaris operating system’s core power management facilities, with particular attention to Sun’s x64 hardware platforms. Earlier at Sun he led the Solaris ABI program: a campaign to develop and deliver a practical approach to binary compatibility for applications built on Solaris. He received a Ph.D. in Computer Science from Cambridge University, for a dissertation which introduced the “Unified Memory Architecture” approach for the integration of high performance graphics subsystems in a general-purpose computing architecture. This idea is now widely applied, notably in the current Intel processor and memory system architecture.

CHARLES REAMS is a Ph.D. student in Computer Science at the University of Cambridge. His research is driven by an interest in quantitative and emergent aspects of programming languages, such as power efficiency and error resistance. He hopes ultimately to move academic work on energy- and cost-efficient scheduling into the commercial world; mathematical underpinnings for this work will appear in the forthcoming technical report, *Computing On The Cheap: Formal Methods for Efficient Scheduling*.

© 2010 ACM 1542-7730/10/0200 \$10.00