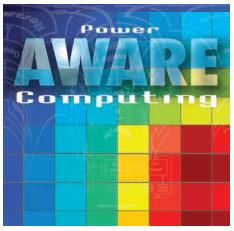
High-Performance, Power-Aware Distributed Computing for Scientific Applications



The PowerPack framework enables distributed systems to profile, analyze, and conserve energy in scientific applications using dynamic voltage scaling. For one common benchmark, the framework achieves more than 30 percent energy savings with minimal performance impact.

Kirk W. Cameron
Rong Ge
Xizhou Feng
Virginia Tech

Computer models deepen our understanding of complex phenomena and indirectly improve our quality of life. Biological simulations of DNA sequencing and protein folding advance healthcare and drug discovery. Mathematical simulations of world economies guide political policy. Physical fluid-flow simulations of global climate allow more accurate forecasting and life-saving weather alerts. Nanoscale simulations enhance underlying computing technologies.

Evolving parallel and distributed systems designed for these demanding scientific simulations will be enormously complex and power hungry. Computers capable of executing one trillion floating-point operations per second (Tflops) have already emerged, and petaflops (Pflops) systems that can execute one quadrillion floating-point operations per second are expected by the end of the decade.

Because supercomputing experts cannot agree on what the core technologies should be, these systems will likely be diverse as well. Nevertheless, most will be built from commercial components in integrated scalable clusters of symmetric multiprocessors. Such solutions will be highly parallel with tens of thousands of CPUs, tera- or petabytes of main memory, and tens of Pbytes of storage. Advanced software will synchronize computation and communication among multiple operating systems and thousands of components.

PERFORMANCE AND POWER DEMANDS

As Figure 1 illustrates, the growing computational demands of scientific applications have led to exponential increases in peak performance (shown as Top5 average Linpack peak, R_{max}), power consumption (shown for several supercomputers), and complexity (shown relative to average number of microprocessors, µP) for large-scale systems using commercial components. Power-performance optimization involves minimizing the current efficiency gap in processing throughput and power consumption.

Performance

Complex parallel and distributed systems are difficult to program for efficient performance. As the number of components increases, such systems achieve a lower percentage of theoretical peak performance. Gordon Bell award-winning applications achieve between 35 and 65 percent of peak performance through aggressive optimizations by a team of experts working collaboratively for months.[1] For the past decade's five most powerful machines, the average percentage of peak performance for Linpack—arguably the most heavily optimized high-performance code suite—ranged between 54 and 71

Published by the IEEE Computer Society
0018-9162/05/$20.00 © 2005 IEEE

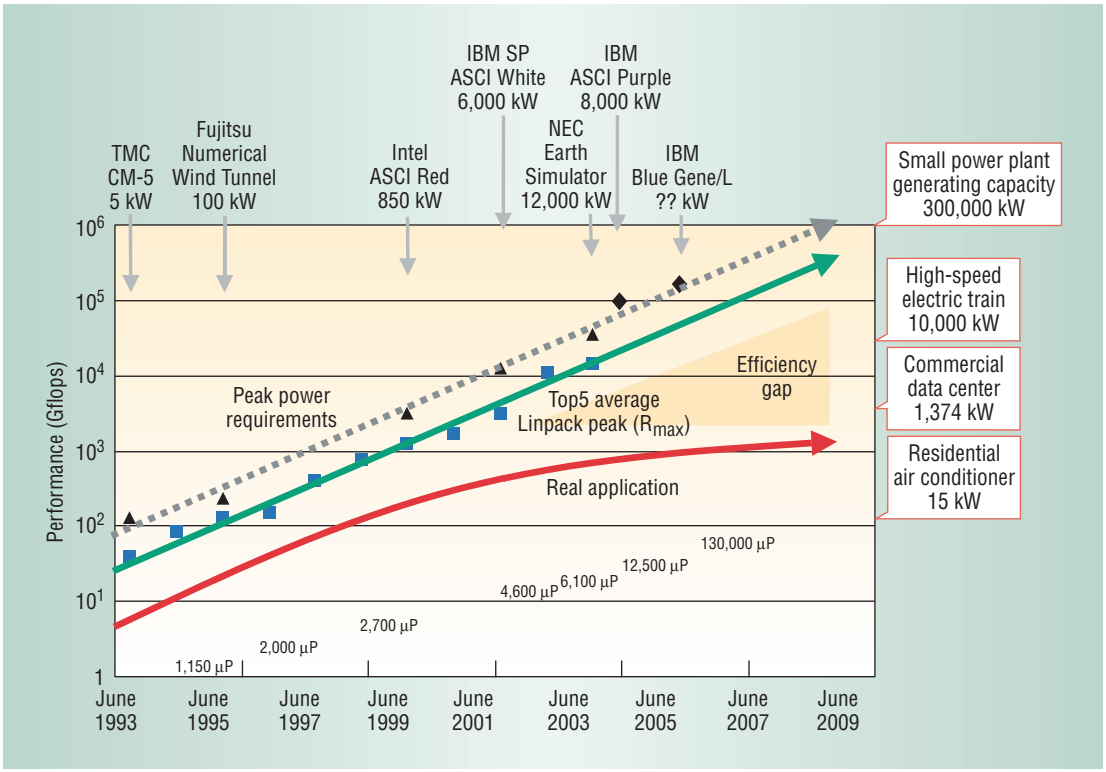


Figure 1. Supercomputer performance and power trends. Scientific applications' growing computational demands have led to exponential increases in system peak performance, power consumption, and complexity requirements.

percent. Typical scientific applications commonly achieve only 5 to 10 percent of peak performance on today's high-performance systems.

Power consumption

Japan's NEC Earth Simulator, which is capable of executing 40 Tflops, requires about 12 megawatts of power. Pflops distributed systems will require considerably more power. A 1998 NASA study[2] predicted power needs of 25 megawatts for such a system, while more recent projections point to 100 megawatts[3]—enough power to supply 1.6 million 60-watt lightbulbs, the lighting requirements of a small city.

The estimated exponential increase in large-scale systems' power requirements shown in Figure 1 is somewhat conservative in that the data, compiled from various sources, ignores air-cooling system power, and single-processor power consumption—for example, the Itanium2 consumes 130 watts—is expected to increase exponentially with clock rates until at least 2010.

Reliability. Reducing distributed system components' power and energy consumption increases reliability. A growing number of scientific simulations require hundreds of hours of continuous execution.[4] Current devices executing these applications fail at an annual rate of 2 to 3 percent. A hypothetical Pflops system of about 12,000 nodes would experience a hardware failure every 24 hours.

The Arrhenius life-stress model implies that the operating temperature of an electronic component reduces its life expectancy and thus its mean time to failure. Every 10°C (18°F) temperature increase reduces component life expectancy by half; reducing a component's operating temperature by the same amount doubles its life expectancy. Decreasing high-performance systems' energy consumption will reduce the loss of time, effort, and experimental data due to aborted program executions or incorrect results from component failure.

Cost. Reducing distributed system components' power and energy consumption also decreases cost. Assuming a rate of $100 per megawatt, a Pflops machine consuming 100 megawatts of power at peak operation would cost $10,000 per hour and could surpass $85 million in a year. More conservative rule-of-thumb predictions of 20 percent peak power consumption imply a lower, but not necessarily manageable, $17 million annual operational cost ($2,000 per hour). These rough estimates do not include air-cooling requirements, which commonly amount to 40 percent of system operating costs. Even small reductions in overall energy consumption would significantly impact Pflops systems' operational costs.

Dynamic voltage scaling makes it possible to scale a processor's frequency, thereby reducing CPU performance and conserving system power.

Power-performance efficiency

Scientific applications demand more power-performance efficiency in parallel and distributed systems. The efficiency gap shown in Figure 1 could be reduced in two ways. Theoretically, improving performance efficiency would cause the red line to move upward. Improving power efficiency while maintaining performance would reduce theoretical peak performance when it is not required and thereby lower the green line without affecting the red line. Distributed applications that suffer from poor performance efficiency despite aggressive optimizations are candidates for increased power efficiency.

POWER-AWARE COMPUTING

Simulators are available to evaluate the power consumption of applications on emerging systems at the circuit-design level, such as Hspice (www.synopsys.com/products/mixedsignal/hspice/hspice.html), and at the architectural level—for example, Wattch,[5] SimplePower,[6] and SoftWatt.[7] Direct measurement at the system level is more difficult; PowerScope,[8] Castle,[9] and other tools attempt to do so using contact resistors, multimeters, and software tools. Indirect approaches estimate power consumption based on hardware counter data, but these options focus on microarchitecture power and energy consumption and are limited in the context of distributed scientific applications.

Hard versus soft constraints

In direct contrast to low-power computing, performance is a hard constraint and power is a soft constraint in high-performance computing. The low-power approach is appropriate for designing systems that run on limited power supplies such as PDAs and laptops but not for overcoming most reliability and cost constraints in high-performance systems. Such attempts attain reasonable performance only insofar as they increase the number and improve the efficiency of low-power components—for example, IBM's Blue Gene/L uses more than 100,000 processors and increases the throughput of floating-point operations in the embedded PowerPC architecture.

Power modes

Power-aware features provide opportunities for more innovative energy-saving techniques. Integrated power-aware components in high-performance systems provide fine-grained control over power consumption. Such components have various programmable energy settings or modes: PCs and monitors stand by and shut down automatically, disk drives spin down when not in use, network interface cards have low-power settings, many commercial microprocessors can dynamically scale down clock rates,[10] and memory power can be conserved separately.[11]

The challenge of power-aware computing is to reduce power and energy consumption while maintaining good performance. Power modes affect a device's rate of electrical consumption. Operating in low-power modes over time can conserve energy, but performance usually decreases during mode transition and low-power operation.

A device's energy consumed reflects the interaction of the application workload, the device's internal structure, and power-mode timing. Determining the optimal duration and frequency of power modes requires predicting the effects of mode-transition penalties and low power on an application's time to solution.

Dynamic voltage scaling

DVS makes it possible to scale a processor's frequency, thereby reducing CPU performance and conserving system power. Researchers have studied power-performance tradeoffs extensively using DVS in the context of real-time systems and interactive mobile applications. For example, Luca Benini and Giovanni De Micheli provide a tutorial on system-level optimization techniques including DVS,[12] while Chung-Hsing Hsu and Ulrich Kremer have inserted DVS control calls at compile time in noninteractive, memory-bound applications.[13]

DVS techniques that exploit memory latency for power savings are very promising in the context of scientific applications because performance impact is controllable and minimal (between 0.69 and 6.14 percent for Spec95 codes), and energy savings are significant (between 9.17 and 55.65 percent).

Distributed systems

Energy savings are possible with controllable, minimal performance loss in interactive cluster-based commercial servers. These power-aware distributed clusters[14] react to workload variations. When full nodes can be shut down, energy savings are significant. These approaches, however, do not meet the needs of scientific applications.

Interactive applications in Web servers are independent tasks that can be scheduled arbitrarily upon arrival. Soft real-time deadlines on task completion allow further flexibility. Power-related policy decisions are centralized in the distributed

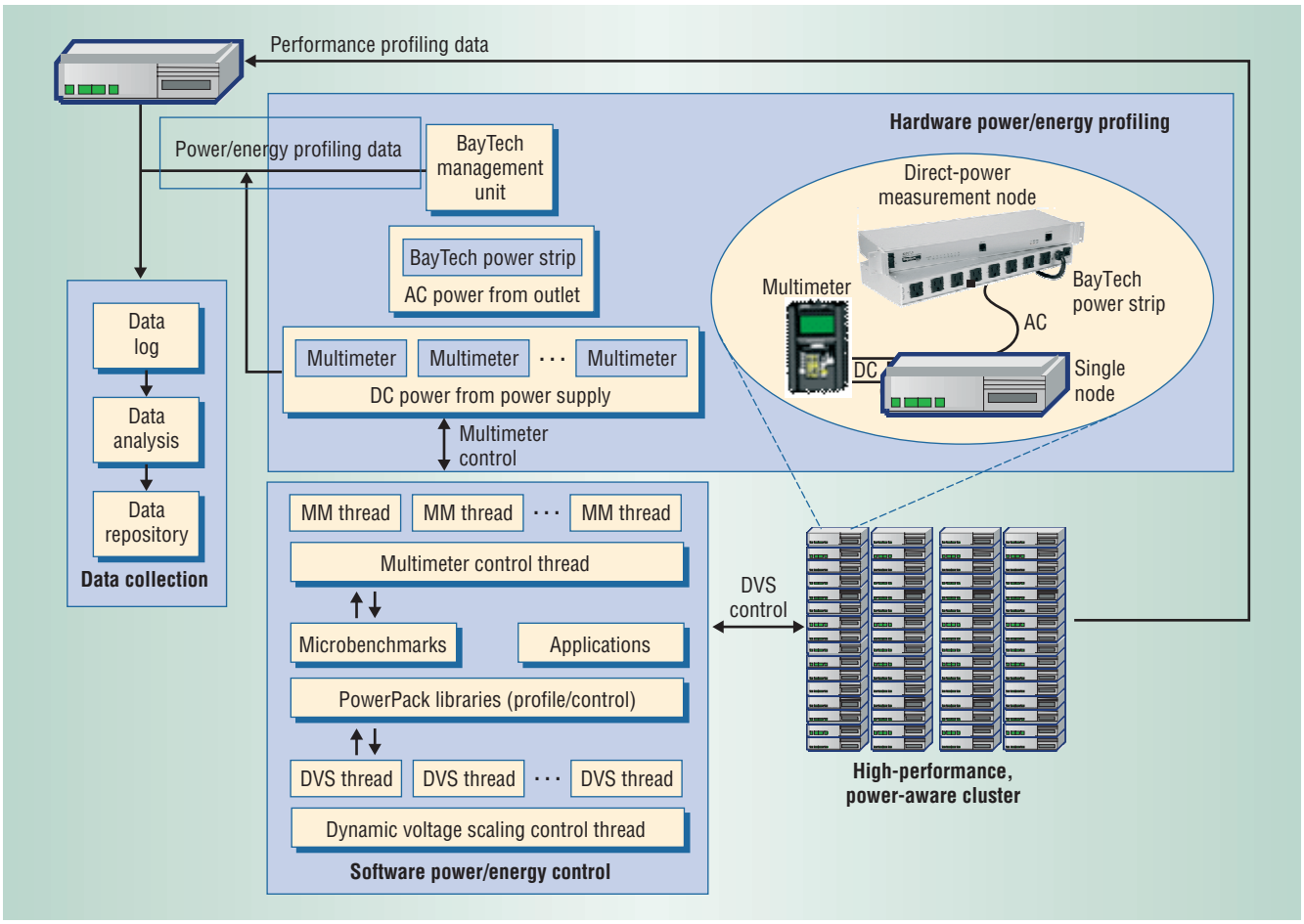


Figure 2. PowerPack 1.0. The framework combines direct power measurements using multimeters, smart power strips, and ACPI-enabled power supplies with software data collection, analysis, and control.

scheduler, which determines utilization using system-level data from the /proc file system and models system load using arrival distributions or predicts it based on past events. Matching system resources to anticipated requests enables power savings in interactive systems.

Distributed scientific applications are noninteractive and often run as batch jobs. Because tasks are usually interdependent and have a user-defined synchronization scheme, future resource utilization cannot typically be predicted based on past observations. Given the low potential for idle nodes, power savings must be obtained by other means such as DVS. As time is critical in high-performance computing, power optimizations must not impact performance significantly.

POWERPACK FRAMEWORK

Reducing energy consumption requires having information about the significant details of an application's performance and synchronization scheme. Centralized control is possible but, given the additional costs in latency to communicate power-aware control information across a distributed system, decentralized control is preferable. To meet these demands, we have created PowerPack, a framework for profiling, analyzing, and controlling energy consumption in parallel scientific applications.

As Figure 2 shows, PowerPack 1.0 combines direct power measurements using multimeters, smart power strips, and ACPI (advanced configuration and power interface)-enabled power supplies with software data collection, analysis, and control. With the framework, developers can profile component performance and power and throttle the CPU during distributed workload slack time to conserve energy while maintaining good performance.

In our profiling experiments, we measured individual power consumption of four node components—CPU, memory, disk, and network interface card—on slave nodes in a 32-node Beowulf cluster.[15] We used an older machine because power-related experiments can damage components. Each slave node has one 933-MHz Intel Pentium III processor, four 256-Mbyte synchronous DRAM modules, one 15.3-Gbyte IBM DTLA-307015 DeskStar hard drive, and one Intel 82559 Ethernet Pro 100 onboard Ethernet controller.

In these studies, CPU power (about 30 watts) accounts for roughly 40 percent of nodal system power under load. On newer 130-watt Itanium machines, our techniques would result in greater energy savings because the CPU accounts for a larger percentage of total system power.

PowerPack library routines let users embed ACPI calls in applications and reduce the CPU's process-

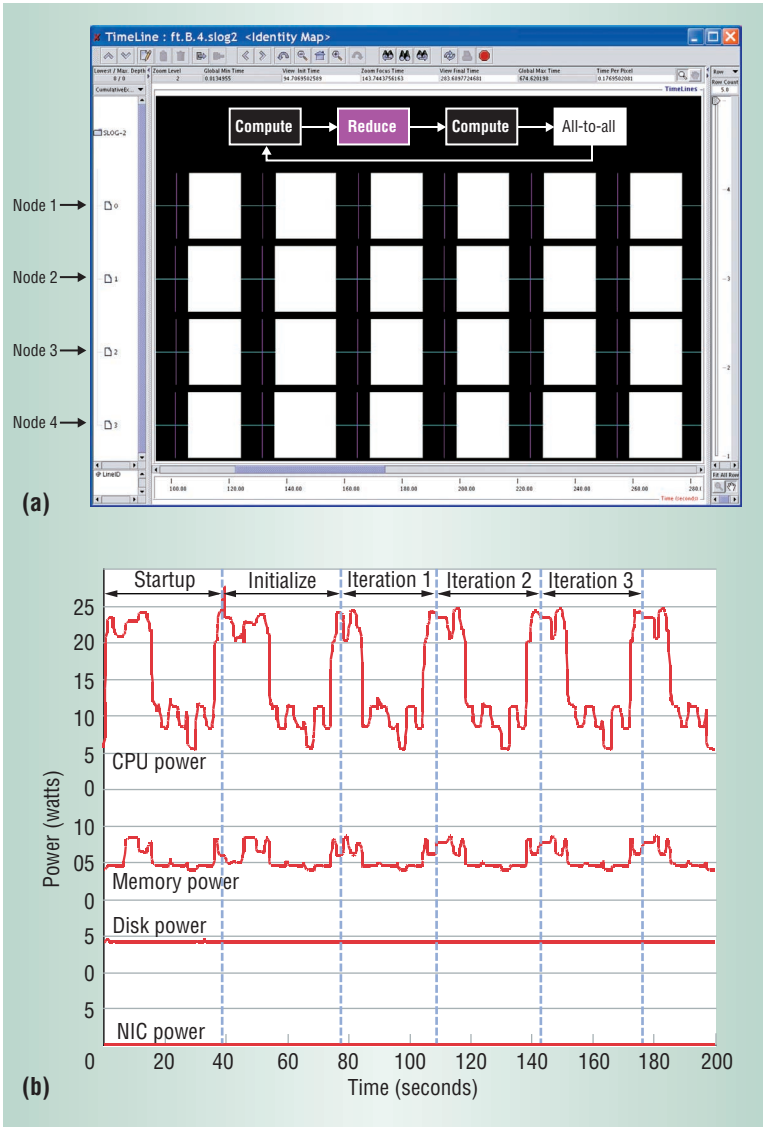


Figure 3. NAS FT benchmark performance and power profiles using PowerPack. (a) Parallel performance on four nodes. (b) Power consumption on one of four nodes.

ing speed via DVS. While in a lower voltage state, the processor consumes less power and performance suffers. To maintain good performance in scientific applications, PowerPack can throttle down CPU speed during idle or slack periods when memory or communication bound phases, not CPU speed, is the application bottleneck.

BENCHMARK EXAMPLE

By carefully applying PowerPack, it is possible to achieve total system energy savings in scientific applications with minimal performance impact. We illustrate our power-aware approach with the Fourier transform (FT) benchmark from NASA's National Aerodynamic Simulation (NAS) parallel benchmark suite.

FT code is communication bound and generally performs poorly on distributed systems due to butterfly data exchanges that rely heavily on message passing interface (MPI) all-to-all communication. Other NAS benchmarks, such as embarrassingly parallel (EP) code, do not conserve energy due to high computational efficiency, or, like conjugate gradient (CG) code, offer significant energy savings but involve complex tradeoffs.[16]

Performance and power profiles

We first obtained performance and power profiles of FT on PowerPack to quantify the power and energy savings possible for a given application. Figure 3a shows a Jumpshot visualization of FT performance on four nodes using the MPI Parallel Environment tool from the MPICH implementation. A single FT performance cycle or phase consists of four basic steps: compute, reduce, compute, all-to-all. Figure 3b shows power profiles of FT on one of four nodes. All-to-all communication dominates execution time and results in long CPU idle periods, indicating potential for energy savings.

To conserve energy with code like FT, the available slack periods must be long enough to outweigh the overhead for moving into and out of DVS power modes. For example, using PowerPack the switching overhead on our DVS-enabled system—composed of 16 1.4-GHz Pentium-M-based laptops—is about 40 microseconds. Such large penalties imply that DVS should be used sparingly. By observing the FT performance and power profiles, we determined that potential energy savings were high using function-level granularity—effectively wrapping calls to throttle the CPU down and back up around the MPI all-to-all communications.

Saving energy without impacting performance

Figure 4 shows the results of throttling down CPU speed for FT on our system. The bars show the normalized values for delay and energy of an internal algorithm that uses phased DVS scheduling based on the previously obtained performance and power profiles, external settings fixed for the duration of application execution (600-1,400 MHz), and cpuspeed—an automatic history-based daemon. The results are normalized to the best performing "machine"—in this case, all nodes running at 1,400 MHz without any DVS.

The figure shows that with PowerPack significant energy savings can be achieved for code such as FT. The amount of total system energy saved varies from 5 to 40 percent, while execution time increases from 0 to 15 percent. The internal scheduling algorithm achieves the most energy savings—about 36 percent—with less than 1 percent performance impact. Another 2 percent energy can be saved by setting all machines to the lowest oper-

ating point (600 MHz), but this increases execution time by more than 15 percent.

The results of cpuspeed and the static settings from 800 to 1,200 MHz show that the energy-performance tradeoff is more complicated than simply selecting a minimum value in each group. Users must determine the desired energy savings and the performance they are willing to sacrifice to achieve it.

COST IMPLICATIONS

Based on current trends in supercomputer design, system operating budgets will eventually rival hardware costs. Consider, for example, the NEC Earth Simulator, which has an estimated hardware price tag of $300 million. Assuming 20 percent of peak power consumption (12 megawatts), its annual operational cost exceeds $2 million ($240 per hour)—less than 1 percent of acquisition cost. However, the annual operating cost of a future 100-megawatt Pflops system, ignoring inflation, would be about $5 million; assuming a generous $300 million budget, this is equivalent to nearly 6 percent of acquisition cost.

On our Beowulf system, which consumes about 1.9 kilowatts under load at a cost of about $5 per day or $1,825 per year, 30 percent energy savings translates to a $550 annual cost savings. On a 100-megawatt Pflops system consuming 20 percent peak power at a cost of $2,000 per hour, or $17 million per year (ignoring air-cooling costs), this would result in a $5 million annual cost savings. Even half this amount would significantly reduce the operating budget of most US supercomputer facilities while maintaining the high performance that demanding scientific applications require.

FUTURE WORK

PowerPack is the first framework to enable application-driven power measurement and management at the component and cluster levels for scientific applications. We have successfully ported its tools and techniques to dual-processor AMD and Intel server-class machines, and we are now in the process of porting PowerPack to a cluster of AMD dual-processor, dual-core servers. Nonetheless, the software is in its early stages.

PowerPack has helped us identify how and when to conserve energy in scientific applications without significantly impacting performance. However, our manual approach requires user intervention and understanding of application performance to be effective. Parallel programming is already difficult, and legacy applications will not benefit from the need to recompile to conserve energy.

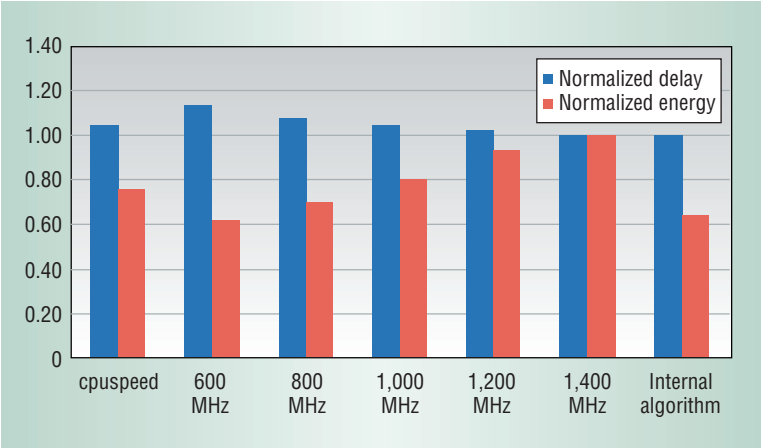


Figure 4. Normalized delay and energy values for NAS FT benchmark on PowerPack. Delay greater than 1 indicates performance loss; energy consumption less than 1 indicates energy savings.

We are currently exploring ways to transparently automate DVS control for scientific applications. For example, we have created a power-aware MPI that transparently throttles DVS during MPI communications; however, communication frequency and power transition costs typically reduce performance considerably. Our current trace-based approach can be automated with existing parallel tools but is not transparent. We are therefore incorporating our $\log_n P$ model[17] of parallel computation into PowerPack to predict runtime performance and thereby transparently automate DVS throttling.

Power-aware components such as disks, network cards, and memory offer additional opportunities for power and energy savings in clusters. However, challenges arise in coordinating the use of these components to conserve energy without seriously affecting performance. Researchers have shown that the effects of coordinating multiple components for interactive workloads are often unpredictable.[18] We plan to explore the tradeoffs of this approach in the future.

Critics of our power-aware approach may argue that operating cost and reliability are not critically important because supercomputers are primarily designed for high performance; slowing peak speed down to conserve power and cost is thus tantamount to designing a lower-performance machine. However, such logic is flawed.

The performance diversity of applications today implies that peak performance is sometimes but not always needed. Further, the number of system components is increasing nearly exponentially and component reliability is not improving as rapidly, making hardware failures more common and large systems less useful. Finally, operating cost and reli-

ability are critical design factors if they prohibit creation of the highest-performance machine possible using commodity components under budget and space constraints.

Today, supercomputers are built from components that offer the highest performance from the commodity marketplace. We cannot rely solely on hardware to reduce the power consumed by cluster-based systems. Increasingly, these components have power-aware features. Only by exploiting these features using system software such as PowerPack is it possible to address the power-performance efficiency gap and maintain exponential increases in parallel application performance. ■

Acknowledgment

PowerPack is an open source project sponsored by the National Science Foundation (CCF #0347683) and the US Department of Energy (DE-FG02-04ER25608). Source code is available for download at the SCAPE Laboratory Web site (http://scape.cs.vt.edu).

References

1. J.C. Phillips et al., "NAMD: Biomolecular Simulation on Thousands of Processors," *Proc. 2002 ACM/IEEE Conf. Supercomputing,* IEEE CS Press, 2002, p. 36.
2. F.S. Preston, *A Petaflops Era Computing Analysis,* tech. report CR-1998-207652, NASA, Mar. 1998; http://adipor.members.epn.ba/p2/NASA.pdf.
3. D.H. Bailey, "Performance of Future High-End Computers," presentation, DOE Mission Computing Conf., 2003; http://perc.nersc.gov/docs/Workshop_05_03/dhb-future-perf.pdf.
4. S.C. Jardin, ed., *DOE Greenbook: Needs and Directions in High Performance Computing for the Office of Science,* tech. report PPPL-4090, Princeton Plasma Physics Lab., June 2005; www.nersc.gov/news/greenbook/N5greenbook-print.pdf.
5. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. 27th Ann. Int'l Symp. Computer Architecture,* IEEE CS Press, 2000, pp. 83-94; www.eecs.harvard.edu/~dbrooks/isca2000.pdf.
6. M.J. Irwin, M. Kandemir, and N. Vijaykrishnan, "SimplePower: A Cycle-Accurate Energy Simulator," *IEEE CS Technical Committee on Computer Architecture Newsletter,* Jan. 2001; http://tab.computer.org/tcca/NEWS/jan2001/irwin.pdf.
7. S. Gurumurthi et al., "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach," *Proc. 8th Int'l Symp. High-Performance Computer Architecture,* IEEE CS Press, 2002, pp. 141-150; www.cs.virginia.edu/~gurumurthi/papers/gurumurthi_softwatt.pdf.
8. J. Flinn and M. Satyanarayanan, "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," *Proc. 2nd IEEE Workshop Mobile Computer Systems and Applications,* IEEE CS Press, 1999, pp. 2-10.
9. R. Joseph and M. Martonosi, "Run-time Power Estimation in High-Performance Microprocessors," *Proc. 2002 Int'l Symp. Low-Power Electronics and Design,* ACM Press, 2001, pp. 135-140; http://parapet.ee.princeton.edu/papers/rjoseph-islped2001.pdf.
10. Intel Corp., *Intel 80200 Processor Based on Intel XScale Microarchitecture,* developer's manual, Mar. 2003; www.intel.com/design/iio/manuals/27341103.pdf.
11. D. Lammers, "IDF: Mobile Rambus Spec Unveiled," *EE Times Online,* 25 Feb. 1999; www.eet.com/story/OEG19990225S0016.
12. L. Benini and G. De Micheli, "System-Level Power Optimization: Techniques and Tools," *ACM Trans. Design Automation of Electronic Systems,* vol. 5, no. 2, 1999, pp. 115-192.
13. C-H. Hsu and U. Kremer, "The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction," *Proc. ACM SIGPLAN 2003 Conf. Programming Language Design and Implementation,* ACM Press, 2003, pp. 38-48.
14. E.V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," *Proc. 17th Int'l Conf. Supercomputing,* ACM Press, 2003, pp. 86-97.
15. X. Feng, R. Ge, and K.W. Cameron, "Power and Energy of Scientific Applications on Distributed Systems," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp.,* IEEE CS Press, 2005, p. 34.
16. K.W. Cameron, X. Feng, and R. Ge, "Performance- and Energy-Conscious Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters," to appear in *Proc. 2005 ACM/IEEE Conf. Supercomputing,* IEEE CS Press, 2005.
17. K.W. Cameron and R. Ge, "Predicting and Evaluating Distributed Communication Performance," *Proc. 2004 ACM/IEEE Conf. Supercomputing,* IEEE CS Press, 2004, p. 43.
18. X. Fan, C.S. Ellis, and A.R. Lebeck, "The Synergy between Power-Aware Memory Systems and Processor Voltage Scaling," *Proc. 3rd Int'l Workshop Power-Aware Computing Systems,* LNCS 3164, Springer-Verlag, 2003, pp. 164-179; www.cs.duke.edu/~alvy/papers/dvs-pacs03.pdf.

Kirk W. Cameron *is an associate professor in the Department of Computer Science and director of the Scalable Performance (SCAPE) Laboratory at Virginia Polytechnic Institute and State University. His research interests include high-performance and grid computing, parallel and distributed systems, computer architecture, power-aware systems, and performance evaluation and prediction. Cameron received a PhD in computer science from Louisiana State University. He is a member of the IEEE and the IEEE Computer Society. Contact him at cameron@vt.edu.*

Rong Ge *is a PhD candidate in the Department of Computer Science and a researcher at the SCAPE Laboratory at Virginia Tech. Her research interests include performance modeling and analysis, parallel and distributed systems, power-aware systems, high-performance computing, and computational science. Ge received an MS in fluid mechanics from Tsinghua University, China, and an MS in computer science from the University of South Carolina. She is a member of the ACM and Upsilon Pi Epsilon. Contact her at ge@cs.vt.edu.*

Xizhou Feng *is a PhD candidate in the Department of Computer Science and Engineering at theUniversity of South Carolina and a researcher at the SCAPE Laboratory at Virginia Tech. His research interests include parallel and distributed systems, high-performance and grid computing, algorithms, bioinformatics, and software engineering. Feng received an MS in engineering thermophysics from Tsinghua University. He is a member of the IEEE and the IEEE Computer Society. Contact him at fengx@cs.vt.edu.*

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING

Learn more about this new publication and become a subscriber today.

www.computer.org/tdsc

Learn how others are achieving systems and networks design and development that are dependable and secure to the desired degree, without compromising performance.

This new journal provides original results in research, design, and development of dependable, secure computing methodologies, strategies, and systems including:

- Architecture for secure systems
- Intrusion detection and error tolerance
- Firewall and network technologies
- Modeling and prediction
- Emerging technologies

Publishing quarterly
Member rate: $31
Institutional rate: $285

IEEE
IEEE COMPUTER SOCIETY