PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications

Rong Ge, *Member*, *IEEE*, Xizhou Feng, *Member*, *IEEE*, Shuaiwen Song, Hung-Ching Chang, *Student Member*, *IEEE*, Dong Li, and Kirk W. Cameron, *Member*, *IEEE*

Abstract—Energy efficiency is a major concern in modern high-performance computing system design. In the past few years, there has been mounting evidence that power usage limits system scale and computing density, and thus, ultimately system performance. However, despite the impact of power and energy on the computer systems community, few studies provide insight to where and how power is consumed on high-performance systems and applications. In previous work, we designed a framework called PowerPack that was the first tool to isolate the power consumption of devices including disks, memory, NICs, and processors in a high-performance cluster and correlate these measurements to application functions. In this work, we extend our framework to support systems with multicore, multiprocessor-based nodes, and then provide in-depth analyses of the energy consumption of parallel applications on clusters of these systems. These analyses include the impacts of chip multiprocessing on power and energy efficiency, and its interaction with application executions. In addition, we use PowerPack to study the power dynamics and energy efficiencies of dynamic voltage and frequency scaling (DVFS) techniques on clusters. Our experiments reveal conclusively how intelligent DVFS scheduling can enhance system energy efficiency while maintaining performance.

Index Terms—Distributed system, CMP-based cluster, energy efficiency, power measurement, system tools, power management, dynamic voltage and frequency scaling.

1 Introduction

Power and energy are primary concerns in modern high-performance computing system design. Operational costs for powering and cooling large-scale systems will soon exceed acquisition costs [26]. Recent architectures such as IBM Blue Gene/L use low-power processor designs to significantly boost the power efficiency of large-scale systems, but still require power budgets of several megawatts. Today, large-scale systems and the computing centers that house them are consuming massive amounts of energy (typically 5-10 megawatts in total) to satisfy the needs of advanced computational science. Consequently, national laboratories such as ORNL have to commission substations from regional power companies to support their growing energy needs [2]. The addition of power substations to support future systems is costly and takes years of planning and coordination. The need for high-performance computing power efficiency is further amplified when considered in the context of looming US policy changes driven by concerns about the energy crisis and global warming [10].

To improve the energy efficiency of high-performance computing systems and applications, it is critical to profile the power consumption of real systems and applications at fine granularity. Many factors influence power and energy consumption in high-performance systems, including each component's electrical specification, the system usage characteristics of the applications, and system software. In addition, power consumption and application performance are tightly coupled and often conflicting and complex. Improving energy efficiency without negatively affecting the performance of an application is challenging. Thus, we need direct power measurement to understand how each system component contributes to total system power consumption. Nevertheless, measuring component power alone is not sufficient to improve our understanding of the interaction between application performance and power consumption. Techniques are needed to correlate the power consumption of a component with the software executing on the system. Fine-grain application and component-level profiling can be used to identify how, when, and where power is consumed by high-performance systems and applications.

1.1 Related Work

Despite the need for power monitoring infrastructures, few tools provide fine-grain power/energy profiling correlated to applications and multiple system components. To date, none have profiled multicore processors or chip multiprocessing (CMP) and dynamic voltage and frequency scaling (DVFS) at this level of granularity. The dearth of tools is primarily due to a lack of readily available, standardized hardware sensors in today's computer servers that are used as the basic building blocks of most high-performance systems.

- *R. Ge is with the Department of Mathematics, Statistics, and Computer Science, Marquette University, Milwaukee, WI 53233. E-mail: rong.ge@marquette.edu.*
- *X. Feng is with the Department of Mathematics, Statistics, and Computer Science, Marquette University, Milwaukee, WI 53233, and with the Network Dynamics and Simulation Science laboratory at Virginia Tech, Blacksburg, VA 24060. E-mail: xizhou.feng@marquette.edu.*
- *S. Song, H.-C. Chang, D. Li, and K.W. Cameron are with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24060. E-mail: {s562673, hcchang}@vt.edu, {lid, cameron}@cs.vt.edu.*

Manuscript received 31 Oct. 2008; revised 17 Mar. 2009; accepted 23 Mar. 2009; published online 23 Apr. 2009.
Recommended for acceptance by F. Petrini.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-10-0436.
Digital Object Identifier no. 10.1109/TPDS.2009.76.

1045-9219/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

Most existing power profiling research is focused on a single computer component such as processor [6], [36], disk [37], memory [36], and networking interface [35]. The approaches used include simulation [6], [11], [21], [29], [35], [36], [37], performance-profile-based estimation [27], [30], and direct measurement [5], [13], [27], [30]. The simulation approach refers to modeling power consumption of various components at the microarchitectural level during design time [6], [33], [34], [36]. The performance-profile-based approach estimates component power consumption using performance logs [27], [30]. Both approaches typically require some direct measurement for parameter calibration and result validation. Although these techniques can provide useful insights such as major power consuming performance events, they are limited to a single computing component without revealing the power profile of the entire system. Related efforts estimate weighted power and energy consumption of threads or applications on a single-node computer system by counting performance events and system activities from system logs or simulation traces [5], [8], [14], [20]. Though useful, these approaches provide little information about the energy usage of individual components.

Some have studied the power efficiency of parallel and distributed systems, at the system or building level [25], [31], [32]. One technique is to measure power using proprietary hardware to instrument power feeds to the infrastructure. For example, the IBM PowerExecutive toolkit [25] uses this approach to monitor the power draw on selected IBM Blades and servers. Another technique is to measure power via panels on the power distribution units. Researchers at Lawrence Berkeley National Laboratory used this technique to investigate the power usage of a Cray XT4 system [31] as well as early studies measuring power for full data centers [32]. Other less scientific approaches, used in the past for ASC Terascale facilities [3], estimate system power consumption based on prior experiences or rules-of-thumb at design and acquisition time. Early studies on power and energy consumption of large-scale distributed systems have led to revised policy considerations at the US Department of Energy and US Environmental Protection Agency (EPA) [1]. However, course-grained power profiling is not particularly useful for determining exactly where and how power is consumed by an application and the individual components in a distributed system.

1.2 Contributions of This Work

To address the need for fine-grained power/energy profiling on typical parallel and distributed systems (i.e., computer clusters), we created a power/energy/performance profiling infrastructure named PowerPack, and used it to evaluate energy efficiency and power-aware techniques for parallel applications [12]. PowerPack is a combination of hardware (e.g., sensors and digital meters) and software (e.g., drivers, instrumentation APIs, benchmarks, and analysis tools) that achieves automatic power and energy profiling at component and code segment granularity. PowerPack provides correlations between system/application activities and system power/energy consumptions on distributed systems. Due to increased system complexity and sensitivity, in this work, we redesigned PowerPack to support emerging multicore, multiprocessor systems with advanced power management capabilities such as DVFS.

Particularly, in this paper, we have made the following contributions:

1. We design, implement, and validate a first-of-its-kind framework for accurate and scalable power profiling and evaluation of multicore, multiprocessor-based distributed systems and applications.
2. Using our framework, we investigate power and energy profiles of parallel scientific applications at a level of detail previously not possible and at a scale not found in the extant literature. We compare and contrast the evolution of power budgets from single-core, single-processor, to multicore, multiprocessor systems.
3. We quantitatively show and analyze how DVFS changes power and energy profiles of applications and their energy efficiency on emergent power scalable systems. We emphasize that profiling the changes in power consumption at this level of granularity has not been previously presented in the extant literature (including in our previous work).

The remainder of this paper is organized as follows: Section 2 presents the design, implementation, and validation of the PowerPack framework. Then, as a case study and proof of concept, the detailed power and energy profiles of the NPB benchmarks on a multicore, multiprocessor-based cluster are provided in Section 3. Next, we analyze the power-performance efficiency of the NPB benchmarks for different single-chip multiprocessing configurations in Section 4. Section 5 extends PowerPack to power-performance evaluation of DVFS scheduling. Finally, Section 6 summarizes our findings and future work.

2 The PowerPack Framework

2.1 Overview

As described above, the PowerPack framework is a set of toolkits composed of both hardware and software components. The hardware components include sensors, meters, circuits, and data acquisition devices that enable direct power measurement and instrumentation. The software components include drivers for various meters and sensors, and user-level APIs for controlling power profiling and code synchronization. Together, these hardware and software components enable two unique features: 1) fine-grained component-level power measurement and 2) automatic synchronization between power profiles and application code segments.

Fig. 1 shows a typical PowerPack deployment for power profiling on a high-performance cluster. PowerPack simultaneously measures system power by component. To obtain isolated component power, we tap a precision sensing resistor into each individual DC power line (explained in detail later in this section) and then measure the voltage difference at two ends of the resistor using a digital meter. All system DC power lines are measured simultaneously and used to derive component power according to a derived mapping between lines and components. To obtain

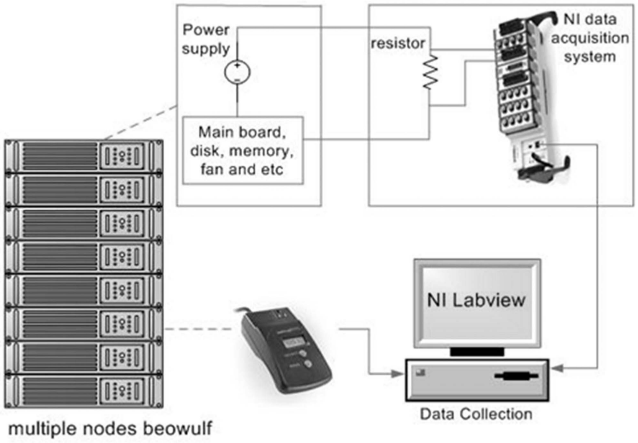


Fig. 1. An implementation of the PowerPack framework on a 9-node cluster with 18 AMD dual-core Opteron processors. AC Power is monitored at the node level using a Watt's Up Pro power meter and/or the ACPI standard interface. Within a node, DC power is monitored for each system component using the NI data acquisition system. PowerPack software automatically synchronizes system and application software events with power profiles. PowerPack AC and DC power profiling software is portable and has been tested on over half a dozen systems.

total system power including AC/DC conversion, AC power is measured via an inline sensor device between the system power cable and the wall. The current version of PowerPack supports various types of power sensors (or meters): 1) National Instruments data acquisition system such as Analog Input Module NI 9205NI and cDAQ chassis NI cDAQ9172 for DC power measurement; 2) Watt's Up Pro power meter for AC power measurement; and 3) Advanced Configuration and Power Interface (ACPI) enabled power supply. The combination of DC measurements and AC measurements allows us to capture and isolate total power usage including inefficiencies in AC to DC conversion. This redundant set of measurements allows us to verify the accuracy of each technique.

The software contained in PowerPack serves two purposes: online data recording and postmortem data analysis. The online data recording components record meter readings and synchronize power profiling with code segments. Previous versions of PowerPack used client-server structures to synchronize power profiling with code segments, i.e., the data collection servers polled the meters and recorded data; then, the client API triggered the server to record data. For improved scalability, the new implementation of PowerPack uses a time-stamp-based approach to synchronize multiple data streams from various meters, sensors, and performance instruments. We note that the gathering of power profiling data is purposely "out-of-band," meaning the data is collected, collated, and analyzed on a separate computer (see Fig. 1). Such measurement ensures power profiling does not impact the system under test. Postmortem data analysis software processes the data and creates the power profiles of applications, systems, and components. The results in this work were obtained using NI devices, Watt's Up Pro power meters, and Baytech power distribution units. However, PowerPack also supports meters from Yokogawa and RadioShack [12]. The latest version of PowerPack software framework leverages any hardware power measurement device to control and correlate the measured data to system components, system software events, and application source code.

PowerPack directly measures one node at a time. To obtain in-depth power consumption of an entire cluster, we use a node remapping approach. Node remapping works as follows: Suppose we are running a parallel application on M nodes. We fix the measurement equipment to one physical node (e.g., node #1) and repeatedly run the same workload M times. Each time we map the tested physical node to a different virtual node. Since all slave nodes are identical (as they should be and we experimentally confirmed for homogeneous clusters), we use the M independent measurements on one node to emulate one measurement on M nodes. For fine-grain analysis of a heterogeneous environment, we can instrument one version of each type of node for coverage.

Except where otherwise noted, all results in this paper were obtained from a 9-node cluster named Dori. Each Dori node contains two dual-core AMD Opteron processors running at 1.8 GHz , six 1 GB SDRAM modules, one Western Digital WD800 SATA hard drive, one Tyan Thunder S2882 motherboard, two CPU fans, and two system fans. Though we limit our results to this system, we have ported PowerPack to a number of other systems with processors varying from Pentium II through Pentium IV and AMD Athlon [12]. The current dual-core dual-processor system was selected to further demonstrate the effectiveness of PowerPack for profiling multicore architectures that have begun to dominate high-performance cluster deployments [28]. While our discussion is specific, our approach is portable to any commodity-based cluster with a standard power supply (e.g., ATX and BTX), and any number or types of processors, disks, memory, and NICs.

2.2 Fine-Grain Systematic Power Measurement

PowerPack uses direct or derived measurements to isolate components within nodal power profiles. Specifically, we isolate CPU, memory, disk, motherboard, CPU fans, and system fans. Using combined AC and DC measurements, we can also isolate the power supply. The remaining components are treated as "others," which includes onboard video card, keyboard, onboard network adapter, etc. Our measurement approach is as follows: if a component is powered through several individual pins, we measure power consumption through each pin and use the sum as the component power; if two or more components are powered through shared pins, we observe the changes on all pins while adding/removing components and running different microbenchmarks to infer the mapping between components and pins. Specifically, here is the technique used for each component.

CPU power. According to our experiments and confirmed by the ATX power supply design guide, the four cores are powered through four +12 VDC pins. Thus, we can profile CPU power consumption by measuring all +12 VDC pins directly.

Disk power. The disk is connected to a peripheral power connection independently and Powered by one +12 VDC pin and one +5 VDC pin. By directly measuring both +12 VDC and +5 VDC pins, we can profile disk power consumption directly.

Memory power. Memory modules are powered through four +5 VDC pins. The power consumption for memory is directly measured from these four pins. In previous work, we relied on a linear extrapolation technique to deduce memory power consumption. For systems where memory is not powered through dedicated pins, we recommend using our previous linear extrapolation techniques [12] to isolate memory power consumption.

Motherboard power. NIC and other onboard components are powered through +3.3 VDC pins. It is challenging to separate NIC power consumption from other onboard components directly. However, our measurements indicate the onboard NIC only consumes a minimal amount of power under maximum load. We verified this by monitoring the total system power consumption changes under saturated network card bandwidth. We verified these findings by consulting the documentation of the NIC. Thus, based on our empirical measurements, we approximate NIC power with a constant value. For simplicity, we treat the power consumption of other onboard components as constant too. We justify this assumption since worker nodes running parallel scientific applications on computational clusters typically do not access onboard components (such as the video card) on slave nodes. Our measurements show that dynamic power usage from the memory and processors far exceed NIC and motherboard power consumption. As mentioned, PowerPack isolates the energy use for CPU, memory, NIC, and disk. Profiling and analysis of other components including PCI devices is left to future work.

CPU and system fans. Integrating multiple cores into a single computing node demands more powerful cooling. In the system under test, there are two CPU fans, with one for each processor, and two system fans on each node of our dual-core dual-processor cluster. Each fan is powered by a +12 VDC pin and a +5 VDC pin.

2.3 Automatic Power Profiling and Code Synchronization

Once the manual instrumentation setup is complete, the process of obtaining and controlling power profiling is fully automated by software. In fact, the PowerPack software includes all the microbenchmarks necessary to isolate power lines in the instrumented node. Additionally, all the experimental data gathered herein are obtained remotely via local intranet. In this section, we describe the software components of PowerPack that automate the entire profiling process and correlate the power profiles with application source code. PowerPack provides a suite of API calls for the application to control and communicate with a meter control process.

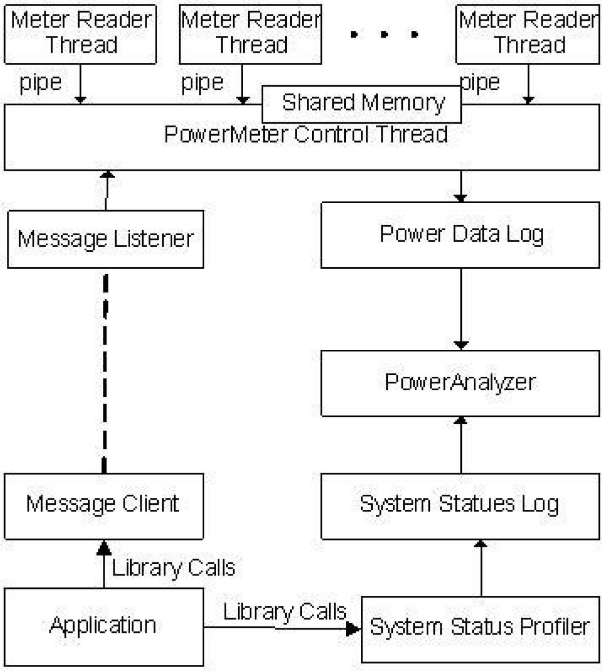


Fig. 2. Software architecture for automatic power and energy profiling. Through modular design, our software automates the process of profiling power data and correlates the results to source code in a distributed system.

The structure of the profiling software is shown in Fig. 2, in which the data collection computer executes a meter control thread and a group of meter read threads. Each meter read thread corresponds to one digital meter. The meter read threads collect readings from the meters and send them to the meter control thread. The meter control thread monitors messages from applications running on the cluster and modifies shared variables of the meter read threads according to messages received.

To synchronize the live power profiling process with the running application, profiled applications trigger message operations through a set of user-level APIs or library calls informing the meter control thread to take corresponding actions to annotate the power profile. Thus, by inserting the power profile API pmeter_start_session and pmeter_end_session before and after the code region of interest, we are able to map the power profile to the source code. In Fig. 3, we list a commonly used subset of the power profile API in PowerPack.

We extended the above approach to work with existing commercial data acquisition software such as the NI LabView system. Instead of writing our own drivers for this meter, we record data samples coming from the NI cDAQ9172 chassis with user-configured LabView modules, then align and merge these samples with other data sources

```
pmeter_init ( char *ip_address, int *port);
//connect to meter control thread

pmeter_log (char *log_file, int *option );
//set power profile log file and options

pmeter_start_session ( char *session_label );
//start a new profile session and label it

pmeter_end_seesion ( );
//stop current profile session

pmeter_finalize( );
//disconnect from the meter control thread
```

Fig. 3. The commonly used PowerPack power meter profile API.

containing synchronization information (e.g., AC power profile from the Watts Up power meter).

3 Component and System Power Profiles

3.1 Systemwide Power Distribution

We begin our analysis with systemwide power distribution for sequential applications on a single compute node. Fig. 4 shows the snapshots of power distribution under two kinds of scenarios—case 1: no user application is running on the system and case 2: the system is running one of three applications from the SPEC CPU 2000 benchmark suite [24] (164.gzip and 171.swim) and the Linux standard file copy command (cp) programs. These three benchmarks are computation intensive (164.gzip), memory access intensive (171.swim), and disk access intensive (cp), respectively. Compared against the power profile when the system is idle, their power profiles reveal how a component's powers change when stressed. For case 2, the system is running four instances of the same program so that each of the four cores simultaneously executes one of them and the load is symmetric. We obtain the following observations from the figure:

1. Since different workloads stress different components, both system power and individual component power vary with workload. Component usage is also reflected in the power profile.
2. The system power under zero user workload (152.2 watts) is more than 72.9 percent of the total system power under load. Reducing power consumption of the power supply and fans could save significant energy. We note that cheap, inefficient power supplies are typical in clusters that use commodity parts. Power supplies traditionally account for less than 2 percent of the acquisition budget of a server node. Improving power supply and fan efficiencies is important but well beyond the scope of our work.
3. When the system is under load, CPU power dominates (e.g., for 164.gzip, CPU power is 56 percent of system power). However, depending on the workload characteristics, disk and memory may also be significant consumers of the total power budget. The components that dominate the power budget in a system should be the first targets of optimizations for power and energy reduction.

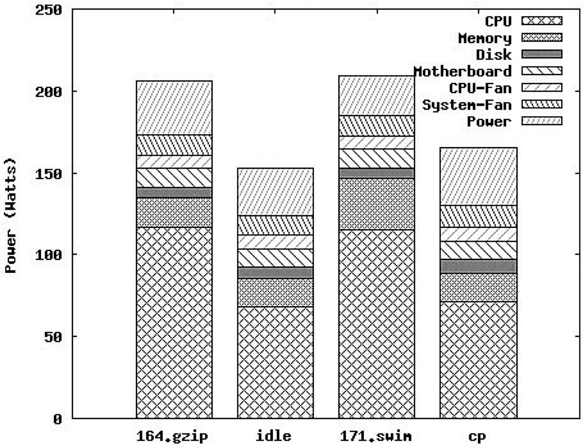


Fig. 4. Power distribution for a single node under different workloads: (a) CPU-bounded workload 164.zip, (b) zero user workload or idle, (c) memory-bounded workload 171.swim, and (d) disk-bounded workload cp.

3.2 Power Profiles of Parallel Applications

As a case study and proof of concept, we profile the power and energy consumption of the NAS parallel benchmarks (Version 3.2) on our cluster using the PowerPack framework. The NAS parallel benchmarks [4] consist of five kernels and three applications that mimic the computation and data movement characteristics of parallel computational fluid dynamics (CFD) applications. We measured CPU, memory, disk, CPU fan, and motherboard power consumption over time for different benchmarks running on different numbers of compute nodes.

3.2.1 Nodal Power Profile of the FT Benchmark

We select a parallel FT benchmark in this study and show its power profile. The FT benchmark exhibits obvious alternating computation phase, memory phase, and communication phase. Therefore, its power profile reveals how components' power change with execution phases for a single application. In particular, the FT benchmark begins with a warm up phase and an initialization phase followed by a certain number of iterations. Each iteration consists of computation (fft), memory access (matrix transpose), all-to-all communication, memory access, computation, and a reduce communication. In Fig. 5, we plot the power profiles of the NPB FT benchmark with problem size B during the first several iterations when running on 16 cores of four nodes. From this point, the illustrated power profiles of a parallel application are for one of the computing nodes unless explicitly stated otherwise.

The power profiles are almost identical for all iterations in which spikes and valleys occur with regular patterns coinciding with the characteristics of different computation stages. In other words, there exist "power phases" corresponding to the workload phases. The CPU power consumption varies from 119 watts in the computation stage to 72 watts in all-to-all communication stage. The memory power consumption varies from 28 watts in the memory stage to 18 watts in communication stage. The power profiles of CPU and memory are

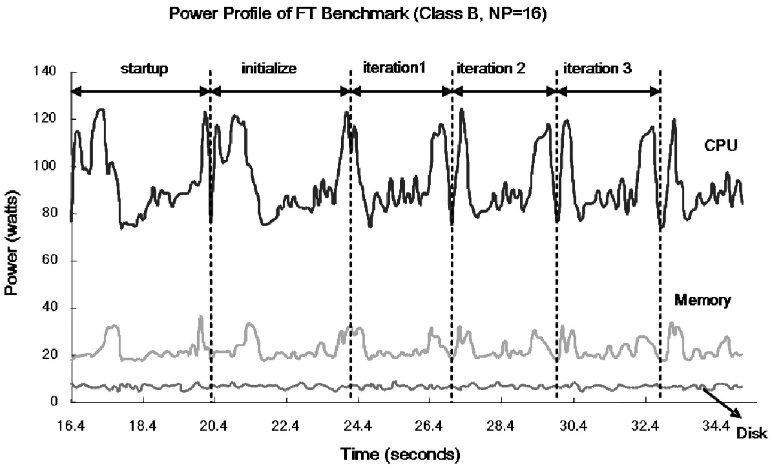


Fig. 5. Power use on one of four nodes for the FT benchmark, class B workload. Here, each node runs four processes with one process per core.

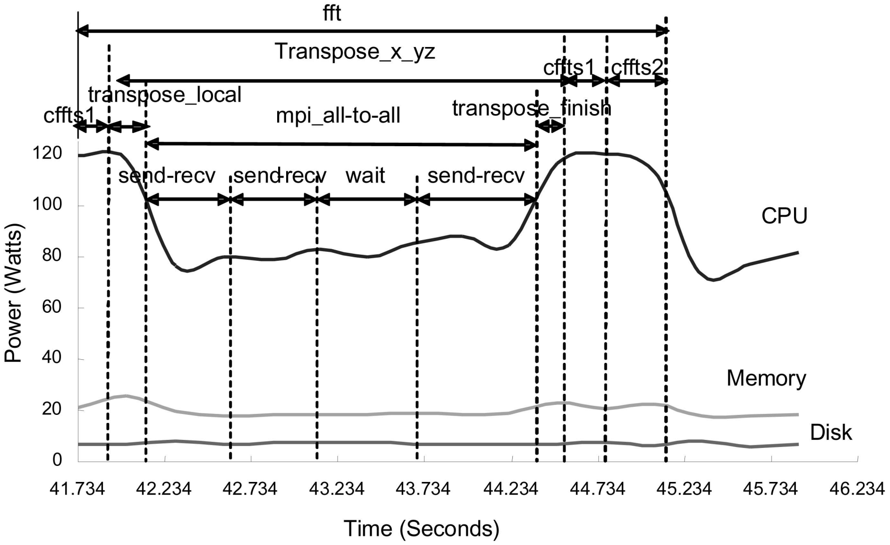


Fig. 6. Mapping between power profile and code segments for FT benchmark with Class B, 16 processes on four nodes. Using code analysis and code power profile synchronization mechanisms provided in PowerPack, we can map the power phases to each individual function and perform detailed power efficiency analysis on selected code segments. This is useful when exploring function-level power-performance optimization.

related: when memory power increases, CPU power typically decreases and vice versa.

We also observed fairly constant power consumption for the disk since the FT benchmark requires few disk accesses. The power consumed by the motherboard (NIC + other chipset components) and fans (CPU and system fans) is constant. For simplification, we will not discuss the disk, motherboard, and fans power consumption in succeeding discussions since none of the benchmarks under study task the disk extensively and we observed that the motherboard and fan power consumption vary little across applications.

3.2.2 Mapping Power Profile to Source Code

PowerPack can correlate an application's power profile to its source code, thereby allowing us to study the power behavior of a specific function or code segment. Fig. 6 shows the mapping between the power profile and the major functions of the FT benchmark. From this figure, we observe the power variations for functions that are compute intensive (cffts-1), memory intensive (transpose-local), and communication intensive (all-to-all). This information can be used to target code segments of application for power reduction. The level of granularity allows us to quantify the ability of the component to minimize energy consumption when not in use. For example, the CPU still consumes almost 41 percent of its peak power during sending and receiving communications. This is likely due to spin locks running on the processor while blocked waiting for a data transmission.

3.2.3 Power Profile Variation with Node and System Size

Scaling a fixed workload to an increasing number of computing cores and nodes may change the workload characteristics (the percentage of CPU computation, memory access, and message communication). We use PowerPack to assess if the change is reflected in the power profiles. We have profiled the power consumption for all NPB benchmarks with different combinations of number of computing cores (up to 16), number of nodes, and problem sizes. In Fig. 7, we provide an overview of the profile variations under different system setups for benchmarks FT, EP, and MG.

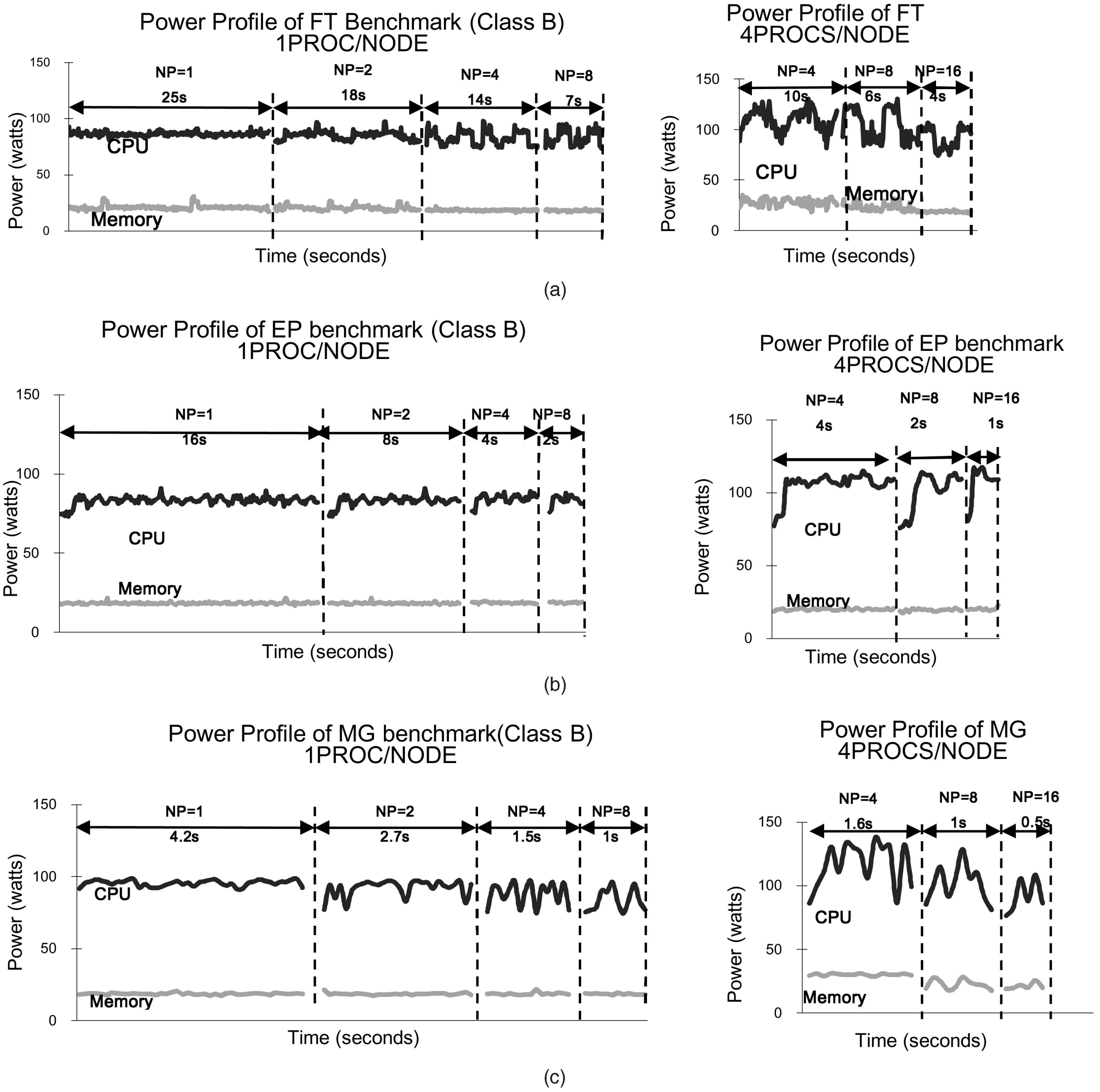


Fig. 7. FT, EP, and MG on different numbers of nodes. NP stands for the number of total simultaneous processes for the program. The figures on the left are with setup ***UP***, and those on the right are with setup ***CMP***. Scaling a fixed workload to increasing number of nodes may change the workload characteristics (the percentage of CPU computation, memory access, and message communication) and the change is reflected in the power profile. (a) Power profiles on one node for FT benchmark. (b) Power profiles on one node for EP benchmark. (c) Power profiles on one node for MG benchmark.

Two parallel computing setups are studied for each benchmark. The first setup assigns one process per node. Under this setup, only one out of the four cores on each used node executes the application while the other three cores are idle. We denote this setup ***UP*** (for chip Uniprocessing). The second setup assigns four processes per node. Under this setup, all the four cores on a node are involved in executing the application. We denote this setup ***CMP*** (for Chip Multiprocessing). Fig. 7 shows segments of synchronized power profiles under these two setups with various numbers of computing cores and nodes; all power profiles correspond to the same computing phase in the application on the same node.

Fig. 7a shows the power profiles on one of the computing nodes for FT with two iterations. For both setups of ***UP*** and ***CMP***, the duration of each phase shortens as the number of computing cores increases since parallelism results in some speedup. Also, since the workload is broken into smaller pieces, CPU and memory peak power decrease as the number of computing cores increase. Although the CPU and memory profiles with two different setups share the same trend as the number of computing cores increases, the absolute power values per node under ***UP*** setup are smaller than those under ***CMP*** setup. For example, when the total number of processors is four ($\mathrm{NP} = 4$), the peak CPU and memory power under ***UP*** setup can be 23 watts and 12 watts smaller, respectively. With ***UP***, a node consumes less power since only one of the four available cores executes FT, while with ***CMP*** all the four cores execute FT. Although with ***UP***, a single node consumes less power, the total system over all the nodes consumes more power since ***UP*** needs four times the node count as needed by ***CMP***. Total system energy consumption and efficiency will be discussed in the next section.

The embarrassingly parallel benchmark (EP) is essentially computation intensive and communication free. Hence, CPU intensity doesn't change as the number of computing cores increases. The fairly constant CPU power consumption in Fig. 7b reflects this property under both setups. However, the absolute CPU power consumption with ***UP*** setup is less than that under ***CMP*** setup. For example, when the total number of computing cores is four, the CPU power consumption under ***UP*** setup is 30 watts smaller. Again, this is explained by the fact that ***UP*** only uses one of the four available cores on a node while ***CMP*** uses all the four codes for executing EP. One interesting observation in this case is memory power doesn't noticeably change with the number of computing cores or with the setup. This is explained by the minimal memory accesses of EP. As we shall see in the next section, fixing the total number of computing cores across ***UP*** and ***CMP*** setups results in the same performance for EP while the total energy consumption under the ***CMP*** setup is less than half of the energy using ***UP***.

The multigrid benchmark (MG) is more computation intensive than FT and also exhibits computational load imbalance across computing cores; i.e., each run exhibits a slightly different power profile depending on the data set it is assigned. Generally though, the average power for MG decreases with an increase in the number of nodes for both CPU and memory. This is due to changes in the computation to communication ratio as the number of computing cores increases. Parallel overhead increases with the number of computing cores which results in a dampening of the power consumption per node. When the total number of computing cores is four, the difference in CPU power for MG under the two setups is about 35 watts, the most among the three benchmarks shown in this section. Such significant difference in CPU power consumption indirectly reflects the computational intensity of MG. Again, we will discuss efficiency metrics for MG in the next section.

4 Energy Efficiency of Parallel Applications

In this section, we apply PowerPack to analyze the energy efficiency of parallel applications. While nodal power (P) describes the rate of energy consumption at a discrete point in time, energy (E) specifies the total number of joules spent in time interval (t_1, t_2), as a sum of products of average power ($\bar{P}$) and delay ($D = t_2 - t_1$) over all the computing nodes:

$$E = \sum_{1}^{\#nodes} \int_{t_1}^{t_2} P(t)dt = \sum_{1}^{\#nodes} \bar{P} \times D. \qquad (1)$$

4.1 Energy Scaling

Equation (1) specifies the relation between power, delay (the final measure of performance), and energy. To reduce energy, we need to reduce the delay, the average power, or both. Within the context of parallel processing, by increasing the number of processors, we can not only speed up the application's execution (decrease delay) but also increase the total power consumption. Depending on the scalability of the application, the energy consumed by an application may be constant, grow slowly, or grow very quickly with the number of processors.

For distributed parallel applications, we use two metrics to compare the energy-performance behavior of different parallel applications such as those of the NPB benchmarks: 1) the speedup (D_1/D_N) for performance scalability, where D_1 is the sequential execution time on one process, and D_N is the parallel execution time with N computing cores in parallel and 2) normalized system energy (E_N/E_1) for energy scalability, or the ratio of energy on a parallel system of N computing cores to energy using sequential execution and a single process.

As in the previous section, we study the performance and energy scalability under two parallel computing setups: ***UP*** and ***CMP***, and compare the values between these two setups. Fig. 8 shows the variations of speedup and energy with the number of total parallel computing cores for the two setups. We observe that energy consumption using the ***CMP*** setup is always less than using the ***UP*** setup for the system under test, while speedups are more complicated. In detail, we identify three energy-performance categories for the codes we measured.

Type I. Under either of the two setups, energy remains constant or approximately constant while speedup increases linearly with the number of parallel computing cores, as shown in Fig. 8a. Between the two setups, performance is about the same for a given number of computing cores but energy is less under ***CMP*** setup. EP, SP, LU, and BT exhibit this behavior. Such applications are computation intensive with minimal or overlapped communication; the execution time decreases proportionally with the number of computing cores; and chip multiprocessing provides the best energy efficiency (about 70 percent energy savings) with similar performance.

Type II. Under either of the two setups, both energy and speedup increase with the number of parallel computing cores, but speedup increases faster. Between the two setups, both speedup and energy consumption under ***UP*** setup are larger than those under ***CMP*** setup. MG and CG share this behavior as shown in Fig. 8b. Such applications are computation intensive, have a large memory footprint, and nonnegligible communication; parallel processing reduces their execution times with an associated energy increase; the large memory footprint causes speedup to decrease when the parallel processing setup (across nodes) shifts to multicore processing (within a node) since more processes share a fixed amount of physical memory.

Type III. Under both ***UP*** and ***CMP*** setups, speedup and energy consumption increase with the number of computing cores at comparable rates. Between the two setups, the speedup curves may cross, e.g., between $NP = 4$ and $NP = 8$ for FT benchmark, as shown in Fig. 8c. As in the other measured codes, the energy consumption under the ***UP*** setup is larger than that under the ***CMP*** setup for a given number of computing cores. FT and IS belong to type III. These applications are communication intensive; performance does not scale well with the number of parallel computing cores; any performance improvement is accompanied by energy

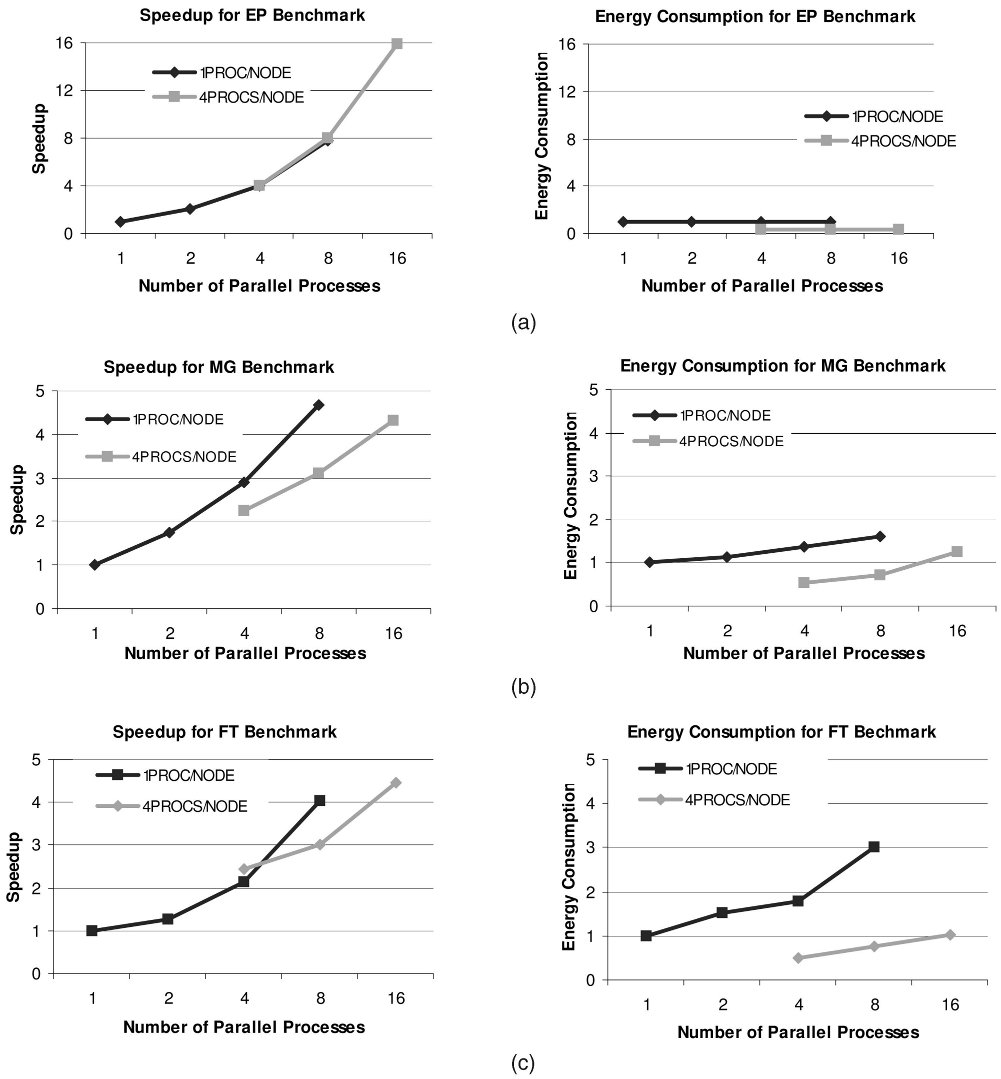


Fig. 8. Energy-performance efficiency. These graphs are normalized by values for performance (i.e., speedup) and total system energy under UP setup where only one process is running on one core of a processor. (a) EP shows linear performance improvement with constant energy consumption. (b) MG is capable of some speedup with the number of parallel processes with a corresponding increase in the amount of total system energy. (c) FT shows only minor performance improvement but relatively more increase in total system energy.

increase. Reducing communication cost can improve the overall performance for such applications, which is supported by the speedups at $NP = 4$. That is, when the parallel computing setup shifts form ***UP*** to ***CMP***, the speedup at $NP = 4$ increases because of reduced communication cost.

Our analysis indicates that energy scaling of parallel applications is strongly tied to parallel scalability. In other words, as applications have good scalability, they also make more efficient use of the energy when using more computing cores. In our study, communication is a major impacting factor, and large communication to computation ratio leads to less scalable performance and less efficient use of energy for both ***UP*** and ***CMP*** setups. Memory contention is another impacting factor that largely determines the difference in performance and energy scalability between ***UP*** and ***CMP*** setups. Our results show the scalability up to 16 cores. As the number of cores increases, the resulting performance and energy scalability will be decided by how communication to computation ratio and memory contention change with the number of cores.

4.2 Resource Scheduling

An application's energy scaling is dependent on its speedup or parallel efficiency. For certain applications such as FT and MG, we can achieve speedup by running on more cores but with increased total energy consumption. Our measurements indicate there are trade-offs between power, energy, and performance that should be considered to determine the best "operating points" or the best configurations in number of cores (NP) based on the user's needs. For performance-constrained systems, the best operating points will be those that minimize delay (D). For power-constrained systems, the best operating points will be those that minimize power (P) or energy (E). For systems where energy efficiency is optimized or power performance must be balanced, the choice of appropriate metric is whether the performance gain was worth the additional energy requirement. The energy-delay product ED^{α} (α is real number and $\alpha \geq 1$) is commonly used as a single metric to weigh the effects of power and performance for a given application under different configurations. The smaller ED^{α} the configuration achieves for an

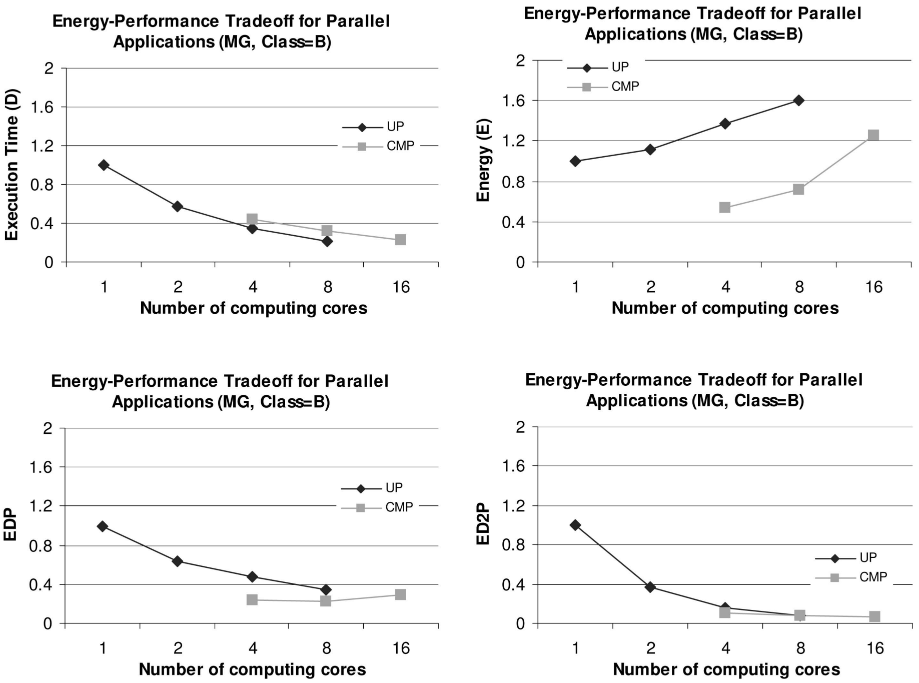


Fig. 9. Energy-performance trade-off of NPB benchmark MG on a high-performance cluster. We compare four different types of energy efficiency metrics: performance-oriented, energy-oriented, and performance-energy tradeoff in EDP and ED2P, under two process configurations (UP and CMP). EDP is the inverse of perf/J and ED2P is the inverse of $\mathrm{Perf}^2/\mathrm{J}$. Though the exact optimal operating points differ under different efficiency metrics, using multiple cores per node is more energy efficient in most cases.

application, the better the efficiency of this configuration for this application. EDP (the energy-delay product where $\alpha = 1$) and ED2P (the energy-delay^2 product where $\alpha = 2$) are popular metrics in the form of ED^α for computer clusters. EDP is equivalent to the inverse of performance per Joule (perf/J) and ED2P is equivalent to the inverse of performance square per Joule (perf^2/J).

We use NPB MG benchmark (class B) as an example to show the relationship between four metrics (normalized E and D, *EDP*, and *ED2P*) and the number of cores. Fig. 9 presents the values under ***UP*** and ***CMP*** setups. To minimize energy (E), we should schedule four cores on a single node to efficiently parallelize MG. To minimize delay (D), we should schedule eight computing cores to execute MG, where each of the eight cores resides on a different node. This setup achieves $4.67\times$ speedup. For energy efficiency, the EDP metric suggests scheduling eight cores on two nodes for $3.12\times$ speedup and $0.72\times$ energy cost. Using the ED2P metric suggests running with 16 cores on four nodes for $4.33\times$ speedup and $1.26\times$ energy cost. For accuracy, the average delay and energy consumption obtained from multiple runs are used in Fig. 9.

5 Power-Performance Efficiency for Power-Aware Cluster

So far, discussions have been limited to "conventional" distributed systems without specific, controllable power modes. Today, many distributed systems have various power modes available to conserve energy. For example, typical AMD Opteron and Intel Xeon processors can scale power through changing of a frequency and voltage pair; this technique is referred to as DVFS. The power consumption on such processors typically changes significantly with frequency [22].

DVFS has been used for power reduction and energy conservation in high-performance distributed systems [7], [15], [16], [18], [19], [23] by scaling down processor frequency during processor slackness, or when slower processor speed does not impact performance significantly. This work assumes the power consumed by the CPU dominates system power, and thus, scaling down frequency can significantly reduce system power. In this work, it is not our intent to study all the different types of CPU scheduling algorithms for effective use of DVFS. Our intention is to demonstrate the insight PowerPack provides to quantitatively explain the power-performance efficiency and energy conservation of applications using DVFS.

The AMD Opteron processors on our experimental cluster have two publicly available (i.e., exposed) frequencies 1,000 and 1,800 MHz. As typical in these types of systems, we were able to experimentally add three other frequencies for use: 1,200, 1,400, and 1,600 MHz. Table 1 shows the power-performance efficiency of the NPB benchmarks with 16 processes on four nodes. The columns "XXX MHz" refer to the CPU speed at which we run the applications. The rows "XX.C.16" refer to the NPB code with problem size C on 16 processes. For each application row (e.g., IS.C.16) and frequency column (e.g., 1,800), there is a pair of numbers in a top cell and a bottom cell. The number in the top cell is the normalized delay and the number in the bottom cell is the normalized energy. These numbers are normalized to the highest frequency (1,800 MHz). We observe that NPB applications show two categories of power-performance efficiency under different processor frequencies.

TABLE 1
Energy-Performance Profiles of NPB Parallel Benchmarks

Code	Frequency (MHz)				
	1800	1600	1400	1200	1000
IS.C.16	1.00	1.07	1.04	1.05	1.16
	1.00	0.97	0.90	0.83	0.95
FT.C.16	1.00	1.05	1.11	1.20	1.30
	1.00	0.95	0.92	0.88	0.99
CG.C.16	1.00	1.04	1.10	1.16	1.26
	1.00	0.95	0.91	0.87	0.97
EP.C.16	1.00	1.23	1.29	1.50	1.80
	1.00	1.00	1.03	1.05	1.29
LU.C.16	1.00	1.00	1.07	1.24	1.22
	1.00	0.92	0.89	0.83	0.92
MG.C.16	1.00	1.05	1.08	1.13	1.20
	1.00	0.97	0.92	0.87	0.95
BT.C.16	1.00	1.05	1.15	1.26	1.41
	1.00	0.95	0.93	0.89	1.03

The columns "XXX MHz" refer to the CPU speed at which we run the applications. The rows "XX.C.16" refer to the NPB code with problem size C on 16 processes. For each application row (e.g., IS.C.16) and frequency column (e.g., 1,800), there is a pair of numbers in a top cell and a bottom cell. The number in the top cell is the normalized delay and the number in the bottom cell is the normalized energy. These numbers are normalized to the highest frequency (1,800 MHz).

1. Decreasing CPU frequency causes a nearly proportional increase in application execution time and possible increase in energy consumption for applications, such as EP and BT.
2. Decreasing CPU frequency causes an increase in application execution time but saves energy. The percentage of energy savings may be comparable to, better than, or less than the percentage of performance degradation. All the other NPB applications belong to this category.

To quantitatively explain why DVFS has such different impacts on application power-performance efficiency, we use PowerPack fine-grain profiling information to study how voltage and frequency scaling impact the power consumption of each component, especially the CPU, during application execution.

First, we study the power profiles at various CPU frequencies when there is no user workload running on the system (idle), as shown in Fig. 10. When CPU frequency decreases from 1,800 to 1,000 MHz, CPU power consumption decreases from about 79 to about 53 Watts, and power consumption on the other components is unchanged. This confirms the typical assumption that scaling down CPU frequency reduces only CPU power but that as a percentage of total system energy the amount of power savings is significant. The 26 Watts CPU power drop from 1,800 to 1,000 MHz comes from both dynamic power and leakage power.

Next, we analyze the power profiles of EP. Fig. 11 shows the profiles of EP at 1,800 and 1,000 MHz. From this figure, we observe: 1) with a fixed CPU frequency, the power consumption of each component does not vary much over time, 2) when scaling CPU frequency from 1,800 down to 1,000 MHz, CPU power decreases noticeably but less than proportionally while other components' power does not change. Since EP is computation intensive, scaling down processor frequency proportionally increases the execution time. As a result, when scaling down CPU frequency from 1,800 to 1,000 MHz, EP consumes more energy despite operating at a lower frequency because the increase in execution time offsets the benefits of reduced CPU power. In fact, scaling the CPU frequency to maximum benefits both energy and performance for EP.

Fig. 12 shows the profiles of FT. Unlike EP which consists mainly of a computation phase, FT presents alternating computation, memory, and communication phases. The impact of voltage and frequency scaling varies with these execution patterns. When scaling down processor frequency from 1,800 to 1,000 MHz, the CPU power drops about 40 Watts from 124 to 84 Watts during computation phases, and drops about 22 Watts from around 82 to 60 Watts during communication phases. Note that processor power consumption with a fixed frequency is larger during communications than during idle time (no user applications running), indicating there are some computations involved during communications. CPU frequency scaling also impacts the memory access pattern and memory power consumption for FT, which is not seen in EP; the memory power profile of FT fluctuates more at 1,000 than at 1,800 MHz. Meanwhile, scaling down processor frequency slightly increases the execution time of FT

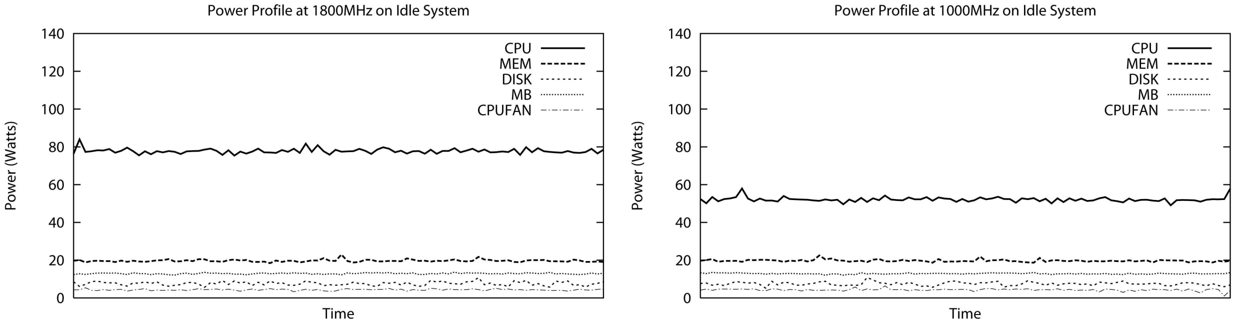


Fig. 10. Power profiles at 1,800 and 1,000 MHz when system is idle. When CPU frequency decreases from 1,800 to 1,000 MHz, CPU power consumption decreases from about 79 to about 53 Watts, and the power consumption of other components is unchanged.

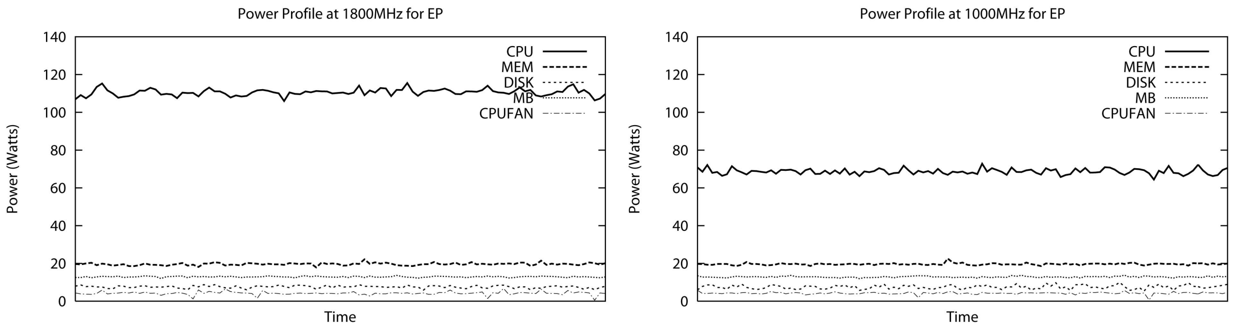


Fig. 11. Power profiles at 1,800 and 1,000 MHz when the system is executing EP. When CPU frequency decreases from 1,800 to 1,000 MHz, CPU power consumption decreases from about 110 Watts to about 69 Watts, and the power consumption of other components is unchanged.

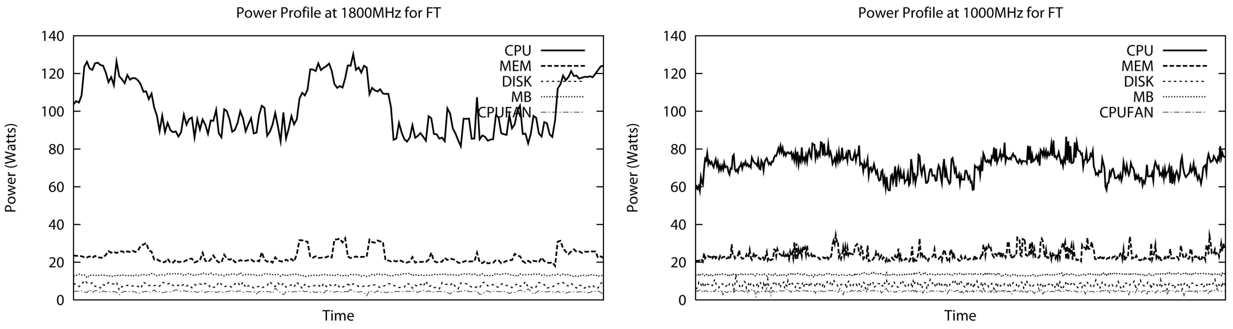


Fig. 12. Power profiles at 1,800 and 1,000 MHz when the system is executing FT. When CPU frequency decreases from 1,800 to 1,000 MHz, CPU power consumption decreases from about 110 to about 69 Watts, and power consumption of other components is unchanged.

because of a large portion of communication whose execution time nearly changes with frequency. Despite the overall increased execution time, scaling down CPU frequency conserves energy for FT.

The power profiles under various voltages and frequencies indicate that adapting the CPU frequency to meet the different computation needs during various execution phases for an application like FT would achieve the best combination of energy and performance. Specifically, if we scale up CPU frequency to its maximum during computation and scale it down during memory and communication, we can potentially save energy with less impact on execution time. Fig. 13 shows the resulting power profiling of an intelligent scheduling following this idea. In this scheduling, the CPU frequency is set to 1,800 MHz during computation and memory-intensive phases, and 1,200 MHz during communication phases. As we can see, the power consumption during communications with intelligent scheduling is about 30 watts less than that with fixed 1,800 MHz, and the execution times are similar. Overall, this scheduling achieves 12.1 percent energy savings with 1.2 percent performance impact. The same idea adopted in [17], [18] achieves significant energy savings with minimal performance impact.

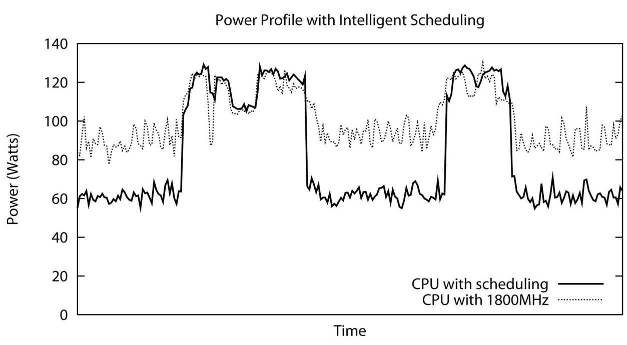


Fig. 13. Power profiles of FT when intelligent DVFS scheduling is employed. Compared to the profiles when CPU is fixed at 1,800 MHz, the power consumption during communication drops about 30 Watts with minimal increase in execution time.

6 Conclusion

We presented PowerPack for power-performance profiling and evaluation of distributed systems and applications at both component level and functional granularity. With the aid of PowerPack, we quantified the power-performance efficiency of applications on current and emerging distributed systems, analyzed the impacts of emergent technologies such as chip multiprocessing (CMP) and DVFS, and discussed the opportunities and approaches for improving power and energy efficiency of distributed systems and applications. We showed that PowerPack provides a systemwide view of power and energy consumption with high fidelity.

We used PowerPack to explore the systemwide power consumption for chip multiprocessors. Our results indicate that multicore systems contribute to higher energy efficiency for a given application, though the impact of multicores on power, energy, and efficiency are closely correlated to the application's workload characteristics. PowerPack was able to directly verify performance and energy efficiency for given applications depending on the applications' memory and communication patterns and their interaction with individual cores.

This work also experimentally showed the phase-based nature of power profiles and their correlation to application execution patterns. Such observations indicate that with intelligent DVFS scheduling, power savings can be achieved in many cases without impacting performance. Additionally, our quantitative presentations of power/energy profiles suggest a rather complex relationship between CPU power consumption and frequency. This observation raises questions on the energy models used in various existing DVFS studies on distributed systems, which either focus on processor dynamic power [9] without considering processor leakage power and power consumed by other system components, or simply assumes the CPU power consumption is proportional to product of frequency and voltage squared [23].

The methodology described in this work can be easily extended to other architectures and measurement equipment. For example, we can directly use the power sensors integrated in emergent computer systems by several manufacturers for more convenient power measurement. As ongoing work, we have adopted PowerPack to thermal profiling as well though we have not tested these extensions at scale due to limited availability of systems. Currently, we are also working with several companies to incorporate PowerPack techniques in their custom infrastructure deployments in large data centers built for computation.

Acknowledgments

This work was supported in part by the US Department of Energy under Grant No. DE-FG02-04ER25608 and the US National Science Foundation under Grant No. CCF-#0347683.

References

[1] T.U.E.P. Agency, http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency_study, 2009.

[2] T.V. Authority, http://www.tva.gov/environment/reports/ornl/index.htm, 2005.

[3] A.M. Bailey, "Accelerated Strategic Computing Initiative (ASCI): Driving the Need for the Terascale Simulation Facility (TSF)," *Proc. Energy Workshop and Exposition,* 2002.

[4] D. Bailey et al., "The NAS Parallel Benchmarks 2.0," Technical Report #NAS-95-020, NASA Ames Research Center, 1995.

[5] F. Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems," *Proc. Ninth ACM SIGOPS European Workshop,* 2000.

[6] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. 27th Int'l Symp. Computer Architecture,* 2000.

[7] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan, "Reducing Power with Performance Contraints for Parallel Sparse Applications," *Proc. First Workshop High-Performance, Power-Aware Computing,* 2005.

[8] J. Chen, M. Dubois, and P. Stenstrom, "SimWattch: Integrating Complete-System and User-Level Performance and Power Simulators," *IEEE Micro,* vol. 27, no. 4, pp. 34-48, July/Aug. 2007.

[9] S. Cho and R. Melhem, "Corollaries to Amdahl's Law for Energy," *Computer Architecture Letters,* vol. 7, no. 1, pp. 25-28. 2008.

[10] U. Congress, www.gpoaccess.gov/plaws/, 2006.

[11] N. Eisley, V. Soterious, and L.-S. Peh, "High-Level Power Analysis for Multi-Core Chips," *Proc. Ninth Int'l Conf. Compilers, Architecture and Synthesis for Embedded Systems (CASES),* 2006.

[12] X. Feng, R. Ge, and K.W. Cameron, "Power and Energy Profiling of Scientific Applications on Distributed Systems," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS),* 2005.

[13] J. Flinn and M. Satyanarayanan, "Powerscope: A Tool for Profiling the Energy Usage of Mobile Applications," *Proc. Second IEEE Workshop Mobile Computer Systems and Applications,* 1999.

[14] J. Flinn and M. Satyanarayanan, "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," *Proc. Second IEEE Workshop Mobile Computer Systems and Applications,* 1999.

[15] V.W. Freeh, D.K. Lowenthal, F. Pan, and N. Kappiah, "Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster," *Proc. 10th ACM Symp. Principles and Practice of Parallel Programming (PPoPP),* 2005.

[16] V.W. Freeh et al., "Exploring the Energy-Time Tradeoff in MPI Programs," *Proc. 19th IEEE/ACM Int'l Parallel and Distributed Processing Symp. (IPDPS),* 2005.

[17] R. Ge and K.W. Cameron, "Power-Aware Speedup," *Proc. IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS),* 2007.

[18] R. Ge, X. Feng, and K.W. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters," *Proc. ACM/IEEE Supercomputing (SC '05),* p. 34, 2005.

[19] R. Ge, X. Feng, W.-C. Feng, and K.W. Cameron, "CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters," *Proc. Int'l Conf. Parallel Processing (ICPP),* 2007.

[20] S. Gurumurthi et al., "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach," *Proc. Eighth Int'l Symp. High-Performance Computer Architecture (HPCA),* 2002.

[21] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," *Proc. 30th Ann. Int'l Symp. Computer Architecture,* 2003.

[22] C.-H. Hsu, "Compiler-Directed Dynamic Voltage and Frequency Scaling for CPU Power and Energy Reduction," PhD dissertation, 2003.

[23] C.-H. Hsu and W.-C. Feng, "A Power-Aware Run-Time System for High-Performance Computing," *Proc. ACM/IEEE Supercomputing (SC),* 2005.

[24] Standard Performance Evaluation Corporation, "The SPEC Benchmark Suite," http://www.spec.org, 2002.

[25] IBM PowerExecutive, http://www-03.ibm.com/systems/management/director/extensions/powerexec.html, 2007.

[26] IDC, Worldwide Server Power and Cooling Expense 2006-2010, 2006.

[27] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-36),* 2003.

[28] H. Meuer, E. Strohmaier, J. Dongarra, and H. Simon, The TOP500 Supercomputer Sites, http://www.top500.org., 2007.

[29] J. Janzen, "Calculating Memory System Power for DDR SDRAM," *Micro Designline,* vol. 10, no. 2, 2001.

[30] R. Joseph, D. Brooks, and M. Martonosi, "Live, Runtime Power Measurements as a Foundation for Evaluating Power/Performance Tradeoffs," *Proc. Workshop Complexity-Effective Design,* 2001.

[31] S. Kamil, J. Shalf, and E. Strohmaier, "Power Efficiency in High Performance Computing," *Proc. Fourth High-Performance, Power-Aware Computing Workshop,* 2008.

[32] LBNL, "Data Center Energy Benchmarking Case Study: Part 5—Case Studies on a Corporate Data Center,"Prepared by Rumsey Engineers for Lawrence Berkeley Nat'l Laboratory, Environmental Energy Technologies Division, p. 20, 2003.

[33] Mentor Graphics Corporation, 1999.

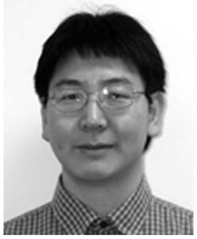
[34] Synopsys Corporation, *Powermill Data Sheet,* 1999.

[35] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," *Proc. 35th Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-35),* 2002.

[36] W. Ye et al., "The Design and Use of Simplepower: A Cycle-Accurate Energy Estimation Tool," *Proc. 37th Design Automation Conf.,* pp. 340-345, 2000.

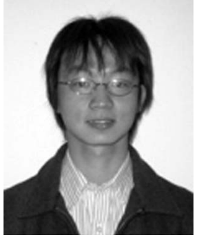
[37] J. Zedlewski et al., "Modeling Hard-Disk Power Consumption," *Proc. Second Conf. File and Storage Technologies,* 2003.

Rong Ge received the PhD degree in computer science from Virginia Tech. She is currently an assistant professor of computer science at Marquette University. Her research interests include performance modeling and analysis, parallel and distributed systems, energy-efficient computing, high-performance computing, and computational science. She is a member of the IEEE and the IEEE Computer Society, and a member of the ACM and Upsilon Pi Epsilon. More details about her research and background can be found at http://www.mscs.mu.edu/~rge.



Xizhou Feng received the PhD degree in computer science from the University of South Carolina. He is currently a research assistant professor of Computer Science at Marquette University and an adjunct faculty in the Network Dynamics and Simulation Science laboratory at Virginia Tech. His research interests include high-performance computing algorithms and systems, distributed systems and cyberinfrastructure, computational biology and bioinformatics, and modeling and simulation of complex systems. He is a member of the IEEE and the IEEE Computer Society, and a member of the ACM. More details about his research and background can be found at http://ndssl.vbi.vt.edu/people/fengx.php.

Shuaiwen Song received the BE degree in software engineering from Dalian University of Technology, China. He is currently working toward the PhD degree in the Department of Computer Science at the Scape Laboratory at Virginia Tech, where he is also a researcher. His research interests include parallel and distributed systems, multicore systems, power-aware and high-performance computing, and performance modeling and analysis. More details about his research and background can be found at http://www.s562673@cs.vt.edu.



Hung-Ching Chang received the BS degree in electronic engineering from Huafan University, Taiwan, in 2003. He has been working toward the PhD degree in computer science at Virginia Polytechnic Institute and State University since 2007. His major research interests include high-performance parallel computing, power-aware computing, and computer architecture. He is a student member of the IEEE.

Dong Li received the BS degree in electronics engineering from BeiHang University, China, in 2001, and the MS degree in computer science from the Royal Institute of Technology, Sweden, in 2006. He is currently working toward the PhD degree at the Virginia Polytechnic Institute and State University. His research interests include power-aware computing in the high-performance computing domain and performance prediction of parallel applications.

Kirk W. Cameron received the BS degree in math from UF in 1994 and the PhD degree in computer science from LSU in 2000. He is currently an associate professor of computer science at Virginia Polytechnic Institute and State University. He directs the SCAPE Laboratory at Virginia Tech, where he pioneered the area of high-performance, power-aware computing to improve the efficiency of high-end systems. He has received numerous awards and accolades for his research and publications including the National Science Foundation Career Award in 2004, the Department of Energy Career Award in 2004, the USC COE Young Investigator Research Award in 2005, the Best Paper Nominee SC06, the VT COE Fellow in 2007, the IBM Faculty Award in 2007, the Uptime Institute Fellow in 2008, and was invited to the 2008 National Academy of Engineering Symposium. He is on the editorial board and the editor for the *IEEE Computer* "Green IT" column. He is a member of the IEEE and the IEEE Computer Society.