PowerPack: Power and Energy Profiling and Analysis of Scientific Applications on Clusters

Rong GE, Xizhou FENG, Kirk W. CAMERON, Leon SONG,
Hungching CHANG, and Dong LI, *Member, IEEE*

***Abstract*--Energy efficiency in high performance applications and systems is critical. In the past few years, there is mounting evidence that energy use limits system scale and density and thus ultimately performance. Despite the impact of power and energy on high performance systems and applications, few studies provide insight to where and how power is consumed. In previous work, we proposed a framework called PowerPack that was the first tool to isolate power consumption of devices including disks, memory, NICs and processors in a high-performance cluster and correlate with application functions. In this work, we significantly extend PowerPack to support systems with multiple processors and multicore chips. We then provide in-depth analyses of high-performance applications on such clusters. These analyses include the impacts of chip multiprocessing on power profiling and energy efficiency, and its interaction with application executions. In addition to these analyses, we discuss the impact of power profiling of dynamic voltage and frequency scaling (DVFS) techniques to demonstrate conclusively how energy is saved while performance maintained.**

***Index Terms*--Distributed system, energy efficiency, power measurement, system tools**

I. INTRODUTION

POWER and energy are now critical constraints in high-performance system design. Advanced architectures such as Blue Gene/L use hundreds of thousands of low power processors to achieve top performance while working within power budgets of several megawatts. Nonetheless, systems and the buildings or centers that house them are consuming massive amounts of energy (typically 5-10 megawatts in total) to satisfy the needs of advanced scientific simulation. Laboratories such as ORNL, are commissioning substations from regional power companies to support their growing system needs [2]. This is both costly and time consuming. Operational costs for power and cooling will soon exceed acquisition costs [22]. The addition of power substations takes years of planning and coordination. Thus, efficient use of emerging high-performance systems resources has never been more important.

Manuscript received (Write the date on which you submitted your paper for review.) This work was supported in part by (sponsor acknowledgment goes here).

Rong Ge is with the Department of Computer Science and Application at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA (telephone: 540-230-7823, e-mail: ge05@vt.edu).

Xizhou Feng is with Virginia Bioinformatics Institute at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA (telephone: 540-231-4298, e-mail: fengx@vt.edu).

Kirk W. Cameron is with the Department of Computer Science and Application at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA (telephone: 540-231-4238, e-mail: cameron@cs.vt.edu).

Leon Song is with the Department of Computer Science and Application at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA.

Hungching Chang is with the Department of Computer Science and Application at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA (e-mail:hcchang@vt.edu).

Dong Li is with the Department of Computer Science and Application at Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA (e-mail: lid@vt.edu).

However, to improve energy efficiency, it is first necessary to measure and analyze the power and energy consumption of existing and emerging systems. Many factors impact power and performance. These factors include from system hardware component features and application execution patterns to parallel computing setups. As power and performance are tightly coupled and often conflicting, improving energy efficiency is complex. We need to measure so as to understand how each system component accounts toward the total system power consumption and how each typical parallel execution pattern affect the components' power. Only then, we are able to recognize the correlation between power and performance and identify opportunities and approaches for energy efficiency.

Despite the need for monitoring infrastructure, few have emerged to deliver fine grain information of system power use at scale correlated to application use and support emergent technologies such as CMP (chip multiprocessing) and DVFS (dynamic voltage and frequency scaling). The dearth of tools is primarily due to a lack of readily available, standardized hardware sensors in today's servers used as the basic building blocks of most high-performance systems.

Most of existing work focuses on power and energy profiling of single computing component on single node computer. Such studies uses simulation [6, 9, 25, 32-35], performance profile based estimation [23, 26], or direct measurement [5, 11, 23, 26]. The simulation work is primarily employed to model power consumptions of various components at micro architectural or lower level at design time [6, 30, 31, 34]. Performance-profile based estimation focuses on general purpose processor power consumption [23, 26]. Direct measurement is mainly used to validate the results obtained by the aforementioned two other approaches. While these works provide important information such as major power consuming performance events, they are limited to a single computing component without revealing information about other major components or the entire system. A number of efforts have been made to estimate energy consumption of threads or applications on a single node computer system by counting performance events [5] or system activity [11]. However, these works provide no information about energy use of individual components and thus give no insights for components' power managements.

Few work has studied the power consumption of distributed system, alas with coarse granularity at system or building level [21, 27, 28]. Some measure power by instrumentating power feeds to the infrastructure. For example, IBM PowerExecutive [21] uses this approach to monitor the power draw on selected IBM BladeCenter and System X servers; Lawrence Berkeley National Laboratory use power panels in power distribution units to monitor the power usage of a Cray XT4 system [27] and conduct landmark studies [28] for data centers. Other approaches, which is used on ASC Terascale facilities [3], estimate the system power consumption according to experience or rule-of-thumb at design and acquisition time. The power studies on large scale distributed systems have led to additional studies as the U.S. Department of Energy and U.S. Environmental Protection Agency have begun to pay more attention to power and energy consumption of servers [1]. However, the granularity of the obtained power information is not particularly useful for determining exactly where and how power is consumed by an application in a distributed system.

Nonetheless, as we have shown in earlier work, power and energy profiling can lead to improved energy efficiency techniques at the system and application level [10]. To address the need for fine-grain profiling of clusters, we created PowerPack [10], an infrastructure that combines hardware sensor devices (e.g. multimeters) with a software framework for correlation to system and application activity. PowerPack is the first attempt that systematically profiles power and energy of applications on distributed systems at fine granularity and including multiple components.

PowerPack was designed and implemented originally on an older Pentium III based system. Due to increased system complexity and sensitivity, in this work, we re-designed PowerPack to support emerging multi-processor, multi-core systems with advanced power management capabilities such as DVFS. Particularly, we 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 chip multiprocessing based distributed systems and applications.

2) Using our framework we provide power and energy profiling at a level of detail previously not possible and at a scale not found in the extant literature. Coupled with previous work, we compare and contrast the evolution of power budgets from single-core, single-processor, to multi-core, multi-processor 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 to the best of our knowledge.

The remaining part of this paper is organized as follows. Section 2 presents the design, implementation, and validation of the PowerPack framework. The detailed power and energy profiles of the NPB benchmarks on a multicore based cluster are provided in Section 3. The power-performance efficiency analysis of the NPB benchmarks with chip multiprocessing is discussed in Section 4. Section 5 extends PowerPack to power-performance evaluation of DVFS scheduling. Finally, Section 6 summarizes our findings and future work.

II. The PowerPack Framework

A. Overview

The PowerPack framework is composed of both hardware and software. The hardware includes commercial products such as digital multi-meters, as well as circuits that we made for power instrumentation. The software includes interfaces with various power meters and user APIs for power profiling and code synchronization. Together, these hardware and software enable two unique features: 1) fine-grain systematic power measurement; 2) automatic power profiling for applications and code segments.

Figure 1 shows a typical PowerPack deployment for power profiling on a multi-node Beowulf cluster. PowerPack simultaneously measures system power and multiple components' power. The system power is measured by power meters instrumented into AC power supplies. The component power measurement requires customized circuits. Specifically, we tap each individual DC power line (explained in detail later in this section) with a precision resistor and then use power meters to

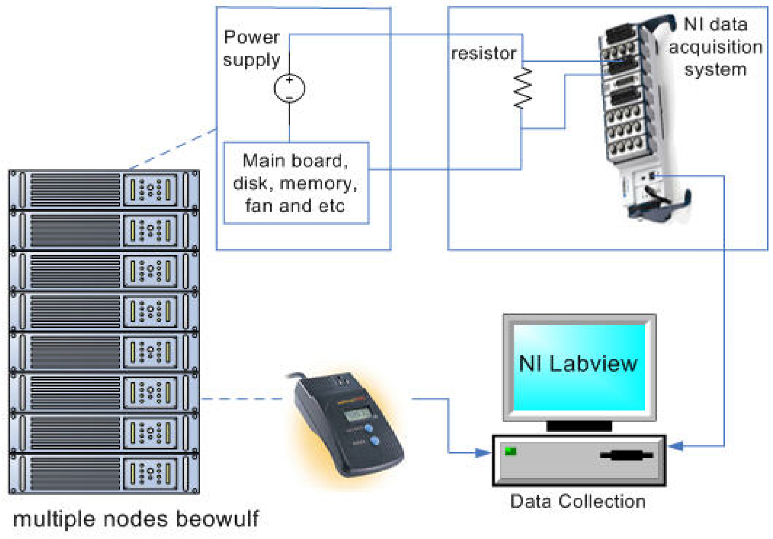


Fig. 1. An implementation of the PowerPack framework on a 9-node cluster with 18 AMD dual-core Opteron processors. AC Power is monitored at the node level using a Watt's Up Pro power meter and/or the ACPI standard interface. Within a node, DC power is monitored for each system component using the NI 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.

measure its current level. Given the resistance and line voltage, we are able to derive the component power. The current version of PowerPack uses various multi-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. Also, this redundant set of measurements allows us to verify the accuracy of each technique.

The software contained in PowerPack serves two purposes: on-line data recording and post mortem data analysis. The software for on-line data recording includes servers that poll meters and record data, and client API that triggers the server to record data. Such client-server structure enables us to synchronize the power profiling with code segments. The software for post mortem data analysis processes the data and reports power profiling of system and components. We show in this work PowerPack supports meters such as NI devices, Watt's Up Pro power meter, and Baytech, but PowerPack also support meters from Yokigawa and RadioShack [10]. Our contribution is a software framework that leverages any hardware power measurement devices to control and correlate the measured data to system software events and source code.

PowerPack currently directly measures one node at a time. To obtain the 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), 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 1.8GHz clock rate, six 1GB 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 [10]. 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 [24]. While our discussion is specific, our approach is portable to any commodity-based cluster with a standard power supply (e.g. ATX, BTX), and any number or types of processors, disks, memory, and NICs.

B. *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 on-board video card, keyboard, on-board network adapter, etc. Our measurement approach is as follows: if a component is powered through individual pins, we measure power consumption through every 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 micro benchmarks 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 +12VDC pins. Thus we can profile CPU power consumption by measuring all +12VDC pins directly.

Disk Power: The disk is connected to a peripheral power connection independently and Powered by one +12VDC pin and one +5VDC pin. By directly measuring both +12VDC and +5VDC 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 [10] to isolate memory power consumption.

Motherboard Power: NIC and other onboard components are powered through +3.3VDA pins. It is challenging to separate NIC power consumption from other onboard components directly. However, our measurements indicate the on-board 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 and we verified these findings by consulting the documentation of the NIC. Thus, based on our empirical measurements,

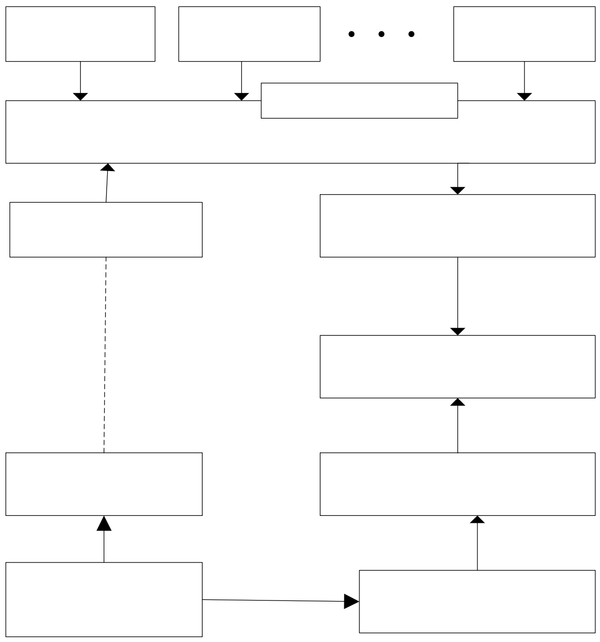


Fig. 2. Software architecture for automatic power and energy profiling. Our software automates the process of profiling power data and correlating the results to source code in a distributed system. Through modular design, the system software is portable to different systems, meter, etc.

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.

C. Automatic Power Profiling and Code Synchronization

Once the manual instrumentation setup is complete, the process of obtaining and controlling power profiling is completely automated by software. In fact, the PowerPack software suite includes all the micro benchmarks necessary to isolate power lines in the instrumented node. Additionally, all the experimental data gathered herein was 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 multimeter control process. The structure of the profiling software is shown in Figure 2.

The data collection computer runs LabVIEW that controls and reads power data from NI cDAQ9172 chassis via a USB interface. This data collection computer executes a meter control thread and a group of meter read threads where each meter read thread corresponds to one multimeter. The meter read threads collect readings from the multimeters and send them to the meter control thread. All meter control thread monitors messages from applications running on the cluster and modifies shared variables of the meter read threads according to messages received.

```
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
```

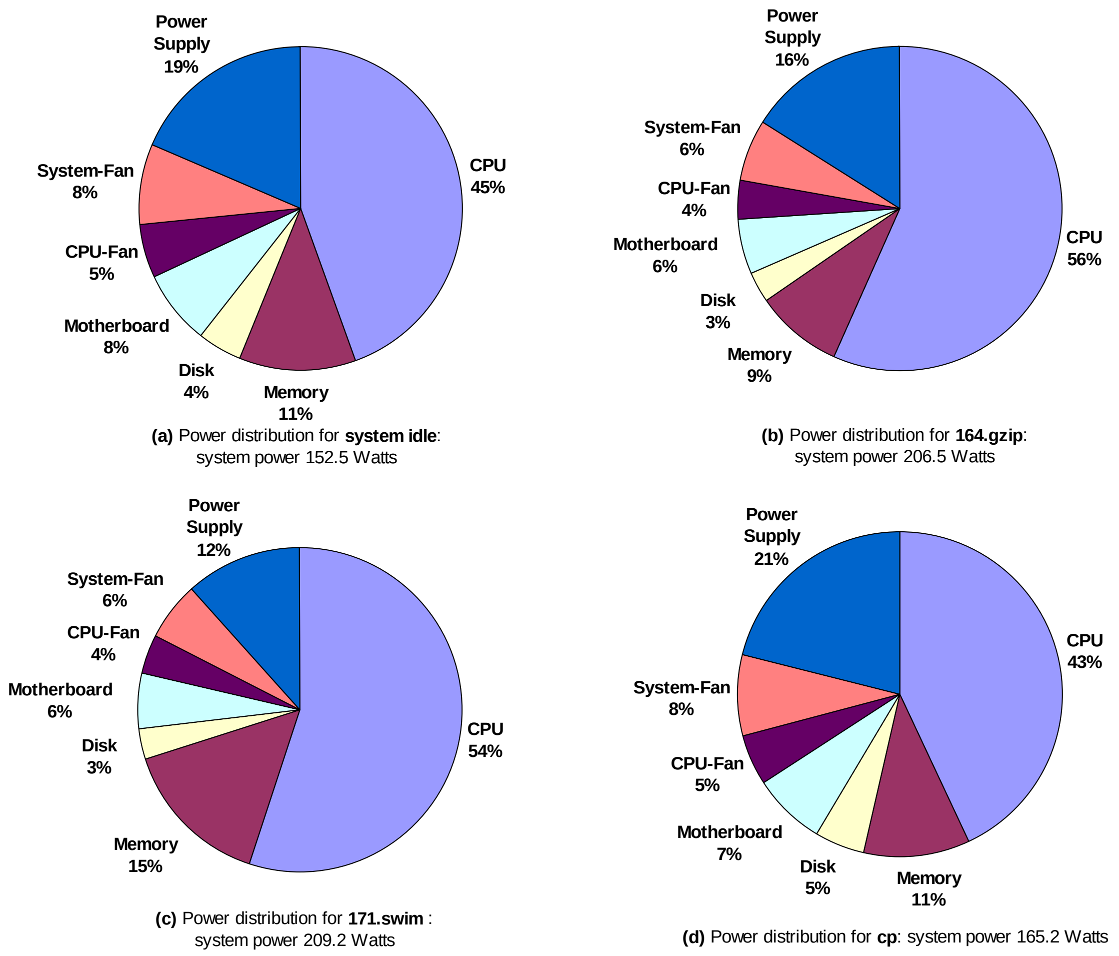
Fig. 3. The commonly used PoewrPack power meter profile API

To synchronize the live power profiling process with the running application, profiled applications trigger message operations through a set of 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 Figure 3, we list a commonly used subset of the power profile API in PowerPack.

III. Component and System Power Profiles

A. System-wide Power Distribution

We begin our analysis of application power profiles with system wide power distribution of sequential applications on a single compute node. Figure 4 shows the snapshots of power distribution of case 1: when no user application is running and case 2: when the system is running one of three applications from the SPEC CPU 2000 benchmark suite [20] (164.gzip and 171.swim) and the Linux standard file copy command (cp) programs. In case 2, when the system is running an application, the system is actually running four copies of the same program so that each of the four cores simultaneously executes one copy and the load is symmetric. We make the following observations from the figure:



(a) Power distribution for **system idle**: system power 152.5 Watts

(b) Power distribution for **164.gzip**: system power 206.5 Watts

(c) Power distribution for **171.swim** : system power 209.2 Watts

(d) Power distribution for **cp**: system power 165.2 Watts

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

1) Both total system power and individual component power vary with workload. Different workloads stress different components in a system. Component usage is reflected in the power profile.

2) The total system power under zero user workload (152.2 watts) is more than 72.9% 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% 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% 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.

B. Power Profiles of Distributed 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 5 kernels and 3 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. Since we have shown the impact of the power supply, fans, and motherboard on power profiles and assume they don't vary with time and applications, we will only report the power consumed by CPU, memory, and disk from this point forward.

1) Nodal Power Profile of the FT Benchmark

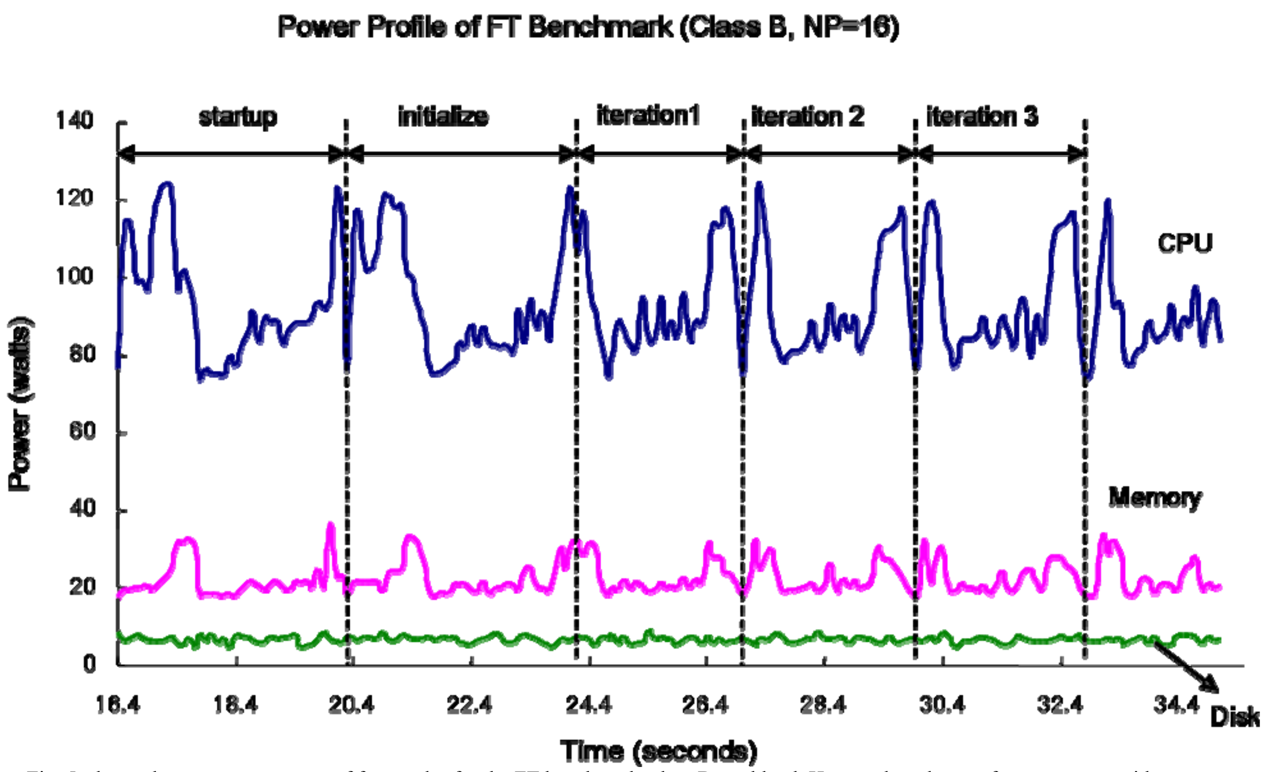


Fig. 5. shows the power use on one of four nodes for the FT benchmark, class B workload. Here each node runs four processes with one process per core.

In Figure 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 4 nodes. 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.

The power profiles are 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 apparent “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 related in that 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.

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. Figure 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

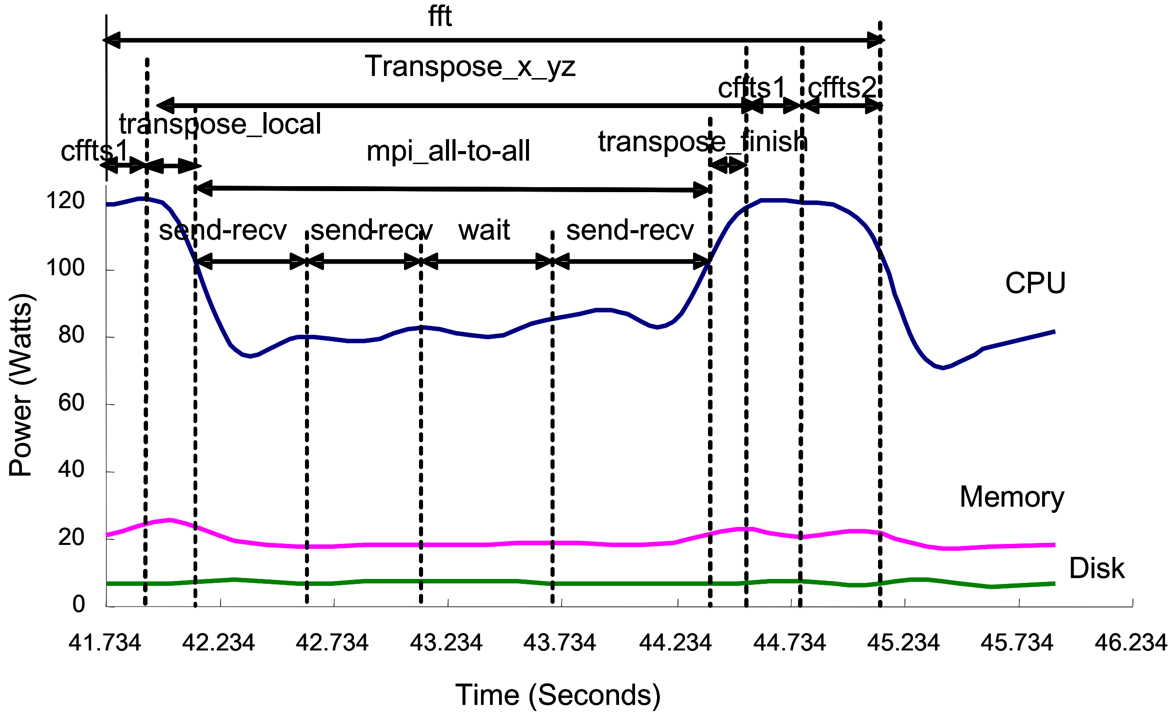


Fig. 6. Mapping between power profile and code segments for FT benchmark with Class B, 16 processes on 4 nodes. Using code analysis and code-power profile synchronization mechanisms provided in PowerPack, we can map the power phases to each individual function and perform detailed power-efficiency analysis on selected code segments. This is useful when exploring function-level power-performance optimization.

portions of code for application of power reduction techniques. This 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% 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.

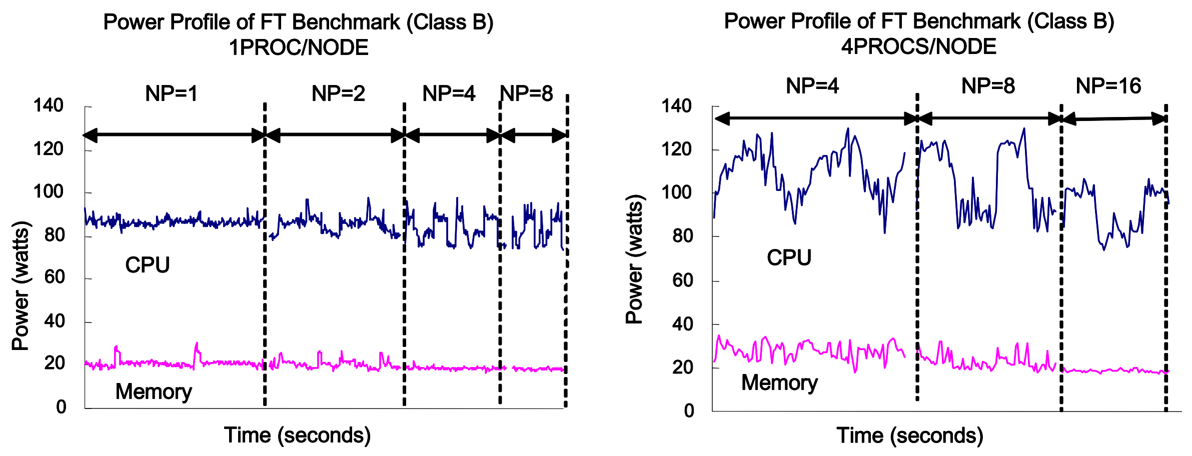
3) Power Profile Variation with Node and System Size

Using the node remapping technique described earlier, PowerPack can provide power profiles for all nodes running the parallel applications in the cluster. 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) and the change is reflected in the power profile.

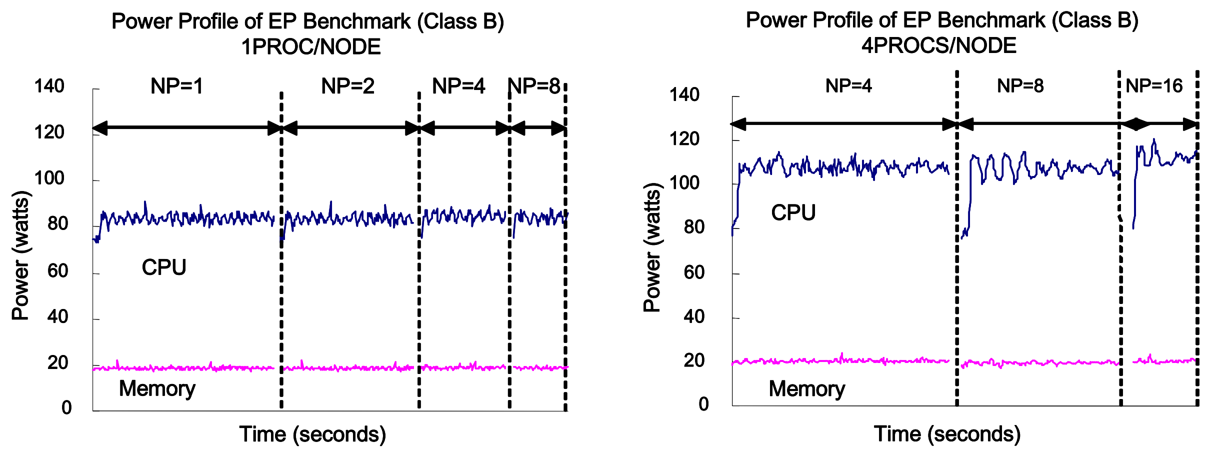
We have profiled the power consumption for all NPB benchmarks with different combinations of number of computing cores (up to 16), number of nodes, and problem sizes. In Figure 7, we provide an overview of the profile variations under different system setups for benchmarks FT, EP, and MG. Two parallel computing setups are studied for each benchmark. One setup is only one process is assigned to a node. Under this setup, only one out of the four cores on a node executes the application while the other three cores are idle. We denote this setup ***UP*** (for chip Uni-Processing). The other setup is four processes are assigned to a node. Under this setup, all the four cores on a node are involved in executing the application, and we denote this setup ***CMP*** (for Chip Multi-Processing). Figure 7 shows segments of synchronized power profiles under the 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.

Figure 8a shows the power profiles 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 decreases with the number of computing cores increase. Although the CPU and memory profiles under two different setups share the same trend as the number of computing cores increases, the absolute values under ***UP*** setup are smaller than those under ***CMP*** setup. For example, when the total number of processors is 4 (NP=4), the peak CPU and memory power under ***UP*** setup can be 23 watts and 12 watts smaller respectively. In the next section, we will discuss the power and energy efficiency of increasing the number of nodes.

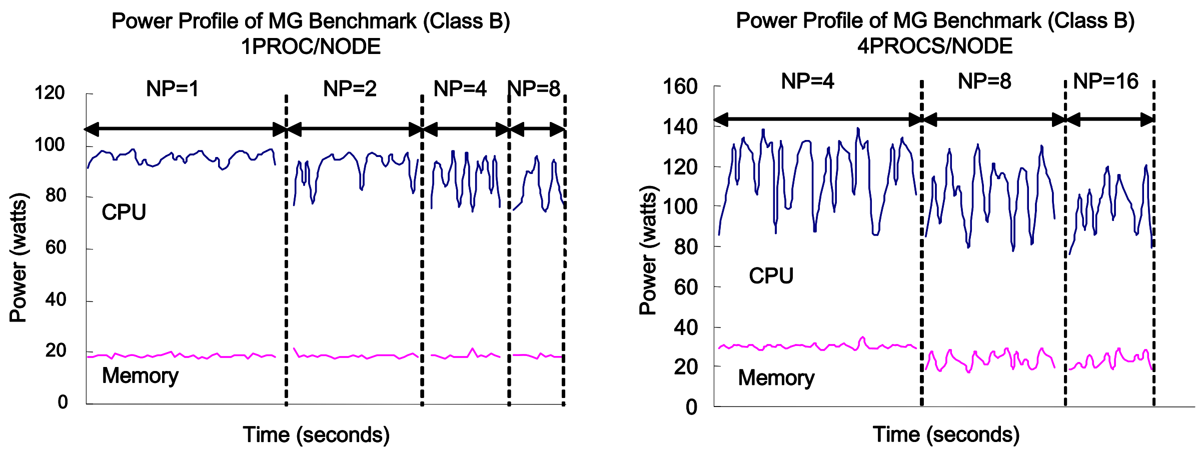
The embarrassingly parallel benchmark (EP) is essentially computation intensive and communication free. Hence, CPU intensity doesn't change as the number of computing cores increases. The fairly constant CPU power consumption in Figure 7b reflects this property under both setups. However, the absolute CPU power consumption under ***UP*** setup is less than that under ***CMP*** setup. For example, when the total number of computing cores is 4, the CPU power consumption under ***UP*** setup is 30 watts smaller. The interesting observation in this case is about the memory power. It doesn't noticeably change with the number of computing cores or with the setup. This is explained by the minimal memory access in EP.



(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

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.

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 4, the difference in CPU power for MG under 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 computation intensiveness of MG. Again, we will discuss efficiency metrics for MG in the next section.

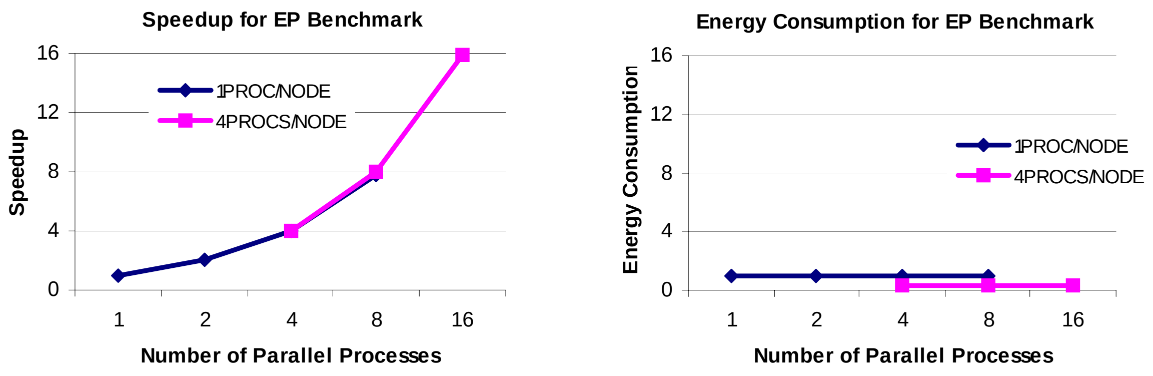
IV. Energy Efficiency of Parallel Applications

In this section, we apply PowerPack to analyze the energy efficiency of parallel applications. While 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 product of the average power ($\overline{P}$) and delay ($D = t_2 - t_1$):

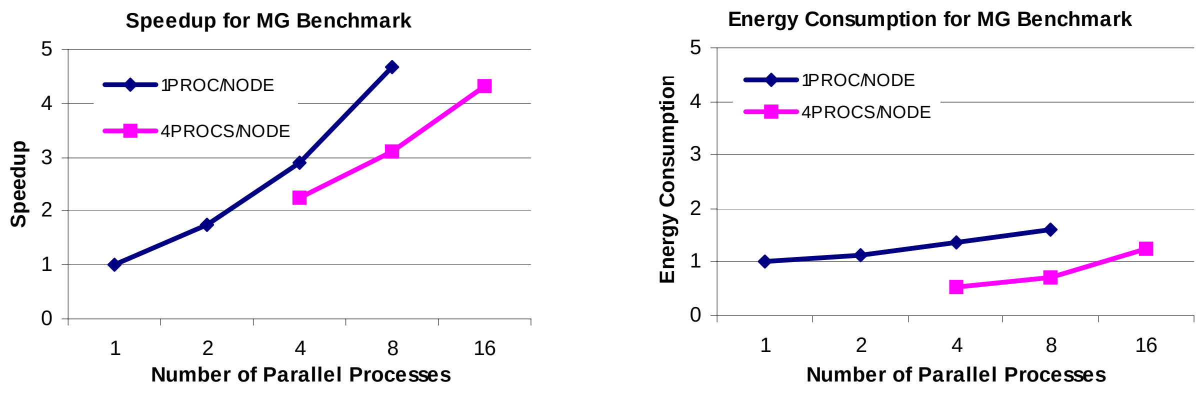
$$E = \int_{t_1}^{t_2} P(t)dt = \bar{P} \times D \tag{1}$$

A. Energy Scaling

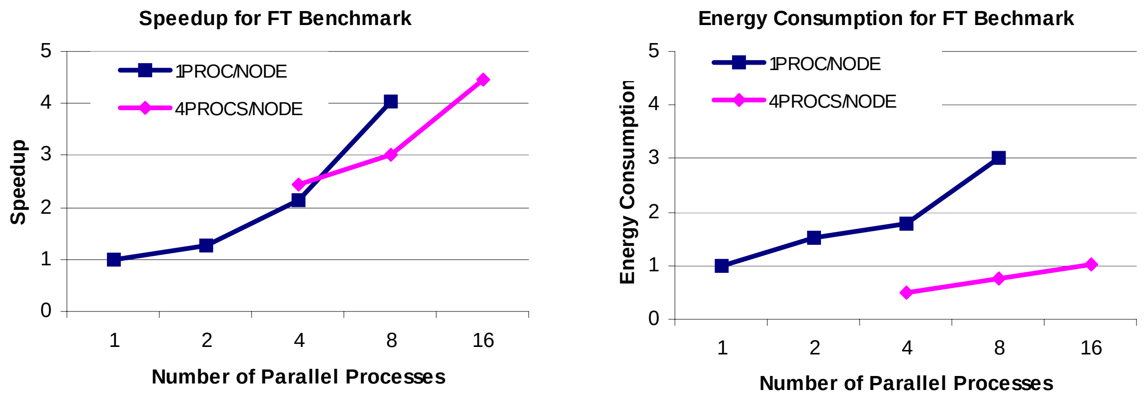
Equation (1) specifies the relation between power, delay (the final measure of performance) and energy. To reduce energy, we need to reduce the delay, the average power, or both. Under the context of parallel processing, by increasing the number of processors, we can speedup the application execution (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



(a)



(b)



(c)

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 1 process is running on one node. (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.

slowly or grow very quickly with the number of processors.

For distributed parallel applications, we can two metrics to compare the energy-performance behavior of different parallel applications such as those of the NPB benchmarks: a) the speedup (D_1/D_N) for performance scalability, where D_1 is the sequential execution time on 1 process, and D_N is the parallel execution time with N computing cores in parallel; and b) 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 subsection, we study the performance and energy scalability under two parallel computing setups: ***UP*** and ***CMP***, and compare the values between these two setups. Figure 8 shows the variations of speedup and energy with the number of total parallel computing cores for the two setups. We observe that energy consumptions using the ***CMP*** setup is always less than those 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 Figure 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 present this behavior. Such applications are computation intensive with minimal or overlapped communication; the execution time decreases proportionally with number of computing cores; and chip multiprocessing provides the best energy efficiency (about 70% 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 Figure 8b. Such applications are computation intensive, have a large memory footprint, and non-negligible 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 multi-core 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 Figure 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 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. For example, given an embarrassingly parallel application such as EP, total energy consumption remains constant within a parallel setup as we scale the number of cores to improve the performance.

B. 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 increase total energy consumption. Our measurements indicate there are tradeoffs between power, energy,

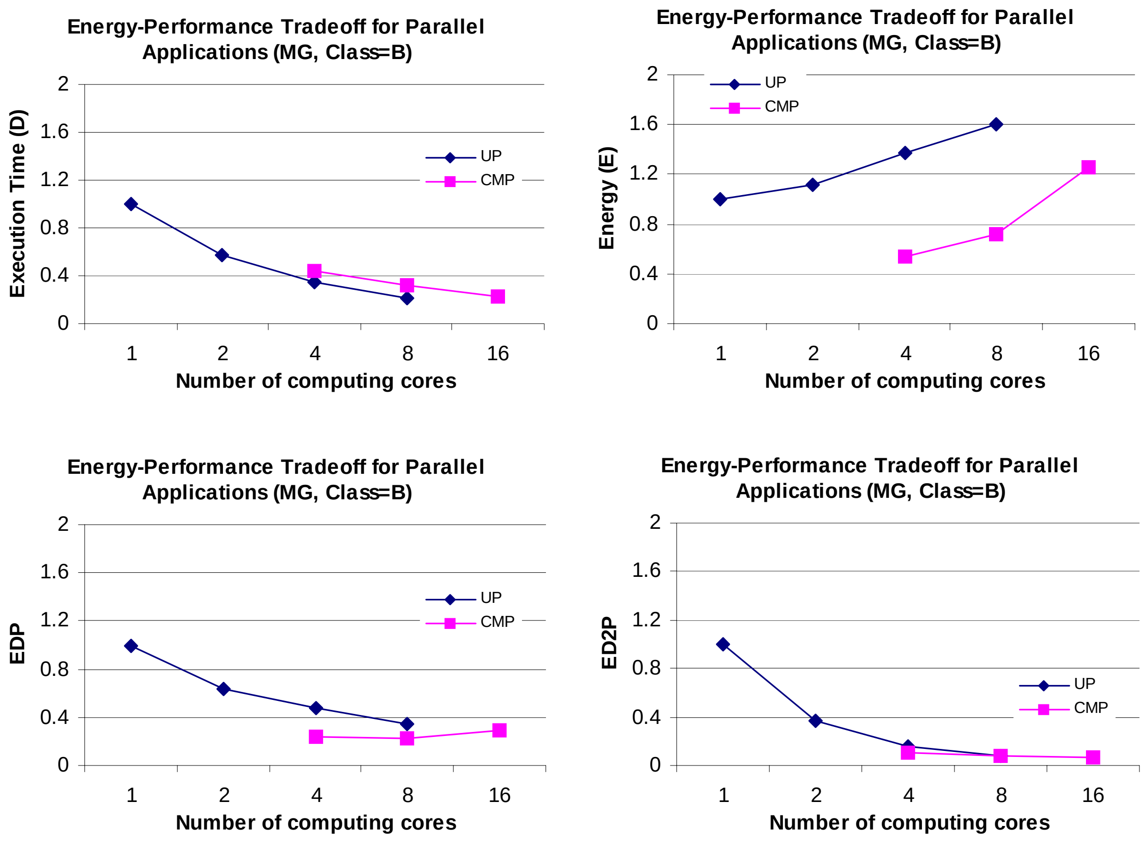


Fig. 9. Energy-performance tradeoff.

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 weight the effects of power and performance for a given application under different configurations. The smaller ED^α the configuration achieves for an application, the better efficiency this configuration is for this application.

We use NPB MG benchmark (class B) as an example to show the relationship between four metrics (normalized *E* and *D*, *EDP*, *ED2P*) and the number of cores. Figure 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 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 3.12X speedup and 0.72X energy cost. Using the ED2P metric suggests running with 16 cores on four nodes for 4.33X speedup and 1.26X energy cost. For accuracy, the average delay and energy consumption obtained from multiple runs are used in Figure 9.

V. 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 have the power modes scalable through a pair of frequency and voltage; this technique is referred as dynamic voltage and frequency scaling (DVFS). The power consumption on such processors significantly changes with frequency [18].

DVFS has been explored for power reduction and energy conservation in high-performance distributed systems [7, 12, 13, 16, 17, 19] by scaling down processor frequency during processor slackness. These works are based on the condition that the power consumed by CPU dominates system power and thus scaling down frequency can effectively reduce CPU and thus system power. Unlike the previous works, our intention in this section is to use PowerPack to reveal how DVFS affects CPU and other component as well as system power consumption, thus quantitatively explain the power-performance efficiency and energy conservation of applications under DVFS.

Table 1. Energy-performance profiles of NPB partial benchmarks. In each cell, the number on the top is the normalized delay and the number at the bottom is the normalized energy. The columns "XXX MHz" refer to the CPU speed at which we run the applications.

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 AMD Opteron processors on our experimental cluster originally have two available frequencies 1000MHz and 1800MHz. We hacked the Linux kernel and added three other frequencies 1200MHz, 1400MHz, and 1600MHz. Table 1 shows the power-performance efficiency of the NPB benchmarks in Class C with 16 processes on 4 nodes. Entry row in the table represents the delay and system energy consumption of a NPB benchmark under various CPU frequencies. Both delay and system energy are normalized to the corresponding values at the highest CPU speed (i.e. 1800MHZ). We observe that NPB applications show two categories of power-performance efficiency under DVFS.

1. Decreasing CPU frequency causes 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 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 applications belong to this category.

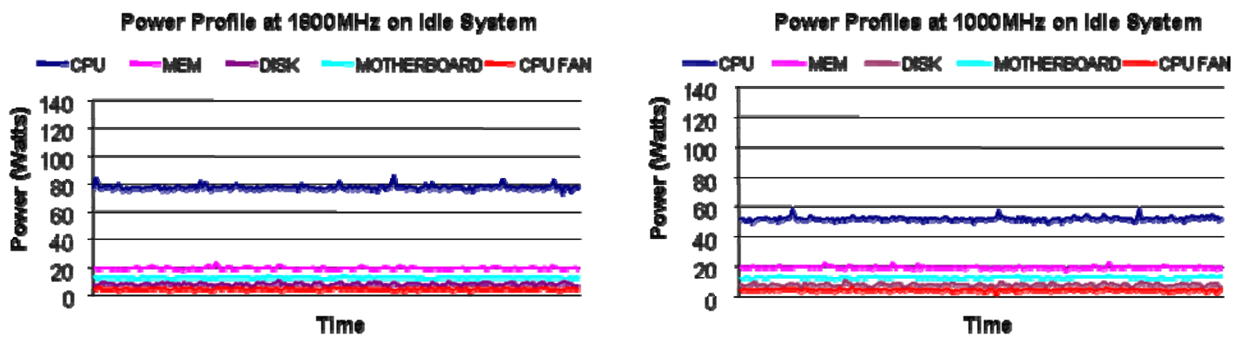


Fig. 10. Power profiles at 1800MHz and 1000MHz when system is idle. When CPU frequency increases from 1000MHz at 1800Hz, CPU power consumption increases from about 79 Watts to about 53 Watts, and power consumptions on other components are unchanged.

To quantitatively explain why DVFS has such different impacts on application power-performance efficiency, we use PowerPack to study how DVFS impacts power consumption of each component, especially CPU, during application execution.

First, we study the power profiles at various CPU frequencies when there is no user workload running on the system, as shown in Figure 10. When CPU frequency decreases from 1800MHz to 1000MHz, CPU power consumption decreases from about 79 Watts to about 53 Watts, and power consumptions on other components are unchanged. This observation concludes that scaling down CPU frequency can significantly and only reduce CPU power. CPU power [29] [14] consists of dynamic power, leakage power and short circuit power. Dynamic power is dominant and approximately proportional to the third power of frequency for DVFS capable processor and leakage power is also a function of frequency. Thus, the 26 Watts CPU power drop from 1800MHz

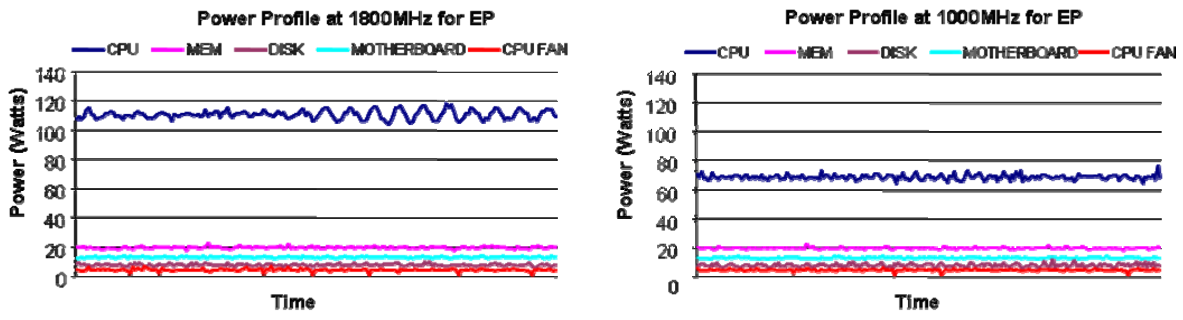


Fig. 11. Power profiles at 1800MHz and 1000MHz when system is executing EP. When CPU frequency decreases from 1800MHz to 1000Hz, CPU power consumption decreases from about 110 Watts to about 69 Watts, and power consumptions on other components are unchanged.

to 1000MHz comes from both dynamic power and leakage power.

Next, we study the power profiles of application EP which power-performance efficiency represents the first category in Table 1. Figure 11 shows the profiles of EP at 1800MHz and 1000MHz. We see from the figures 1) under a certain CPU frequency, the power consumption of each component does not vary much over time, 2) When scaling CPU frequency from 1800MHz down to 1000MHz, CPU power noticeably but less than proportionally decrease while other components' powers do not change. Meanwhile, As EP is computation intensive, scaling down processor frequency proportionally increases the execution time. As a result, scaling down CPU frequency from 1800MHz to 1000MHz for EP execution consumes more, instead of conserve, energy because the increased execution time consumes energy which offsets benefits of reduced CPU power. In fact, scaling the CPU frequency to maximum benefits both energy and performance.

Figure 12 shows the profiles of FT which represent the other category in Table 1. Unlike EP which presents only computation phase, FT presents alternating computation, memory, and communication phases. The impact of DVFS varies with these execution patterns. When scaling down processor frequency from 1800MHz to 1000MHz, the CPU power drops about 40 Watts from 124 Watts to 84 Watts during computation phases, and drops about 22 Watts from around 82

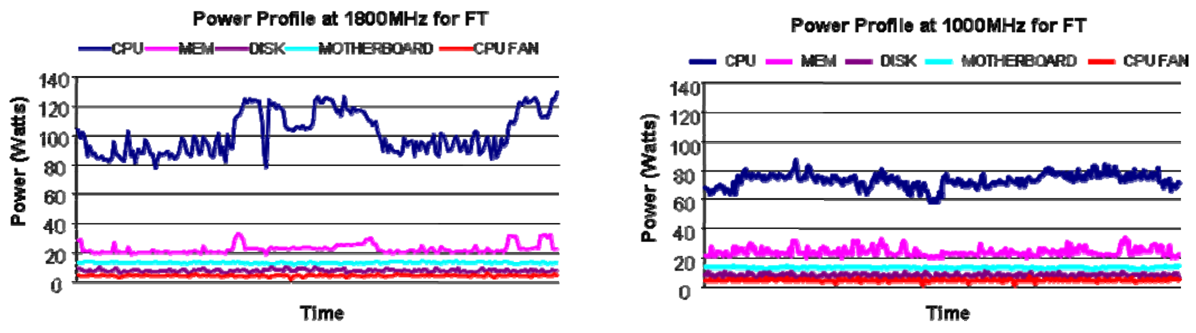


Fig. 12. Power profiles at 1800MHz and 1000MHz when system is executing FT. When CPU frequency decreases from 1800MHz to 1000Hz, CPU power consumption decreases from about 110 Watts to about 69 Watts, and power consumptions on other components are unchanged.

Watts to 60 Watts during communication phases. Note that processor power consumption under a certain frequency is larger during communications than during idle time (no user applications running). 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 1000MHz than at 1800MHz. Meanwhile, scaling down processor frequency slightly increases execution time of FT. The increased time is not big enough to offset the benefit of power reduction toward energy conservation. As a result, scaling down CPU frequency conserves energy for FT.

The power profiles with DVFS quantitatively conclude that adapting the CPU frequency to meet the different computation needs presented in various execution phases for an application such as FT would achieve the best combination of energy and performance. Specifically, we scale up CPU frequency to maximum during computation and scale down CPU frequency to proper values during memory and communication. Work [15, 16] adopts such idea and achieves significant energy savings with minimal performance impact.

VI. Conclusion

We presented PowerPack for power-performance profiling and evaluation of distributed systems and applications at component level and function 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 and DVFS, and discussed the opportunities and approaches for optimizations.

In this work, we found the impacts of chip multiprocessing on power, energy, and efficiency are closely correlated to application characteristics. Generally speaking, chip multiprocessing saves energy. However, whether it achieves best performance and efficiency for applications depends on the applications' required memory footprint as well as communication patterns. Additionally, the presented work shows conclusively that the phased-based nature of power profiles and their correlation to application execution indicates power savings can be achieved in many cases without reducing performance using DVFS. Last, when analytically evaluating the impacts of DVFS on energy consumption of applications, existing work either focuses on processor dynamic power [8] 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 square [19]. Nevertheless, our quantitative presentations suggest a rather more complex relationship between CPU power consumption with its frequency.

The methodology described in this work can be 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.

REFERENCES

1. The US Environmental Protection Agency. Available from: http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency_study.
2. Tennessee Valley Authority. 2005; Available from: http://www.tva.gov/environment/reports/ornl/index.htm.
3. Anna Maria Bailey. *Accelerated Strategic Computing Initiative (Asci): Driving the Need for the Terascale Simulation Facility (Tsf)*. in Proceedings of *Energy 2002 Workshop and Exposition*. 2002. Palm Springs, CA.
4. David Bailey, Tim Harris, William Saphir, Rob van der Wijngaart, et al., *The Nas Parallel Benchmarks 2.0*, 1995.Document
5. Frank Bellosa. *The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems*. in Proceedings of *Proceedings of 9th ACM SIGOPS European Workshop*. 2000. Kolding, Denmark.
6. David Brooks, Vivek Tiwari and Margaret Martonosi. *Wattch: A Framework for Architectural-Level Power Analysis and Optimizations*. in Proceedings of *27th International Symposium on Computer Architecture*. 2000. Vancouver, BC.
7. Guilin Chen, Konrad Malkowski, Mahmut Kandemir, and Padma Raghavan, *Reducing Power with Performance Contraints for Parallel Sparse Applications*, in *The First Workshop on High-Performance, Power-Aware Computing*. 2005: Denver, Colorado.
8. Sangyeun Cho and Rami Melhem, *Corollaries to Amdahl's Law for Energy.* Computer Architecture Letters, 2008. **7**(1): p. 25-28.
9. Noel Eisley, Vassos Soterious and Li-Shiuan Peh. *High-Level Power Analysis for Multi-Core Chips*. in Proceedings of *Proceedings of the 9th International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. 2006. Seoul, Korea.
10. Xizhou Feng, Rong Ge and Kirk W. Cameron. *Power and Energy Profiling of Scientific Applications on Distributed Systems (Ipdps 05)*. in Proceedings of *19th IEEE International Parallel and Distributed Processing Symposium*. 2005. Denver, CO.
11. Jason Flinn and M. Satyanarayanan, *Powerscope: A Tool for Profiling the Energy Usage of Mobile Applications*, in *the Second IEEE Workshop on Mobile Computer Systems and Applications*. 1999.
12. Vincent W. Freeh, David K. Lowenthal, Feng Pan, and Nandani Kappiah, *Using Multiple Energy Gears in Mpi Programs on a Power-Scalable Cluster*, in *10th Acm Symposium on Principles and Practice of Parallel Programming (Ppopp)*. 2005.

13. Vincent W. Freeh, David K. Lowenthal, Rob Springer, Feng Pan, et al., *Exploring the Energy-Time Tradeoff in Mpi Programs*, in *19th Ieee/Acm International Parallel and Distributed Processing Symposium (Ipdps)*. 2005: Denver, Colorado.
14. Rong Ge, *Theories and Techniques for Efficient High-End Computing*. 2007: PH.D. dissertation.
15. Rong Ge and Kirk W. Cameron, *Power-Aware Speedup*
Long Beach, Ca, in *Ieee International Parallel & Distributed Processing Symposium (Ipdps) 2007*. 2007.
16. Rong Ge, Xizhou Feng and Kirk W. Cameron, *Performance-Constrained Distributed Dvs Scheduling for Scientific Applications on Power-Aware Clusters*, in *Proceedings of the Acm/Ieee Supercomputing 2005 (Sc'05)*. 2005. p. 34.
17. Rong Ge, Xizhou Feng, Wu-Chun Feng, and Kirk W. Cameron, *Cpu Miser: A Performance-Directed, Run-Time System for Power-Aware Clusters*, in *International Conference in Parallel Processing (Icpp) 2007 (to Appear)*. 2007: Xian, China.
18. Chung-Hsing Hsu, *Compiler-Directed Dynamic Voltage and Frequency Scaling for Cpu Power and Energy Reduction, Ph.D. Dissertation*. 2003.
19. Chung-Hsing Hsu and Wu-chun Feng, *A Power-Aware Run-Time System for High-Performance Computing*, in *Proceedings of the Acm/Ieee Supercomputing 2005 (Sc'05)*. 2005.
20. http://www.spec.org, *The Spec Benchmark Suite*. 2002, Standard Performance Evaluation Corporation.
21. IBM. *Powerexecutive*. 2007; Available from: http://www-03.ibm.com/systems/management/director/extensions/powerexec.html.
22. IDC, *Worldwide Server Power and Cooling Expense 2006-2010*, 2006.Document #203598.
23. Canturk Isci and Margaret Martonosi. *Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data*. in Proceedings of *36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*. 2003. San Diego, California.
24. Hans Meuer Erich Strohmaier Jack Dongarra and Horst Simon, *The Top500 Supercomputer Sites: Http://Www.Top500.Org*. 2007.
25. Jeff Janzen, *Calculating Memory System Power for Ddr Sdram.* Micro Designline, 2001. **10**(2).
26. Russ Joseph, David Brooks and Margaret Martonosi, *Live, Runtime Power Measurements as a Foundation for Evaluating Power/Performance Tradeoffs. In Workshop on Complexity-Effective Design*, in *Workshop on Complexity-effective Design*. 2001: Goteborg, Sweden.
27. Shoaib Kamil, John Shalf and Erich Strohmaier. *Power Efficiency in High Performance Computing*. in Proceedings of *the fourth high-performance, power-aware computing workshop*. 2008. Miami, FL.
28. LBNL, *Data Center Energy Benchmarking Case Study: Part 5 - Case Studies on a Corporate Data Center*, 2003.Document
29. Jacob R. Lorch and Alan Jay Smith, *Software Strategies for Portable Computer Energy Management.* IEEE Personal Communications Magazine, 1998. **5**: p. 60-73.
30. Mentor Graphics Corporation, 1999.
31. Synopsys Corporation, *Powermill Data Sheet.* 1999.
32. Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik. *Orion: A Power-Performance Simulator for Interconnection Networks*. in Proceedings of *35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-35)*. 2002. Istanbul, Turkey.
33. Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik, *Orion: A Power-Performance Simulator for Interconnection Networks*, in *35th Annual Ieee/Acm International Symposium on Microarchitecture (Micro-35)*. 2002: Istanbul, Turkey.
34. W. Ye, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, et al., *The Design and Use of Simplepower: A Cycle-Accurate Energy Estimation Tool*, in *37th Design Automation Conference*. 2000. p. 340-345.
35. John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng, et al., *Modeling Hard-Disk Power Consumption*, in *Proc. Second Conference on File and Storage Technologies*. 2003.