

Exploring the Energy-Time Tradeoff in High-Performance Computing

Feng Pan Vincent W. Freeh Daniel M. Smith
Department of Computer Science, North Carolina

State University, Raleigh, NC 27695-7534

{fpan2, vwfreeh, dmsmith2}@unity.ncsu.edu

Abstract

High-performance computing is and has always been performance oriented. However, a consequence of the push towards maximum performance is increased energy consumption, especially at supercomputing centers. Moreover, as peak performance is rarely attained, some of this energy consumption results in little or no performance gain. In addition, large energy consumption costs the government a significant amount of money and wastes natural resources.

This paper investigates the tradeoff between energy and performance. Through the use of processors that support frequency and voltage scaling, we measured the performance and energy consumption of programs from three popular benchmark sets. We took multiple measurements for each program using different frequency and voltage settings. Results show that for many programs, a significant decrease in energy is possible with a small increase in time. We believe that this justifies further investigation into parallel HPC (*e.g.*, MPI) applications.

1 Introduction

High-performance computing (HPC) tends to push performance at all costs. Unfortunately, the “last drop” of performance tends to be the most expensive. For example, the last 10% increase in performance requires a disproportionately large amount of resources. The current policy at most of the nation’s supercomputing centers, which are dedicated to the execution of large-scale scientific applications, appears to be “performance at all costs.” Unfortunately, these centers consume a large amount of energy. This unchecked energy consumption costs the government a significant amount of money and wastes natural resources. Moreover, it is unlikely that supercomputing centers can continue limitless consumption of resources. In particular, energy consumption—and the resultant heat dissipation—is becoming an important limiting factor.

Large energy consumption at supercomputing centers might be acceptable if *all* energy were used profitably. However, architectural trends are such that achieving peak performance is becoming more difficult. While modern processors can issue multiple instructions in a single cycle, the average number issued is much lower than the maximum. For example, on a 3-issue Pentium 4 with Hyper-Threading

Technology, the average number of operations retired on the SPEC CPU2000 benchmarks is less than one [RF04, TT03]. In general, sustained performance is much less than peak performance. This is despite the large effort aimed at improving performance on high-performance architectures, including techniques such as instruction scheduling, memory access re-ordering, prefetching, and simultaneous multithreading.

This paper examines potential energy savings in HPC applications. We observe that if processor efficiency is poor, then processors, while running at peak speed and consuming maximum energy, must not be the bottleneck resource. For example, if the memory subsystem cannot supply data fast enough, the processor will be throttled by it. In such a case, a slower processor may achieve similar performance, which means that the extra energy consumed by a faster processor is wasted. Consequently, energy can be saved without a significant performance degradation.

Our eventual goal is to save energy in parallel HPC programs. As a first step, this paper determines the tradeoff between energy and time in sequential HPC programs. It also motivates the energy saving potential in parallel HPC applications.

1.1 Low-Energy Supercomputing

Energy-aware computing has flourished over the last decade in many areas, especially mobile devices. However, as explained above, the HPC community has vastly different goals. Because computational scientists are using HPC to gain maximum performance, many will likely resist any mechanism that decreases performance (*i.e.*, increases time to completion). However, we believe there are two primary reasons why HPC programmers will reduce performance to save energy. First, we believe that supercomputing centers are not immune to economic pressures. Therefore, the large and growing energy consumption of current clusters will eventually come to bear on its users. This may manifest itself in programmers utilizing lower-energy machines such as BlueGene/L [Adi02] or Green Destiny [WWF02] (which is a cluster of Transmeta processors). Such machines have better performance per unit energy than a conventional machine. However, they also offer significantly lower performance.

Second, computational scientists would be more likely to consider energy reduction if the effect on program performance were small. As peak performance on HPC machines is never achieved, and typically not closely achieved, we believe

that a slight performance degradation to save a good percentage of energy is possible—and may be acceptable to computational scientists. In particular, in some applications it is possible to save energy with virtually no performance loss (*e.g.*, *mcg* and *facerec* from the SPEC benchmark set).

1.2 Our Approach

This paper investigates the tradeoff between energy and performance (execution time). Processors are now available that support frequency and voltage scaling, providing multiple *operating points*. These operating points offer different levels of performance and energy consumption. Processor performance is roughly proportional to clock speed or frequency, *i.e.*, $performance \propto f$. On the other hand, power consumption is roughly proportional to frequency times voltage squared, *i.e.*, $power \propto fV^2$. Therefore, *energy efficiency*, (*i.e.*, instructions per joule) increases as frequency and voltage decrease. Using a processor with both frequency and voltage scaling, we evaluate the tradeoff between energy and time by executing a program at several different operating points, each with a different energy efficiency.

Towards this goal, we analyzed the NAS and SPEC suites to determine the relationship between voltage and frequency settings and execution time. Our results show that approximately 94% of our tests have what we call an *energy-time tradeoff*, meaning that a decrease in energy is possible but it comes at the cost of increased time. In the other programs, the highest operating point consumes the least energy and executes in the fastest time. Not every energy-time tradeoff is desirable, as some offer little energy savings and large time penalties. However, approximately half of these tests show a savings that is equal to or better than the penalty (*e.g.*, 10% less energy and 10% more time), and some are much better than that.

Because the case for power savings in desktop and cluster computers has not been made, scalable microprocessors are essentially available only in portable computers. Consequently, our test machines are laptop computers. However, this paper presents the first part of a case for power savings in high-performance computing centers. We expect that frequency and voltage scaling soon will become common in desktop machines.

The rest of this paper is organized as follows. Section 2 describes related work, and Section 3 provides details on our experimental methodology. Next, Section 4 gives results of performance measurements. Finally, Section 5 summarizes and describes future work.

2 Related Work

There has been a voluminous amount of research performed in the general area of energy management. In this section we describe some of the closely related research. We divide the related work into two categories: approaches for whole systems and for specific devices.

2.1 Whole System Approaches

Many have worked on saving energy in the entire system. This subsection details some of these projects.

Operating System Related ECOSystem [ZELV03] attempts to implement a power management system without the need to rewrite application software. The goal is to achieve a user specified battery lifetime. An energy accounting model is implemented, called the *currency* model. The currency model attempts to attribute energy usage to individual applications, as well as to specific components of the machine. Each application is allocated a certain amount of currency, which corresponds to the total energy it is allowed to consume in each epoch. The case for a closer relationship between the operating system and power management is further explored by Vahdat et al. [VLE00, Ell99]. This includes a case for treating energy as a first class resource in operating systems. Perhaps the best endorsement of operating system controlled power consumption comes from the ACPI (Advanced Configuration and Power Interface) standard [CCC⁺00]. It is an evolution of several existing methods including BIOS power management, the APM (Advanced Power Management) API, and a smart battery interface.

In general, the goal of the OS is to conserve energy for the entire set of processes. Our approach differs in that we are concerned with saving energy in a *single* parallel program.

Performance Counters There has also been significant effort put into software architectures that facilitate power management. The Observer architecture implements a monitoring and kernel instrumentation package as an extension to Linux [BBM98]. The main idea here is to design a system that collects information that is relevant to making power management decisions. Another project uses the hardware counters of the microprocessor to do energy accounting on individual processes [Bel00]. The Castle project uses hardware counters to estimate power consumption [JM01]. Finally, one can use program counter techniques to determine when to transition the disk to a lower energy level [GHL04].

Work on performance counter approaches are complementary to our approach. In particular, we may be able to use performance counters to determine where to transition to lower energy modes.

Application Related Flinn and Satyanarayanan [FS99b, FS99a] show that a collaborative relationship between the operating system and applications can yield significant power savings. They attempt to extend the lifetime of the battery to some user-specified goal. The basis for this work is a tool called PowerScope, which maps energy consumption to specific components. It analyzes specific processes and functions and uses statistical sampling to expose which components are consuming energy. This information is then used to direct program activity towards reaching an energy consumption goal by extending the Odyssey platform [NSN⁺97] to support energy adaptation.

	Compaq Presario 700	Compaq Presario 2100
Processor	Mobile AMD Athlon 4	Mobile AMD Athlon XP-M 2500+
Frequency Range (MHz)	300–1400	1130–1862
Voltage Range (V)	1.20–1.45	1.20–1.45
Voltage set points	6	6
Bus rate (MHz)	200	266
L1 Cache combined (KB)	128	128
L2 Cache (KB)	256	512
Memory (MB)	256	512

Table 1: Configuration of test machines.

Like performance counter approaches, application-related approaches are complementary to our approach. In the future, we will likely allow the application to explicitly transition between different energy modes.

Architecture Related There are a few high-performance computing clusters that are designed with energy in mind. One is BlueGene/L [Adi02], which uses a “system on a chip” to reduce energy. Another is Green Destiny [WWF02], which uses low-power Transmeta nodes.

This approach sacrifices performance in order to save energy by using less powerful machines. The approach we advocate is to start with powerful machines and find regions in programs to reduce energy.

Compiler Directed Hsu et al. [HK03] propose a compiler-directed algorithm to determine the appropriate operating points for memory-bound portions of the program. Like our approach, this uses physical power measurements are used instead of simulation.

2.2 Specific Device Approaches

Many have worked on saving energy in different devices. This subsection details some of these projects.

Processor Many modern processor architectures allow different frequency/voltage settings. This work developed into dynamic voltage scaling (DVS) [FRM01, Gru01, PBB98, PLS01, IKH01], which has come to mean the simultaneous changing of clock speed and voltage to reduce power consumption. DVS takes advantage of the fact that peak processing power is not always necessary to adequately service the average system load. Typically, DVS optimizes the energy \times delay product. This creates a system that more efficiently uses energy, but is still powerful and responsive.

In this paper, we investigate the relationship between frequency/voltage and execution time. DVS techniques are complementary, and in fact we will likely utilize them in our future work.

Disk Disks consume large amounts of energy on some architectures. Many have studied disk spindown to save energy

(*e.g.*, [HLS96, DKB95, Wil92, BAD⁺92, LKHA94]). In general, the idea is to determine when there is a large time period in which there are no disk requests and transition to a lower energy level. There has also been work in creating burstiness to save energy consumed by disks [PS03].

Memory and Network In some architectures, individual memory banks can be powered down [DSK⁺02, LFZE00]. The idea is to potentially place data intelligently in banks so that some banks will not be accessed. In some devices the network card has multiple energy states. One way to save energy is to use the energy-saving mechanisms defined by 802.11b [Com99]. One improvement to 802.11b is the Bounded Slowdown Protocol [KB02], which uses minimal energy given a desired maximum increase in round trip time. In addition, Yan et al. [YKW⁺04] leverage TCP to save energy in large file downloads. Finally, Kravets investigated power-aware mechanisms for end-to-end communication in wireless networks [KSC99].

Energy consumed by memory banks and network cards is important primarily in mobile devices. In desktop or cluster processors, memory energy consumption is relatively small. Hence, this work is orthogonal to ours.

3 Methodology

This section describes the experimental methodology. It first describes the details of the test machines, and then explains the techniques used to collect data. Tests were conducted on two Compaq Presario laptop computers equipped with AMD Mobile Athlon processors, which support frequency and voltage scaling. Table 1 displays some details of the test machines; we refer to the slower one as “old” and the faster one as “new.” There are many more frequency settings than voltage settings. Frequency scaling alone scales performance and energy equivalently. Consequently, there is an increase in energy efficiency (*e.g.*, instructions per joule) only when both frequency and voltage are scaled. Therefore, for the purposes of testing, the interesting operating points are those where both frequency and voltage change. When there are several frequencies possible for each voltage setting, tests use the highest frequency because it provides the greatest performance.

The primary difference between machines is that the new one has a processor that is 33% faster. The machines also dif-

fer in the memory subsystem. The old laptop bus operates at 200 MHz, whereas the new laptop operates at 266 MHz and uses double data rate (DDR) memory. The consequence of this is that the old laptop has a maximum memory bandwidth of approximately 430 MB/s, established by the *stream* benchmark [McC]. The new laptop achieves 40% greater throughput of 600 MB/s. We tested the memory bandwidth at each operating point and there is not a significant difference—less than 5% from highest to lowest.

The machines were configured with Linux 2.6. The advanced configuration and power interface (ACPI) was configured and used extensively. In particular, ACPI provides information about the battery state. Most useful for these experiments is information about the remaining battery capacity. Additionally, the *cpufreq* kernel module provides an interface to control the frequency-voltage setting. To set the operating point, one writes a string representation of the desired frequency in MHz to a file in the *sys* filesystem. For example, the following command sets the operating point to 1000 MHz:

```
echo "1000" >
/sys/devices/system/cpu/cpu0/cpufreq
```

Although there may be several frequencies for a particular voltage setting, there is only one voltage for any frequency.

All the tests were run using battery power in order to use the power consumption information provided through the ACPI interface (which is not available when operating on A/C power). All tests were begun with a fully-charged battery to eliminate any discrepancies that may be caused by a non-linear battery discharge rate. The ACPI implementation in the old laptop reports the battery capacity in milliwatt-hours. The new laptop reports two values, one in millivolts and the other in milliamp-hours; we converted these into energy. ACPI measurements are relatively coarse-grain. Tests lasting less than two minutes have highly varying results. Therefore, most tests run for at least 10 minutes and usually more.

Both laptops use lithium-ion (LiOn) batteries. Such batteries have extremely strict specific power delivery and charging characteristics, such that these phases must be closely monitored and tightly controlled [Pana]. For example, LiOn cells exhibit a flat voltage decay, followed by a steep drop at the end of the discharge cycle; cells allowed to “fall off” this drop can be damaged. Each multi-cell battery pack contains a controller IC, responsible for monitoring charge rates, voltage levels, and cell temperatures [Panb]. In an ACPI-compatible battery pack, this IC maintains a serial communication channel to the system board in order to transmit the aforementioned values.

Testing must be done while drawing power from the battery, and we wish to begin every test with a full battery. Therefore, we must recharge the battery after every test. In order to automate this testing, we built a *soft* power switch. The laptop power supply plug is inserted into the switch and the lead from the switch goes to the laptop. The switch is connected to the parallel or serial port of the laptop. A write to the port will turn the power on or off.

The backlight for the screen was turned off and there was minimal background processing during testing. Tests were

conducted according to the following script. First, the appropriate frequency and voltage setting is made. Second, the AC power is disconnected. Next several initial values are saved. In particular, wall clock time was obtained using *gettimeofday*, cycle count was obtained using the *rdtsc* instruction, and energy in the battery was obtained through ACPI. Additionally, hardware performance counters are set to measure memory accesses in order to compute memory bandwidth. Fourth, the program is executed. After the program completes, final values are collected and differences are calculated. Finally, the AC power is reconnected and the script waits until the battery is recharged before starting the next test.

We experimented with three different benchmark sets: NAS, SPEC integer, and SPEC floating point. The NAS suite is a popular high-performance computing benchmark, consisting of 8 scientific benchmarks including application areas such as sorting, spectral transforms, and fluid dynamics. In contrast, the 12 SPEC integer benchmarks are non-scientific applications that are CPU and/or memory intensive. The 14 SPEC floating part benchmarks are a mixture of both scientific and non-scientific programs. For example, *mesa* and *facerec* are non-scientific, graphics programs, whereas *swim* and *mgrid* are well-known scientific benchmarks.

4 Experimental Results

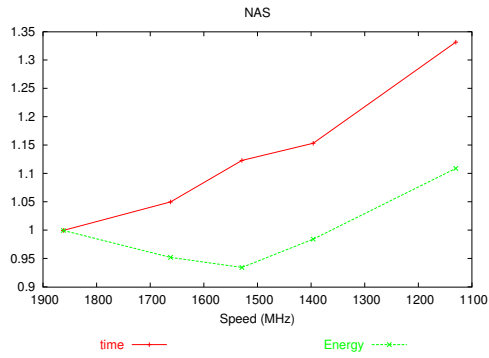
The primary goal of this paper is to examine the energy-time tradeoff for high-performance applications. Space constraints do not permit the presentation of all the results. For more information, including full results, please see our accompanying technical report [PF04]. Unless specifically noted, results are from the new laptop. While the energy-time tradeoff differs between machines, our results show that both laptops save a significant amount of energy on a substantial subset of the programs tested. We believe that this provides evidence, albeit not conclusive, that the results from our experiments will be typical among machines with multiple operating points.

As described in Section 3, we conducted tests over the three different sets of benchmark programs on each of the two laptops described above. Also, each program was run at every operating point. For each test, we measured the time and energy consumed. Additionally, using performance counters, we measured the number of cycles, micro-operations retired, and memory accesses.

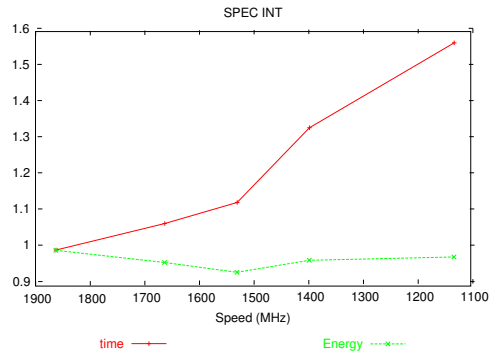
Below we discuss the overall results. Then, we look at a few representative applications in detail. Lastly, we show that these results we have observed on a single machine are likely to be seen in parallel configurations.

4.1 Overall Results

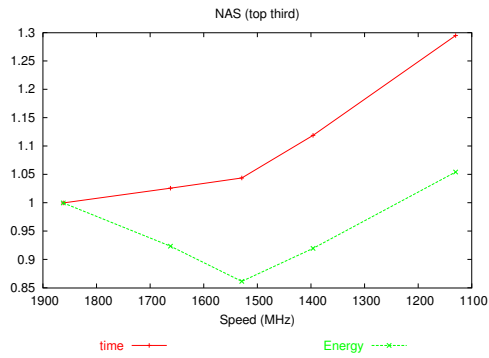
All of our tests show that for a given program, using the highest operating point takes the least time. On the other hand, the lower operating points use less power because the CPU—the dominant power consumer—uses less power. However, in terms of energy, the results vary. At a lower operating point a given program runs longer; if the decrease in power exceeds



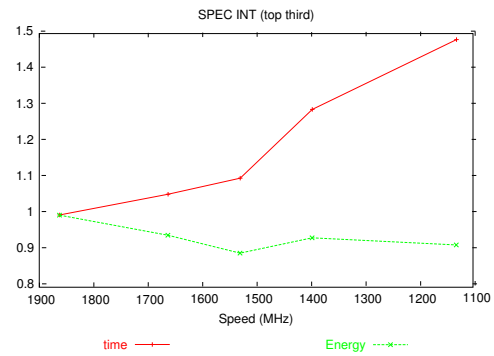
(a) All



(a) All



(b) Top third



(b) Top third

Figure 1: Normalized, aggregate plots of NAS set.

Figure 2: Normalized, aggregate plots of SPEC INT set.

the increase in time, a lower operating point uses less energy. This is the case in most of the programs we tested, where one of the lower operating points results in the least energy consumed.¹ However, for a handful of programs, the time increase exceeds the power decrease, so the highest operating point also consumes the least energy.

Figures 1(a), 2(a), and 3(a) plot the normalized aggregate results for each program set on the new laptop. The x-axis plots the operating point in terms of frequency from highest to lowest. There are two lines; the increasing line is elapsed time and the initially decreasing line is energy consumed. All values are normalized to those of the highest operating point; thus, all lines begin at 1 on the left-hand side. For the NAS programs, the time and energy diverge from 1 about equally. This means the energy savings is approximately equal to the time delay. For example, at the third operating point, the energy used and time taken are 91% and 112% of full, respectively. The SPEC sets also show an increase in time, but show little decrease in energy. This is because there is a high variance in the energy-time tradeoff among programs. The SPEC sets contain a few programs for which there is little energy savings

¹Which operating point resulted in the lowest energy varies between programs.

because the highest operating point uses the least energy or close to it. Overall, the aggregate plots suggest that one needs to be selective about which programs to try to save energy.

To investigate further, we plot for each set the programs that rank in the top 1/3 of energy-time tradeoff. Figures 1(b), 2(b), and 3(b) show the results. The NAS subset looks best, with 15% average energy savings for a time delay of less than 5% at the third operating point. The SPEC FP subset is similar, with 12% savings for 4% delay. Finally, the SPEC INT subset provides a 10% savings for a 10% delay, similar to the tradeoff provided by the full NAS set.

The slowest two operating points do not generally offer an energy savings. While the power is decreased, the time delay is so great that the energy savings is small if any. Only two of the 34 programs do not have an energy-time tradeoff: *crafty* from SPEC INT and *sixtrack* from SPEC FP. The complete results are shown in [PF04]. The next section evaluates individual programs in detail.

4.2 Detailed Results

In this section we analyze six programs in detail: the best and worst in terms of energy-time tradeoff from each program set. The energy of each point is plotted on the y-axis and the time

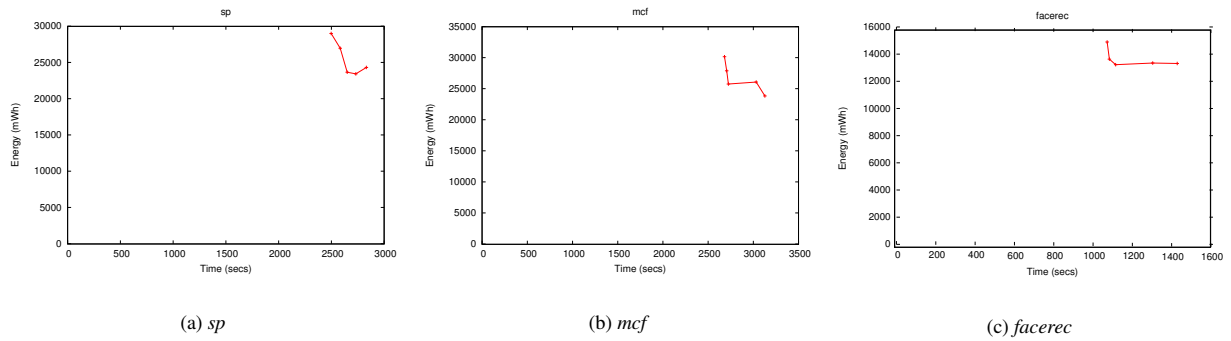


Figure 4: Best energy-time tradeoff in each set.

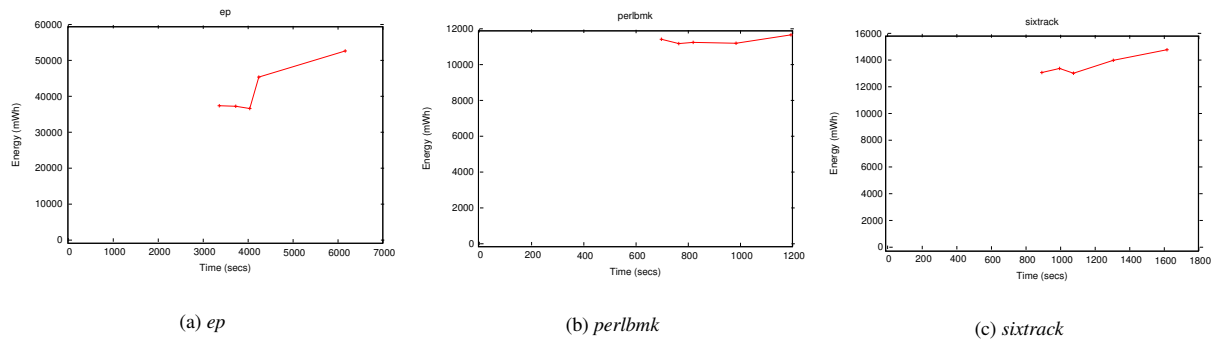


Figure 5: Worst energy-time tradeoff in each set.

is plotted on the x-axis. We find this plot presents the tradeoff most vividly. The higher of two points uses more energy and the rightmost takes more time. Therefore, a near vertical slope indicates an energy savings with little time delay, and a horizontal slope indicates a time penalty and no energy savings.

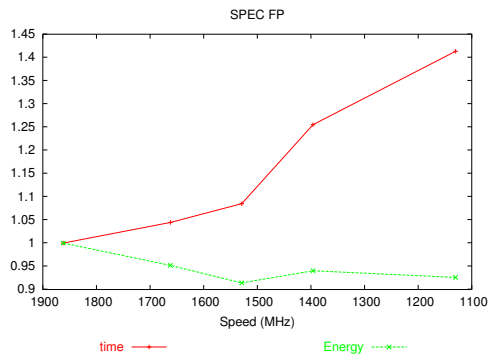
The programs shown in Figure 4 have the best energy-time tradeoff in each sets: NAS (*sp*), SPEC INT (*mcf*), and SPEC FP (*swim*). In these “vertical” applications, the execution time advantage of the highest operating point is small. However, the energy *penalty* for this ultimate performance is large. Consider for example the *sp* benchmark, in Figure 4(a). Using the third operating point (1529 MHz) yields about a 6% increase in execution time compared to the highest operating point, while the corresponding decrease in energy consumption is nearly 20%.

Next, we examine how a vertical energy-time shape occurs. Our results show that programs use the essentially same number (within 1%) of micro-operations regardless of the operating point. However, the number of cycles that an execution takes can change, especially in the vertical applications. For example, consider the *mcf* application at the two highest operating points (1862 and 1662 MHz), in which the performance gain is less than 1%. Using the lower operating point with a clock rate that is 89% of the highest, the execution has 90% as many cycles (approximately 5.0 to 4.5 trillion). Because the number of micro-operations does not change, the perfor-

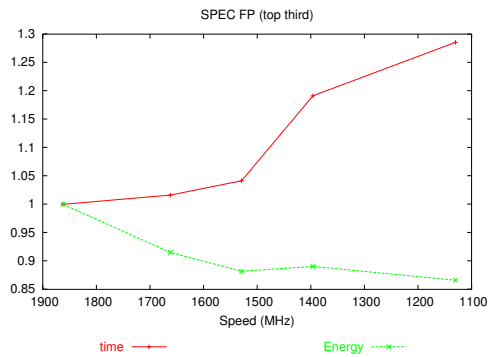
mance, in micro-operations per cycle (UPC), increases as the frequency decreases. The additional cycles in the higher operating point do not perform useful work. This indicates that the CPU is not the performance bottleneck. Below we examine this and, not surprisingly, determine that memory is the bottleneck.

On the other hand, Figure 5 shows the programs that do not exhibit an energy penalty for the ultimate performance. Instead, in these programs, the highest operating point results in nearly the lowest energy consumed. We call these “horizontal” programs.

Figure 6 shows the memory bandwidth achieved by these six programs. The figure shows the memory bandwidth normalized to the highest operating point. The x-axis shows the operating point of the processor. For reference a “straight line” is plotted that is proportional to frequency and normalized to one at the highest operating point. The normalized plot shows that the memory bandwidth scales with the operating point in the horizontal programs, *ep*, *perlbnk*, and *sixtrack*. This clearly shows that the processor is the bottleneck resource. On the other hand, the vertical programs, *sp*, *mcf*, and *facerec*, achieve at least 94% of their maximum memory bandwidth at the third highest operating point. This shows that the memory is limiting the performance. Following these plots from right to left, the memory bandwidth appears to plateau in the verti-



(a) All



(b) Top third

Figure 3: Normalized, aggregate plots of SPEC FP set.

cal applications before the highest operating point, which corresponds to the curve becoming vertical. The higher operating point does not result in higher performance—the energy-time tradeoff suggests using the lower point because the memory is the bottleneck.

In addition to memory bandwidth, we evaluated the processor efficiency in terms of micro-operations per cycle. The Athlon processors decode x86 instructions into one or more micro-operations that are executed by the RISC core of the processor. Therefore, UPC (micro-operations per cycle) is a better indicator of performance than IPC (instructions per cycle). Figure 7 shows UPC normalized to the highest operating point. The vertical applications have an increase in UPC at lower operating points, which is consistent with other observations above. The horizontal applications achieve the same processor efficiency for the three highest operating points. Thus, the overall performance is directly proportional to frequency in these applications.

Figure 8 plots the energy efficiency of our targeted programs in work per unit energy, *e.g.*, micro-operations per Joule. The plot is normalized to the efficiency of the highest operating point. The plot shows that for vertical programs, which have a good tradeoff, the energy efficiency increases as

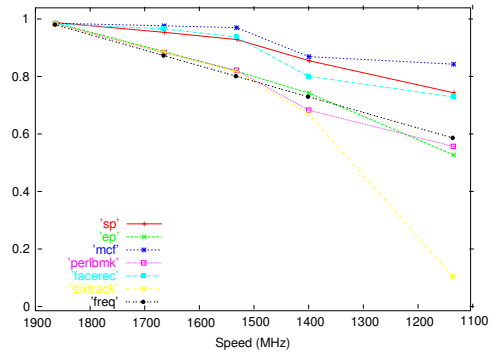


Figure 6: Memory bandwidth.

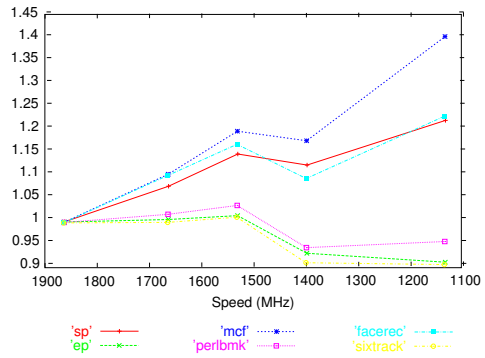


Figure 7: Micro-operations per cycle.

the performance decreases and vice versa for horizontal programs that have a poor energy tradeoff. The vertical programs are less efficient than their companion horizontal programs, *e.g.*, in the NAS set *sp* is less efficient than *ep*. This is not a surprise because a lack of an energy tradeoff means that the program is using its energy well. Again, these results illustrate how an energy tradeoff comes about.

We also conducted tests on the old laptop. The new machine is much faster and has more memory resources than the old machine. As stated above, the new laptop has a 33% faster bus, 100% faster memory, and twice the L2 cache and memory. Therefore, some programs run much better on the new machine than the old, which makes the results quite different. For example, the *ep* program from the NAS set overwhelms the memory subsystem on the old laptop. The *ep* program is a horizontal program on the new laptop. The program's memory bandwidth scale with processor speed indicating that the program is processor-bound. On the other hand, *ep* is a pure vertical program on the old laptop—the time and memory bandwidth are essentially the same for all operating points while energy decreases. Consequently, the behavior of these programs could hardly be more dissimilar. Therefore, given the resources of each laptop, the programs should not be considered the same program for the purposes of analysis.

Figure 9 shows the two laptops executing *sp* from the NAS set, but using different sizes. The *sp.B* data set is 4 times

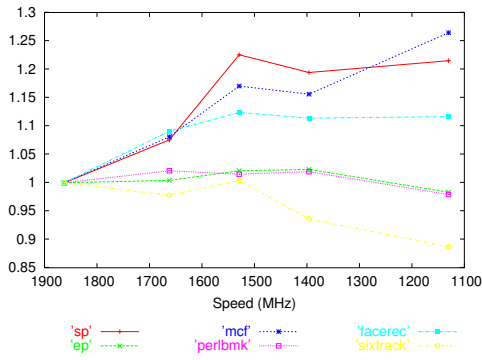


Figure 8: Micro-operations per Joule.

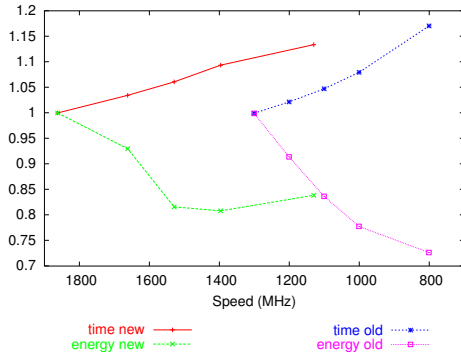


Figure 9: NAS *sp* on old and new laptop.

larger than *sp.A*. The curves are similar; this similarity occurs because the bottlenecks are comparable. In the case of *sp*, the memory bandwidth for the top three operating points is roughly the same on both laptops. In general, the old laptop tends to have more vertical programs because it has a slower bus and less memory. Even though the processor is also slower in the old laptop, the balance between the processor and memory is different.

4.3 Parallel Tests

While this paper focuses on sequential HPC applications, our eventual goal is to support parallel HPC applications, as this is where much of the total energy at supercomputer centers is consumed. In general, there should be *more* opportunity to save energy in parallel HPC applications. This is because not only is there the possibility that the CPU is not the bottleneck—which means that an individual processor can be scaled—but also that a given node is not the bottleneck—which means that it can be executed at a lower operating point without any performance penalty.

Figure 10 shows the energy-time tradeoff for two distributed 2-node MPI programs, *cg* and *ep*. Because a cluster of identical machines is not available to us, the two nodes are the new laptop and a desktop machine. These two machines have different CPU speeds, memory speeds, *etc.* However,

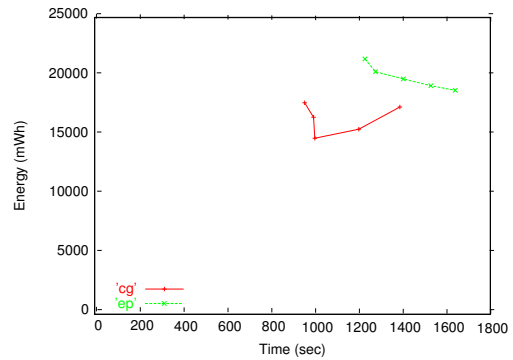


Figure 10: Scatter plot of energy-time for MPI versions of NAS programs *cg* and *ep*.

only the laptop has multiple operating points. It is important to note that the laptop is much slower, and so there is no energy savings possible due to another node being the bottleneck, because the laptop is in fact the bottleneck node. Keep in mind that if either node could be scaled, then the faster machine could be scaled with little or no performance penalty.

The figure shows that, as expected, there is an energy-time tradeoff in MPI programs. The energy-time tradeoff is in fact greater than that of their sequential counterparts. While we have only tested a handful of programs, and therefore is by no means a comprehensive study, the results convince us that a full-scale study of energy consumption in MPI applications is warranted. This study is currently underway.

5 Summary and Future Work

This paper has investigated the tradeoff between energy and performance. We analyzed the NAS and SPEC suites to determine the relationship between frequency and voltage settings and execution time. By executing each program at multiple operating points, we determined that most (94%) of the programs in these suites can consume less energy when they are run at a lower operating point. In other words, they have what we term an *energy-time tradeoff*. While computational scientists typically want high performance at all costs, we believe that economic pressures will eventually force users to consider energy as a limiting factor.

However, our work is only a first step—only a subset of the benchmarks consume significantly less energy with a small increase in time, which is the desired case—and more work is needed to selectively apply frequency and voltage scaling. Computational scientists will be much more willing to consider energy-saving techniques if they result in a relatively small increase in execution time. Indeed, we found that benchmarks such as *mcf* and *facerec* have, at some operating points, virtually no increase in time but a significant reduction in energy.

This paper determined the energy-time tradeoff in sequential HPC programs. Our eventual goal is to save energy in parallel HPC programs. While much future work is required,

we believe that in fact there is *more* opportunity to save energy in parallel programs. Our next step is to undertake an investigation of the energy-time tradeoff in parallel programs.

References

- [Adi02] N.D. Adiga et al. An overview of the BlueGene/L supercomputer. In *Supercomputing 2002*, November 2002.
- [BAD⁺92] M. Baker, S. Asami, E. Deprit, J. Ousterhout, and M. Seltzer. Non-volatile memory for fast, reliable file systems. In *Proceedings of the 5th ASPLOS*, 1992.
- [BBM98] L. Benini, A. Bogliolo, and G. De Micheli. Monitoring system activity of OS-directed dynamic power management. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '98*, 1998.
- [Bel00] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *Proceedings of the 9th ACM SIGOPS European Workshop*, September 2000.
- [CCC⁺00] Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation. Advanced configuration and power interface specification, revision 2.0. July 2000.
- [Com99] IEEE Computer Society LAN/MAN Standards Committee. IEEE Std 802.11: Wireless LAN medium access control and physical layer specification. Technical report, August 1999.
- [DKB95] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proc. 2nd USENIX Symp. on Mobile and Location-Independent Computing*, 1995.
- [DSK⁺02] V. Delaluz, A. Sivasubramanian, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Scheduler-based DRAM energy management. In *Proc. Design Automation Conf. (DAC '02)*, Jun 2002.
- [Ell99] C.S. Ellis. The case for higher-level power management. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, March 1999.
- [FRM01] K. Flautner, S. Reinhardt, and T. Mudge. Automatic performance-setting for dynamic voltage scaling. In *Proceedings of the 7th Conference on Mobile Computing and Networking MOBICOM '01*, July 2001.
- [FS99a] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles*, pages 48–63, 1999.
- [FS99b] J. Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.
- [GHL04] Chris Gniady, Y Charlie Hu, and Yung-Hsiang Lu. Program counter based techniques for dynamic power management. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*, February 2004.
- [Gru01] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '01*, August 2001.
- [HK03] C-H. Hsu and U. Kremer. The design, implementation, and evaluation of a compiler algorithm for cpu energy reduction. In *Proceedings of ACM SIGPLAN Conference on Programming Languages, Design, and Implementation*, June 2003.
- [HLS96] D. P. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Mobile Computing and Networking*, pages 130–142, 1996.
- [IKH01] C. Im, H. Kim, and S. Ha. Dynamic voltage scheduling technique for low-power multimedia applications using buffers. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '01*, August 2001.
- [JM01] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '01*, August 2001.
- [KB02] Ronny Krashinsky and Hari Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. In *Mobicom 2002*, Atlanta, GA, September 2002.
- [KSC99] R. Kravets, K. Schwan, and K. Calvert. Power-aware communication for mobile computers. In *Proc. 6th International Workshop on Mobile Multimedia Communications*, Nov 1999.
- [LFZE00] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In *Architectural Support for Programming Languages and Operating Systems*, pages 105–116, 2000.
- [LKHA94] K. Li, R. Kumpf, P. Horton, and T. E. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter*, pages 279–291, 1994.
- [McC] John D. McCalpin. Stream: Sustainable memory bandwidth in high performance computers. <http://www.cs.virginia.edu/stream/>.
- [NSN⁺97] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems and Principles*, pages 276–287, October 1997.
- [Pana] Panasonic. Lithium ion batteries: Individual data sheet.
- [Panb] Panasonic. Overcharge/overdischarge/overcurrent safety circuits.
- [PBB98] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '98*, pages 76–81, August 1998.
- [PF04] Feng Pan and Vincent W. Freeh. Energy-time tradeoff on laptop computers. Technical Report TBA, North Carolina State University, Department of Computer Science, February 2004.
- [PLS01] J. Pouwelse, K. Langendoen, and H. Sips. Energy priority scheduling for variable voltage processors. In *Proceedings of the International Symposium on Low-Power Electronics and Design ISPLED '01*, August 2001.
- [PS03] Athanasios E. Papatthasiou and Michael L. Scott. Energy efficiency through burstiness. In *WMCOSA*, October 2003.
- [RF04] Michael C. Rosier and Vincent W. Freeh. An evaluation of hyper-threading technology. Technical Report TBA, North Carolina State University, Department of Computer Science, February 2004.
- [TT03] Nathan Tuck and Dean M. Tullsen. Initial observations of the simultaneous multithreading Pentium 4 processor. In *Twelfth International Conference on Parallel Architectures and Compilation Techniques*, pages 26–35, 2003.
- [VLE00] A. Vahdat, A. Lebeck, and C. Ellis. Every joule is precious: The case for revisiting operating system design for energy efficiency. *SIGOPS European Workshop*, 2000.
- [Wil92] John Wilkes. Predictive power consumption. Technical Report HPL-CSP-92-5, Hewlett-Packard Labs, Feb 1992.
- [WWF02] M. Warren, E. Weigle, and W. Feng. High-density computing: A 240-node beowulf in one cubic meter. In *Supercomputing 2002*, November 2002.
- [YKW⁺04] Haijin Yan, Rupa Krishnan, Scott A. Watterson, David K. Lowenthal, and Kang Li. A theoretical and experimental study of energy-saving mechanisms for TCP downloads. Technical report, University of Georgia, February 2004.
- [ZELV03] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. Currentcy: Unifying policies for resource management. In *USENIX 2003 Annual Technical Conference*, June 2003.