

Towards Energy Efficient Scaling of Scientific Codes

Yang Ding, Konrad Malkowski, Padma Raghavan, Mahmut Kandemir
Department of Computer Science & Engineering
Pennsylvania State University, University Park, PA 16802, USA
{yding, malkowsk, raghavan, kandemir}@cse.psu.edu

Abstract

Energy consumption is becoming a crucial concern within the high performance computing community as computers expand to the peta-scale and beyond. Although the peak execution rates on tuned dense matrix operations in supercomputers have consistently increased to approach the peta-scale regime, the linear scaling of peak execution rates has been achieved at the expense of cubic growth in power with systems already appearing in the megawatt range. In this paper, we extend the ideas of algorithm scalability and performance iso-efficiency to characterize the system-wide energy consumption. The latter includes dynamic and leakage energy for CPUs, memories and network interconnects. We propose analytical models for evaluating energy scalability and energy efficiency. These models are important for understanding the power consumption trends of data intensive applications executing on a large number of processors. We apply the models to two scientific applications to explore opportunities when using voltage/frequency scaling for energy savings without degrading performance. Our results indicate that such models are critical for energy-aware high-performance computing in the tera- to peta-scale regime.

1 Introduction

The high performance computing community has traditionally focused on improving the number of instructions executed per second. As we approach peta-scale computing, it is becoming increasingly important for architecture designers to balance power requirements and computational performance potential. Therefore, the focus is shifting from the pursuit of highest possible performance in terms of floating-point operations (f-ops) per second, to a metric that captures power/energy efficiency of the systems such as f-ops per Watt or f-ops per Joule.

The importance of power consumption in the context of scientific computing is attributable to several factors. First, high power consumption can lead to high temperature, which in turn requires sophisticated and costly cooling techniques for large-scale parallel architectures. Second,

high temperatures threaten reliability of computing infrastructure and consequently integrity of data. Finally, peak power consumption imposes constraints on power supply design and increases overall system cost. All these factors motivate the work on power reduction in data intensive scientific applications.

Recent work in high performance computing [6, 18, 12, 27, 20, 32] has considered use of voltage scaling to save power without unduly affecting performance. However, many outstanding issues remain. One of the most important of these issues is the *energy scalability* of scientific applications. More specifically, it is very important to understand power-aware performance characteristics when very large numbers of CPUs are used.

In this paper we extend the ideas of performance scalability and iso-efficiency to characterize the system-wide energy consumption, which includes dynamic and leakage energy for processors, memories and network interconnects. We first propose *analytical models* for evaluating energy scalability and energy efficiency of scientific workloads. We then use these models to explore opportunities for energy savings using voltage/frequency scaling without degrading performance. Our major contributions can be summarized as follows:

- We introduce the concept of *energy scalability* in formal terms and study its behavior using two scientific applications. In addition, we also study the concept of *energy efficiency/iso-efficiency* and present experimental data, using base numbers collected from vendors.
- We extend our analytical models to show how voltage/frequency scaling can be used for studying the tradeoffs between performance and power consumption when using a large number of CPUs. In particular, we discuss how the performance gain obtained through parallelization can be translated to power savings.

To our knowledge, this is the first paper to discuss – in formal terms – energy scalability and energy efficiency of scientific applications executing on thousands of CPUs. We believe our results will help application designers and code implementers to select the right combination of power-saving techniques based on their performance constraints and power budgets.

Performance scalability has been extensively studied be-

fore. Grama et al [10] provide the formal definitions for several metrics such as speedup, efficiency and iso-efficiency. Our work extends these concepts into the energy domain. Ge and Cameron [9] present a power-aware speedup model for distributed systems by decomposing the workload with the degree of parallelism and on-chip/off-chip characteristics. While such a model can predict both the performance and energy consumption in certain cases, it does not investigate the general trend for increasing energy consumption as system scales up. Several other studies [11, 28] attempt enhanced speedup definitions to extend Amdahl’s Law to multicore chips, especially for heterogeneous single chip multi-processors. However, they only focus on performance. Jiang et al [17] propose a metric called energy-resource efficiency, which is built on top of the performance efficiency to guide performance-energy tradeoff. In comparison, we use a new set of metrics to address the energy scalability with more detailed power models. Many recent papers identified power as a performance-limiting factor. For example, Cameron et al [6] propose a framework called PowerPack for application-driven power measurement and management for scientific applications. Feng et al [8] use such a framework and study the power-performance efficiency of the NAS parallel benchmarks on a 32-node Beowulf cluster. Springer et al [32] focus on clusters with CPU power control modes and investigate the near-optimal scheduling to minimize the execution time under certain energy consumption limit, where processors can dynamically change frequency and voltage.

The rest of this paper is organized as follows. Section 2 introduces the methodology for modeling power-aware performance, proposes new energy-related metrics, and explains the baseline settings. Section 3 studies the energy scalability and energy efficiency with two scientific applications as examples. Section 4 considers dynamic voltage/frequency scaling (DVFS) and the tradeoff between DVFS and CPU shutdown. Finally, the concluding remarks are given in Section 5.

2 Methodology for Modeling Power-aware Performance

In this section, we describe methods used for exploring energy-efficiency and power saving opportunities for peta-scale systems. Our architecture abstraction assumes that each node has a CPU, a memory hierarchy (which can contain both hardware-managed conventional caches and software-managed memories), and a network controller. Such a system architecture is similar to many high performance computers such as Blue Gene/L [1].

2.1 Modeling Energy Scalability and Energy Efficiency

Let T_1 be the application execution time on a single CPU and T_p be the execution time on p CPUs. Several important

metrics used in the literature [10, 29] include *performance speedup* (PS) and *performance efficiency* (PE), which can be defined as follows:

$$PS = \frac{T_1}{T_p}, \tag{1}$$

$$PE = \frac{PS}{p} = \frac{T_1}{pT_p}. \tag{2}$$

Analogous to these definitions of performance metrics, we define new metrics to study energy scalability of applications running on large-scale parallel machines. Let E_1 be the energy consumption of the application when running on a single CPU, and let E_p be the average energy *per CPU* when running the same application on p CPUs. Thus, the total energy for the parallel system with p CPUs is pE_p . Based on these definitions of E_1 and E_p , we define *energy scaling* (ES) and *energy efficiency* (EE) as follows:

$$ES = \frac{E_1}{E_p}, \tag{3}$$

$$EE = \frac{ES}{p} = \frac{E_1}{pE_p}. \tag{4}$$

As we can see from the definitions, energy efficiency is actually the ratio between the total energy consumptions of a single CPU system and a parallel system with p CPUs. Also, energy scaling can potentially be incorporated into the study of thermal issues such as power density, though these issues are not in the scope of this paper.

Dynamic energy and *leakage energy* are the two major components of the total energy consumption [26, 19, 22]. Dynamic energy is dissipated due to circuit switching activities. Leakage energy on the other hand is consumed as long as circuit is on, due to the leakage current. In old technology generations, the focus has primarily been on dynamic energy and leakage energy could be neglected. However, leakage energy is expected to increase each generation and will become a major component of the total energy in the near future [4, 15]. Dynamic energy and leakage energy have different attributes; for example, leakage energy has two important factors, execution time and the number/size of resources used, whereas dynamic energy is more or less bounded by the instruction mix and data layout. Therefore, we study their contributions to the total energy consumption separately whenever possible and appropriate in our study. We use $E_{i,d}$ and $E_{i,l}$ to denote the average dynamic energy per CPU and average leakage energy per CPU, respectively, in a system with i CPUs. Note that both $E_{i,d}$ and $E_{i,l}$ also include other components such as memory and network interface associated with one CPU. Therefore, we can define energy scaling and energy efficiency for dynamic energy and leakage energy separately as follows:

Table 1. Basic performance and energy variables and their default values.

variable	description	default value
t_c	number of time units for a single basic computation	0.031 or 0.33
t_m	number of average time units per memory access	10
t_s	number of average time units for communication setup	182
t_w	number of average time units to transmit a word over network	34
$e_{c,d}$	dynamic energy units for computations per unit time	80 or 0.8
$e_{c,l}$	CPU leakage energy units per unit time	50 or 0.2
$e_{m,d}$	dynamic energy units for memory accesses per unit time	2.5
$e_{m,l}$	memory leakage energy units per unit time	0.9
e_l	link energy units per unit time	5

$$ES_d = \frac{E_{1,d}}{E_{p,d}}, \quad (5) \quad ES_l = \frac{E_{1,l}}{E_{p,l}}, \quad (6)$$

$$EE_d = \frac{E_{1,d}}{pE_{p,d}}, \quad (7) \quad EE_l = \frac{E_{1,l}}{pE_{p,l}}. \quad (8)$$

In this work, we consider the total energy consumption as the sum of the energy consumption incurred by CPU operations, memory accesses, and inter-processor communications. Note that, there are many components that may affect the system performance/power characteristics, we only consider these three dominating components in our study for simplicity. This formulation gives us the opportunity to study different optimizations and also allows us the freedom to disregard hardware details.

2.2 Power and Performance Baseline

To calculate various components of performance and energy, we define the basic performance and energy variables and obtain their default values from vendors as shown in Table 1.

- CPU – We consider two types of processors: a low-power core and a high-end workstation-like processor. We refer to the energy numbers of the PowerPC440 core used in the Blue Gene/L to represent the low-power space. The PowerPC440 core uses 130nm process technology and consumes approximately 1W of power on average. We estimate the dynamic power as 0.8W and leakage power as 0.2W. To represent the high-end processors, we consider the Intel Extreme Quad-Core Processor QX6700 processor under 65nm technology. The processor power is rated at 130W, and we estimate the leakage power to be 50W based on its power figure in its advanced HALT low power mode [14]. Note that the real numbers may be slightly different from our estimations. Nevertheless, the values used here reasonably represent the two different types of processors and their process technologies. We refer to these two types of processors as the *low-power processor* and *high-power processor* in the rest of the paper.

- Memory – We estimate the memory power consumption based on the power calculator datasheets from Micron Inc [25]. We simulate a 1GB DDR3 memory subsystem consisting of eight 1 Gigabit, 16 data pin modules. A 64-bit memory bus is assumed to be under 80% utilization, with

70% reads and 30% writes of these transactions. In addition, our calculation assumes even distribution of accesses across all memory modules. As a result, the average power of the memory subsystem is 3.4W and the leakage power of the memory subsystem is 0.9W. Note that caches are critical in the modern memory hierarchy. However, we do not include their power numbers into $e_{m,d}$ and $e_{m,l}$ because the CPU power numbers already consider on-chip caches.

- Network Interface – We refer to Mellanox’s datasheets [23, 24], which estimate the power consumption of their high-speed infiniband network solution to be 5W per link, and 10-12W per dual port adapter. We assume that the network port consumes energy whether data is transmitted or not. We further assume that the majority of the power consumed by the network adapter per port is spent on maintaining the link. Therefore, the energy consumption for the network is simply the product of its power number and the program execution time, i.e., we do not separate its dynamic energy and leakage energy.

We obtain the performance constants from published sources as well. We use t_c to represent time spent in one basic operation such as a floating-point addition, and t_m is the time to perform one memory access. Also, t_s is the startup time in cycles incurred for each network message, and t_w is the cost in cycles to transfer one double-precision floating-point number over the network.

Table 2. Performance reference numbers from Top500.

	PowerPC 440	Intel Core Duo
N	1769471	1769471
F(N)	3694×10^{15}	3694×10^{15}
Sustained F-Ops/Peak F-Ops	0.76	0.76
Peak F-Ops	2.8×10^9	21.28×10^9
Sustained F-Ops	2.14×10^9	16.27×10^9
Frequency (GHz)	0.7	2.66
Cycles/F-Op	0.33	0.16
Cycles/F-Op Normalized	0.33	0.031

We estimated the constants of t_c for high-power processor and low-power processor based on the published values in the Top500 project [34] for the Intel Core Duo processors and PowerPC440 processors when running the LINPACK benchmark. Table 2 shows the problem size as well as the calculations used, estimating t_c as the number of cy-

cles to perform one floating-point operation. N represents the problem size, and $F(N)$ is $\frac{2}{3} \times N^3 + N^2$. Peak F-Ops performance is calculated assuming one F-Ops per SIMD slot per processor cycle. The t_c values for the Intel cores are normalized to the cycle time of the PowerPC440 processor (Cycles/F-op Normalized in Table 2). Additionally, for the Intel multicore processor, we use the peak theoretical f-ops rate for the whole processor. The ratio of max f-ops to peak f-ops is approximately 0.76 for the PowerPC-based Blue Gene/L. We assume that the same ratio holds for the high-power processors, which is corroborated by performance ratio obtained by many computers on the Top500 list. We estimate network message startup time t_s and transfer time per double-precision floating-point number t_w based on one hop ping-pong latency for Blue Gene/L [2].

3 Studying Energy Scalability and Energy Efficiency

In this paper, we study both *strong scaling* (fixed problem size when increasing the number of processors) and *weak scaling* (increasing the problem size and the number of processors together) in the context of high-performance computers. The performance benefits that can be obtained via strong scaling are bounded by the program’s serial fraction following Amdahl’s Law. Also, the parallelization overheads such as data communications limit the performance benefits of strong scaling further. In this work, we study the energy benefits obtained from strong scaling and compare it with the corresponding performance benefits. For weak scaling, people have studied the problem of iso-efficiency, which is how to increase the problem size together with the number of processors in order to keep the performance efficiency constant. We discuss energy iso-efficiency and investigate its relationship with performance iso-efficiency in Section 4.

3.1 General Formulations

For a given application, assume Γ to be the number of basic computations, Φ to be the number of communications and Π to be the amount of data transmitted. We can calculate the execution time of this application on a single CPU and on a system with p CPUs as follows:

$$T_1 = t_c\Gamma + t_m\Gamma, \quad (9)$$

$$T_p = t_c\frac{\Gamma}{p} + t_m\left(\frac{\Gamma}{p} + \Pi\right) + t_s\Phi + t_w\Pi. \quad (10)$$

In the above equations, Γ is a function of the problem size, while Φ and Π are functions of both the number of processors and the problem size. Note that the number of memory accesses is calculated as the summation of the number of computations and the number of data elements transmitted. This is because, for each computation, some data is read or updated; also, the memory accesses are needed in

communication to save the data. Recall that we use the expression “memory access” to refer to an access to the memory hierarchy. Therefore, accesses to caches are also captured in this abstraction. We decide not to incorporate a separate cache module because it is difficult in general to predict cache behaviors at the algorithmic level. Using a single number for the memory access is a simplification. One can obtain such a number based on profiling and sampling the real execution [8, 12, 35].

Based on Equations 9 and 10, we can write the following expressions for performance speedup (PS) and performance efficiency (PE):

$$PS = \frac{T_1}{T_p} = \frac{t_c\Gamma + t_m\Gamma}{t_c\frac{\Gamma}{p} + t_m\left(\frac{\Gamma}{p} + \Pi\right) + t_s\Phi + t_w\Pi}, \quad (11)$$

$$\begin{aligned} PE &= \frac{T_1}{pT_p} = \frac{t_c\Gamma + t_m\Gamma}{t_c\Gamma + t_m(\Gamma + \Pi p) + t_s\Phi p + t_w\Pi p} \\ &= \frac{1}{1 + \frac{(t_s\Phi + (t_m + t_w)\Pi)p}{(t_c + t_m)\Gamma}}. \end{aligned} \quad (12)$$

Note that the parallelization overhead, which is important in studying the iso-efficiency, can be represented as follows:

$$T_0 = pT_p - T_1 = (t_s\Phi + (t_m + t_w)\Pi)p. \quad (13)$$

Similarly, the following equations represent the energy consumption E_1 and E_p as defined earlier.

$$E_1 = t_c\Gamma e_{c,d} + t_m\Gamma e_{m,d} + T_1(e_{c,l} + e_{m,l}), \quad (14)$$

$$E_p = t_c\frac{\Gamma}{p}e_{c,d} + t_m\left(\frac{\Gamma}{p} + \Pi\right)e_{m,d} + T_p(e_{c,l} + e_{m,l} + e_l). \quad (15)$$

Based on these two equations, we can derive the energy scaling (ES) and energy efficiency (EE) as follows:

$$ES = \frac{E_1}{E_p} = \frac{t_c\Gamma e_{c,d} + t_m\Gamma e_{m,d} + T_1(e_{c,l} + e_{m,l})}{t_c\frac{\Gamma}{p}e_{c,d} + t_m\left(\frac{\Gamma}{p} + \Pi\right)e_{m,d} + T_p(e_{c,l} + e_{m,l} + e_l)}. \quad (16)$$

$$\begin{aligned} EE &= \frac{E_1}{pE_p} \\ &= \frac{t_c\Gamma e_{c,d} + t_m\Gamma e_{m,d} + T_1(e_{c,l} + e_{m,l})}{t_c\Gamma e_{c,d} + t_m(\Gamma + \Pi p)e_{m,d} + T_p p(e_{c,l} + e_{m,l} + e_l)} \\ &= \frac{1}{1 + \frac{t_m\Pi p e_{m,d} + (pT_p - T_1)(e_{c,l} + e_{m,l}) + T_p p e_l}{t_c\Gamma e_{c,d} + t_m\Gamma e_{m,d} + T_1(e_{c,l} + e_{m,l})}}. \end{aligned} \quad (17)$$

As a result, the energy overhead due to parallelization can be defined as follows:

$$\begin{aligned} E_0 &= pE_p - E_1 \\ &= t_m\Pi p e_{m,d} + (pT_p - T_1)(e_{c,l} + e_{m,l}) + pT_p e_l. \end{aligned} \quad (18)$$

With such formulations, we can also investigate the iso-efficiency for energy consumption, i.e., the workload increase needed when increasing the number of CPUs in order to keep the energy efficiency constant. Such functions can be obtained similarly from Equation 18.

3.2 Case Studies

In this section, we describe the scientific codes used in this study, and present the trends of energy metrics with these applications. The performance scalability of these applications was discussed in detail in several books [10, 29]. However, the cost of memory accesses was ignored in the previous studies, which may hurt the accuracy of the estimation. We extend their formulations and propose new models for energy scalability and energy efficiency. The interconnection architecture is assumed to be a hypercube or mesh with cut-through routing in estimating the communication costs, as in the study by Kumar et al [10]. Due to space limit, we only illustrate fast fourier transform and dense matrix-vector multiplication here. Discussion on other kernels such as sparse matrix-vector multiplication can be found in our technical report [7].

- **Fast Fourier Transform (FFT)**

A fast Fourier Transform (FFT) is a divide-and-conquer algorithm to compute the Fourier Transforms. A variety of methods have been developed for FFT. For the purpose of our discussion, we consider the simplest one-dimensional, unordered and radix-2 binary exchange algorithm [10]. Binary-exchange algorithm for an n -point FFT has $\log n$ iterations. All the data points are updated in each iteration. Thus, the total number of computations is $n \log n$. In a parallel system with p CPUs, each CPU updates n/p data points on each iteration. The computations in the first $\log p$ iterations need data communication and the rest are local. In each communication step, n/p data points are transmitted. The number of communications and the number of data elements transmitted are summarized in Table 3.

Table 3. Different cost components for different applications.

Application	Γ	Φ	Π
FFT	$n \log n$	$\log p$	$\frac{n}{p} \log n$
DMVM	n^2	$\log p$	$\frac{n}{\sqrt{p}} \log p$

We apply strong scaling to the FFT application with the problem size fixed at $256M$ -point FFT and varying the number of CPUs as powers of 2 from 2 to 2^{16} . Following the general equations in Section 3.1 and using the cost expressions in Table 3, the performance speedup and energy scaling results are given in Figures 1(a) and 1(c) for the high-power processor and low-power processor cases, respectively. Figures 1(b) and 1(d) show the performance efficiency and energy efficiency plots. Note that the default settings in Table 1 are used for the basic variables.

We see from Figure 1 that energy scaling is worse than the performance speedup as the number of CPUs increases.

Similarly, the performance efficiency grows faster than energy efficiency. Note that the performance efficiency curves in Figures 1(b) and 1(d) are not same because the low-power processor runs slower than the high-power processor. However, they look similar because the memory latency dominates the execution time in the application. When comparing the different results for the low-power processor and high-power processor, we see that the differences between performance metrics and energy metrics are much larger with the low-power processor. The reason is that network energy overhead takes a much larger fraction when using the low-power processor comparing with the high-power processor. Similarly, the energy efficiency is lower when using the low-power processor. This is again because memory and network consumes the majority of the energy consumption, resulting in significant energy overhead.

- **Dense Matrix-Vector Multiplication (DMVM)**

The serial runtime of multiplying a dense matrix of dimension $n \times n$ with a vector is $t_c n^2$, where t_c is the time for a single multiply-add operation. We assume a 2-D partitioning method for the parallel matrix-vector multiplication. In this case, the $n \times n$ matrix is divided into p blocks to map onto p CPUs, and each block has $(n/\sqrt{p}) \times (n/\sqrt{p})$ elements. The vector is also distributed among the p CPUs. The parallel computation consists of aligning the vector, column-wise broadcasting, and row-wise reduction [10].

Using the cost expressions in Table 3, Figure 2 plots the results when multiply a $16K \times 16K$ matrix with a $16K$ vector. Again, we see that the energy efficiency scales worse than the performance efficiency. Also, the energy scaling keeps increasing at a rate less than that of the speedup. The variations, which we observed with FFT, between the low-power processor and high-power processor still hold.

4 Voltage/Frequency Scaling

Dynamic voltage/frequency scaling (DVFS) has been widely used to decrease power consumption under certain performance bounds [5, 13, 16, 30]. In this section, we study the potential benefits of employing DVFS techniques in our target large-scale parallel system. Note that although it is possible to apply DVFS to network links [31] and memory modules [3], such techniques are not commonly used in practice; thus, we limit our discussion to CPUs.

To investigate the impacts of DVFS techniques on energy consumption and energy scalability, we extend the energy scalability metrics to include the voltage/frequency levels. Assuming that $E_{p,\alpha}$ denotes the energy consumption per CPU for a parallel system with p CPUs running at the frequency of αf_{max} , the following equations give the corresponding energy scaling and energy efficiency metrics:

$$ES_\alpha = \frac{E_{1,1}}{E_{p,\alpha}}, \quad (19)$$

$$EE_\alpha = \frac{ES_\alpha}{p} = \frac{E_{1,1}}{pE_{p,\alpha}}. \quad (20)$$

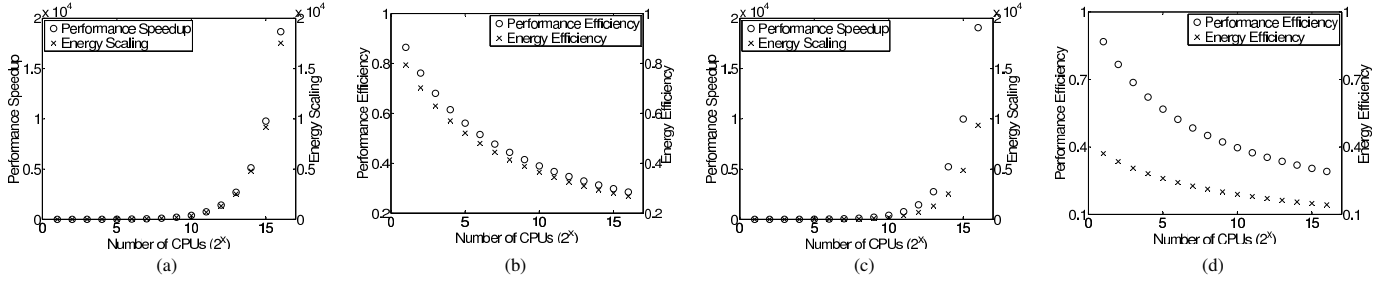


Figure 1. Performance and energy results for FFT. (a) and (b) for the high-power processor; (c) and (d) for the low-power processor.

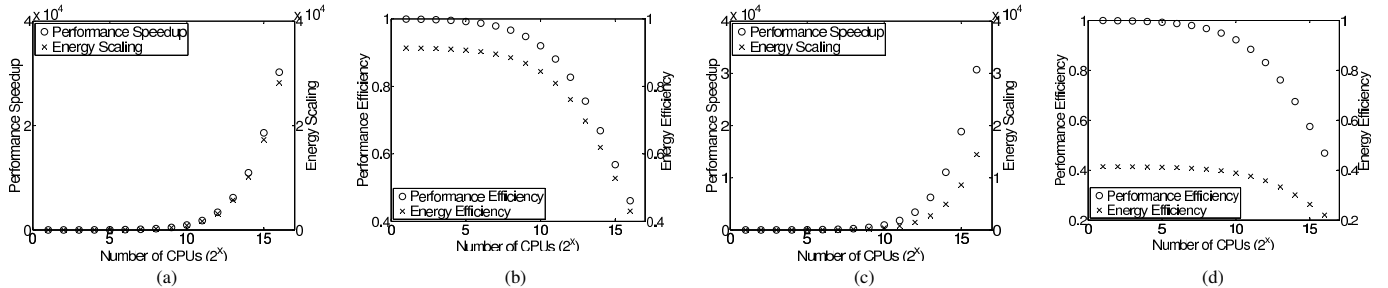


Figure 2. Performance and energy results for DMVM. (a) and (b) for the high-power processor; (c) and (d) for the low-power processor.

Decreasing the supply voltage on a fixed system configuration has impacts on both the operating frequency and the basic power numbers. In the CMOS technology, these impacts can be represented using the following equations [19, 22]:

$$f_{max} \propto \frac{(V_{dd} - V_{th})^\alpha}{V_{dd}}, \quad (21)$$

$$P = P_{dyn} + P_{leak} = ACV_{dd}^2 f + I_{leak} V_{dd}. \quad (22)$$

In these equations, V_{dd} and V_{th} represent supply voltage and sub-threshold voltage respectively. A is the gate activity factor, C is the total capacitance, and f is the operating frequency. I_{leak} is the leakage current, which mainly consists of subthreshold and gate-oxide leakage [22]. The energy consumption is simply the product of the total power number and the execution time. The latter can in turn be calculated as the number of cycles divided by the operating frequency.

We consider a realistic case of DVFS under $130nm$ process technology, where the range for V_{dd} is $0.7V \sim 1.5V$ and the interval between stages is $0.1V$. Based on the analytical models captured by Equations 21 and 22, we obtain the corresponding frequency and power numbers. These numbers are verified against the simulation results on HSPICE [33], a popular digital circuit simulator.

- We first study using different voltage levels on a parallel system with both problem size and the number of CPUs fixed. We take dense matrix-vector multiplication (DMVM) application as an example, assuming the problem size of

$16K$ and 256 low-power processors. The results for performance speedup and energy scaling are illustrated in Figure 3. All the results are normalized to the values when $V_{dd} = 1.5V$ (the maximum supply voltage we assumed for $130nm$ process technology). When the supply voltage is decreased, the performance speedup drops as expected. The figure also shows energy scaling decreases. This is because the energy saving on low-power processors through DVFS is less than the increased leakage on memory and link. To further illustrate this phenomenon, we alternate the CPU power numbers to $15W$ and $130W$, and the results are shown in Figure 4. Comparing with Figure 3, the trends for energy scaling in these cases differ. As the supply voltage decreases, the energy scaling goes up to a certain point and then decreases. This is due to the dominate factor changes between CPU energy savings and system leakage energy increase. As the number of processors is not changed in this study, the results for performance efficiency and energy efficiency are similar, following Equations 19 and 20.

- We now study the potential energy savings in strong scaling, where the problem size is fixed and the number of processors increases. Let us assume the voltage/frequency level of x is selected (from among a set of discrete voltage/frequency levels) for a system with p CPUs, i.e., the CPU setting is V_x/f_x . We further assume that the total energy consumption for the system with this operating point is $pE_p = e_x$ and execution time is $T_p = t_x$. Now, we can express the question to be addressed as follows: *Is it possible to use a larger number of CPUs ($p' > p$) with a smaller voltage/frequency (say V_y/f_y) to achieve the same execu-*

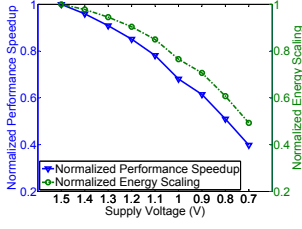


Figure 3. DVFS results of performance speedup and energy scaling for DMVM.

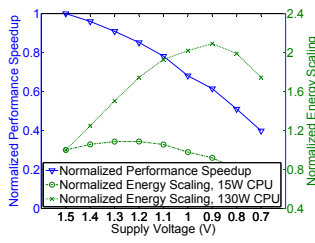


Figure 4. DVFS results for DMVM with increased CPU power.

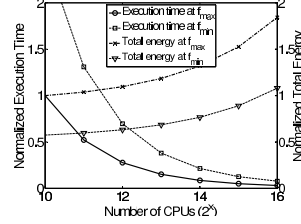


Figure 5. Execution time and total energy of DMVM using 130W high-power processor with DVFS.

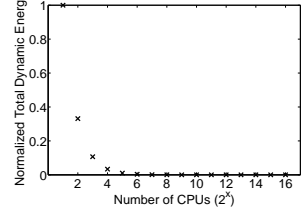
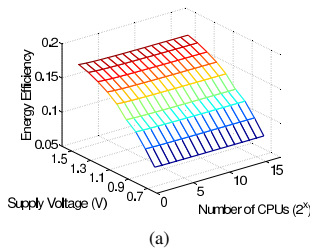
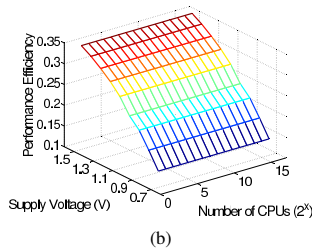


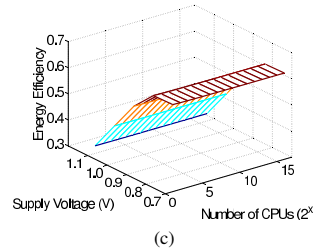
Figure 6. Normalized total dynamic energy of FFT with strong scaling and DVFS.



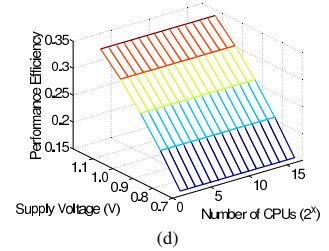
(a)



(b)



(c)



(d)

Figure 7. Energy efficiency and performance efficiency under weak scaling with DVFS for the DMVM application. (a) and (b) for the low-power processor; (c) and (d) for the high-power processor.

tion time t_x (i.e., $\frac{T_p}{f_x} = \frac{T_{p'}}{f_{y'}}$), but with lower total energy consumption than e_x ?

Figure 5 presents results, for DMVM, that help designers find answers to the question raised above for the high-power processor case. For each processor count, we calculated the execution time and energy consumption values for the maximum available voltage/frequency level (V_{max}/f_{max}) and the minimum available voltage/frequency level (V_{min}/f_{min}). The results shown are normalized to those when 2^{10} CPUs are running at maximum frequency (corresponding to 1.5V supply voltage).

If we fix an execution time value on this plot, determine the corresponding processor counts (p and p') from the execution time curves, and calculate the difference between the corresponding energy consumption results, this gives us the amount of energy we can save by using p' CPUs with V_{min}/f_{min} level, over p CPUs with V_{max}/f_{max} level. For example, we can see from Figure 5 that if $p = 2^{10}$, more than 40% energy saving can be obtained when $p' = 2$, 896 and running such number of CPUs at minimum frequency (corresponding to 0.7V supply voltage).

In order to investigate the *ideal* power saving potentials, we next assume that an infinite number of voltage/frequency levels are available (i.e. continuous scaling is possible). In this case, we try to make the execution time on p CPUs the same as that of one CPU by voltage/frequency scaling, i.e., $T_1/f_{max} = T_p/f_\alpha$ (T_1 and T_p represent the execution time *before applying DVFS* for serial system and parallel

system respectively), and study corresponding energy savings. In other words, parallelization in this case is not used to achieve speedups but to save energy instead. However, under these assumptions, Equations 21 and 22 become infeasible to use in deriving frequency from decreased supply voltage and deriving constants in leakage current. We choose a linear DVFS scenario [16], where both voltage and frequency are scaled linearly. For instance, if the V_{dd} is decreased to $x\%V_{max}$, the frequency is also decreased to $x\%f_{max}$. Also, we decided to study only the dynamic energy behavior in this case. Such a study is also important as, for most of the processors used today, dynamic energy still dominates the total energy consumption. The results for the FFT application are shown in Figure 6. As we can see, the total dynamic energy can be saved significantly if one chooses to exploit parallelism for saving energy only.

- We have also studied the change in both performance efficiency and energy efficiency under weak scaling when using iso-efficiency functions for performance with different supply voltage levels available. For example, with targeted performance efficiency at 0.35 for the DMVM application, the results are shown in Figure 7. Note that when studying the high-power processor, we assume that the highest supply voltage is 1.1V following the product specifications for 65nm process technology in industry. Figures 7(a) and 7(b) show that energy efficiency has a similar curve to that of the performance efficiency, though the former has smaller values. By comparison, Figures 7(c) and

7(d) indicate an opposite trend, namely, when the supply voltage is decreased, energy efficiency increases and performance efficiency decreases. This is due to the different power numbers of different types of processors. More specifically, when the low-power processor is used, the energy gain by DVFS is not obvious, compared with other energy consumption components. An observation when comparing these figures is that the energy efficiency value which can be maintained seems to be smaller than the performance efficiency value that can be maintained.

5 Conclusions

We have extended the well-known models for performance scalability and efficiency to include energy consumption, considering both strong scaling for fixed problem size and weak scaling for iso-efficiency. We introduce the concept of energy scalability in formal terms and study its behavior using several scientific applications. We apply the analytical models to investigate the energy savings by combining strong scaling and weak scaling with dynamic voltage/frequency scaling. Such models enable us to estimate the benefits and to find the settings with which the original problem solution time is maintained, while overall system energy consumption is reduced significantly. For example, the results in our case study show that running the same application on more CPUs at lower frequency can save as much as 40% energy.

In this work, we have tried to make the formulation of energy models simple so that it can be easily used at design time. This simplicity also brings several limitations. For example, our models do not consider specialized hardware optimizations such as possible computation/communication overlap. Also, we have only applied them to limited kernels for which performance can reasonably be captured with complexity analysis. Extending these models to general application requires more complex techniques. As a possible solution, we can utilize the testing results based on several configurations and use regression methods [21] to develop the estimation functions, which can make our models still applicable.

Acknowledgements

This work is supported in part by NSF grants CNS 0720749, CCF 0444345, and CNS 0720645. We are very grateful to the anonymous reviewers for their comments and suggestions.

References

- [1] N. R. Adiga, et al. An overview of the Blue Gene/L supercomputer. In Proceedings of the Conference on Supercomputing, 2002.
- [2] N. R. Adiga, et al. Blue Gene/L torus interconnection network. IBM Journal of Research and Development, volume 49, (2-3), 2005.
- [3] H. Ben Fradj, C. Belleudy and M. Auguin. Multi-bank main memory architecture with dynamic voltage frequency scaling for system energy optimization. In Proceedings of the EUROMICRO Conference on Digital System Design, 2006.

- [4] S. Borkar. Design challenges of technology scaling. In IEEE Micro 19 (4), July 1999.
- [5] T. Burd and R. W. Brodersen. Design issues for dynamic voltage scaling. In Proceedings of the International Symposium on Low Power Electronics and Design, 2000.
- [6] K. W. Cameron, R. Ge and X. Feng. High-performance, power-aware distributed computing for scientific applications. In IEEE Computer, Vol. 38, No. 11, 2005.
- [7] Y. Ding, K. Malkowski, P. Raghavan, M. Kandemir. Towards energy efficient scaling of scientific codes. Technical Report No. 07-012, Department of Computer Science & Engineering, Pennsylvania State University, 2007.
- [8] X. Feng, R. Ge and K. W. Cameron. Power and energy profiling of scientific applications on distributed systems. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005.
- [9] R. Ge and K. W. Cameron. Power-aware speedup. In Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, 2007.
- [10] A. Grama, A. Gupta, G. Karypis and V. Kumar. Introduction to Parallel Computing, 2nd edition, The Addison-Wesley Longman Publishing, 2003.
- [11] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. In Technical Report CS-TR-2007-1593, University of Wisconsin, April 2007.
- [12] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster. In Proceedings of the 20th International Parallel and Distributed Processing Symposium, 2006.
- [13] C. Hsu, U. Kremer and M. Hsiao. Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. In Proceedings of the International Symposium on Low Power Electronics and Design, 2001.
- [14] Intel Core 2 Extreme Quad-Core Processor QX6700 Δ Sequence Datasheet. <http://www.intel.com/design/processor/datashts/315592.htm>
- [15] International Technology Roadmap for Semiconductors 2002. <http://public.itrs.net>
- [16] C. Isci, A. Buyuktosunoglu, C-Y. Cher, P. Bose and M. Martonosi. An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget. In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006.
- [17] N. Jiang, J. Pisharath and A. Choudhary. Characterizing and improving energy-delay tradeoffs in heterogeneous communication systems. In Proceedings of the International Symposium on Signals, Circuits and Systems, 2003.
- [18] N. Kappiah, V. W. Freeh and D. K. Lowenthal. Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs. In Proceedings of the Conference on Supercomputing, 2005.
- [19] N. S. Kim, et al. Leakage current: Moore's Law meets static power. In IEEE Computer, 36(12):68-75, Dec. 2003.
- [20] H. Kimura, M. Sato, Y. Hotta, T. Boku and D. Takahashi. Empirical study on reducing energy of parallel programs using slack reclamation by DVFS in a power-scalable high performance cluster. In Proceedings of the IEEE International Conference on Cluster Computing, 2006.
- [21] B. Lee and D. Brooks. A tutorial in spatial sampling and regression strategies for microarchitectural analysis. IEEE Micro Special Issue: Hot Tutorials, 2007.
- [22] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In Proceedings of the International Symposium on High-Performance Computer Architecture, 2006.
- [23] Mellanox DataSheet. InfiniHost Dual-Port 10Gb/s InfiniBand HCA Cards with PCI-X, http://www.mellanox.com/pdf/products/hca/InfiniHost_HCA_Card_pb_1.pdf
- [24] Mellanox White Paper. InfiniBand in the Enterprise Data Center, http://www.mellanox.com/pdf/whitepapers/InfiniBand_EDS.pdf
- [25] Micron Technical Note. TN-41-01: Calculating Memory System Power For DDR3. Aug 2007.
- [26] T. Mudge. Power: a first-class architectural design constraint. IEEE Computer, Vol.34, Iss.4, Apr 2001
- [27] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi and Y. Hotta. MegaProto: 1 TFlops/10kW Rack Is Feasible Even with Only Commodity Technology In Proceedings of the conference on Supercomputing, 2005.
- [28] J. M. Paul and B. H. Meyer. Amdahl's law revisited for single chip systems. In International Journal of Parallel Programming, 35(2):101-123, 2007.
- [29] M. J. Quinn. Parallel Programming in C with MPI and OpenMP. McGraw-Hill Education, 2003.
- [30] B. Rountree, D. K. Lowenthal, S. H. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz. Bounding energy consumption in large-scale MPI programs. In Proceedings of the Conference on Supercomputing, 2007.
- [31] L. Shang, L-S. Peh and N. K. Jha. Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks In Proceedings of the 9th International Symposium on High-Performance Computer Architecture, 2003.
- [32] R. Springer, D. K. Lowenthal, B. Rountree and V. W. Freeh. Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster. In Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2006.
- [33] Synopsys, Inc. <http://www.synopsys.com/>
- [34] The TOP500 project. <http://www.top500.org/>
- [35] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. Statistical sampling of microarchitecture simulation. ACM Transaction on Modeling and Computer Simulation 16(3), July 2006.