On the Interplay of Parallelization, Program Performance, and Energy Consumption

Sangyeun Cho, *Member*, *IEEE*, and Rami G. Melhem, *Fellow*, *IEEE*

Abstract—This paper derives simple, yet fundamental formulas to describe the interplay between parallelism of an application, program performance, and energy consumption. Given the ratio of serial and parallel portions in an application and the number of processors, we derive optimal frequencies allocated to the serial and parallel regions in an application to either minimize the total energy consumption or minimize the energy-delay product. The impact of static power is revealed by considering the ratio between static and dynamic power and quantifying the advantages of adding to the architecture capability to turn off individual processors and save static energy. We further determine the conditions under which one can obtain both energy and speed improvement, as well as the amount of improvement. While the formulas we obtain use simplifying assumptions, they provide valuable theoretical insights into energy-aware processor resource management. Our results form a basis for several interesting research directions in the area of energy-aware multicore processor architectures.

Index Terms—Multicore processor, Amdahl's law, dynamic voltage and frequency scaling (DVFS), energy-delay product (EDP).

1 Introduction

A surge of attention is being paid to parallel processing with the recent emergence of commodity multicore processors. Microprocessors carrying two to eight general processing cores are commercially available [2], [15], [21], [31], [32] and projections suggest that future technologies will allow integrating many more cores, potentially in the order of hundreds to thousands, in a single chip [4], [22]. Abundant on-chip processing elements and much reduced processor-to-processor communication overhead will offer an unprecedented environment for efficient and affordable parallel processing on every desktop.

While the increased amount of on-chip computing resources promises higher performance through parallel execution of applications, suppressing the power and energy consumption remains an even more stringent constraint to the design and management of such processors [5], [26]. The allowed power level has been and will be limited to the same constant value ($\sim$150W) and the chip operating voltage will not be significantly lowered in the future, according to the ITRS projection [19]. Battery capacity and efficiency are not improving as quickly as the number of transistors in a chip increases. Henceforth, many previously developed low-power and low-energy ideas will play an even more significant role in future multicore processors, including highly beneficial dynamic voltage and frequency scaling (DVFS or simply DVS) techniques [11], [35], [36] and the capability of turning off individual processor cores [20].

• *S. Cho and R.G. Melhem are with the Department of Computer Science, University of Pittsburgh, 5407 Sennott Square, 210 S. Bouquet Street, Pittsburgh, PA 15260. E-mail: {cho, melhem}@cs.pitt.edu.*

Manuscript received 21 Oct. 2008; revised 23 Jan. 2009; accepted 19 Feb. 2009; published online 27 Feb. 2009.
Recommended for acceptance by D. Bader.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-10-0427.
Digital Object Identifier no. 10.1109/TPDS.2009.41.

This paper presents a theoretical study on the interplay of parallelization, program performance, and energy consumption. It has been previously pointed out that parallel processing can be used to lower energy instead of improving performance. For instance, Borkar [5] suggested that a perfect two-way parallelization would lead to half the clock frequency (and voltage), one-quarter of the energy consumption, and one-eighth of the power density compared with the sequential execution given the same execution time constraint. Little work has been done, however, to understand how parallelizing an application would enable us to achieve both speedup and energy improvement or how to obtain the best energy improvement given an application and a processor architecture. We will address the following specific questions:

1. What is the maximum energy improvement due to parallelization, and how can we determine the processor speeds to achieve that improvement?
2. How does static power affect the energy-optimal program speedup and energy consumption?
3. Given a target speedup, how do we set the processor speeds to minimize energy?
4. What is the condition for obtaining the minimum energy-delay product?

Our goal in this work is to follow a simple analytical approach, similar to the one used in Amdahl's law [3], to derive formulas that describe the behavior of the system. For example, according to Amdahl's law, the maximum program speedup due to parallelization is

$$Speedup = \frac{1}{s + \frac{p}{N}}, \tag{1}$$

where $(s + p) = 1, s$ (p) is the ratio of the serial (parallel) portion in the program, and N is the number of processors. Using the same input parameters, we can calculate how parallelization improves energy consumption:

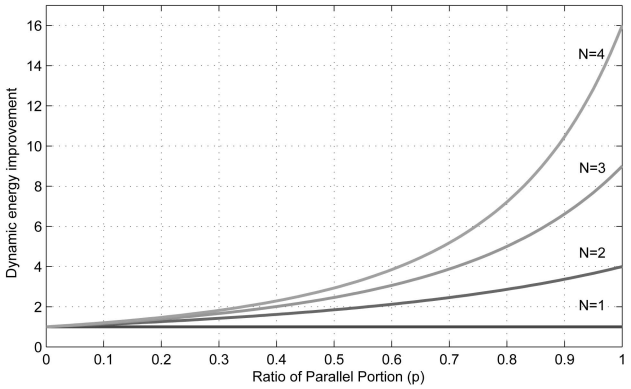

Fig. 1. Achievable dynamic energy improvement assuming $\alpha = 3$ and using 1, 2, 3, and 4 processors given the parallel portion's ratio of a program.

$$Improvement\ in\ Dynamic\ Energy = \frac{1}{\left(s + \frac{p}{N^{(\alpha-1)/\alpha}}\right)^{\alpha}}, \quad (2)$$

when the parallel program execution time is identical to the sequential execution time and the dynamic power consumption of a processor running at frequency f is proportional to f^{α}. The above equation, whose detailed derivation will be shown in Section 3, illustrates that more parallelism (larger p and smaller s) and more processors (larger N) help reduce energy. Fig. 1 presents a plot of (2). For a typical value of α, the energy improvement function in (2) is growing faster than the speedup function in (1) with p or N.[1]

The result obtained in this paper reveals interesting relationships between processor speeds in the sequential and parallel regions of a program and between dynamic and static energy consumption. For instance, when individual processors cannot be turned off, the total energy is minimized when the dynamic energy is equal to $1/(\alpha - 1)$ times the static energy regardless of the value of N or s. Moreover, we find that minimum energy is achieved at processor speeds smaller than the maximum speeds only when $\lambda < (\alpha - 1)/N$, where λ is the ratio of the static power consumption to the dynamic power consumption at the maximum processor speed. Under this condition, the ratio between the processor speed of the serial and parallel section to minimize energy is $N^{1/\alpha}$. When the condition does not hold, the program's serial section must be executed at full speed. If $\lambda > (\alpha - 1)$, processor speeds in both the serial and the parallel sections must be set to the maximum speed to achieve the minimum total energy. The above conditions are greatly relaxed if we can turn off individual processors to save static energy consumption.

The rest of this paper is organized as follows: Section 2 presents our problem formulation and the two machine models that we will consider in the paper. Sections 3 and 4 study the problem of minimizing energy consumption for the two machine models, followed by a discussion on energy-delay product in Section 5. Section 6 further discusses how our derivations can be used in situations when there are constant-speed operations (such as memory access) whose latencies do not depend on the processor speed. Sections 3, 4, and 5 assume that the processor speeds simply determine the execution time of a program. Related work will be discussed and contrasted with our work in Section 7 and conclusions will be summarized in Section 8.

1. In literature α is between 2 and 3, typically 3 [30], [35].

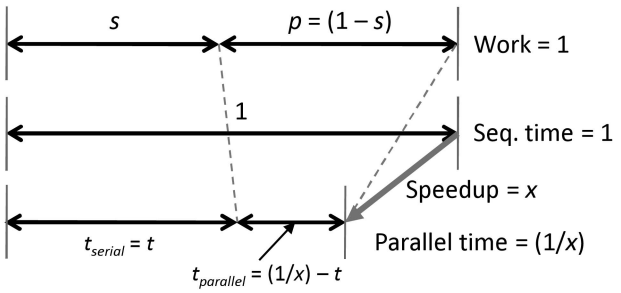

Fig. 2. Normalized "work" and "time." "Parallel time" is partitioned into serial and parallel regions.

2 Problem Formulation and Machine Models

2.1 Problem Formulation and Assumptions

We have an application model identical to that of Amdahl's law. An application has a serial section that can be executed by a single processor and a parallel section that can be executed by any number of processors in the system, i.e., fully parallelizable. When the number of processors employed is N, the speedup of the parallel section is N. We do not consider the overhead of processor-to-processor communications.

We normalize the sequential execution time of the program to be 1, in order to present our derivation in an intuitive way. Similarly, we normalize the amount of work (i.e., number of cycles) in the program to be 1. Therefore, the maximum clock frequency, F_{max}, has a relative speed of 1 and the program has the serial portion whose amount of work is represented with s, and the parallel portion with p (or $1 - s$). Fig. 2 shows this arrangement. The program speedup is denoted by x and the resulting program execution time with $y = 1/x$. The clock frequencies for the two regions in the work, namely, s and p, are calculated as follows:

$$f_s = \frac{s}{t}, \quad (3)$$

$$f_p = \frac{1 - s}{(y - t) \cdot N}. \quad (4)$$

We assume a DVFS scheme where voltage and frequency are changed linearly. To be general, we also assume that the power consumption of each processor consists of two components, a frequency-dependent component that can be controlled by changing the frequency of the processor (DVFS), and a frequency-independent component that is not controlled by DVFS. We call these two components as "dynamic" and "static," respectively. There have been many studies to either theoretically or empirically model the dependence of the power consumption on the operating frequency, and most of these studies conclude that this dependence can be approximated by $C \cdot f^{\alpha}$, for some value of $\alpha \geq 2$ [30], [35]. The frequency-independent component is a constant that depends on the technology and system

architecture. However, assuming that $E_{max} = C \cdot F_{max}^{\alpha}$ is the energy of the frequency-dependent component at the maximum processor speed, the energy for the frequency-independent component can be normalized with regard to E_{max} and expressed as $\lambda \cdot E_{max}$. In this paper, we further normalize E_{max} to 1 so that the static power consumption is simply λ. Although the power model described above is approximate, its simplicity allows us to compare the effect of parallelism on the choice of processor speed policies using closed-form formulas. It is common to use simplified power models to derive DVFS policies [33].

Clearly, the benefit of any DVFS technique largely depends on the value of λ. Specifically, for systems with relatively large λ, the frequency-independent component of power dominates the total energy consumption, and thus, applying DVFS techniques will not produce any appreciable energy savings. However, in systems with relatively small λ, applying DVFS techniques can reduce the total system energy consumption. Many DVFS algorithms have been proposed recently and demonstrated to reduce power consumption [11], [18], [25], [29], [35], [36], and many commercial microprocessors are now equipped with DVFS capabilities [8], [9], [16], [27].

For a given problem, s is fixed, and for a given architecture, N, α, and λ are fixed. Hence, the energy consumption, E, is a function of t and y. Specifically,

$$E(t, y) = t \cdot f_s^{\alpha} + N \cdot (y - t) \cdot f_p^{\alpha} + N \cdot \lambda \cdot y. \tag{5}$$

In (5), the three terms represent energy for the serial portion, energy for the parallel portion, and energy for the static power consumption during the whole execution time, respectively. We do not consider the processor temperature, and hence, the term for the static energy is the product of per-processor power consumption rate, λ, the number of processors, N, and the total execution time, y. We do not assume a specific interprocessor network topology and do not consider energy consumption of the interprocessor network.

In our problem formulation, we assumed that the processor speed (or processor's clock frequency) solely determines the runtime of a program. However, certain "constant-speed" operations, such as memory access (caused by a cache miss in cache-based systems) and I/O processing, may take a fixed amount of time that is independent of the processor speeds. Consequently, increasing or decreasing the processor speed will have "sublinear" effect (rather than "linear") on performance [14]. For clear presentation and intuitive discussion, we will not consider the impact of the constant-speed operations on program execution time (and hence, energy consumption) in the following three sections. However, Section 6 will specifically discuss how constant-speed operations affect our derivations and intuitions learned.

Finally, it is important to note that this work pays little attention to practical issues of parallelizing an application or mapping serial and parallel regions of an application to multiple cores. We assume (as Amdahl's law suggests) that an application is perfectly parallelized given its parallelism and work described by a parallel code region can be perfectly distributed to an arbitrary number of processors.

2.2 Two Machine Models

In the problem formulation in (5), we assumed that processors consume static energy in both the serial and the parallel regions. That is, even when a processor is not assigned a task to execute (and thus, sits idle), it consumes static energy. Naturally, one would regard an idle processor as an opportunity to save energy consumption if processors can be turned off when not busy, given a mechanism to turn off individual processors. Because low-energy consumption via suppressing static power will be increasingly important, viable mechanisms such as separate power supply designs are actively explored in the context of multicore processors in industry [20]. Hence, we will study in this work two machine models: one without and one with the capability to turn off individual processors. Throughout this paper, we refer to these two machine models $\mathcal{M}_A$ and $\mathcal{M}_B$.

For the simplicity of our derivation, we assume that $\mathcal{M}_B$ can turn off or on processors without any overhead. Given the same processor speed setting, the dynamic energy consumption of $\mathcal{M}_B$ is the same as that of $\mathcal{M}_A$: sum of the first two terms in (5). Only the static energy part need be replaced by $t \cdot \lambda + N \cdot (y - t) \cdot \lambda$.

We further assume that processors can run at an arbitrary clock frequency, subject to the maximum frequency F_{max} in both machine models. While processor clock frequencies in real chip implementations are typically discrete, previous work showed that energy savings using discrete frequencies closely match that of continuous frequencies [18]. The speedup x one would achieve with parallelization and processor speed scaling is subject to

$$x \leq \frac{1}{s + \frac{p}{N}} \tag{6}$$

according to Amdahl's law in (1). It is clear that this condition is equivalent to $f_s, f_p \leq F_{max} = 1$, and $y \geq s + \frac{p}{N}$. We will discuss the two machine models in detail in the following two sections.

3 Machine Model $\mathcal{M}_A$: Processors Cannot Be Turned Off Individually

3.1 The Case of $x = 1$

We first obtain the minimum energy consumption when $x = 1$ (or $y = 1$), i.e., program execution time is unchanged. Imposing the condition $x = 1$ is similar to setting a deadline, which is the original sequential execution time, to finish the computation. While one can further reduce the dynamic energy by slowing down processors, considering the case of $x = 1$ provides us with interesting insights as well as a basis for later discussions. To minimize the total energy, we rewrite (5) as

$$E(t) = t\left(\frac{s}{t}\right)^{\alpha} + N(1 - t)\left(\frac{1 - s}{(1 - t)N}\right)^{\alpha} + N\lambda. \tag{7}$$

Next, we obtain the derivative of $E(t)$ with respect to t,

$$\frac{dE(t)}{dt} = \frac{-(\alpha - 1) \cdot s^{\alpha}}{t^{\alpha}} + \frac{(\alpha - 1) \cdot (1 - s)^{\alpha}}{(1 - t)^{\alpha} \cdot N^{(\alpha - 1)}}, \tag{8}$$

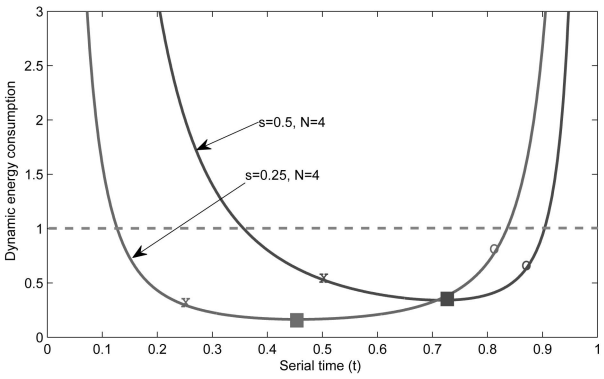

Fig. 3. Dynamic energy consumption versus serial time for two cases, $s = 0.25$ and $s = 0.5$, when $N = 4$. The bound of t is marked with "X" (when $f_s = F_{max} = 1$) and "O" (when $f_p = F_{max} = 1$). The minimum energy point in each curve (at $t = t^*$) is marked with a filled rectangle.

and then, we compute the value of t which minimizes $E(t)$ by setting $dE(t)/dt$ to 0 and obtain

$$\frac{t}{1-t} = \frac{s}{1-s} \cdot N^{\frac{(\alpha-1)}{\alpha}}. \tag{9}$$

Hence, the value of t which minimizes $E(t)$ is

$$t^* = \frac{s}{s + \frac{p}{N^{(\alpha-1)/\alpha}}}. \tag{10}$$

We are ready to obtain the values of f_s and f_p, which minimize $E(t)$ using (3), (4), and (10). Specifically,

$$f_s^* = \frac{s}{t^*} = s + \frac{p}{N^{(\alpha-1)/\alpha}}, \tag{11}$$

$$f_p^* = \left(s + \frac{p}{N^{(\alpha-1)/\alpha}}\right) \cdot N^{-\frac{1}{\alpha}}, \tag{12}$$

$$= f_s^* / N^{\frac{1}{\alpha}}. \tag{13}$$

Both f_s^* and f_p^* are a function of s and N in (11) and (12), and (13) shows the relationship between f_s and f_p when $E(t)$ is minimized. Interestingly, the ratio between the two frequencies, f_s^*/f_p^*, is a function of N, but not s.

Finally, from (7) and (10), we obtain the minimum energy consumption

$$E_{min} = E(t^*) = \left(s + \frac{p}{N^{(\alpha-1)/\alpha}}\right)^{\alpha} + N \cdot \lambda. \tag{14}$$

Fig. 1 depicts the maximum energy improvement due to parallelization (E_{min}^{-1}) when the number of processors is varied between one and four, assuming $\alpha = 3$ and $\lambda = 0$. It is clear that energy improvement is a function monotonically increasing with p and N. The curves are also higher than those given by Amdahl's law (not shown). Fig. 3 shows how the overall energy ($E(t)$) changes as we adjust t. It also presents t^*, the value of t that minimizes $E(t)$. Note that the optimal solution obtained for f_s^* and f_p^* is feasible since both the frequencies are smaller than the maximum frequency $F_{max} = 1$.

3.2 The Case of Unrestricted x

Amdahl's law explores the effect of parallelization on speedup, and we have described the effect of parallelization on energy consumption when the program execution time is unchanged, i.e., $x = 1$. We note that the fixed value of x, in turn, fixes the static energy consumption. We have thus focused only on the dynamic energy consumption. Hence, the optimal processor speeds in (11) and (12) are independent of λ. In this section, we relax the program execution time constraint and revisit the same problem of minimizing the total energy consumption. Unrestricting x will expose the impact of static power consumption on the optimal speed of a program's serial and parallel sections.

We begin by setting the derivatives of (5) with respect to both t and y to zero as follows:

$$\begin{aligned} &\frac{\partial E}{\partial t} = \frac{-(\alpha-1)\cdot s^{\alpha}}{t^{\alpha}} + \frac{(\alpha-1)\cdot(1-s)^{\alpha}}{(y-t)^{\alpha}\cdot N^{(\alpha-1)}} = 0 \implies \\ &\frac{t}{y-t} = \frac{s}{1-s}\cdot N^{(\alpha-1)/\alpha}, \end{aligned} \tag{15}$$

$$\begin{aligned} &\frac{\partial E}{\partial y} = \left(\frac{(\alpha-1)\cdot(1-s)^{\alpha}}{(y-t)^{\alpha}\cdot N^{\alpha-1}} - \lambda N\right) = 0 \implies \\ &(y-t) = \left(\frac{\alpha-1}{\lambda}\right)^{\frac{1}{\alpha}} \cdot \frac{1-s}{N}. \end{aligned} \tag{16}$$

Solving (15) and (16) for t and y gives

$$t^* = \left(\frac{\alpha-1}{\lambda N}\right)^{\frac{1}{\alpha}} \cdot s, \tag{17}$$

$$y^* = \left(\frac{\alpha-1}{\lambda N}\right)^{\frac{1}{\alpha}} \cdot \left(s + \frac{p}{N^{(\alpha-1)/\alpha}}\right). \tag{18}$$

With t^* and y^*, we can use (3) and (4) to calculate the optimum frequencies

$$f_s^* = \left(\frac{\lambda N}{\alpha-1}\right)^{\frac{1}{\alpha}}, \tag{19}$$

$$f_p^* = \left(\frac{\lambda}{\alpha-1}\right)^{\frac{1}{\alpha}} = f_s^* / N^{\frac{1}{\alpha}}, \tag{20}$$

from which we can compute the minimum energy. Again, note that f_s^* and f_p^* are independent of s and that the ratio between them is $N^{\frac{1}{\alpha}}$. An interesting observation is that at f_s^* and f_p^*, the dynamic energy is given by

$$\begin{aligned} E_{dynamic} &= t^* \cdot f_s^{\alpha} + N \cdot (y^* - t^*) \cdot f_p^{\alpha} \\ &= \frac{1}{\alpha-1} \cdot N\lambda \cdot y^*, \end{aligned} \tag{21}$$

which is equal to $1/(\alpha-1)$ of the static energy, $E_{static} = N\lambda \cdot y^*$. In other words, the total energy consumption is minimized when the dynamic energy consumption is $1/(\alpha-1)$ times the static energy. This relation holds during the execution of both the serial and the parallel sections of the program.

The above solution is only applicable if both f_s^* and f_p^* are smaller than F_{max}, however, necessitating that $\lambda \leq (\alpha-1)/N$. If λ, the ratio between the static and dynamic power, is large so that it is not possible to maintain the aforementioned relation between the static and

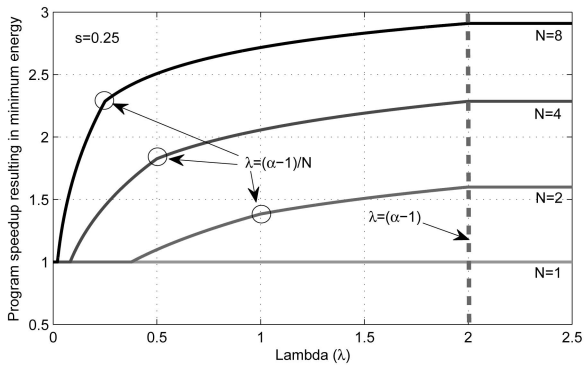

Fig. 4. λ changes the speedup of a program when its energy consumption is minimized. We assume that $\alpha = 3$.

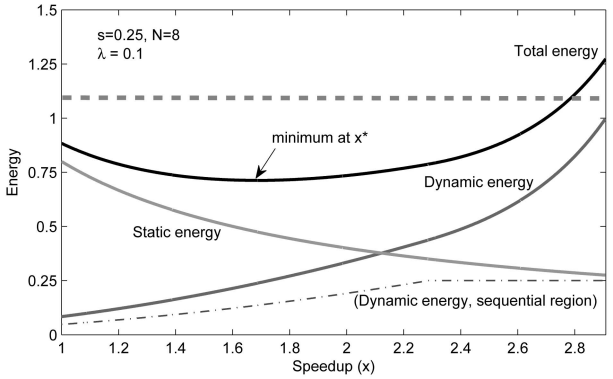

Fig. 5. Optimal energy at the program speedup of x when $\alpha = 3$. The thick dotted line shows the sequential machine's energy consumption $(1 + \lambda)$.

dynamic energy, we should set $f_s = 1$ and find the values of y and f_p that minimize the total energy consumption. Denoting these values by y^{**} and f_p^{**}, we obtain

$$y^{**} = s + \left(\frac{p}{N}\right) \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{\frac{1}{\alpha}}, \tag{22}$$

$$f_p^{**} = \left(\frac{\lambda}{\alpha - 1}\right)^{\frac{1}{\alpha}}. \tag{23}$$

Again, these values result in the dynamic power consumption being $1/(\alpha - 1)$ times the static power consumption during the execution of the parallel portion of the program.

In order to summarize the relationship between λ and speedup that results in minimum energy consumption, we show that relationship in Fig. 4. In this figure, the values of λ are divided into three regions. When $\lambda \leq (\alpha - 1)/N$, the solution for the optimum energy consumption problem is given by (18), (19), and (20). When $(\alpha - 1)/N < \lambda \leq (\alpha - 1)$, the solution is given by $f_s = 1$, (22), and (23). Finally, when $\lambda > (\alpha - 1)$, the solution is given by $f_s = f_p = 1$, and the speedup is that given by Amdahl's law (1).

3.3 Optimal Energy Consumption Given a Speedup

We have thus far considered the problem of calculating the optimal speeds of processors (hence, program speedup) to minimize the total energy consumption given p, λ, and N. In this section, we consider the problem of how to set the speeds of the processors (f_s and f_p) to minimize the total energy consumption when the target program speedup x (or, equivalently, y) is specified. Because the static energy $N\lambda \cdot y$ is immediately determined given x, we only need to minimize the dynamic energy while meeting the program speedup requirement and our solution derived from (3), (4), (18), (19), and (20) is as follows:

$$\text{If} \quad x \leq \frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}}, \quad f_s^* = x f_{s,x=1}^*, f_p^* = x f_{p,x=1}^*, \tag{24}$$

$$\text{if} \quad \frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}} < x \leq \frac{1}{s + \frac{p}{N}}, \quad f_s^* = 1, f_p^* = \frac{px}{N(1 - sx)}, \tag{25}$$

where $f_{s,x=1}^*$ and $f_{p,x=1}^*$ are the optimal frequencies when $x = 1$, as given in (11) and (12), respectively. We call the interval in (24) as the *linear frequency scaling interval* because the energy-optimal f_s and f_p can be obtained by simply scaling $f_{s,x=1}^*$ and $f_{p,x=1}^*$ by a factor of x. We also note that the upper bound of the condition in (24) is, in fact, equivalent to $\lambda \leq (\alpha - 1)/N$.

Fig. 5 shows how the minimum energy consumption changes as we target a different program speedup, along with the contribution of the dynamic and static energy consumption. It is noticeable from the plot that the dynamic energy of the sequential region saturates at around $x = 2.3$. This is due to the inability to scale f_s beyond F_{max}. Finally, when $f_s = f_p = 1$ (i.e., at the maximum speedup), the dynamic energy is 1; it is the same as that of sequential execution.

We point out that static power consumption plays an important role in determining the minimum energy consumption of an application. Fig. 6 depicts the improvement ratio of the minimum energy at different program speedups, relative to the baseline sequential execution of a given application. The plot clearly demonstrates that a smaller λ leads to a larger energy improvement ratio at any selected program speedup. Moreover, the largest energy improvement ratio occurs at a smaller program speedup. In other words, one can slow down processor speeds further to benefit from reducing dynamic energy to a greater degree before static energy starts to offset the benefit, if λ is small.

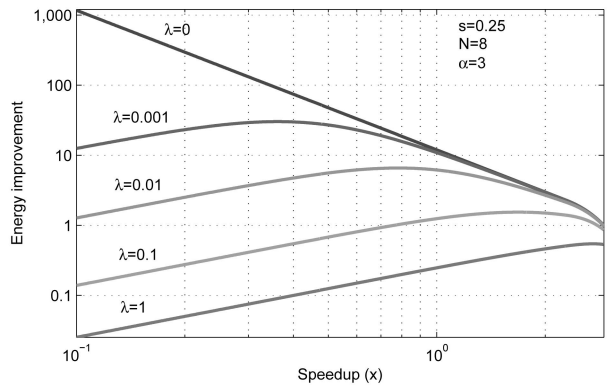

Fig. 6. Energy improvement at different speedups over sequential execution.

The formulas derived in this section also show that, for a given processor implementation (λ, α, and N), the minimum total energy is a monotonically decreasing function of the number of processors, N, as long as the parallel section of the code can be executed on N processors. Hence, from the point of view of the total energy consumption, all available processors should be used. It should be noted, however, that this result and all the results given in this section assume that the N processors in the system consume static power, even during the serial section of the code.

4 Machine Model $\mathcal{M}_B$: Processors Can Be Turned Off Individually

For machine model $\mathcal{M}_A$ considered in the previous section, there was no reason for using fewer than the available N processors during parallel execution. However, because $\mathcal{M}_B$ allows us to turn off individual processors, there may be an advantage of not using all the available processors to execute the parallel sections of programs. Hence, in this section, we use a variable, n, to denote the number of processors used during the parallel program sections. Consequently, we rewrite the energy equation with t, y, and n as parameters:

$$E(t, y, n) = t f_s^{\alpha} + n(y - t) f_p^{\alpha} + t\lambda + n(y - t)\lambda. \tag{26}$$

We note that $1 \leq n \leq N$, where N is the total number of processors that can be used. While n assumes discrete values, we will treat it as a continuous function when we derive the conditions for optimal energy consumption.

4.1 Conditions for Optimal Energy Consumption

We will begin from (26) to derive the conditions for obtaining the minimum total energy consumption

$$\begin{aligned} \frac{\partial E}{\partial t} &= \frac{s^{\alpha} \cdot (1 - \alpha)}{t^{\alpha}} - \frac{(1 - s)^{\alpha}(1 - \alpha)}{n^{(\alpha-1)}(y - t)^{\alpha}} + \lambda - n\lambda = 0 \Longrightarrow \\ \frac{s^{\alpha}(1 - \alpha)}{t^{\alpha}} &= \frac{(1 - s)^{\alpha}(1 - \alpha)}{n^{(\alpha-1)}(y - t)^{\alpha}} + n\lambda - \lambda. \end{aligned} \tag{27}$$

Similarly, we have

$$\begin{aligned} \frac{\partial E}{\partial y} &= \frac{(1 - s)^{\alpha}}{n^{(\alpha-1)}} \cdot \frac{(1 - \alpha)}{(y - t)^{\alpha}} + n\lambda = 0 \Longrightarrow \\ y &= t + \frac{p}{n} \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{1/\alpha}. \end{aligned} \tag{28}$$

Note that we assume $\lambda > 0$ in the above. Finally, we have

$$\begin{aligned} \frac{\partial E}{\partial n} &= \frac{(1 - s)^{\alpha}}{n^{\alpha}} \cdot \frac{(1 - \alpha)}{(y - t)^{(\alpha-1)}} + (y - t) \cdot \lambda = 0 \Longrightarrow \\ n &= \frac{p}{y - t} \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{1/\alpha}. \end{aligned} \tag{29}$$

The result in (29) is, however, identical to (28). We will discuss why this is the case, after first obtaining t^*, f_s^*, and f_p^*.

From (27) and (28), we obtain t^* which minimizes the total energy consumption

$$t^* = s \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{1/\alpha}. \tag{30}$$

It is interesting to observe that t^* for machine model $\mathcal{M}_B$ in (30) is quite similar to that of $\mathcal{M}_A$ in (17), except that it does not have n in the scene. That is, the energy-optimal time allocated to the serial section (thus the speed of the single processor used) does not depend on the number of processors, n. It is only dependent on the amount of work s, the formula for dynamic power f^{α}, and the ratio of static power to the dynamic power λ.

Now, the optimal execution time y^* is obtained from (28) and (30) in the following, as well as the optimal frequencies to the serial and the parallel sections

$$y^* = s \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{1/\alpha} + \frac{p}{n} \cdot \left(\frac{\alpha - 1}{\lambda}\right)^{1/\alpha}, \tag{31}$$

$$f_s^* = \frac{s}{t^*} = \left(\frac{\lambda}{\alpha - 1}\right)^{1/\alpha}, \tag{32}$$

$$f_p^* = \frac{p}{(y^* - t^*) \cdot n} = \left(\frac{\lambda}{\alpha - 1}\right)^{1/\alpha}. \tag{33}$$

The above equations illustrate that the optimal energy is obtained when $f_s^* = f_p^*$, i.e., all the processors are given the same speed during program execution regardless of the serial or the parallel section. Moreover, the optimal processor speeds f_s^* and f_p^* do not depend on n or s; they are immediately determined by λ and α. At first, this seemed counterintuitive to us; why are f_s^* and f_p^* not dependent on n or s?

To address the above question, consider the same value of f_s^* and $f_p^*, (\frac{\lambda}{\alpha-1})^{1/\alpha}$. We call this value as the *energy-optimal processor speed* in that it leads to the smallest total energy consumption for a processor if performance is not considered, given the dynamic power consumption rate f^{α}, and the static power consumption rate λ. If we do not consider program execution time, as long as the total amount of work is distributed to processors and they are kept busy, we obtain the minimum energy consumption. Hence, the solution does not depend on the number of processors used. That our solution is independent of n is indirectly verified by the redundant result we obtain for $\frac{\partial E}{\partial n}$ (in (29)), which is identical to (28).

The above result can be clarified by a simple example. Assume that the s cycles in the serial section of a program are executed by a processor P_1, at a speed f_s, while the $(1 - s)$ cycles in the parallel section are executed by two processors P_1 and P_2 at speed f_p (see Fig. 7a). Because only executing processors incur static energy, the energy consumption for the system is equal to that consumed when P_1 executes the $(1 - s)/2$ cycles executed by P_2 at speed f_p after it finishes executing its own s cycles at speed f_s and $(1 - s)/2$ cycles at speed f_p (see Fig. 7b). However, because of the convexity of the power function, we know that, with a single processor

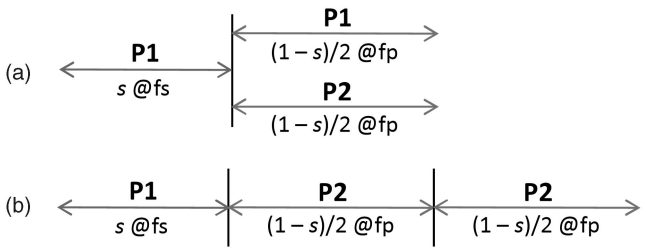

Fig. 7. Running a program having a serial portion s with (a) two processors and (b) one processor.

executing, the optimal energy consumption occurs when the processor executes at a fixed speed, which implies that $f_s = f_p = f_{opt}$. In this example, to find the optimum speed f_{opt} which minimizes the total energy $E = s \cdot f_{opt}^{\alpha-1} + (1-s) \cdot f_{opt}^{\alpha-1} + \lambda/f_{opt} = f_{opt}^{\alpha-1} + \lambda/f_{opt}$, we set $\frac{dE}{df} = (\alpha-1) \cdot f^{\alpha-2} - \lambda/f^2 = 0$ to obtain $f_{opt} = (\lambda/(\alpha-1))^{1/\alpha}$. This argument is valid for $n > 2$ as well.

The results obtained in (31), (32), and (33) are only valid if f_s^* and f_p^* are smaller than $F_{max} = 1$. That is when

$$\left(\frac{\lambda}{\alpha-1}\right)^{1/\alpha} < 1 \quad \text{or} \quad \lambda < (\alpha-1). \tag{34}$$

This condition is greatly relaxed compared with that of $\mathcal{M}_A$, where the condition was $\lambda < (\alpha-1)/N$. If condition (34) does not hold, i.e., processors consume too much static power, both f_s^* and f_p^* should be set to 1, the maximum speed, to minimize the program execution time and the static energy consumption.

Given the same processor speeds used in program execution, it is clear that the ratio between the energy consumption in the program's serial section and the parallel section is s/p. The minimum energy consumption including both dynamic and static energy is

$$\begin{aligned} E_{total} &= E_{dynamic} + E_{static} \\ &= \left(\frac{\alpha-1}{\lambda}\right)^{1-\frac{1}{\alpha}} + \lambda \cdot \left(\frac{\alpha-1}{\lambda}\right)^{1/\alpha}, \end{aligned} \tag{35}$$

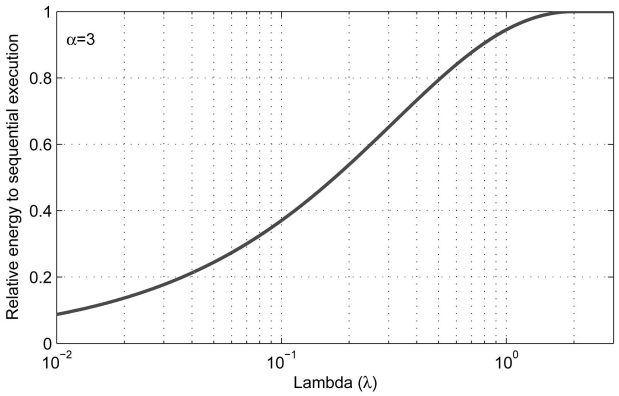

Fig. 8. Minimum total energy at different λ values for $\mathcal{M}_B$. X axis is log-scale.

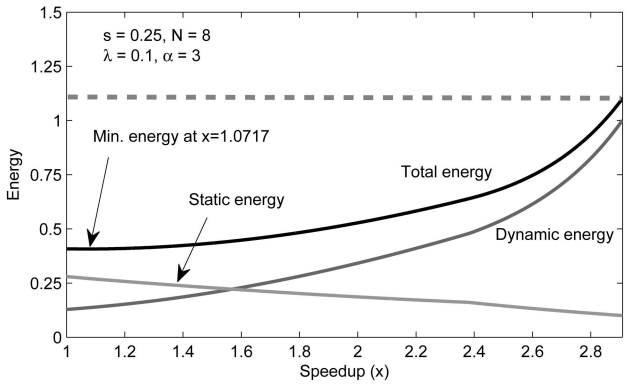

Fig. 9. Optimal energy at the program speedup of x. The thick dotted line shows the sequential machine's energy consumption $(1+\lambda)$.

which, as in the case of model $\mathcal{M}_A$, is achieved when the static energy is $(\alpha-1)$ times the dynamic energy. Fig. 8 depicts the minimum energy consumption of machine model $\mathcal{M}_B$ at different λ values. The plot demonstrates that the minimum energy consumption decreases monotonically with λ.

4.2 Optimal Energy Consumption Given a Speedup

In this section, we consider the problem of obtaining the minimum total energy consumption when a desired speedup x (equivalently, program execution time y) is specified. This problem is equivalent to finding a value of t which minimizes the total energy (26), given a value of y and n, and accordingly, we have the same derivation as in (27). Unfortunately, the derivation in (27) by itself does not give us a closed-form formula to directly solve; hence, we resort to numerical methods to present our results in this section.

Fig. 9 shows how the minimum energy consumption changes as we target a different program speedup, x, along with the contributions of the dynamic and static energy consumption. At the maximum speedup dictated by Amdahl's law, when $f_s = f_p = F_{max} = 1$, the dynamic energy consumption reaches 1 (i.e., same as the dynamic energy consumption of sequential execution) and the total energy reaches $1+\lambda$ (i.e., same as the total energy consumption of sequential execution).

Fig. 10 further shows how the improvement ratio of the minimum energy changes with the different value of

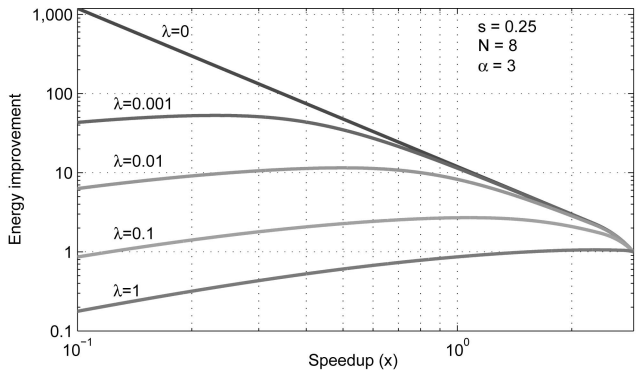

Fig. 10. Energy improvement at different speedups over sequential execution.

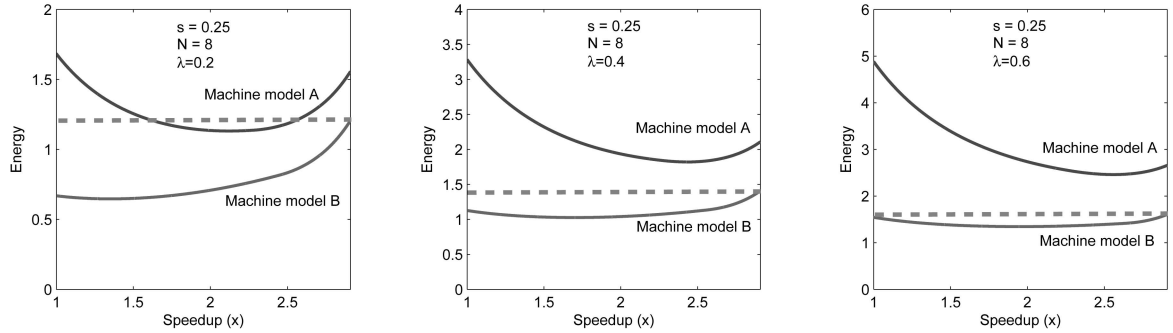

Fig. 11. Minimum energy consumption at different speedups for $\mathcal{M}_A$ and $\mathcal{M}_B$. Plots for three different λ values are shown: 0.2, 0.4, and 0.6 from left. The thick dotted line shows the sequential machine's energy consumption $(1+\lambda)$.

program speedup. As we saw in Fig. 6, it is again confirmed that the value of λ determines the shape and direction of each energy improvement-speedup curve.

4.3 Energy Advantage of Turning Off Idle Processors

Fig. 11 compares the two machine models, $\mathcal{M}_A$ and $\mathcal{M}_B$, with respect to the minimum energy consumption at different program speedups. It is shown that the energy consumption of $\mathcal{M}_B$ is strictly lower than that of $\mathcal{M}_A$ at any desired speedup. At the maximum program speedup, $\mathcal{M}_B$ has the same energy consumption as sequential execution, regardless of the number of processors used.

Finally, we consider the energy consumption of the two machine models at the maximum program speedup given by Amdahl's law, $1/(s+p/N)$. Under this condition, $\mathcal{M}_A$ consumes $1+(s+p/N)\cdot N\lambda$ while $\mathcal{M}_B$ consumes $(1+\lambda)$. Hence, the ratio between the consumption in the two machine models at the maximum program speedup is

$$\frac{E(\mathcal{M}_A)}{E(\mathcal{M}_B)} = 1 + \frac{\lambda}{1+\lambda}\cdot(N-1)\cdot s. \tag{36}$$

Fig. 12 depicts this ratio as we vary the number of processors. It is shown that a larger λ value leads to a higher ratio between the energy consumption values of the two machine models. That is, as expected, the advantage of turning off processors increases when the static power is larger. Furthermore, when more processors are introduced, the ratio also grows, suggesting that $\mathcal{M}_A$ consumes much more energy as we scale the system than $\mathcal{M}_B$.

5 Minimizing Energy-Delay Product

In the previous sections, we have considered the problem of obtaining the minimum energy given the two machine models $\mathcal{M}_A$ and $\mathcal{M}_B$. In many systems, however, it is desirable to strike a trade-off between energy consumption and performance by minimizing the energy-delay product rather than the total energy. We study the problem of obtaining the minimum energy-delay product (or ED) in this section.

5.1 $\mathcal{M}_A$: Processors Cannot be Turned Off Individually

The ED for $\mathcal{M}_A$ is expressed as follows:

$$ED(t,y) = \big(tf_s^\alpha + N(y-t)f_p^\alpha + N\lambda y\big)y. \tag{37}$$

In order to minimize $ED(t,y)$, we obtain $\frac{\partial ED}{\partial t}$ and $\frac{\partial ED}{\partial y}$, and set them to zero and obtain

$$t^* = \left(\frac{\alpha-2}{2N\lambda}\right)^{1/\alpha}\cdot s, \tag{38}$$

$$y^* = \left(\frac{\alpha-2}{2N\lambda}\right)^{1/\alpha}\cdot\left(s+\frac{p}{N^{(\alpha-1)/\alpha}}\right), \tag{39}$$

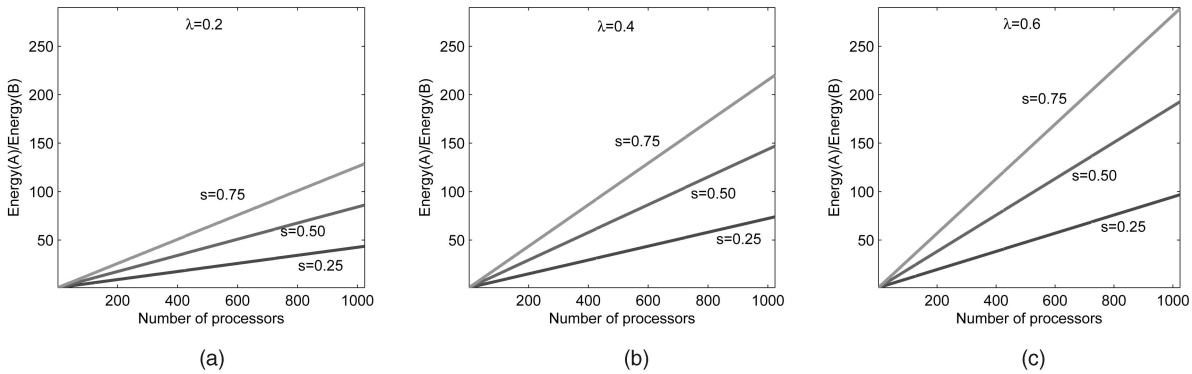

Fig. 12. Ratio of the total energy of $\mathcal{M}_A$ to that of $\mathcal{M}_B$ at the maximum program speedup Amdahl's law dictates. Plots for three different λ values are shown: 0.2, 0.4, and 0.6 from left.

from which we get

$$f_s^* = \left(\frac{2N\lambda}{\alpha-2}\right)^{1/\alpha}, \quad (40)$$

$$f_p^* = \left(\frac{2\lambda}{\alpha-2}\right)^{1/\alpha}. \quad (41)$$

It is shown that $f_p^* = f_s^*/N^{\frac{1}{\alpha}}$ as in (20). In fact, by comparing the values obtained in this section for t^*, y^*, f_s^*, and f_p^* with (17), (18), (19), and (20), we observe that they are the same equations with the replacement of $(\alpha-1)$ with $(\alpha-2)/2$. This similarity also appears in the calculation of energy. Specifically, we can compute the dynamic energy when ED is minimized to be

$$E_{dynamic} = \frac{2}{\alpha-2} \cdot N\lambda \cdot y^*, \quad (42)$$

which is equal to $2/(\alpha-2)$ of the static energy, $E_{static} = N\lambda \cdot y^*$.

In other words, while the total energy consumption is minimized when the dynamic energy consumption is $1/(\alpha-1)$ times the static energy consumption, the energy-delay product is minimized when the dynamic energy consumption is $2/(\alpha-2)$ times the static energy consumption.

The processor speeds given by (40) and (41) for the minimum energy-delay product are only valid when they are less than the maximum speed, $F_{max} = 1$. That is, when $\lambda < (\alpha-2)/2N$. When the static energy is high to the point of violating this condition, we should use $f_s^* = 1$, which gives $t = s$. Hence, as was done in Section 3.2, ED should be rewritten as a function of y and the optimum value for f_p should be obtained by solving $\frac{\partial ED}{\partial y} = 0$.

5.2 $\mathcal{M}_B$: Processors Can be Turned off Individually

It was demonstrated in Section 4 that, when it is possible to turn off individual processors, the minimum total energy consumption occurs when the processor speeds in the parallel and the serial execution sections are equal and are given by $(\lambda/(\alpha-1))^{1/\alpha}$. Given that this result is independent of the number of processors that are used, then the goal of minimizing the energy-delay product can be achieved when as many processors as possible are used to execute the parallel section to minimize completion time.

In fact, if we write the energy-delay product for $\mathcal{M}_B$ as

$$ED(t,y,n) = \big(tf_s^\alpha + n(y-t)f_p^\alpha + t\lambda + n(y-t)\lambda\big)y, \quad (43)$$

and differentiate ED with respect to t, n, and y, and set the result to zero, the first two equations that we obtain are identical to (27) and (29) while the third equation that we obtain from $\frac{\partial ED}{\partial n} = 0$ is different from (28). Recall that $\frac{\partial E}{\partial n} = 0$ and $\frac{\partial E}{\partial y} = 0$ resulted in two dependent equations, namely (29) and (28), which led to the result that the minimization of energy in model $\mathcal{M}_B$ is independent of n. Thus, when minimizing ED, the equations resulting from $\frac{\partial ED}{\partial t} = 0$ and $\frac{\partial ED}{\partial n} = 0$ give us the same solution for t^* and y^* as in the case of minimizing the energy. The equation resulting from $\frac{\partial ED}{\partial y} = 0$, however, can be shown to be only satisfied when $n = \infty$ (or alternatively, $y^* - t^* = 0$). That is, the energy-delay product is minimized when $n = \infty$.

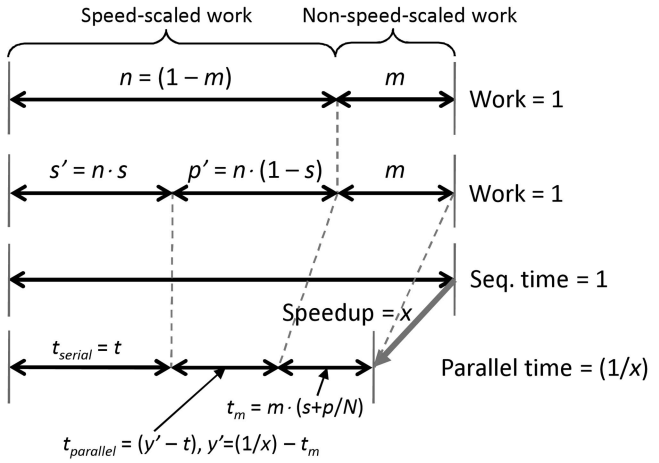

Fig. 13. "Work" is partitioned into speed-scaled work ("n") and non-speed-scaled work ("m"). Given N processors, t_m, the parallel time for m is $m \cdot (s + p/N)$ regardless of processor speeds.

In many systems, the number of processors is fixed to some value N. This is particularly true in CMPs with a fixed number of cores. In these cases, the energy-delay product is minimized when $n = N$. To obtain the values of t and y (and, hence, f_s and f_p that minimize $ED(t,y)$), (43) should be written with $n = N$ and differentiated with respect to t and y. Unfortunately, the equations resulting from $\frac{\partial ED}{\partial t} = 0$ and $\frac{\partial ED}{\partial y} = 0$ in this case cannot be solved analytically. Numeric techniques are needed to obtain a solution.

6 Effect of Constant-Speed Operations

In our problem formulation in Section 2 and derivations in the previous three sections, we assumed that processor speed (processor's clock frequency) solely determines the runtime of a program region. In this section, we will discuss how constant-speed operations, such as memory access and I/O processing, affect program execution time and, hence, our derivations.

Taking memory access as an example of constant-speed operations, a simple single-core performance model that relates performance to the processor frequency is [7], [28]

$$T(f) = W_{core} \cdot \frac{1}{f} + T_{mem}, \quad (44)$$

where W_{core} is the work for the core and T_{mem} is the time for memory, unaffected by the processor frequency f. In (44), it is clearly shown that the program execution time is split into a frequency-dependent part (first term) and a frequency-independent part (second term).

The same intuition about the constant-speed operations in (44) can be applied to our parallel workload formulation. Fig. 13 presents a modified problem formulation after incorporating the constant-speed operations. Let us assume that the constant-speed operations (e.g., memory access) are distributed uniformly across the work and their aggregated amount is m ($m < 1$). With this assumption, the total work is split into two parts, *speed-scaled work* and *non-speed-scaled work*, whose amount is n (same as $1-m$) and m, respectively. If we assume that memory accesses from

multiple processor cores can be overlapped, then the total amount of time spent on the constant-speed operations (t_m) is: $m \cdot (s + p/N)$. As was the case with a single processor core example in (44), given N, the time spent on constant-speed memory operations with N processors does not change with the processor speed.

Because the effect of the constant-speed operations is a fixed offset in the parallel execution time, our derivations and discussions in Sections 3, 4, and 5 can still apply to this new problem formulation. A simple way to directly apply our derivations in this new formulation is to "translate" the variables used in the two problem formulations. For instance, as shown in Fig. 13, s', the serial section in the speed-scaled work, is a scaled down value of s (Fig. 2) with the factor of $n, n < 1$. Similarly, $p' = n \cdot p$. The program execution time is the sum of the time for the speed-scaled section and the constant value t_m, given N. Hence, a target program speedup x can be translated into x' for the speed-scaled work, $x' = (x^{-1} - t_m)^{-1}$. If one wishes to vary N, N can become a variable as well, as in Section 4.

In summary, introducing constant-speed operations in the workload model does not fundamentally change our problem formulation and the framework for deriving solutions. Given that translating a problem formulated using Fig. 13 into a problem based on our original problem formulation (Fig. 2) is straightforward, we do not pursue in this paper the task of deriving new formulas that incorporate the impact of constant-speed operations.

7 Related Work

Amdahl's law [3], being of a very simple form, has inspired much work in the domain of computer architecture and parallel processing [1], [12]. Using Amdahl's law, recently, Hill and Marty [13] looked into the trade-off between processor core types and sizes in a multicore processor, parallelism in applications, and processor performance (i.e., large, high-performance, power-hungry cores versus small, low-performance, low-power cores). They demonstrate the performance advantage of a heterogeneous architecture featuring large, high-performance processor cores, and many small cores. They did not, however, consider power or energy consumption. Woo and Lee [34] extended [13] to consider energy efficiency of processor architectures using different core types and confirmed that a heterogeneous architecture with a full-blown core along with many small, power-efficient cores is a viable alternative to homogeneous many-core architectures. They assume that processor cores have nominal power consumption levels and do not consider scaling voltage and speed of the processor cores. In our previous work [6], we considered the problem of optimizing energy when running a parallel application and presented a problem formulation that became a foundation of this work. However, we [6] did not study a machine model where individual processor cores can be turned off, or the energy-delay metric.

Energy saving techniques that utilize available timing slack with DVFS have been extensively studied, especially in the domain of real-time task scheduling [18], [25], [29], [35], [36]. While much previous work has been on improving energy on uniprocessor systems, Zhu et al. [36] introduces the concept of *slack sharing* on multiprocessor systems and Mishra et al. [25] utilizes the static slack based on the degree of parallelism in a schedule. Compared with previous heuristic-oriented energy-aware task scheduling strategies, our work in this paper focused on understanding the interaction between parallelization, performance, and energy consumption by constructing an analytical model to directly derive the minimum energy given a parallelized application.

More recently, Ge et al. [11] studied distributed performance-directed DVFS strategies for use in HPC clusters, which exploit fine-grained timing slacks to save energy. Ge and Cameron [10] studied *power-aware speedup*, defined as the ratio between the single-processor performance at the lowest available on-chip frequency and the parallel execution time. They partition a given workload into a portion of the workload accessing on-chip data and another portion accessing off-chip data. The main goal of their study is to derive a more accurate parallel speedup model for modern processors capable of DVFS.

Isci et al. [17] studied chip-wide power management strategies to meet the maximum power consumption allowed for the processor and showed that their best strategy, when backed up by a per-core DVFS mechanism, is able to achieve the maximum power consumption budget without degrading the overall chip throughput significantly. While their goal was to meet the specified power budget in a multicore processor and used multiprogrammed workloads for their study, they also confirmed the observation, like ours, that the role of DVFS in multicore processors is of growing significance.

Finally, Li and Martínez [23] presented an analytical model to derive power consumption and performance when nominal parallel efficiency and the number of processors in a chip multiprocessor are given. However, they do not consider serial and parallel regions in a parallel application separately and abstract the performance of parallel execution with "parallel efficiency." In their later work [24], they proposed a dynamic power performance adaptation technique using heuristics-based DVFS and sleep mode control. Unlike our work which focuses on the analytical approach and latency-oriented parallel applications, they targeted applications requiring a steady-state rate (e.g., multimedia application), and studied a search-based solution to the problem of power minimization.

8 Conclusions

In this paper, we developed an analytical framework to study the trade-offs between parallelization, program performance, and energy consumption. Although this framework is based on many simplifying assumptions, some of which are inherited from Amdahl's law and some of which are specific to variable-speed processors, it provides interesting insights on these trade-offs. The main simplification inherited from Amdahl's law is that the parallel section of an application is fully parallelizable. The simplifications assumed about the processors include:

1. ignoring the overhead of changing speeds and turning off processors;

2. the number of cycles (i.e., program execution time) is proportional to the processor speed;
3. the ratio of static to dynamic power consumption λ is independent of the processor temperature; and
4. the dynamic power consumption at frequency f is proportional to f^{α}, where f can continuously vary between 0 and a maximum speed.

The framework is kept intentionally simple to obtain general results that provide broad insights on the effect of the different parameters of the problem.

We considered two machine models; one assumes that individual processors cannot be turned off independently, and the other assumes that they can. In the first machine model, our analysis shows that total energy is minimized when the static energy is equal to $(\alpha - 1)$ times the dynamic energy while the energy-delay product is minimized when the static energy is $(\alpha - 2)/2$ times the dynamic energy. Both the minimum energy and the minimum energy-delay are obtained when the speed of the serial section f_s is $N^{(1/\alpha)}$, the speed in the parallel section, f_p. The values of f_s that minimize the energy and the energy-delay product are independent of s (the program characteristics) and are given by $(N\lambda/(\alpha - 1))^{1/\alpha}$ and $(2N\lambda/(\alpha - 2))^{1/\alpha}$, respectively. For an application that requires a specific quality of service (i.e., a specific speedup), the analytical framework also provides a simple, yet powerful, way of determining the processors speeds that minimize the energy or the energy-delay product. It also provides for a simple way to determine the effect of the static/dynamic power ratio on the aforementioned trade-offs.

When processors can be individually turned off, the analysis indicates that the minimum total energy is independent of the number of processors used for executing the parallel section, while the energy-delay product is minimized when the maximum number of available processors are used during the parallel execution section. For the energy optimization, it is shown that the minimum energy is obtained when the speeds during the serial and the parallel sections are equal, which again results in the static energy being equal to $(\alpha - 1)$ times the dynamic energy.

The demonstrated substantial power advantage that can be gained from turning off individual processors is a great incentive to designing multicore processors with the capability of turning off individual processors. The framework, in its current form, however, does not show the advantage of scaling the speeds of cores individually. This is mainly due to the uniform parallelism assumed in the Amdahl's execution model. A more complex execution model is needed to demonstrate the advantage of individual processor speed scaling.

Acknowledgments

This work was supported in part by a grant from Intel and by US National Science Foundation (NSF) grants CCF-0702236, CNS-0524634, and ITR-0325353.

References

[1] G.S. Almasi and A. Gottlieb, *Highly Parallel Computing*, second ed. Benjamin/Cummings Publishing Company, 1994.
[2] AMD Dual-Core Processors, http://www.amd.com, 2009.
[3] G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *Proc. Am. Federation of Information Processing Soc. (AFIPS) Conf.*, pp. 483-485, 1967.
[4] K. Asanović, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, and K.A. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley," Technical Report UCB/EECS-2006-183, Univ. of California, Berkeley, Dec. 2006.
[5] S. Borkar, "Microarchitecture and Design Challenges for Gigascale Integration," *Proc. Int'l Symp. Microarchitecture (MICRO)*, Dec. 2004.
[6] S. Cho and R.G. Melhem, "Corollaries to Amdahl's Law for Energy," *IEEE Computer Architecture Letters (CAL)*, vol. 7, no. 1, pp. 25-28, Jan. 2008.
[7] K. Choi, R. Soma, and M. Pedram, "Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Trade Off Based on the Ratio of Off-Chip Access to On-Chip Computation Times," *Proc. Design Automation and Test in Europe Conf. (DATE)*, Feb. 2004.
[8] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar, "An Integrated Quad-Core Opteron Processor," *Proc. Int'l Solid-State Circuits Conf. (ISSCC)*, pp. 102-103, Feb. 2007.
[9] J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, E. Chan, Y. Chan, D. Plass, S. Chu, H. Le, L. Clark, J. Ripley, S. Taylor, J. Dilullo, and M. Lanzerotti, "Design of the Power6 Microprocessor," *Proc. Int'l Solid-State Circuits Conf. (ISSCC)*, pp. 96-97, Feb. 2007.
[10] R. Ge and K.W. Cameron, "Power-Aware Speedup," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS)*, pp. 1-10, Mar. 2007.
[11] R. Ge, X. Feng, and K.W. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters," *Proc. Conf. Supercomputing*, pp. 34-44, Nov. 2005.
[12] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, fourth ed. Morgan Kaufmann Publishers, 2007.
[13] M.D. Hill and M.R. Marty, "Amdahl's Law in the Multiore Era," *Computer*, vol. 41, no. 7, pp. 33-38, July 2008.
[14] C.-H. Hsu and W.-C. Feng, "Effective Dynamic Voltage Scaling through CPU-Boundedness Detection," *Proc. Workshop Power-Aware Computing Systems (PACS)*, Dec. 2004.
[15] Intel, "A New Era of Architectural Innovation Arrives with Intel Dual-Core Processors," *Technology@Intel Magazine*, pp. 1-11, May 2005.
[16] Intel, Intel XScale Microarchitecture, technical summary, 2000.
[17] 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," *Proc. Int'l Symp. Microarchitecture (MICRO)*, pp. 347-358, Dec. 2006.
[18] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," *Proc. Int'l Symp. Low-Power Electronics and Design (ISLPED)*, pp. 197-202, Aug. 1998.
[19] ITRS (International Technology Roadmap for Semiconductors), 2005 ed., http://public.itrs.net, 2005.
[20] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie, "Comparison of Split- Versus Connected Core Supplies in the POWER6 Microprocessor," *Proc. Int'l Solid-State Circuits Conf. (ISSCC)*, pp. 298-299, 604, Feb. 2007.
[21] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *IEEE Micro*, vol. 25, no. 2, pp. 21-29, Mar./Apr. 2005.
[22] M. LaPedus, "Intel Tips Teraflops Programmable Processor," *EETimes*, Sept. 26, 2006.
[23] J. Li and J.F. Martínez, "Power-Performance Considerations of Parallel Computing on Chip Multiprocessors," *ACM Trans. Architecture and Code Optimization (TACO)*, vol. 2, no. 4, pp. 397-422, Dec. 2005.
[24] J. Li and J.F. Martínez, "Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors," *Proc. Int'l Symp. High Performance Computer Architecture (HPCA)*, pp. 77-87, Feb. 2006.
[25] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem, "Energy Aware Scheduling for Distributed Real-Time Systems," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS)*, pp. 21-29, Apr. 2003.

[26] R. Ronen, A. Mendelson, K. Lai, S.-L. Lu, F. Pollack, and J.P. Shen, "Coming Challenges in Microarchitecture and Architecture," *Proc. IEEE,* vol. 89, no. 3, pp. 325-340, Mar. 2001.
[27] N. Sakran, M. Yuffe, M. Mehalel, J. Doweck, E. Knoll, and A. Kovacs, "The Implementation of the 65 nm Dual-Core 64b Merom Processors," *Proc. Int'l Solid-State Circuits Conf. (ISSCC),* pp. 106-107, 590, Feb. 2007.
[28] K. Seth, A. Anantaraman, F. Mueller, and E. Rotenberg, "FAST: Frequency-Aware Static Timing Analysis," *Proc. Real-Time Systems Symp. (RTSS),* Dec. 2003.
[29] D. Shin, J. Kim, and S. Lee, "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications," *IEEE Design and Test of Computers,* vol. 18, no. 2, pp. 20-30, Mar./Apr. 2001.
[30] A. Sinha and A.P. Chandrakasan, "JouleTrack—A Web Based Tool for Software Energy Profiling," *Proc. Design Automation Conf. (DAC),* pp. 220-225, June 2001.
[31] B. Sinharoy, R.N. Kalla, J.M. Tendler, R.J. Eickemeyer, and J.B. Joyner, "POWER5 System Microarchitecture," *IBM J. Research & Development,* vol. 49, nos. 4/5, pp. 505-521, July-Sept. 2005.
[32] T. Takayanagi, J.L. Shin, B. Petrick, J.Y. Su, H. Levy, H. Pham, J. Son, N. Moon, D. Bistry, U. Nair, M. Singh, V. Mathur, and A.S. Leon, "A Dual-Core 64-bit UltraSPARC Microprocessor for Dense Server Applications," *IEEE J. Solid-State Circuits,* vol. 40, no. 1, pp. 7-18, Jan. 2005.
[33] R. Teodorescu and J. Torrellas, "Variation-Aware Application Scheduling and Power Management for CMPs," *Proc. Int'l Symp. Computer Architecture (ISCA),* pp. 363-374, June 2008.
[34] D.-H. Woo and H.-H.S. Lee, "Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era," *Computer,* vol. 41, no. 12, pp. 24-31, Dec. 2008.
[35] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," *Proc. Symp. Foundations of Computer Science (FOCS),* pp. 374-382, Oct. 1995.
[36] D. Zhu, R. Melhem, and B.R. Childers, "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems," *Proc. Real-Time Systems Symp. (RTSS),* pp. 84-94, Dec. 2001.

Sangyeun Cho (S'95-M'04) received the BS degree in computer engineering from Seoul National University in 1994 and the PhD degree in computer science from the University of Minnesota in 2002. In 1999, he joined the System LSI Division of Samsung Electronics Co., Giheung, Korea, and contributed to the development of Samsung's flagship embedded processor core family CalmRISC(TM). He was a lead architect of CalmRISC-32, a 32-bit microprocessor core, and designed its memory hierarchy including caches, DMA, and stream buffers. Since 2004, he has been with the Computer Science Department at the University of Pittsburgh as an assistant professor. His research interests are in the area of computer architecture and embedded systems, with particular focus on performance, power, and reliability aspects of memory hierarchy design for next-generation multicore processors. He is a member of the IEEE.

Rami G. Melhem received the BE degree in electrical engineering from Cairo University in 1976, the MA degree in mathematics and the MS degree in computer science from the University of Pittsburgh in 1981, and the PhD degree in computer science from the University of Pittsburgh in 1983. He was an assistant professor at Purdue University prior to joining the faculty of the University of Pittsburgh in 1986, where he is currently a professor of computer science and electrical engineering. His research interests include power management, real-time and fault-tolerant systems, optical networks, high-performance computing, and parallel computer architectures. He served on the program committees of numerous conferences and workshops. He was on the editorial board of the *IEEE Transactions on Computers* and the *IEEE Transactions on Parallel and Distributed Systems*. He is serving on the advisory boards of the IEEE technical committees on computer architecture. He is the editor for the *Springer Book Series in Computer Science* and is on the editorial board of the *Computer Architecture Letters*, the *International Journal of Embedded Systems*, and the *Journal of Parallel and Distributed Computing*. He is a fellow of the IEEE and the IEEE Computer Society, and a member of the ACM.