Energy-Performance Trade-off Analysis of Parallel Algorithms

Vijay Anand Korthikanti and Gul Agha
University of Illinois Urbana Champaign
{vkortho2, agha}@illinois.edu

Abstract

Energy consumption by computer systems has emerged as an important concern, both at the level of individual devices (limited battery capacity in mobile systems) and at the societal level (the production of Green House Gases). In multicore architectures, applications may be executed on a variable number of cores and these cores may operate at different frequencies. The performance and energy cost of a parallel algorithm executing on a multicore architecture have different trade-offs, depending on how many cores the algorithm uses, at what frequencies these cores operate, and the structure of the algorithm. We show how algorithm designers and software developers can analyze the energy-performance trade-off in parallel algorithms. We believe that such analyses should be applied to parallel algorithms to facilitate energy conservation.

1 Introduction

Our work is motivated by two observations. First and foremost, computers are consuming an increasingly significant amount of energy: in the US alone, estimates already put the figure at 13% of the total electricity usage [25]. Thus computer use represents a significant source of Green House Gasses, and creates a critical problem for sustainability. Second, limited energy storage capacity is a critical problem for mobile devices.

As CPU's hit the power wall, multicore architectures have been proposed as a way to increase computation cycles while keeping power consumption constant. It turns out that in multicore architectures, it is possible to scale the speed of the individual cores or leave them idle, thus reducing their energy consumption. Because the relation between power and frequency of a core is nonlinear, on a sequential processor, energy consumption can be reduced by lowering the frequency at which the processor runs. However, lowering the frequency in a uniprocessor leads to an increase in time taken by an algorithm (i.e., decrease in performance). The picture is more complex in the case of parallel processors. Parallel computing involves some serial subcomputations, some parallel computation, and interaction between the parallel subcomputations. Thus parallel performance and energy cost are dependent not only on the number of cores (and the frequency at which they operate), they are also dependent on the structure of the parallel algorithm.

By increasing the number of cores, the computation that is required at each core may be reduced, which may in turn improve performance. Specifically, for parallel algorithms in which there is no interaction between the computations at each core, doubling cores halves the computation per core. If the frequency of each of two slower cores is ≈ 0.8 of the frequency of a faster core, the two cores consume about the same amount of energy as the faster core, while their overall performance is about 60% higher. However, parallel algorithms in general involve interaction between subcomputations. Thus, as the number of cores increases, the need for interaction between cores also increases. This in turn means that more energy is required for interaction. In particular, if we fix an energy budget, using more cores eventually leads to a decrease in the amount of energy left for computation at each core; in turn, this means that the cores have to run at lower speeds, thereby decreasing performance.

Note that measuring the energy consumed and the performance of a parallel algorithm lead to different results. For one, power consumption (i.e., the rate of energy consumption) by a core is (typically) proportional to the cube of its frequency. For another, the energy and performance characteristics of communication (and of shared memory accesses) and computation differ between different parallel algorithms. For example, in some algorithms, communication time may be masked by overlapping communication and computation (e.g., see [2]). However, the energy required for communication may be unaffected, whether or not such overlapping can be done. Thus the degree of parallelism used for performance maximization will generally different from that required for energy conservation.

We believe that examining the relation between the performance of parallel algorithms and their *energy requirements* on *multicore processors* may be facilitated by analyzing some scalability metrics. Such metrics can provide programmers with intuitions about the energy required by different parallel algorithms, thus *guiding the*

choice of algorithm, architecture, the number of cores to use and the frequency at which to operate them. Moreover, such analyses would help in the *design of more energy efficient algorithms*.

We have studied how to analyze the energy consumption of algorithms in two commonly used models of parallel computation–namely, the shared memory model [20] and the message-passing model [18]. Note that our approach addresses the problem in the spirit of scalability analysis of parallel algorithms–as distinct from practical performance analysis on specific architectures. The problem of defining metrics to quantify energy performance trade-offs was also posed as an important open problem in a recent NSF workshop [8, 16]. In this position paper, we argue for a research agenda emphasizing energy analysis of algorithms and provide a high-level view of our approach as a starting point.

2 Scalability Metrics

Traditionally, *scalability* of a parallel algorithm measures the relation between performance and the number of cores used [21]. We believe this notion can be applied to energy costs. We propose that three energy related scalability metrics should be considered, each for a different purpose.

Let a problem instance be the execution of a given algorithm for a given input size. The performance of a problem instance is the inverse of the time required for the completion of the problem instance. We define the following scalability metrics:

Energy scalability under Iso-performance: For a given problem instance and a fixed performance requirement, *energy scalability under iso-performance* provides the optimal number of cores required to minimize the energy consumption [18, 20]. This metric may be used to conserve energy in real time applications with a fixed performance target.

Performance Scalability under Iso-energy or Energy-bounded Scalability: For a given problem instance and a fixed energy budget, *energy-bounded scalability* provides the optimal number of cores required to maximize performance [19]. Energy-bounded scalability may be used in mobile devices which have strong energy constraints.

Energy Efficient Scalability: For a given problem instance, *energy efficient scalability* provides the optimal number of cores required and the frequency at which these cores should operate in order to maximize energy efficiency (the performance/energy ratio). This metric may be used to provide energy efficient computing. Note that this metric can be generalized by associating an arbitrary utility function with both performance and energy costs, and analyzing scalability with respect to the utility values. Energy utility is useful metric where, for example, the value of a result varies nonlinearly with response time.

3 Model and Assumptions

Estimating energy cost requires a computational model and an energy model. Two types of computation models (and their corresponding architectures) have been proposed: shared memory [17] and message passing [11]. A model of energy cost can be associated with any parallel computational model. By considering a model of each type, we illustrate how this can be done.

3.1 Parallel Computation Model

Consider the current generation of shared memory multicore architectures [3]. Such multicore architectures use a hierarchical shared memory. Although the Parallel Random Access Machine (PRAM) models shared memory architectures [17], the PRAM model contains no notion of the memory hierarchy. More recently, several models emphasizing memory hierarchies have been proposed [7, 5, 4]. In particular, the *Parallel External Memory* (PEM) model is an extension of the PRAM model which includes a single level of memory hierarchy [4]. A more general model is the so-called *Multicore* model [7]; in this model, multiple levels of the memory hierarchy are represented. For simplicity, we consider the PEM model; adding energy cost to this model is sufficient to illustrate performance-energy trade-offs.

The PEM model [4] assumes there are P cores and a two-level memory hierarchy consisting of an external memory (main memory) shared by all the cores, and P internal memories (caches) local to each core. All caches are of a fixed size M, partitioned in blocks of size B, and owned by a single core (only the owner can access a cache). To perform an operation on data, a core must have the data in its own cache. Data is transferred between the main memory and the cache in blocks of size B. Multiple cores may access distinct blocks of the shared memory concurrently. There are three variants of the PEM model (as in the case of the PRAM model); these variants differ in whether or not the same block of shared memory may be concurrently read from, and if so, if it can also be concurrently written to.

Along similar lines, for message passing multicore architectures [22], we may extend the EM model [1] to message-passing architectures. In the EM model, each core has a local two level memory hierarchy (a cache, and a slower Random Access Memory). We define *Message-Passing EM* (MEM) model as one in which the cores communicate through an interconnection network. We

may consider the case where message sends and receives take a constant amount of time and energy; this is not unreasonable as in many architectures, the time consumed for sending and receiving a message is high relative to the fraction spent en-route. Other assumptions (e.g., communication time and energy cost varies by distance) may also be reasonable to consider.

Note that the computation time T_{busy} on a given core will be proportional to the number of cycles including cache accesses μ executed on the core. Let X be the frequency of a core, then:

$$T_{busy} = \textit{(number of cycles)} \times \frac{1}{X} \tag{1}$$

We denote the time for which a given core is active (not idle) as T_{active}.

3.2 Energy Model

The following equation approximates the power consumption in a CMOS circuit:

$$P = C_L V^2 f + I_L V \tag{2}$$

where C_L is the load capacitance, V is the supply voltage, I_L is the leakage current, and f is the operational frequency. The first term corresponds to the dynamic power-consumption component of the total power consumption, while the second term corresponds to the leakage power consumption.

Recall that a linear increase in the voltage supply leads to a linear increase in the frequency of the core. However, a linear increase in the voltage supply also leads to a non-linear (typically cubic) increase in power consumption. Thus, for simplicity, we model the dynamic and leakage energies consumed by a core, E, to be the result of the above mentioned critical factors:

$$E_{dynamic} = E_d \cdot T_{busy} \cdot X^3 \tag{3}$$

$$E_{leakage} = E_l \cdot T_{active} \cdot X \tag{4}$$

where E_d and E_l are some hardware constants [9].

We may assume that both memory accesses (both reads and writes) and message transfers consume some constant amounts of energy. Because recent processors have introduced efficient support for low power modes that can reduce the power consumption to near zero, it is reasonable to assume that the energy consumed by idle cores is zero.

4 Approach

Given an computation model which specifies performance and energy consumption, we can compute the various energy scalability metrics discussed in section 2. This involves analyzing an algorithm to deduce an equation (called the PE equation) which relates the algorithm's performance, energy costs, number and frequency of cores, to the input size. Once the equation is determined, it can be analyzed to compute various optima.

We have analyzed some algorithms using this approach. Due to lack of space, we briefly illustrate how one may evaluate energy scalability metrics. Specifically, we analyze energy scalability under iso-performance for parallel addition on the PEM model with four simplifying assumptions. First, the frequency of the cores can be varied using a frequency (voltage) probe and cores switch to *idle* state if there is no computation left on them. Second, all active cores operate at the same frequency. Third, the computation and cache access time of the cores can be scaled by scaling the frequency of the cores. And finally memory access (both read and write) cannot be scaled and thus takes constant time. The last assumption implies that there is no contention on shared memory.

We now show how the PE equation for parallel addition can be formulated. First, compute the critical path of the parallel algorithm and partition it in to CPU cycles, memory accesses, synchronization breaks, and message transfers (in message passing model). Next, scale the CPU cycles (reduce the frequency) of the critical path in such a way that the performance of the parallel algorithm matches the given performance target. Now we can frame an expression for energy consumption assuming that the cores are running at the reduced frequency X. The energy expression has four components namely energy for computation E_{comp}, energy for memory accesses E_{mem}, energy for message transfers E_{mes} and leakage energy E_{leak}. They are defined as follows:

$$\begin{aligned} E_{comp} &= E_d \cdot \mu_{comp} \cdot X^2 \\ E_{mem} &= E_m \cdot \mu_{mem} \\ E_{mes} &= E_t \cdot \mu_{mes} \\ E_{leak} &= E_l \cdot T_{active} \cdot X \end{aligned}$$

where E_m is energy consumed for single memory access (both read and write), E_t is energy consumed for single message transfer between the cores, and μ_{comp}, μ_{mem}, μ_{mes} are the number of computations, number of memory accesses and number of message transfers respectively at all cores. Note that E_{comp} is lower if the cores run at a lower frequency, while E_{leak} may increase as the active cores take longer to finish. E_{mem} amd E_{mes} may increase as more cores are used since the computation is more distributed. Considering the structure of the given parallel algorithm, we evaluate number of computation cycles (μ_{comp}), number of memory accesses (μ_{mem}), number of message transfers (μ_{mes}) and total active time (T_{active}) at all the cores as a function of

the input size, number of cores and the initially obtained reduced frequency X. We then analyze the equation to obtain the number of cores required for minimum energy consumption as a function of input size. In particular, we compute the appropriate number of cores that are required to guarantee a required level of *performance*.

Example: Adding Numbers Initially, all N numbers are stored contiguously in the main memory and caches of all P cores are empty. Without loss of generality, we assume that the input size N is a multiple of the number of cores P. In the first phase of the algorithm, each core transfers (N/P) numbers from memory to their own caches and computes their sum. The transfer and summation of (N/P) numbers by each core happens in a series of steps. In each step, core transfers a block of numbers B from main memory to its cache and computes the sum of B numbers and the result obtained in the previous step. At the end of the first phase, each of P cores possesses a partial sum. With access to a distinct additional auxiliary block of main memory by each core, in CREW PEM model, the sum of P partial sums is efficiently computed in parallel in a tree fashion in $\log(P)$ steps (for simplicity, assuming P to be some power of 2). In the first step, half of the cores transfer their partial sums in parallel to their respective auxiliary blocks in main memory. The other half of the cores then read in parallel the elements that were stored in the auxiliary blocks of the first half, and sum it with their local partial sum. The same step is recursively performed until there is only one core left. At the end of the computation, one core will store the sum of all N numbers. Figure 1 depicts the execution of addition algorithm for the case $P = 4$.

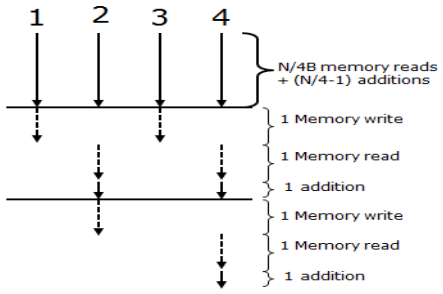

Figure 1: Example scenario: Adding N numbers using 4 cores; execution of 4th core represents the critical path; embarrassingly parallel application and represents a broad class of tree algorithms

Due to lack of space, we just provide the expression for the energy components we obtain using our methodology.

$$X' = F \cdot \frac{(\frac{N}{P} - 1 + log(P)) \cdot \beta}{T \cdot F - (\frac{N}{B \cdot P} + 2 \cdot \log(P)) \cdot M_c}$$

$$T_{active} = \frac{M_c}{F}(\frac{N}{B} + 3 \cdot (P-1)) + \frac{\beta}{X'} \cdot (N-1)$$

$$E_{comp} = E_d \cdot (N-1) \cdot \beta \cdot X'^2$$

$$E_{mem} = E_m \cdot ((N/B) + 2 \cdot (P-1))$$

$$E_{leak} = E_l \cdot T_{active} \cdot X'$$

where T is the performance target, F is maximum frequency of a single core, N is the input size for the parallel algorithm, P is number of cores being used by the parallel algorithm, M_c is number of cycles executed at maximum frequency for single memory access time, and β represents number of cycles required per addition.

We now analyze the energy expression obtained above for the addition algorithm to evaluate energy scalability under iso-performance. While we could differentiate the function with respect to the number of cores to compute the minimum, this results in a rather complex expression. Instead, we analyze the graphs expressing energy scalability under iso-performance.

Note that the energy expression is dependent on many variables. We can simplify a couple of these parameters without loss of generality. We set leakage energy constant as $E_l = 1$. We express all energy values with respect to this normalized energy value.

In order to graph the differential, we make some assumptions about the other parameters. While these assumptions compromise generality, we can consider the sensitivity of the analysis to a range of values for these parameters. One parameter is the the energy consumed for single cycle at maximum frequency as a multiple of leakage energy constant. We assume this ratio to be 10, i.e., that $E_d \cdot F^2 = 10 \cdot E_l$. It turns out that this parameter is not very significant for our analysis; in fact, large variations in the parameter do not significantly affect the shapes of the graphs for the parallel algorithms we have studied.

Another parameter, k, represents the ratio of the energy consumed for single memory accesses, E_m, and the energy consumed for executing a single instruction at the maximum frequency. Thus, $E_m = k \cdot E_d \cdot F^2$. We fix the required performance T to be that of the running time of the sequential algorithm at maximum frequency F and analyze the sensitivity of our results to a range of values of k. The sequential algorithm for this problem is trivial: it takes (N/B) memory accesses and $N-1$ additions to compute the sum of N numbers. The running time of the sequential algorithm is given by $T_{seq} = \beta \cdot (N-1) \cdot (1/F) + (N/B) \cdot (M_c/F)$.

Fig. 2 plots energy E as a function of input size and number of cores. We can see that for any input size N, initially energy decreases with increasing M and later on increases with increasing M. As explained earlier, this behavior can be understood by the fact that energy for computation decreases with an increase in number

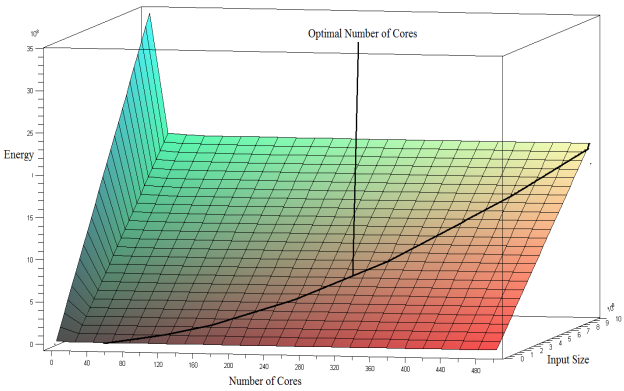

Figure 2: Addition: Energy curve with energy on Z axis, number of cores on X axis and input size on Y axis with $k = 1000$, $\beta = 2$, $M_c = 1000$. Black curve on the XY plane is the plot of optimal number of cores required for minimum energy consumption with varying input size(10^7 to 10^9).

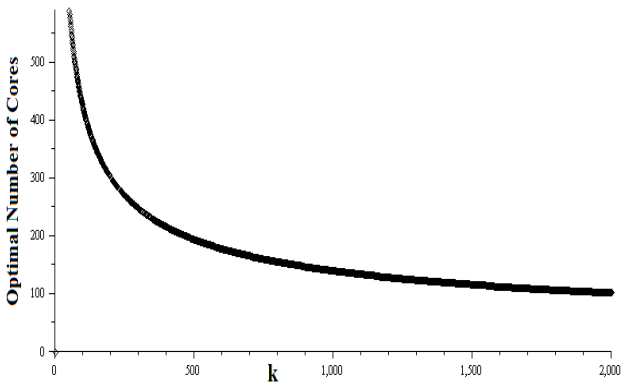

Figure 3: Sensitivity analysis: optimal number of cores on Y axis, and k (ratio of the energy consumed for single memory accesses and the energy consumed for executing a single instruction at the maximum frequency) on X axis with input size $N = 10^8$.

of cores running at reduced frequencies, and energy for memory accesses increases with increasing cores. Furthermore, we can see that increasing the input size leads to an increase in the optimal number of cores required for minimum energy consumption.

We now consider the sensitivity of this analysis with respect to the ratio k. Fig. 3 plots the optimal number of cores required for minimum energy consumption by varying k for an input size 10^8. The results show that for a given input size in the range considered, the optimal number of cores required for minimum energy consumption decreases with increasing k. (The curve approximates a negative exponential with a negative coefficient). We observe that this trend remains the same for whole of the input range (10^7 to 10^9).

The above graph analysis depicts the exact behavior of optimal number of cores as function of input size for the given input range and appears to generalize to larger input sizes. We do not provide an analytic expression for the asymptotic behavior of the optimal number of cores as a function of input size as the energy expression is rather complex.

5 Related Work

Cho and Melhem analyze the effect of parallelizing an algorithm on its energy consumption [10]. The idea is to assume that an algorithm has been divided into serial and parallel components and then evaluate the impact of using more cores to reduce energy consumption. Other researchers have not looked at energy consumption but considered the *rate* at which energy is expended (i.e., power is consumed). This line of work has provided tools to monitor power consumption, and controllers to reduce the number of cores used (*concurrency throttling*) or change the frequency at which these cores operate (*dynamic voltage frequency scaling*) [12, 23, 15, 27, 13]. Information from such monitors can also be used to profile algorithms in order to optimize energy consumption.

Bingham and Greenstreet proposed a generic energy complexity metric, ET^α, to represent the energy-time trade-off in CMOS technology [6]. Prior to this, other researchers had promoted the use of the ET [14] and ET^2 [24] metrics. These metrics try to abstract the voltage/frequency scaling issues from programmers, while enabling them to reason about the energy complexity of the computation. In contrast, we explicitly represent frequency in our model and view both concurrency throttling and voltage/frequency scaling as two orthogonal control knobs to control energy. More critically, none of these models account for the energy required for memory accesses or for message passing)–which can be a significant portion of the total energy consumed. A good discussion of the motivation for understanding energy-performance trade-off can be found in [16].

6 Research Agenda

We believe energy costs of algorithms should form an important research agenda for Computer Science. This includes analyzing algorithms for energy consumption as well as developing new algorithms which are more energy efficient. It could be argued that the computation models we have considered and the architectural assumptions we have made are too simple. Some of the ways in which this is true include the following. We assumed that the cores are homogeneous, but given the limitations im-

posed on parallel performance by Amdahl's law, cores are likely to be heterogeneous. We assumed a simple memory hierarchy, ignored memory contention in shared memory models, and considered communication costs to be uniform in message passing models. None of these assumptions is realistic and others may wish to consider one or the other generalization.

More interestingly, we believe that it would be useful to understand how different parallel algorithms can be categorized into scalability classes with respect to these metrics. For example, parallel PRIM algorithm for computing minimum spanning on message-passing architectures, energy-bounded scalability is $O(n)$, where n is the input size [19]. One way to achieve such a categorization is by doing an asymptotic analysis with respect to the input size assuming an arbitrarily large number of cores. We should point out that we are not convinced that such an asymptotic analysis is meaningful in the case of shared memory models, as such architectures are themselves not scalable [26]. However, asymptotic analyses of energy scalability metrics may provide insight into the behavior of parallel algorithms on message-passing models.

Acknowledgments

The authors would like thank Soumya Krishnamurthy (Intel) for motivating us to look at this problem. We would also like to thank George Goodman and Bob Kuhn (Intel), and MyungJoo Ham (Illinois) for helpful feedback and comments.

References

[1] AGGARWAL, A., AND VITTER, JEFFREY, S. The input/output complexity of sorting and related problems. *Commun. ACM 31*, 9 (1988), 1116–1127.

[2] AGHA, G., AND KIM, W. Parallel programming and complexity analysis using actors. *Massively Parallel Programming Models 0* (1997), 68.

[3] ALFS, G., AND KNUPFFER, N. Intel corporation's multicore architecture briefing. *Intel Press Room* (2008).

[4] ARGE, L., GOODRICH, M. T., NELSON, M. J., AND SITCHINAVA, N. Fundamental parallel algorithms for private-cache chip multiprocessors. In *SPAA* (2008), pp. 197–206.

[5] BENDER, M. A., FINEMAN, J. T., GILBERT, S., AND KUSZMAUL, B. C. Concurrent cache-oblivious b-trees. In *SPAA* (2005), pp. 228–237.

[6] BINGHAM, B. D., AND GREENSTREET, M. R. Computation with energy-time trade-offs: Models, algorithms and lower-bounds. In *ISPA* (2008), pp. 143–152.

[7] BLELLOCH, G. E., CHOWDHURY, R. A., GIBBONS, P. B., RAMACHANDRAN, V., CHEN, S., AND KOZUCH, M. Provably good multicore cache performance for divide-and-conquer algorithms. In *SODA* (2008), pp. 501–510.

[8] CAMERON, K. W., PRUHS, K., IRANI, S., RANGANATHAN, P., AND BROOKS, D. Report of the science of power management workshop. *Workshop on the Science of Power Management* (2009).

[9] CHANDRAKASAN, A., SHENG, S., AND BRODERSEN, R. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuits 27*, 4 (1992), 473–484.

[10] CHO, S., AND MELHEM, R. Corollaries to Amdahl's Law for Energy. *Computer Architecture Letters 7*, 1 (2008), 25–28.

[11] CULLER, D., KARP, R., PATTERSON, D., SAHAY, A., SCHAUSER, K., SANTOS, E., SUBRAMONIAN, R., AND VON EICKEN, T. LogP: Towards a Realistic Model of Parallel Computation. *ACM SIGPLAN Notices 28*, 7 (1993), 1–12.

[12] CURTIS-MAURY, M., SHAH, A., BLAGOJEVIC, F., NIKOLOPOULOS, D., DE SUPINSKI, B., AND SCHULZ, M. Prediction Models for Multi-dimensional Power-Performance Optimization on Many Cores. In *PACT* (2008), pp. 250–259.

[13] GE, R., FENG, X., AND CAMERON, K. Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters. In *SC* (2005), IEEE Computer Society Washington, DC, USA.

[14] GONZALEZ, R., AND HOROWITZ, M. A. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits 31* (1995), 12771284.

[15] ISCI, C., BUYUKTOSUNOGLU, A., CHER, C., BOSE, P., AND MARTONOSI, M. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *MICRO* (2006), vol. 9, pp. 347–358.

[16] KANT, K. Toward a science of power management. *Computer 42*, 9 (Sept. 2009), 99 –101.

[17] KARP, M., AND RAMACHANDRAN, V. Parallel algorithms for shared-memory machines. *Handbook of Theoretical Computer Science* (1990), 869–941.

[18] KORTHIKANTI, V. A., AND AGHA., G. Analysis of Parallel Algorithms for Energy Conservation in Scalable Multicore Architectures. In *International Conference on Parallel Processing (ICPP)* (2009).

[19] KORTHIKANTI, V. A., AND AGHA, G. Energy bounded scalability analysis of parallel algorithms. *Technical Report, Department of Computer Science, University of Illinois at Urbana Champaign* (2009).

[20] KORTHIKANTI, V. A., AND AGHA, G. Towards optimizing energy costs of algorithms for shared memory architectures. In *SPAA* (2010).

[21] KUMAR, V., GRAMA, A., GUPTA, A., AND KARYPIS, G. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin-Cummings, 1994.

[22] LANGER, M. Demonstrations of the 48-core research prototype. *Intel Press Room* (2009).

[23] LI, J., AND MARTINEZ, J. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In *HPCA* (2006), pp. 77–87.

[24] MARTIN, A. J. Towards an energy complexity of computation. *Information Processing Letters 39* (2001), 181187.

[25] MILLS, M. P. The internet begins with coal. *Green Earth Society, USA* (1999).

[26] MURPHY, R. On the effects of memory latency and bandwidth on supercomputer application performance. *IISWC* (Sept. 2007), 35–43.

[27] PARK, S., JIANG, W., ZHOU, Y., AND ADVE, S. Managing Energy-Performance Tradeoffs for Multithreaded Applications on Multiprocessor Architectures. In *SIGMETRICS* (2007), ACM New York, NY, USA, pp. 169–180.