J Supercomput (2012) 60:223–247
DOI 10.1007/s11227-010-0416-0

Energy efficient scheduling of parallel tasks on multiprocessor computers

Keqin Li

Published online: 12 March 2010

Abstract In this paper, scheduling parallel tasks on multiprocessor computers with dynamically variable voltage and speed are addressed as combinatorial optimization problems. Two problems are defined, namely, minimizing schedule length with energy consumption constraint and minimizing energy consumption with schedule length constraint. The first problem has applications in general multiprocessor and multicore processor computing systems where energy consumption is an important concern and in mobile computers where energy conservation is a main concern. The second problem has applications in real-time multiprocessing systems and environments where timing constraint is a major requirement. Our scheduling problems are defined such that the energy-delay product is optimized by fixing one factor and minimizing the other. It is noticed that power-aware scheduling of parallel tasks has rarely been discussed before. Our investigation in this paper makes some initial attempt to energy-efficient scheduling of parallel tasks on multiprocessor computers with dynamic voltage and speed. Our scheduling problems contain three nontrivial subproblems, namely, system partitioning, task scheduling, and power supplying. Each subproblem should be solved efficiently, so that heuristic algorithms with overall good performance can be developed. The above decomposition of our optimization problems into three subproblems makes design and analysis of heuristic algorithms tractable. A unique feature of our work is to compare the performance of our algorithms with optimal solutions analytically and validate our results experimentally, not to compare the performance of heuristic algorithms among themselves only experimentally. The harmonic system partitioning and processor allocation scheme is used, which divides a multiprocessor computer into clusters of equal sizes and schedules tasks of similar sizes together to increase processor utilization. A three-level energy/time/power allocation scheme is adopted for a given schedule, such that

K. Li (✉)
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

the schedule length is minimized by consuming given amount of energy or the energy consumed is minimized without missing a given deadline. The performance of our heuristic algorithms is analyzed, and accurate performance bounds are derived. Simulation data which validate our analytical results are also presented. It is found that our analytical results provide very accurate estimation of the expected normalized schedule length and the expected normalized energy consumption and that our heuristic algorithms are able to produce solutions very close to optimum.

Keywords Energy consumption · List scheduling · Parallel task · Performance analysis · Power-aware scheduling · Simulation · Task scheduling

1 Introduction

To achieve higher computing performance per processor, microprocessor manufacturers have doubled the power density at an exponential speed over decades, which will soon reach that of a nuclear reactor [31]. Such increased energy consumption causes severe economic, ecological, and technical problems. A large-scale multiprocessor computing system consumes millions of dollars of electricity and natural resources every year, equivalent to the amount of energy used by tens of thousands U.S. households [9]. A large data center such as Google can consume as much electricity as a city. Furthermore, the cooling bill for heat dissipation can be as high as 70% of the above cost [8]. A recent report reveals that the global information technology industry generates as much greenhouse gas as the world's airlines, about 2% of global carbon dioxide (CO_2) emissions.[1] Despite sophisticated cooling facilities constructed to ensure proper operation, the reliability of large-scale multiprocessor computing systems is measured in hours, and the main source of outage is hardware failure caused by excessive heat. It is conceivable that a supercomputing system with 10^5 processors would spend most of its time checkpointing and restarting [11].

There has been increasing interest and importance in developing high-performance and energy-efficient computing systems. There are two approaches to reducing power consumption in computing systems (see [4, 30, 31] for comprehensive surveys). The first approach is the method of thermal-aware hardware design. Low power consumption and high system reliability, availability, and usability are main concerns of modern high-performance computing system development. In addition to the traditional performance measure using FLOPS, the Green500 list uses FLOPS per Watt to rank the performance of computing systems, so that the awareness of other performance metrics such as energy efficiency and system reliability can be raised.[2] All the current systems which can achieve at least 400 MFLOPS/W are clusters of low-power processors, aiming to achieve high performance/power and performance/space. For instance, the IBM Roadrunner, currently the world's fastest computer, which achieves top performance of 1.456 PFLOPS, is also the fourth most energy efficient supercomputer in the world with an operational rate of 444.94 MFLOPS/W.[3] Intel's Tera-scale

[1] http://www.foxnews.com/story/0,2933,479127,00.html.

[2] http://www.green500.org/.

[3] http://en.wikipedia.org/wiki/IBM_Roadrunner.

research project has developed the world's first programmable processor that delivers supercomputer-like performance from a single 80-core chip which uses less electricity than most of today's home appliances and achieves over 16.29 GFLOPS/W.[4]

The second approach to reducing energy consumption in computing systems is the method of power-aware software design, by using a mechanism called *dynamic voltage scaling* (equivalently, dynamic frequency scaling, dynamic speed scaling, dynamic power scaling). Many modern components allow voltage regulation to be controlled through software, e.g., the BIOS or applications such as PowerStrip. It is usually possible to control the voltages supplied to the CPUs, main memories, local buses, and expansion cards.[5] Processor power consumption is proportional to frequency and the square of supply voltage. A power-aware algorithm can change supply voltage and frequency at appropriate times to optimize a combined consideration of performance and energy consumption. There are many existing technologies and commercial processors that support dynamic voltage (frequency, speed, power) scaling. SpeedStep is a series of dynamic frequency scaling technologies built into some Intel microprocessors that allow the clock speed of a processor to be dynamically changed by software.[6] LongHaul is a technology developed by VIA Technologies which supports dynamic frequency scaling and dynamic voltage scaling. By executing specialized operating system instructions, a processor driver can exercise fine control on the bus-to-core frequency ratio and core voltage according to how much load is put on the processor.[7] LongRun and LongRun2 are power management technologies introduced by Transmeta. LongRun2 has been licensed to Fujitsu, NEC, Sony, Toshiba, and NVIDIA.[8]

Dynamic power management at the operating system level refers to supply voltage and clock frequency adjustment schemes implemented while tasks are running. These energy conservation techniques explore the opportunities for tuning the energy-delay tradeoff [29]. Power-aware task scheduling on processors with variable voltages and speeds has been extensively studied since mid 1990s. In a pioneering paper [32], the authors first proposed the approach to energy saving by using fine grain control of CPU speed by an operating system scheduler. The main idea is to monitor CPU idle time and to reduce energy consumption by reducing clock speed and idle time to a minimum. In a subsequent work [34], the authors analyzed offline and online algorithms for scheduling tasks with arrival times and deadlines on a uniprocessor computer with minimum energy consumption. These research have been extended in [2, 6, 16, 19–21, 35] and inspired substantial further investigation, much of which focus on real-time applications, namely, adjusting the supply voltage and clock frequency to minimize CPU energy consumption while still meeting the deadlines for task execution. In [1, 12, 13, 15, 17, 22, 23, 25, 27, 28, 33, 37–39] and many other related works, the authors addressed the problem of scheduling independent or precedence

[4] http://techresearch.intel.com/articles/Tera-Scale/1449.htm.

[5] http://en.wikipedia.org/wiki/Dynamic_voltage_scaling.

[6] http://en.wikipedia.org/wiki/SpeedStep.

[7] http://en.wikipedia.org/wiki/LongHaul.

[8] http://en.wikipedia.org/wiki/LongRun.

constrained tasks on uniprocessor or multiprocessor computers where the actual execution time of a task may be less than the estimated worst-case execution time. The main issue is energy reduction by slack time reclamation.

There are two considerations in dealing with the energy-delay tradeoff. On the one hand, in high-performance computing systems, power-aware design techniques and algorithms attempt to maximize performance under certain energy consumption constraints. On the other hand, low-power and energy-efficient design techniques and algorithms aim to minimize energy consumption while still meeting certain performance goals. In [3], the author studied the problems of minimizing the expected execution time given a hard energy budget and minimizing the expected energy expenditure given a hard execution deadline for a single task with randomized execution requirement. In [5], the author considered scheduling jobs with equal requirements on multiprocessors. In [26], the authors investigated the problem of system value maximization subject to both time and energy constraints.

In [18], we addressed scheduling sequential tasks on multiprocessor computers with dynamically variable voltage and speed as combinatorial optimization problems. A sequential task is executed on one processor. We defined the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on multiprocessor computers. The first problem has applications in general multiprocessor and multi-core processor computing systems where energy consumption is an important concern and in mobile computers where energy conservation is a main concern. The second problem has applications in real-time multiprocessing systems and environments such as parallel signal processing, automated target recognition, and real-time MPEG encoding, where timing constraint is a major requirement. Our scheduling problems are defined such that the energy-delay product is optimized by fixing one factor and minimizing the other.

In this paper, we address scheduling parallel tasks on multiprocessor computers with dynamically variable voltage and speed as combinatorial optimization problems. A parallel task can be executed on multiple processors. We define the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint for parallel tasks on multiprocessor computers. We notice that power-aware scheduling of parallel tasks has rarely been discussed before; all previous studies were on scheduling sequential tasks which require one processor to execute. Our investigation in this paper makes some initial attempt to energy-efficient scheduling of parallel tasks on multiprocessor computers with dynamic voltage and speed.

Our scheduling problems contain three nontrivial subproblems, namely, system partitioning, task scheduling, and power supplying. Each subproblem should be solved efficiently, so that heuristic algorithms with overall good performance can be developed. These subproblems and our strategies to solve them are described as follows.

- *System Partitioning*—Since each parallel task requests for multiple processors, a multiprocessor computer should be partitioned into clusters of processors to be assigned to the tasks. We use the harmonic system partitioning and processor allocation scheme, which divides a multiprocessor computer into clusters of equal sizes and schedules tasks of similar sizes together to increase processor utilization.

- *Task Scheduling*—Parallel tasks are scheduled together with system partitioning, and it is NP-hard even scheduling sequential tasks without system partitioning. Our approach is to divide a list of tasks into sublists such that each sublist contains tasks of similar sizes which are scheduled on clusters of equal sizes. Scheduling such parallel tasks on clusters is no more difficult than scheduling sequential tasks and can be performed by list scheduling algorithms.
- *Power Supplying*—Tasks should be supplied with appropriate powers and execution speeds such that the schedule length is minimized by consuming given amount of energy or the energy consumed is minimized without missing a given deadline. We adopt a three-level energy/time/power allocation scheme for a given schedule, namely, optimal energy/time allocation among sublists of tasks (Theorems 7 and 8), optimal energy allocation among groups of tasks in the same sublist (Theorems 5 and 6), and optimal power supplies to tasks in the same group (Theorems 3 and 4).

The above decomposition of our optimization problems into three subproblems makes design and analysis of heuristic algorithms tractable. Our analytical results provide very accurate estimation of the expected normalized schedule length and the expected normalized energy consumption. A unique feature of our work is to compare the performance of our algorithms with optimal solutions analytically and validate our results experimentally, not to compare the performance of heuristic algorithms among themselves only experimentally. Such an approach is consistent with traditional scheduling theory. We find that our heuristic algorithms are able to produce solutions very close to optimum.

The rest of the paper is organized as follows. In Sect. 2, we present the power consumption model used in this paper. In Sect. 3, we introduce our scheduling problems, show the strong NP-hardness of our scheduling problems, derive lower bounds for the optimal solutions, and find an energy-delay tradeoff theorem. In Sects. 4 and 5, we describe the harmonic system partitioning and processor allocation scheme and list scheduling algorithms used to schedule sublists of tasks of similar sizes on clusters of equal sizes. In Sects. 6 and 7, we discuss optimal power supplies to tasks in the same group and optimal energy allocation among groups of tasks in the same sublist. In Sect. 8, we discuss optimal energy/time allocation among sublists of tasks, analyze the performance of our heuristic algorithms, and derive accurate performance bounds. In Sect. 9, we present simulation data which validate our analytical results. Finally, we conclude the paper in Sect. 10.

2 The power consumption model

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well-designed circuit is dynamic power consumption p (i.e., the switching component of power), which is approximately $p = aCV^2 f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [7]. Since $s \propto f$, where s is the processor speed, and $f \propto V^\gamma$ with

$0 < \gamma \leq 1$ [36], which implies that $V \propto f^{1/\gamma}$, we know that power consumption is $p \propto f^{\alpha}$ and $p \propto s^{\alpha}$, where $\alpha = 1 + 2/\gamma \geq 3$. It is clear from $f \propto V^{\gamma}$ and $s \propto V^{\gamma}$ that linear change in supply voltage results in up to linear change in clock frequency and processor speed. It is also clear from $p \propto V^{\gamma+2}$ and $p \propto f^{\alpha}$ and $p \propto s^{\alpha}$ that linear change in supply voltage results in at least quadratic change in power supply and that linear change in clock frequency and processor speed results in at least cubic change in power supply.

Assume that we are given n independent parallel tasks to be executed on m identical processors. Task i requires π_i processors to execute, where $1 \leq i \leq n$, and any π_i of the m processors can be allocated to task i. We call π_i the *size* of task i. It is possible that in executing task i, the π_i processors may have different execution requirements (i.e., the numbers of CPU cycles or the numbers of instructions executed on the processors). Let r_i represent the maximum execution requirement on the π_i processors executing task i. We use p_i to represent the power supplied to execute task i. For ease of discussion, we will assume that p_i is simply s_i^{α}, where $s_i = p_i^{1/\alpha}$ is the execution speed of task i. The execution time of task i is $t_i = r_i/s_i = r_i/p_i^{1/\alpha}$. Note that all the π_i processors allocated to task i have the same speed s_i for duration t_i, although some of the π_i processors may be idle for some time. The energy consumed to execute task i is $e_i = \pi_i p_i t_i = \pi_i r_i p_i^{1-1/\alpha} = \pi_i r_i s_i^{\alpha-1}$.

We would like to mention a number of important observations. First, since $s_i/p_i \propto s_i^{-(\alpha-1)}$ and $s_i/p_i \propto V^{-2}$, the processor energy performance, measured by speed per Watt,[9] is at least quadratically proportional to the voltage and speed reduction. Second, since $w_i/e_i \propto s_i^{-(\alpha-1)}$ and $w_i/e_i \propto V^{-2}$, where $w_i = \pi_i r_i$ is the amount of work to be performed for task i, the processor energy performance, measured by work per Joule [32], is at least quadratically proportional to the voltage and speed reduction. Third, the relation $e_i \propto p_i^{1-1/\alpha} \propto V^{(\gamma+2)(1-1/\alpha)} = V^2$ implies that linear change in supply voltage results in quadratic change in energy consumption. Fourth, the equation $e_i = w_i s_i^{\alpha-1}$ implies that linear change in processor speed results in at least quadratic change in energy consumption. Fifth, the equation $e_i = w_i p_i^{1-1/\alpha}$ implies that energy consumption reduces at a sublinear speed as power supply reduces. Finally, we observe that $e_i t_i^{\alpha-1} = \pi_i r_i^{\alpha}$ and $p_i t_i^{\alpha} = r_i^{\alpha}$, namely, for a given parallel task, there exist energy-delay and power-delay tradeoffs. Later, we will extend such tradeoff to a set of parallel tasks, i.e., the energy-delay tradeoff theorem.

3 Lower bounds and energy-delay tradeoff

Given n independent parallel tasks with task sizes $\pi_1, \pi_2, \ldots, \pi_n$ and task execution requirements $r_1, r_2, \ldots, r_n$, the problem of *minimizing schedule length with energy consumption constraint* E on a multiprocessor computer with m processors is to find the power supplies $p_1, p_2, \ldots, p_n$ and a nonpreemptive schedule of the n parallel tasks on the m processors such that the schedule length is minimized and the total energy consumed does not exceed E.

[9] See footnote 2.

Given n independent parallel tasks with task sizes $\pi_1, \pi_2, \ldots, \pi_n$ and task execution requirements $r_1, r_2, \ldots, r_n$, the problem of *minimizing energy consumption with schedule length constraint* T on a multiprocessor computer with m processors is to find the power supplies $p_1, p_2, \ldots, p_n$ and a nonpreemptive schedule of the n parallel tasks on the m processors such that the total energy consumption is minimized and the schedule length does not exceed T.

When all the π_i's are identical, the above scheduling problems are equivalent to scheduling sequential tasks discussed in [18]. Since both scheduling problems are NP-hard in the strong sense for all rational $\alpha > 1$ in scheduling sequential tasks, our problems for scheduling parallel tasks are also NP-hard in the strong sense for all rational $\alpha > 1$. Hence, we will develop fast polynomial-time heuristic algorithms to solve these problems.

We will compare the performance of our algorithms with optimal solutions analytically. Since it is infeasible to compute optimal solutions in reasonable amount of time, we derive lower bounds for the optimal solutions in Theorems 1 and 2. These lower bounds can be used to evaluate the performance of heuristic algorithms when they are compared with optimal solutions.

Let $W = w_1 + w_2 + \cdots + w_n = \pi_1 r_1 + \pi_2 r_2 + \cdots + \pi_n r_n$ denote the total amount of work to be performed for the n tasks. The following theorem gives a lower bound for the optimal schedule length T^* for the problem of minimizing schedule length with energy consumption constraint.

Theorem 1 *For the problem of minimizing schedule length with energy consumption constraint in scheduling parallel tasks, we have the following lower bound*:

$$T^* \geq \left(\frac{m}{E} \left(\frac{W}{m} \right)^{\alpha} \right)^{1/(\alpha-1)}$$

for the optimal schedule length.

Proof Imagine that each parallel task i is broken into π_i sequential tasks, each having execution requirement r_i. It is clear that any schedule of the n parallel tasks is also a legitimate schedule of the $n' = \pi_1 + \pi_2 + \cdots + \pi_n$ sequential tasks. However, it is more flexible to schedule the n' sequential tasks, since the π_i sequential tasks obtained from parallel task i do not need to be scheduled simultaneously. Hence, the optimal schedule length of the n' sequential tasks is no longer than the optimal schedule length of the n parallel tasks. It has been proven in [18] that the optimal schedule length of sequential tasks is at least

$$\left(\frac{m}{E} \left(\frac{R'}{m} \right)^{\alpha} \right)^{1/(\alpha-1)},$$

where R' is the total execution requirement of the sequential tasks. It is clear that $R' = \pi_1 r_1 + \pi_2 r_2 + \cdots + \pi_n r_n = W$. □

The following theorem gives a lower bound for the minimum energy consumption E^* for the problem of minimizing energy consumption with schedule length constraint.

Theorem 2 *For the problem of minimizing energy consumption with schedule length constraint in scheduling parallel tasks, we have the following lower bound*:

$$E^* \geq m\left(\frac{W}{m}\right)^{\alpha} \frac{1}{T^{\alpha-1}}$$

for the minimum energy consumption.

Proof Using an argument similar to that in the proof of Theorem 1, we break each parallel task i into π_i sequential tasks, each having execution requirement r_i. The minimum energy consumption of the n' sequential tasks is no more than the minimum energy consumption of the n parallel tasks. It has been proven in [18] that the minimum energy consumption of sequential tasks is at least

$$m\left(\frac{R'}{m}\right)^{\alpha} \frac{1}{T^{\alpha-1}}.$$

This proves the theorem. □

The lower bounds in Theorems 1 and 2 essentially state the following important theorem.

$\mathrm{ET}^{\alpha-1}$ Lower Bound Theorem (Energy-Delay Tradeoff Theorem) *For any execution of a set of parallel tasks with total amount of work W on m processors with schedule length T and energy consumption E, we must have the following tradeoff*:

$$ET^{\alpha-1} \geq m\left(\frac{W}{m}\right)^{\alpha},$$

by using any scheduling algorithm.

Therefore, our scheduling problems are defined such that the energy-delay product is optimized by fixing one factor and minimizing the other.

4 System partitioning

To schedule a list of n independent parallel tasks, algorithm H_c-A, where A is a list scheduling algorithm to be presented in the next section, divides the list into c sublists according to task sizes (i.e., numbers of processors requested by tasks), where $c \geq 1$ is a positive integer constant. For $1 \leq j \leq c-1$, we define sublist j to be the sublist of tasks with $m/(j+1) < \pi_i \leq m/j$, i.e., sublist j contains all tasks whose sizes are in the interval $I_j = (m/(j+1), m/j]$. We define sublist c to be the

sublist of tasks with $0 < \pi_i \leq m/c$, i.e., sublist c contains all tasks whose sizes are in the interval $I_c = (0, m/c]$. The partition of $(0, m]$ into intervals $I_1, I_2, \ldots, I_j, \ldots, I_c$ is called *the harmonic system partitioning scheme* whose idea is to schedule tasks of similar sizes together. The similarity is defined by the intervals $I_1, I_2, \ldots, I_j, \ldots, I_c$. For tasks in sublist j, processor utilization is higher than $j/(j+1)$, where $1 \leq j \leq c-1$. As j increases, the similarity among tasks in sublist j increases, and processor utilization also increases. Hence, the harmonic system partitioning scheme is very good at handling small tasks.

Algorithm H_c-A produces schedules of the sublists sequentially and separately. To schedule tasks in sublist j, where $1 \leq j \leq c$, the m processors are partitioned into j *clusters*, and each cluster contains m/j processors. Each cluster of processors is treated as one unit to be allocated to one task in sublist j. This is basically the harmonic system partitioning and processor allocation scheme. Therefore, scheduling parallel tasks in sublist j on the j clusters where each task i has processor requirement π_i and execution requirement r_i is equivalent to scheduling a list of sequential tasks on j processors where each task i has execution requirement r_i. It is clear that scheduling of the list of sequential tasks on j processors can be accomplished by using algorithm A, where A is a list scheduling algorithm.

5 Task scheduling

When a multiprocessor computer with m processors is partitioned into $j \geq 1$ clusters, scheduling tasks in sublist j is essentially dividing sublist j into j groups of tasks, such that each group of tasks are executed on one cluster. Such a partition of sublist j into j groups is essentially a schedule of the tasks in sublist j on m processors with j clusters. Once a partition (i.e., a schedule) is determined, we can use the methods in Sects. 6–8 to find power supplies.

We propose to use the list scheduling algorithm and its variations to solve the task scheduling problem. Tasks in sublist j are scheduled on j clusters by using the classic list scheduling algorithm [10] and by ignoring the issue of power supplies. In other words, the task execution times are simply $r_1, r_2, \ldots, r_n$, and tasks are assigned to the j clusters (i.e., groups) by using the list scheduling algorithm, which works as follows to schedule a list of tasks $1, 2, 3, \ldots$.

- *List Scheduling* (LS): Initially, task k is scheduled on cluster (or group) k, where $1 \leq k \leq j$, and tasks $1, 2, \ldots, j$ are removed from the list. Upon the completion of a task k, the first unscheduled task in the list, i.e., task $j+1$, is removed from the list and scheduled to be executed on cluster k. This process repeats until all tasks in the list are finished.

Algorithm LS has many variations, depending on the strategy used in the initial ordering of the tasks. We mention several of them here.

- *Largest Requirement First* (LRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $r_1 \geq r_2 \geq \cdots \geq r_n$.
- *Smallest Requirement First* (SRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $r_1 \leq r_2 \leq \cdots \leq r_n$.

- *Largest Size First* (LSF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_n$.
- *Smallest Size First* (SSF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $\pi_1 \leq \pi_2 \leq \cdots \leq \pi_n$.
- *Largest Task First* (LTF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $\pi_1^{1/\alpha} r_1 \geq \pi_2^{1/\alpha} r_2 \geq \cdots \geq \pi_n^{1/\alpha} r_n$.
- *Smallest Task First* (STF): This algorithm is the same as the LS algorithm, except that the tasks are arranged so that $\pi_1^{1/\alpha} r_1 \leq \pi_2^{1/\alpha} r_2 \leq \cdots \leq \pi_n^{1/\alpha} r_n$.

We call algorithm LS and its variations simply as list scheduling algorithms.

6 Task level power supplying

As mentioned earlier, our scheduling problems consist of three components, namely, system partitioning, task scheduling, and power supplying. Our strategies for scheduling parallel tasks include two basic ideas. First, tasks are divided into c sublists, where each sublist contains tasks of similar sizes, and the sublists are scheduled separately. Second, for each sublist j, the m processors are partitioned into $j \geq 1$ clusters and tasks in sublist j are partitioned into j groups such that each cluster of processors execute one group of tasks. Once a partition (and a schedule) is given, power supplies which minimize the schedule length within energy consumption constraint or the energy consumption within schedule length constraint can be determined. We adopt a three-level energy/time/power allocation scheme for a given schedule, namely, optimal power supplies to tasks in the same group (Theorems 3 and 4 in Sect. 6), optimal energy allocation among groups of tasks in the same sublist (Theorems 5 and 6 in Sect. 7), and optimal energy/time allocation among sublists of tasks (Theorems 7 and 8 in Sect. 8).

We first consider optimal power supplies to tasks in the same group. In fact, we discuss task level power supplying in a more general case, i.e., when n parallel tasks have to be scheduled sequentially on m processors. This may happen when $\pi_i > m/2$ for all $1 \leq i \leq n$. In this case, the m processors are treated as one unit, i.e., a cluster, to be allocated to one task. Of course, for each particular task i, only π_i of the m allocated processors are actually used and consume energy. It is clear that the problem of minimizing schedule length with energy consumption constraint E is simply to find the power supplies $p_1, p_2, \ldots, p_n$ such that the schedule length

$$T = \frac{r_1}{p_1^{1/\alpha}} + \frac{r_2}{p_2^{1/\alpha}} + \cdots + \frac{r_n}{p_n^{1/\alpha}}$$

is minimized and the total energy consumed $e_1 + e_2 + \cdots + e_n$ does not exceed E, i.e.,

$$\pi_1 r_1 p_1^{1-1/\alpha} + \pi_2 r_2 p_2^{1-1/\alpha} + \cdots + \pi_n r_n p_n^{1-1/\alpha} \leq E.$$

Let $M = \pi_1^{1/\alpha} r_1 + \pi_2^{1/\alpha} r_2 + \cdots + \pi_n^{1/\alpha} r_n$. The following result gives the optimal power supplies when the n tasks are scheduled sequentially.

Theorem 3 *When the n tasks are scheduled sequentially, the schedule length is minimized when task i is supplied with power* $p_i = (E/M)^{\alpha/(\alpha-1)}/\pi_i$, *where* $1 \le i \le n$. *The optimal schedule length is* $T = M^{\alpha/(\alpha-1)}/E^{1/(\alpha-1)}$.

Proof We can minimize T by using the Lagrange multiplier system

$$\nabla T(p_1, p_2, \ldots, p_n) = \lambda \nabla F(p_1, p_2, \ldots, p_n),$$

where T is viewed as a function of $p_1, p_2, \ldots, p_n$, λ is the Lagrange multiplier, and F is the constraint $\pi_1 r_1 p_1^{1-1/\alpha} + \pi_2 r_2 p_2^{1-1/\alpha} + \cdots + \pi_n r_n p_n^{1-1/\alpha} - E = 0$. Since

$$\frac{\partial T}{\partial p_i} = \lambda \frac{\partial F}{\partial p_i},$$

that is,

$$r_i \left(-\frac{1}{\alpha}\right) \frac{1}{p_i^{1+1/\alpha}} = \lambda \pi_i r_i \left(1 - \frac{1}{\alpha}\right) \frac{1}{p_i^{1/\alpha}},$$

$1 \le i \le n$, we get

$$p_i = \frac{1}{\lambda(1-\alpha)\pi_i},$$

which implies that

$$\sum_{i=1}^{n} \frac{\pi_i r_i}{(\lambda(1-\alpha)\pi_i)^{1-1/\alpha}} = E,$$

$$\frac{1}{\lambda(1-\alpha)} = \left(\frac{E}{M}\right)^{\alpha/(\alpha-1)},$$

and

$$p_i = \frac{1}{\pi_i} \left(\frac{E}{M}\right)^{\alpha/(\alpha-1)}$$

for all $1 \le i \le n$. Consequently, we get the optimal schedule length

$$T = \sum_{i=1}^{n} \frac{r_i}{p_i^{1/\alpha}} = \sum_{i=1}^{n} \pi_i^{1/\alpha} r_i \left(\frac{M}{E}\right)^{1/(\alpha-1)} = M \left(\frac{M}{E}\right)^{1/(\alpha-1)} = \frac{M^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}}.$$

This proves the theorem. □

It is clear that on a unicluster computer with time constraint T, the problem of minimizing energy consumption with schedule length constraint is simply to find the power supplies $p_1, p_2, \ldots, p_n$ such that the total energy consumption

$$E = \pi_1 r_1 p_1^{1-1/\alpha} + \pi_2 r_2 p_2^{1-1/\alpha} + \cdots + \pi_n r_n p_n^{1-1/\alpha}$$

is minimized and the schedule length $t_1 + t_2 + \cdots + t_n$ does not exceed T, i.e.,

$$\frac{r_1}{p_1^{1/\alpha}} + \frac{r_2}{p_2^{1/\alpha}} + \cdots + \frac{r_n}{p_n^{1/\alpha}} \le T.$$

The following result gives the optimal power supplies when the n tasks are scheduled sequentially.

Theorem 4 *When the n tasks are scheduled sequentially, the total energy consumption is minimized when task i is supplied with power* $p_i = (M/T)^\alpha/\pi_i$, *where* $1 \le i \le n$. *The minimum energy consumption is* $E = M^\alpha/T^{\alpha-1}$.

Proof We can minimize E by using the Lagrange multiplier system

$$\nabla E(p_1, p_2, \ldots, p_n) = \lambda \nabla F(p_1, p_2, \ldots, p_n),$$

where E is viewed as a function of $p_1, p_2, \ldots, p_n$, λ is the Lagrange multiplier, and F is the constraint

$$\frac{r_1}{p_1^{1/\alpha}} + \frac{r_2}{p_2^{1/\alpha}} + \cdots + \frac{r_n}{p_n^{1/\alpha}} - T = 0.$$

Since

$$\frac{\partial E}{\partial p_i} = \lambda \frac{\partial F}{\partial p_i},$$

that is,

$$\pi_i r_i \left(1 - \frac{1}{\alpha}\right) \frac{1}{p_i^{1/\alpha}} = \lambda r_i \left(-\frac{1}{\alpha}\right) \frac{1}{p_i^{1+1/\alpha}},$$

$1 \le i \le n$, we get

$$p_i = \frac{\lambda}{(1-\alpha)\pi_i},$$

which implies that

$$\sum_{i=1}^{n} r_i \left(\frac{(1-\alpha)\pi_i}{\lambda}\right)^{1/\alpha} = T,$$

$$\frac{1-\alpha}{\lambda} = \left(\frac{T}{M}\right)^\alpha,$$

and

$$p_i = \frac{1}{\pi_i}\left(\frac{M}{T}\right)^\alpha$$

for all $1 \le i \le n$. Consequently, we get the minimum energy consumption

$$E = \sum_{i=1}^{n} \pi_i r_i p_i^{1-1/\alpha} = \sum_{i=1}^{n} \pi_i r_i \frac{1}{\pi_i^{1-1/\alpha}} \left(\frac{M}{T}\right)^{\alpha-1} = M\left(\frac{M}{T}\right)^{\alpha-1} = \frac{M^\alpha}{T^{\alpha-1}}.$$

This proves the theorem. □

7 Group level energy allocation

Now, we consider optimal energy allocation among groups of tasks in the same sublist. Again, we discuss group level energy allocation in a more general case, i.e., scheduling n parallel tasks on m processors, where $\pi_i \leq m/j$ for all $1 \leq i \leq n$ with $j \geq 1$. In this case, the m processors can be partitioned into j clusters such that each cluster contains m/j processors. Each cluster of processors are treated as one unit to be allocated to one task. Assume that the set of n tasks is partitioned into j groups such that all the tasks in group k are executed on cluster k, where $1 \leq k \leq j$. Let M_k denote the total $\pi_i^{1/\alpha} r_i$ of the tasks in group k. For a given partition of the n tasks into j groups, we are seeking power supplies that minimize the schedule length. Let E_k be the energy consumed by all the tasks in group k. The following result characterizes the optimal power supplies.

Theorem 5 *For a given partition $M_1, M_2, \ldots, M_j$ of the n tasks into j groups on a multiprocessor computer partitioned into j clusters, the schedule length is minimized when task i in group k is supplied with power $p_i = (E_k/M_k)^{\alpha/(\alpha-1)}/\pi_i$, where*

$$E_k = \left(\frac{M_k^\alpha}{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha} \right) E$$

for all $1 \leq k \leq j$. The optimal schedule length is

$$T = \left(\frac{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha}{E} \right)^{1/(\alpha-1)}$$

for the above power supplies.

Proof We observe that by fixing E_k and supplying power $p_i = (E_k/M_k)^{\alpha/(\alpha-1)}/\pi_i$ to task i in group k according to Theorem 3, the total execution time of the tasks in group k can be minimized to

$$T_k = \frac{M_k^{\alpha/(\alpha-1)}}{E_k^{1/(\alpha-1)}}.$$

Therefore, the problem of finding power supplies $p_1, p_2, \ldots, p_n$ that minimize the schedule length is equivalent to finding $E_1, E_2, \ldots, E_j$ that minimize the schedule length. It is clear that the schedule length is minimized when all the j clusters complete their execution of the j groups of tasks at the same time T, that is, $T_1 = T_2 = \cdots = T_j = T$, which implies that

$$E_k = \frac{M_k^\alpha}{T^{\alpha-1}}.$$

Since $E_1 + E_2 + \cdots + E_j = E$, we have

$$\frac{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha}{T^{\alpha-1}} = E,$$

that is,

$$T = \left(\frac{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha}{E}\right)^{1/(\alpha-1)}$$

and

$$E_k = \left(\frac{M_k^\alpha}{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha}\right)E.$$

The theorem is proven. □

The following result gives the optimal power supplies that minimize energy consumption for a given partition of the n tasks into j groups on a multiprocessor computer.

Theorem 6 *For a given partition $M_1, M_2, \ldots, M_j$ of the n tasks into j groups on a multiprocessor computer partitioned into j clusters, the total energy consumption is minimized when task i in group k is executed with power $p_i = (M_k/T)^\alpha/\pi_i$, where $1 \le k \le j$. The minimum energy consumption is*

$$E = \frac{M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha}{T^{\alpha-1}}$$

for the above power supplies.

Proof By Theorem 4, the energy consumed by tasks in group k is minimized as $E_k = M_k^\alpha/T^{\alpha-1}$ without increasing the schedule length T by supplying power $p_i = (M_k/T)^\alpha/\pi_i$ to task i in group k. The minimum energy consumption is simply $E = E_1 + E_2 + \cdots + E_j = (M_1^\alpha + M_2^\alpha + \cdots + M_j^\alpha)/T^{\alpha-1}$. □

Notice that our results in Sects. 3, 6, 7 include those results in [18] as special cases. In other words, when $\pi_i = 1$ for all $1 \le i \le n$, Theorems 1–6 and the energy-delay tradeoff theorem become the results in [18].

8 Performance analysis

To use algorithm H_c-A to solve the problem of minimizing schedule length with energy consumption constraint E, we need to allocate the available energy E to the c sublists. We use $E_1, E_2, \ldots, E_c$ to represent an energy allocation to the c sublists, where sublist j consumes energy E_j, and $E_1 + E_2 + \cdots + E_c = E$. By using any of the list scheduling algorithms to schedule tasks in sublist j, we get a partition of the tasks in sublist j into j groups. Let R_j be the total execution requirement of tasks in

sublist j, $R_{j,k}$ be the total execution requirement of tasks in group k, and $M_{j,k}$ be the total $\pi_i^{1/\alpha} r_i$ of tasks in group k, where $1 \le k \le j$. Theorem 7 provides optimal energy allocation to the c sublists for minimizing schedule length with energy consumption constraint in scheduling parallel tasks by using scheduling algorithm H_c-A, where A is a list scheduling algorithm.

We define the *performance ratio* as $\beta = T/T^*$ for heuristic algorithms that solve the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer. The following theorem gives the performance ratio when algorithm H_c-A is used to solve the problem of minimizing schedule length with energy consumption constraint.

Theorem 7 *For a given partition $M_{j,1}, M_{j,2}, \ldots, M_{j,j}$ of the tasks in sublist j into j groups produced by a list scheduling algorithm A, where $1 \le j \le c$, and an energy allocation $E_1, E_2, \ldots, E_c$ to the c sublists, the scheduling algorithm H_c-A produces the schedule length*

$$T = \sum_{j=1}^{c} \left(\frac{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}}{E_j} \right)^{1/(\alpha-1)}.$$

The energy allocation $E_1, E_2, \ldots, E_c$ which minimizes T is

$$E_j = \left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}} \right) E,$$

where $N_j = M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}$ for all $1 \le j \le c$, and the minimized schedule length is

$$T = \frac{(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}},$$

by using the above energy allocation. The performance ratio is

$$\beta \le \left(\left(\left(\sum_{j=1}^{c} \frac{R_j}{j} \right) + cr^* \right) \Big/ \left(\frac{W}{m} \right) \right)^{\alpha/(\alpha-1)},$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement.

Proof By Theorem 5, for a given partition $M_{j,1}, M_{j,2}, \ldots, M_{j,j}$ of the tasks in sublist j into j groups, the schedule length T_j for sublist j is minimized when task i in group k is supplied with power $p_i = (E_{j,k}/M_{j,k})^{\alpha/(\alpha-1)}/\pi_i$, where

$$E_{j,k} = \left(\frac{M_{j,k}^{\alpha}}{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}} \right) E_j$$

for all $1 \le k \le j$. The optimal schedule length is

$$T_j = \left(\frac{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}}{E_j} \right)^{1/(\alpha-1)}$$

for the above power supplies. Since algorithm H_c-A produces the schedule length $T = T_1 + T_2 + \cdots + T_c$, we have

$$T = \sum_{j=1}^{c} \left(\frac{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}}{E_j} \right)^{1/(\alpha-1)}.$$

By the definition of N_j, we obtain

$$T = \left(\frac{N_1}{E_1}\right)^{1/(\alpha-1)} + \left(\frac{N_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + \left(\frac{N_c}{E_c}\right)^{1/(\alpha-1)}.$$

To minimize T, we use the Lagrange multiplier system

$$\nabla T(E_1, E_2, \ldots, E_c) = \lambda \nabla F(E_1, E_2, \ldots, E_c),$$

where λ is the Lagrange multiplier, and F is the constraint $E_1 + E_2 + \cdots + E_c - E = 0$. Since

$$\frac{\partial T}{\partial E_j} = \lambda \frac{\partial F}{\partial E_j},$$

that is,

$$N_j^{1/(\alpha-1)} \left(-\frac{1}{\alpha - 1} \right) \frac{1}{E_j^{1/(\alpha-1)+1}} = \lambda,$$

$1 \le j \le c$, we get

$$E_j = N_j^{1/\alpha} \left(\frac{1}{\lambda(1-\alpha)} \right)^{(\alpha-1)/\alpha},$$

which implies that

$$E = \left(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}\right) \left(\frac{1}{\lambda(1-\alpha)} \right)^{(\alpha-1)/\alpha}$$

and

$$E_j = \left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}} \right) E$$

for all $1 \le j \le c$. By using the above energy allocation, we have

$$\begin{aligned} T &= \sum_{j=1}^{c} \left(\frac{N_j}{E_j} \right)^{1/(\alpha-1)} \\ &= \sum_{j=1}^{c} \frac{N_j^{1/(\alpha-1)}}{\left(\left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}} \right) E \right)^{1/(\alpha-1)}} \end{aligned}$$

$$= \sum_{j=1}^{c} \frac{N_j^{1/\alpha}(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{1/(\alpha-1)}}{E^{1/(\alpha-1)}}$$

$$= \frac{(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}}.$$

For any list scheduling algorithm A, we have

$$R_{j,k} \le \frac{R_j}{j} + r^*$$

for all $1 \le j \le c$ and $1 \le k \le j$. Since $\pi_i \le m/j$ for every task i in sublist j, we get

$$M_{j,k} \le \left(\frac{m}{j}\right)^{1/\alpha} R_{j,k} \le \left(\frac{m}{j}\right)^{1/\alpha} \left(\frac{R_j}{j} + r^*\right).$$

Therefore,

$$N_j \le m\left(\frac{R_j}{j} + r^*\right)^{\alpha},$$

$$N_j^{1/\alpha} \le m^{1/\alpha}\left(\frac{R_j}{j} + r^*\right),$$

and

$$N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha} \le m^{1/\alpha}\left(\left(\sum_{j=1}^{c} \frac{R_j}{j}\right) + cr^*\right),$$

which implies that

$$T \le m^{1/(\alpha-1)}\left(\left(\sum_{j=1}^{c} \frac{R_j}{j}\right) + cr^*\right)^{\alpha/(\alpha-1)} \frac{1}{E^{1/(\alpha-1)}}.$$

By Theorem 1, we get

$$\beta = \frac{T}{T^*} \le \left(\left(\left(\sum_{j=1}^{c} \frac{R_j}{j}\right) + cr^*\right) \Big/ \left(\frac{W}{m}\right)\right)^{\alpha/(\alpha-1)}.$$

This proves the theorem. □

Theorems 5 and 7 give the power supply to the task i in group k of sublist j as

$$\frac{1}{\pi_i}\left(\frac{E_{j,k}}{M_{j,k}}\right)^{\alpha/(\alpha-1)}$$

$$= \frac{1}{\pi_i}\left(\left(\frac{M_{j,k}^{\alpha}}{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}}\right)\left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}}\right)\frac{E}{M_{j,k}}\right)^{\alpha/(\alpha-1)}$$

for all $1 \le j \le c$ and $1 \le k \le j$.

We notice that the performance bound given in Theorem 7 is pessimistic mainly due to the overestimation of the π_i's in sublist j to m/j. One possible remedy is to use $(m/(j+1)+m/j)/2$ as an approximation to π_i. Also, as the number of tasks gets large, the term cr^* may be removed. This gives rise to the following performance bound for β:

$$\left(\left(\sum_{j=1}^{c} \frac{R_j}{j}\left(\frac{2j+1}{2j+2}\right)^{1/\alpha}\right) \Big/ \left(\frac{W}{m}\right)\right)^{\alpha/(\alpha-1)}. \tag{1}$$

Our simulation shows that the modified bound in (1) is more accurate than the performance bound given in Theorem 7.

To use algorithm H_c-A to solve the problem of minimizing energy consumption with schedule length constraint T, we need to allocate the time T to the c sublists. We use $T_1, T_2, \ldots, T_c$ to represent a time allocation to the c sublists, where tasks in sublist j are executed within deadline T_j, and $T_1 + T_2 + \cdots + T_c = T$. Theorem 8 provides optimal time allocation to the c sublists for minimizing energy consumption with schedule length constraint in scheduling parallel tasks by using scheduling algorithm H_c-A, where A is a list scheduling algorithm.

We define the *performance ratio* as $\beta = E/E^*$ for heuristic algorithms that solve the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer. The following theorem gives the performance ratio when algorithm H_c-A is used to solve the problem of minimizing energy consumption with schedule length constraint.

Theorem 8 *For a given partition $M_{j,1}, M_{j,2}, \ldots, M_{j,j}$ of the tasks in sublist j into j groups produced by a list scheduling algorithm A, where $1 \le j \le c$, and a time allocation $T_1, T_2, \ldots, T_c$ to the c sublists, the scheduling algorithm H_c-A consumes the energy*

$$E = \sum_{j=1}^{c}\left(\frac{M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}}{T_j^{\alpha-1}}\right).$$

The time allocation $T_1, T_2, \ldots, T_c$ which minimizes E is

$$T_j = \left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}}\right)T,$$

where $N_j = M_{j,1}^{\alpha} + M_{j,2}^{\alpha} + \cdots + M_{j,j}^{\alpha}$ for all $1 \le j \le c$, and the minimized energy consumption is

$$E = \frac{(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{\alpha}}{T^{\alpha-1}},$$

by using the above time allocation. The performance ratio is

$$\beta \le \left(\left(\left(\sum_{j=1}^{c} \frac{R_j}{j}\right) + cr^*\right) \Big/ \left(\frac{W}{m}\right)\right)^{\alpha},$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ *is the maximum task execution requirement.*

Proof By Theorem 6, for a given partition $M_{j,1}, M_{j,2}, \ldots, M_{j,j}$ of the tasks in sublist j into j groups, the total energy E_j consumed by sublist j is minimized when task i in group k is executed with power $p_i = (M_{j,k}/T_j)^\alpha/\pi_i$, where $1 \le j \le c$ and $1 \le k \le j$. The minimum energy consumption is

$$E_j = \frac{M_{j,1}^\alpha + M_{j,2}^\alpha + \cdots + M_{j,j}^\alpha}{T_j^{\alpha-1}}$$

for the above power supplies. Since algorithm H_c-A consumes the energy $E = E_1 + E_2 + \cdots + E_c$, we have

$$E = \sum_{j=1}^{c} \left(\frac{M_{j,1}^\alpha + M_{j,2}^\alpha + \cdots + M_{j,j}^\alpha}{T_j^{\alpha-1}} \right).$$

By the definition of N_j, we obtain

$$E = \frac{N_1}{T_1^{\alpha-1}} + \frac{N_2}{T_2^{\alpha-1}} + \cdots + \frac{N_c}{T_c^{\alpha-1}}.$$

To minimize E, we use the Lagrange multiplier system

$$\nabla E(T_1, T_2, \ldots, T_c) = \lambda \nabla F(T_1, T_2, \ldots, T_c),$$

where λ is the Lagrange multiplier, and F is the constraint $T_1 + T_2 + \cdots + T_c - T = 0$. Since

$$\frac{\partial E}{\partial T_j} = \lambda \frac{\partial F}{\partial T_j},$$

that is,

$$N_j \left(\frac{1-\alpha}{T_j^\alpha} \right) = \lambda,$$

$1 \le j \le c$, we get

$$T_j = N_j^{1/\alpha} \left(\frac{1-\alpha}{\lambda} \right)^{1/\alpha},$$

which implies that

$$T = \left(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha} \right) \left(\frac{1-\alpha}{\lambda} \right)^{1/\alpha}$$

and

$$T_j = \left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha}} \right) T$$

for all $1 \le j \le c$. By using the above time allocation, we have

$$\begin{aligned} E &= \sum_{j=1}^{c} \frac{N_j}{T_j^{\alpha-1}} \\ &= \sum_{j=1}^{c} \frac{N_j}{\left(\left(\frac{N_j^{1/\alpha}}{N_1^{1/\alpha}+N_2^{1/\alpha}+\cdots+N_c^{1/\alpha}}\right)T\right)^{\alpha-1}} \\ &= \sum_{j=1}^{c} \frac{N_j^{1/\alpha}(N_1^{1/\alpha}+N_2^{1/\alpha}+\cdots+N_c^{1/\alpha})^{\alpha-1}}{T^{\alpha-1}} \\ &= \frac{(N_1^{1/\alpha}+N_2^{1/\alpha}+\cdots+N_c^{1/\alpha})^{\alpha}}{T^{\alpha-1}}. \end{aligned}$$

Similar to the proof of Theorem 7, we have

$$N_1^{1/\alpha}+N_2^{1/\alpha}+\cdots+N_c^{1/\alpha} \le m^{1/\alpha}\left(\left(\sum_{j=1}^{c}\frac{R_j}{j}\right)+cr^*\right),$$

which implies that

$$E \le m\left(\left(\sum_{j=1}^{c}\frac{R_j}{j}\right)+cr^*\right)^{\alpha}\frac{1}{T^{\alpha-1}}.$$

By Theorem 2, we get

$$\beta = \frac{E}{E^*} \le \left(\left(\left(\sum_{j=1}^{c}\frac{R_j}{j}\right)+cr^*\right)\Big/\left(\frac{W}{m}\right)\right)^{\alpha}.$$

This proves the theorem. □

Theorems 6 and 8 give the power supply to task i in group k of sublist j as

$$\frac{1}{\pi_i}\left(\frac{M_{j,k}}{T_j}\right)^{\alpha} = \frac{1}{\pi_i}\left(\frac{M_{j,k}(N_1^{1/\alpha}+N_2^{1/\alpha}+\cdots+N_c^{1/\alpha})}{N_j^{1/\alpha}T}\right)^{\alpha}$$

for all $1 \le j \le c$ and $1 \le k \le j$.

Again, we adjust the performance bound given in Theorem 8 to

$$\left(\left(\sum_{j=1}^{c}\frac{R_j}{j}\left(\frac{2j+1}{2j+2}\right)^{1/\alpha}\right)\Big/\left(\frac{W}{m}\right)\right)^{\alpha}. \tag{2}$$

Our simulation shows that the modified bound in (2) is more accurate than the performance bound given in Theorem 8.

9 Numerical and simulation data

To validate our analytical results, extensive simulations are conducted. In this section, we demonstrate some numerical and experimental data.

We define the *normalized schedule length* (NSL) as

$$\mathrm{NSL} = \frac{T}{\left(\frac{m}{E}\left(\frac{W}{m}\right)^{\alpha}\right)^{1/(\alpha-1)}}.$$

When T is the schedule length produced by a heuristic algorithm H_c-A according to Theorem 7, the normalized schedule length is

$$\mathrm{NSL} = \left(\frac{(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{\alpha}}{m\left(\frac{W}{m}\right)^{\alpha}}\right)^{1/(\alpha-1)}.$$

NSL is an upper bound for the performance ratio $\beta = T/T^*$ for the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer. When the π_i's and the r_i's are random variables, T, T^*, β, and NSL all become random variables. It is clear that for the problem of minimizing schedule length with energy consumption constraint, we have $\bar{\beta} \le \overline{\mathrm{NSL}}$, i.e., the expected performance ratio is no larger than the expected normalized schedule length. (We use $\bar{x}$ to represent the expectation of a random variable x.)

We define the *normalized energy consumption* (NEC) as

$$\mathrm{NEC} = \frac{E}{m\left(\frac{W}{m}\right)^{\alpha}\frac{1}{T^{\alpha-1}}}.$$

When E is the energy consumed by a heuristic algorithm H_c-A according to Theorem 8, the normalized energy consumption is

$$\mathrm{NEC} = \frac{(N_1^{1/\alpha} + N_2^{1/\alpha} + \cdots + N_c^{1/\alpha})^{\alpha}}{m\left(\frac{W}{m}\right)^{\alpha}}.$$

NEC is an upper bound for the performance ratio $\beta = E/E^*$ for the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer. For the problem of minimizing energy consumption with schedule length constraint, we have $\bar{\beta} \le \overline{\mathrm{NEC}}$.

Notice that the expected normalized schedule length $\overline{\mathrm{NSL}}$ and the expected normalized energy consumption $\overline{\mathrm{NEC}}$ are determined by A, c, m, n, α, and the probability distributions of the π_i's and r_i's. In our simulations, the algorithm A is chosen as LS; the parameter c is set as 20; the number of processors is set as $m = 128$; the number of tasks is set as $n = 1{,}000$; and the parameter α is set as 3. The particular choices of these values do not affect our general observations and conclusions. For convenience, the r_i's are treated as independent and identically distributed (i.i.d.) continuous random variables uniformly distributed in $[0, 1)$. The π_i's are i.i.d. discrete random variables. We consider three types of probability distributions of task sizes with about the same expected task size $\bar{\pi}$. Let a_b be the probability that $\pi_i = b$, where $b \ge 1$.

- Uniform distributions in the range $[1..u]$, i.e., $a_b = 1/u$ for all $1 \le b \le u$, where u is chosen such that $(u+1)/2 = \bar{\pi}$, i.e., $u = 2\bar{\pi} - 1$.
- Binomial distributions in the range $[1..m]$, i.e.,

$$a_b = \frac{\binom{m}{b} p^b (1-p)^{m-b}}{1-(1-p)^m}$$

for all $1 \le b \le m$, where p is chosen such that $mp = \bar{\pi}$, i.e., $p = \bar{\pi}/m$. However, the actual expectation of task sizes is

$$\frac{\bar{\pi}}{1-(1-p)^m} = \frac{\bar{\pi}}{1-(1-\bar{\pi}/m)^m},$$

which is slightly greater than $\bar{\pi}$, especially when $\bar{\pi}$ is small.
- Geometric distributions in the range $[1..m]$, i.e.,

$$a_b = \frac{q(1-q)^{b-1}}{1-(1-q)^m}$$

for all $1 \le b \le m$, where q is chosen such that $1/q = \bar{\pi}$, i.e., $q = 1/\bar{\pi}$. However, the actual expectation of task sizes is

$$\frac{1/q - (1/q+m)(1-q)^m}{1-(1-q)^m} = \frac{\bar{\pi} - (\bar{\pi}+m)(1-1/\bar{\pi})^m}{1-(1-1/\bar{\pi})^m},$$

which is less than $\bar{\pi}$, especially when $\bar{\pi}$ is large.

In Tables 1 and 2, we show and compare the analytical results with simulation data. For each $\bar{\pi}$ in the range $10, 15, 20, \ldots, 60$, and each probability distribution of task sizes, we generate 200 sets of n tasks, produce their schedules by using algorithm H_c-LS, calculate their NSL (or NEC) and the bound (1) (or bound (2)), report the average of NSL (or NEC) which is the experimental value of $\overline{\text{NSL}}$ (or $\overline{\text{NEC}}$), and report the average of bound (1) (or bound (2)) which is the numerical value of analytical results. The 99% confidence interval of all the data in the same table is also given.

We have the following observations from our simulations.

- $\overline{\text{NSL}}$ is less than 1.4, and $\overline{\text{NEC}}$ is less than 1.95, except the case for uniform distribution with $\bar{\pi} = 45$. Since $\overline{\text{NSL}}$ and $\overline{\text{NEC}}$ only give upper bonds for the expected performance ratios, the performance of our heuristic algorithms are even better, and our heuristic algorithms are able to produce solutions very close to optimum.
- The performance of algorithm H_c-A for A other than LS (i.e., LRF, SRF, LSF, SSF, LTF, STF) is very close (within $\pm 1\%$) to the performance of algorithm H_c-LS. Since these data do not provide further insight, they are not shown here.
- The performance bound (1) is very close to $\overline{\text{NSL}}$, and the performance bound (2) is very close to $\overline{\text{NEC}}$. Our analytical results provide very accurate estimation of the expected normalized schedule length and the expected normalized energy consumption.

Table 1 Simulation data for expected NSL

$\bar{\pi}$	Uniform		Binomial		Geometric	
	Simulation	Analysis	Simulation	Analysis	Simulation	Analysis
10	1.1311531	1.1846499	1.0711008	1.0636341	1.2176904	1.3172420
15	1.1262486	1.1493186	1.0794990	1.0549572	1.2042597	1.2607125
20	1.1377073	1.1495630	1.0991387	1.0820476	1.2260825	1.2718070
25	1.1963542	1.2221468	1.1179888	1.1164336	1.2472974	1.2887673
30	1.1925090	1.2028694	1.1377585	1.1406375	1.2650054	1.3045373
35	1.2671006	1.3060567	1.1627916	1.1730722	1.2758955	1.3126316
40	1.3724390	1.4507239	1.2108560	1.2372959	1.2822935	1.3162972
45	1.4036446	1.4835721	1.2629891	1.3070823	1.2863025	1.3173556
50	1.3963575	1.4611373	1.2486138	1.2775513	1.2907750	1.3198693
55	1.3667205	1.4084232	1.2095924	1.2158904	1.2915822	1.3179406
60	1.3275050	1.3448166	1.2823717	1.3218361	1.2953585	1.3205274

(99% confidence interval $\pm 0.289\%$)

Table 2 Simulation data for expected NEC

$\bar{\pi}$	Uniform		Binomial		Geometric	
	Simulation	Analysis	Simulation	Analysis	Simulation	Analysis
10	1.2796678	1.4030164	1.1486957	1.1318542	1.4833754	1.7352263
15	1.2696333	1.3238851	1.1659599	1.1137040	1.4469347	1.5848844
20	1.2935587	1.3196059	1.2091747	1.1717433	1.5002288	1.6140745
25	1.4304500	1.4922751	1.2505756	1.2465521	1.5614698	1.6698993
30	1.4221576	1.4470699	1.2940850	1.3000633	1.5917814	1.6898822
35	1.6006477	1.6981148	1.3515095	1.3745887	1.6257366	1.7213599
40	1.8792217	2.0984836	1.4663274	1.5313174	1.6441691	1.7334538
45	1.9726518	2.2049910	1.5960920	1.7100668	1.6554994	1.7372576
50	1.9496911	2.1343696	1.5591070	1.6324087	1.6652267	1.7404442
55	1.8710998	1.9885360	1.4628489	1.4780403	1.6717055	1.7421288
60	1.7632633	1.8092858	1.6380633	1.7373764	1.6737146	1.7375970

(99% confidence interval $\pm 0.553\%$)

10 Concluding remarks

We have made some initial attempt to address energy-efficient scheduling of parallel tasks on multiprocessor computers with dynamic voltage and speed as combinatorial optimization problems. We defined the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint for independent parallel tasks on multiprocessor computers. We argued that each heuristic algorithm should solve three nontrivial subproblems efficiently, namely, system partitioning, task scheduling, and power supply-

ing. By using the harmonic system partitioning and processor allocation method, the list scheduling algorithms, and a three-level energy/time/power allocation scheme, we have developed heuristic algorithms which are able to produce schedules very close to optimum. In doing so, we have also established lower bounds for the optimal solutions and have found an energy-delay tradeoff theorem.

There are several further research directions. In addition to independent parallel tasks in this paper, our scheduling problems can be extended to precedence constrained parallel tasks. Investigation can also be directed toward scheduling parallel tasks on multiprocessors with discrete voltage/speed settings [14, 24].

Acknowledgements Thanks are due to the reviewers whose comments led to improved organization and presentation of the paper. A preliminary version of the paper was presented on the *6th Workshop on High-Performance Power-Aware Computing* (HPPAC 2010) held on April 19, 2010, Atlanta, Georgia, USA, in conjunction with the *24th International Parallel and Distributed Processing Symposium* (IPDPS 2010).

References

1. Aydin H, Melhem R, Mossé D, Mejía-Alvarez P (2004) Power-aware scheduling for periodic real-time tasks. IEEE Trans Comput 53(5):584–600
2. Bansal N, Kimbrel T, Pruhs K (2004) Dynamic speed scaling to manage energy and temperature. In: Proceedings of the 45th IEEE symposium on foundation of computer science, pp 520–529
3. Barnett JA (2005) Dynamic task-level voltage scheduling optimizations. IEEE Trans Comput 54(5):508–520
4. Benini L, Bogliolo A, De Micheli G (2000) A survey of design techniques for system-level dynamic power management. IEEE Trans Very Large Scale Integr (VLSI) Syst 8(3):299–316
5. Bunde DP (2006) Power-aware scheduling for makespan and flow. In: Proceedings of the 18th ACM symposium on parallelism in algorithms and architectures, pp 190–196
6. Chan H-L, Chan W-T, Lam T-W, Lee L-K, Mak K-S, Wong PWH (2007) Energy efficient online deadline scheduling. In: Proceedings of the 18th ACM–SIAM symposium on discrete algorithms, pp 795–804
7. Chandrakasan AP, Sheng S, Brodersen RW (1992) Low-power CMOS digital design. IEEE J Solid-State Circuits 27(4):473–484
8. Feng W-C (2005) The importance of being low power in high performance computing. CTWatch Quarterly 1(3). Los Alamos National Laboratory, August
9. Gara A et al (2005) Overview of the Blue Gene/L system architecture. IBM J Res Dev 49(2/3):195–212
10. Graham RL (1969) Bounds on multiprocessing timing anomalies. SIAM J Appl Math 2:416–429
11. Graham SL, Snir M, Patterson CA (eds) (2005) Getting up to speed: the future of supercomputing. Committee on the Future of Supercomputing, National Research Council, National Academies Press, Washington
12. Hong I, Kirovski D, Qu G, Potkonjak M, Srivastava MB (1999) Power optimization of variable-voltage core-based systems. IEEE Trans Comput-Aided Des Integr Circuits Syst 18(12):1702–1714
13. Im C, Ha S, Kim H (2004) Dynamic voltage scheduling with buffers in low-power multimedia applications. ACM Trans Embed Comput Syst 3(4):686–705
14. Intel (2004) Enhanced Intel SpeedStep technology for the Intel Pentium M processor—white paper, March
15. Krishna CM, Lee Y-H (2003) Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. IEEE Trans Comput 52(12):1586–1593
16. Kwon W-C, Kim T (2005) Optimal voltage allocation techniques for dynamically variable voltage processors. ACM Trans Embed Comput Syst 4(1):211–230
17. Lee Y-H, Krishna CM (2003) Voltage-clock scaling for low energy consumption in fixed-priority real-time systems. Real-Time Syst 24(3):303–317

18. Li K (2008) Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. IEEE Trans Parallel Distrib Syst 19(11):1484–1497
19. Li M, Yao FF (2006) An efficient algorithm for computing optimal discrete voltage schedules. SIAM J Comput 35(3):658–671
20. Li M, Liu BJ, Yao FF (2006) Min-energy voltage allocation for tree-structured tasks. J Comb Optim 11:305–319
21. Li M, Yao AC, Yao FF (2006) Discrete and continuous min-energy schedules for variable voltage processors. Proc Natl Acad Sci USA 103(11):3983–3987
22. Lorch JR, Smith AJ (2004) PACE: a new approach to dynamic voltage scaling. IEEE Trans Comput 53(7):856–869
23. Mahapatra RN, Zhao W (2005) An energy-efficient slack distribution technique for multimode distributed real-time embedded systems. IEEE Trans Parallel Distrib Syst 16(7):650–662
24. Qu G (2001) What is the limit of energy saving by dynamic voltage scaling. In: Proceedings of the international conference on computer-aided design, pp 560–563
25. Quan G, Hu XS (2007) Energy efficient DVS schedule for fixed-priority real-time systems. ACM Trans Embed Comput Syst 6(4): Article no 29
26. Rusu C, Melhem R, Mossé D (2002) Maximizing the system value while satisfying time and energy constraints. In: Proceedings of the 23rd IEEE real-time systems symposium, pp 256–265
27. Shin D, Kim J (2003) Power-aware scheduling of conditional task graphs in real-time multiprocessor systems. In: Proceedings of the international symposium on low power electronics and design, pp 408–413
28. Shin D, Kim J, Lee S (2001) Intra-task voltage scheduling for low-energy hard real-time applications. IEEE Des Test Comput 18(2):20–30
29. Stan MR, Skadron K (2003) Guest editors' introduction: power-aware computing. IEEE Comput 36(12):35–38
30. Unsal OS, Koren I (2003) System-level power-aware design techniques in real-time systems. Proc IEEE 91(7):1055–1069
31. Venkatachalam V, Franz M (2005) Power reduction techniques for microprocessor systems. ACM Comput Surv 37(3):195–237
32. Weiser M, Welch B, Demers A, Shenker S (1994) Scheduling for reduced CPU energy. In: Proceedings of the 1st USENIX symposium on operating systems design and implementation, pp 13–23
33. Yang P, Wong C, Marchal P, Catthoor F, Desmet D, Verkest D, Lauwereins R (2001) Energy-aware runtime scheduling for embedded-multiprocessor SOCs. IEEE Des Test Comput 18(5):46–58
34. Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. In: Proceedings of the 36th IEEE symposium on foundations of computer science, pp 374–382
35. Yun H-S, Kim J (2003) On energy-optimal voltage scheduling for fixed-priority hard real-time systems. ACM Trans Embed Comput Syst 2(3):393–430
36. Zhai B, Blaauw D, Sylvester D, Flautner K (2004) Theoretical and practical limits of dynamic voltage scaling. In: Proceedings of the 41st design automation conference, pp 868–873
37. Zhu D, Melhem R, Childers BR (2003) Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE Trans Parallel Distrib Syst 14(7):686–700
38. Zhu D, Mossé D, Melhem R (2004) Power-aware scheduling for AND/OR graphs in real-time systems. IEEE Trans Parallel Distrib Syst 15(9):849–864
39. Zhuo J, Chakrabarti C (2008) Energy-efficient dynamic task scheduling algorithms for DVS systems. ACM Trans Embed Comput Syst 7(2): Article no 17