Journal of Computer and System Sciences 82 (2016) 174–190

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Power and performance management for parallel computations in clouds and data centers

Keqin Li *

Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 24 December 2014
Received in revised form 27 June 2015
Accepted 1 July 2015
Available online 21 July 2015

Keywords:
Cloud computing
Data center
Energy-efficient scheduling
Parallel task
Performance analysis
Precedence constraint
Simulation

ABSTRACT

We address scheduling independent and precedence constrained parallel tasks on multiple homogeneous processors in a data center with dynamically variable voltage and speed as combinatorial optimization problems. We consider the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on multiple processors. Our approach is to use level-by-level scheduling algorithms to deal with precedence constraints. We use a simple system partitioning and processor allocation scheme, which always schedules as many parallel tasks as possible for simultaneous execution. We use two heuristic algorithms for scheduling independent parallel tasks in the same level, i.e., SIMPLE and GREEDY. We adopt a two-level energy/time/power allocation scheme, namely, optimal energy/time allocation among levels of tasks and equal power supply to tasks in the same level. Our approach results in significant performance improvement compared with previous algorithms in scheduling independent and precedence constrained parallel tasks.

1. Introduction

1.1. Motivation

Modern supercomputing centers and data centers consume astonishing amount of power and generate huge amount of pollution. For instances, the Tianhe-2 system in the National Super Computer Center in Guangzhou (China), the world's fastest supercomputing system as of November 2014, has power consumption of 17,808 kW [27]. Assuming an electricity charge of 0.15 USD per kWh, the system has electricity cost of more than 23 million USD per year. It is well known that the extra costs of cooling and electrical losses are at about similar or even higher scale. The four Facebook's data centers consumed about 678 million kWh of energy in 2012, costing more than 25 million USD per data center [25]. The mix of fuels consumed to generate the energy used by Facebook consisted of 34% coal, 22% nuclear, 19% renewable, 15% natural gas, and 10% "uncategorized". It is also well known that among 100 units of source energy, only 33 units of electrical energy can be successfully generated and delivered to data centers. Therefore, energy consumption in data centers are actually tripled in terms of natural resources consumption. Furthermore, the amount of carbon emissions associated with Facebook's Prineville, Oregon, data center in 2012 was 104,000 metric tons of greenhouse gases. Sustainable high-performance computing and cloud computing has been a grand challenge for human civilization and an extremely important research direction.

* Fax: +845 257 3996.
E-mail address: lik@newpaltz.edu.

http://dx.doi.org/10.1016/j.jcss.2015.07.001
0022-0000/

Traditionally, parallel applications are run on high-performance computers (e.g., supercomputers and clusters), which are very expensive and not available to ordinary people. Cloud computing provides the tools and technologies such as computing resource sharing and virtualization to support data and computing intensive parallel applications with much more affordable prices compared to traditional parallel computing infrastructures [22]. Cloud computing is also the most effective approach to green computing due to resource virtualization, automation software, pay-per-use and self-service, and multitenancy [26]. While traditional parallel computing pursues the goal of high performance, cloud computing provides the opportunity to effective and efficient management of power and performance for parallel computations in green clouds and data centers simultaneously. Such balanced consideration of system performance and energy consumption for large-scale parallel applications is very important, since such applications typically consume significant computing resources, times, and energy. Furthermore, cloud computing has the capability to provide energy-efficient methods and algorithms to deal with the power-performance tradeoff at the data center level, when numerous users submit their service requests at the same time. Energy-efficient cloud computing [10] is the issue to be considered in this paper.

1.2. Related research

Increased energy consumption causes severe economic, ecological, and technical problems. Power conservation is critical in many computation and communication environments and has attracted extensive research activities. Reducing processor energy consumption has been an important and pressing research issue in recent years. There has been increasing interest and importance in developing high-performance and energy-efficient computing systems [17–19]. There exists an explosive body of literature on power-aware computing and communication. The reader is referred to [1,8,9,63–65,77] for comprehensive surveys.

Software techniques for power reduction are supported by a mechanism called *dynamic voltage scaling* [24]. 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 [62]. In a pioneering paper [66], the authors first proposed the approach to energy saving by using fine grain control of CPU speed by an operating system scheduler. In a subsequent work [68], 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 [4–6,12,32,35,47–49,69] and inspired substantial further investigation, much of which focus on real-time applications. In [3,21,23,28,30,34,38,50,52,54,57,58,60,61,67,73–76] and many other related work, the authors addressed the problem of scheduling independent or precedence 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 [7], 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 [11], the author considered scheduling jobs with equal requirements on multiprocessors. In [14], the authors investigated the relationship among parallelization, performance, and energy consumption, and the problem of minimizing energy-delay product. In [20], the authors addressed joint minimization of carbon emission and maximization of profit. In [33,37], the authors attempted joint minimization of energy consumption and task execution time. In [46], the authors studied the problem of scheduling a bag-of-tasks application, i.e., a collection of independent stochastic tasks on a heterogeneous platform with deadline and energy consumption budget constraints. In [59], the authors simultaneously addressed three constraints, i.e., energy, deadline, and reward, for both homogeneous and heterogeneous applications. In [71], the authors devised a novel reliability maximization with energy constraint algorithm, which can effectively balance the tradeoff between high reliability and energy consumption. In [78], the authors considered task scheduling on clusters with significant communication costs. System level power management has been investigated in [16,29,36,51,55]. Other studies were reported in [2,53,72].

In [40–45], we addressed energy and time constrained power allocation and task scheduling on multiprocessors with dynamically variable voltage and frequency and speed and power as combinatorial optimization problems. In [40,43], we studied the problems of scheduling independent sequential tasks. In [41,44], we studied the problems of scheduling independent parallel tasks. In [42], we studied the problems of scheduling precedence constrained sequential tasks. In [45], we studied the problems of scheduling precedence constrained parallel tasks.

1.3. Our contributions

In this paper, we address scheduling independent and precedence constrained parallel tasks on multiple homogeneous processors in a data center with dynamically variable voltage and speed as combinatorial optimization problems. In particular, we consider the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on multiple processors. The first problem has applications

in a parallel computing environment where energy consumption is an important concern. The second problem has applications in a parallel computing environment where system performance is a major requirement. Notice that optimizing both energy consumption and system performance are conflicting objectives. Therefore, our scheduling problems are defined such that the power-performance tradeoff is dealt with by fixing one factor and minimizing the other.

Our scheduling problems contain four nontrivial subproblems, namely, precedence constraining, 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.

- *Precedence Constraining* – Precedence constraints make design and analysis of heuristic algorithms very difficult. We propose to use level-by-level scheduling algorithms to deal with precedence constraints [45]. Since tasks in the same level are independent of each other, they can be scheduled by any of the efficient algorithms previously developed for scheduling independent tasks. Such decomposition of scheduling precedence constrained tasks into scheduling levels of independent tasks makes analysis of level-by-level scheduling algorithms much easier and clearer than analysis of other algorithms.
- *System Partitioning* – Since each parallel task requests for multiple processors which are simultaneously allocated, the available multiple processors in a data center should be partitioned into clusters of processors to be assigned to the tasks. We use a simple system partitioning and processor allocation scheme, which always schedules as many parallel tasks as possible for simultaneous execution.
- *Task Scheduling* – Parallel tasks are scheduled together with system partitioning and precedence constraining, and it is NP-hard even scheduling independent sequential tasks without system partitioning and precedence constraint. We use two heuristic algorithms for scheduling independent parallel tasks in the same level, i.e., SIMPLE and GREEDY developed in [39].
- *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 two-level energy/time/power allocation scheme, namely, optimal energy/time allocation among levels of tasks (Theorems 5 and 6), and equal power supply to tasks in the same level (Theorems 3 and 4).

The above decomposition of our optimization problems into four 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. Such an approach is consistent with traditional scheduling theory.

We notice that energy-efficient scheduling of parallel tasks with precedence constraints has rarely been discussed before analytically [45,56]; most previous studies were on scheduling sequential tasks which require one processor to execute, or independent tasks which have no precedence constraint. Our investigation in this paper continues to make initial attempt to energy-efficient scheduling of parallel tasks with precedence constraints on multiple processors in a data center with dynamic voltage and speed. The approach in this paper results in significant performance improvement as compared with the algorithms in [41] in scheduling independent parallel tasks and the algorithms in [45] in scheduling precedence constrained parallel tasks.

Recent researches have indicated that power consumption of many components in a system such as memories, disks, networks, and so on is also significant. However, in this paper, we primarily focus on processor energy consumption, which can be modeled accurately by available equations. Power and performance management for other components in clouds and data centers supporting parallel computations require additional modeling and deserve separate studies. Other important issues such as heterogeneous processors in a data center are also beyond the scope of this paper and deserve separate papers to study.

The rest of the paper is organized as follows. In Section 2, we provide background information, including the power and task models, definitions of our research problems, and lower bounds for optimal solutions. In Section 3, we present and analyze the class of equal-speed algorithms for scheduling independent parallel tasks. In Section 4, we discuss optimal energy/time allocation among levels of tasks and analyze the performance of our level-by-level scheduling algorithms. In Section 5, we demonstrate simulation data. We conclude the paper in Section 6.

2. Models and problems

In this section, we present background information, including the power and task models, definitions of our problems, and lower bounds for optimal solutions.

2.1. Power and task models

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 [13]. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V \propto f^{\phi}$ for some constant $\phi > 0$ [70]. The processor execution speed s is usually linearly proportional to the clock frequency, namely, $s \propto f$. For ease of discussion, we will assume that $V = bf^{\phi}$ and $s = cf$, where b and c are some constants. Hence, we know that power consumption is $p = aCV^2 f = ab^2 Cf^{2\phi+1} = (ab^2 C/c^{2\phi+1})s^{2\phi+1} = \xi s^{\alpha}$, where $\xi = ab^2 C/c^{2\phi+1}$ and $\alpha = 2\phi + 1$. For instance, by setting $b = 1.16$, $aC = 7.0$, $c = 1.0$, $\phi = 0.5$, $\alpha = 2\phi + 1 = 2.0$, and $\xi = ab^2 C/c^{\alpha} = 9.4192$, the value of p calculated by the equation $p = aCV^2 f = \xi s^{\alpha}$ is reasonably close to that in [31] for the Intel Pentium M processor.

Assume that we are given a parallel computation or application with a set of n precedence constrained parallel tasks. The precedence constraints can be specified as a partial order $\prec$ over the set of tasks $\{1, 2, \ldots, n\}$, or a task graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of tasks and E is a set of arcs representing the precedence constraints. The relationship $i \prec j$, or an arc (i, j) from i to j, means that task i must be executed before task j, i.e., task j cannot be executed until task i is completed. A parallel task i, where $1 \le i \le n$, is specified by π_i and r_i explained below. The integer π_i is the number of processors requested by task i, i.e., 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) due to imbalanced load distribution. Let r_i represent the maximum execution requirement on the π_i processors executing task i. The product $w_i = \pi_i r_i$ is called the *work* of task i.

We are also given m homogeneous and identical processors in a data center. To execute a task i, any π_i of the m processors in the data center can be allocated to task i. Several tasks can be executed simultaneously in the data center, with the restriction that the total number of active processors (i.e., processors allocated to tasks being executed) at any moment cannot exceed m.

In a more general setting, we can consider scheduling u parallel applications represented by task graphs $G_1, G_2, \ldots, G_u$ respectively, on m processors in a data center. Notice that multiple task graphs can be viewed as a single task graph with disconnected components. Therefore, our task model can accommodate multiple parallel applications.

We use p_i to represent the power supplied to task i and s_i to represent the speed to execute task i. It is noticed that the constant ξ in $p_i = \xi s_i^{\alpha}$ only linearly scales the value of p_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 $e_i = \pi_i p_i t_i = \pi_i r_i p_i^{1-1/\alpha} = \pi_i r_i s_i^{\alpha-1} = w_i s_i^{\alpha-1}$, where $w_i = \pi_i r_i$ is the amount of work to be performed for task i.

It is worth to mention that in a real processor, the supply voltage, clock frequency, and execution speed can only take a finite set of discrete values. However, for analytical tractability, these quantities have been treated as continuous variables, as done by most researchers in the literature, ever since the original research in [68]. We will adopt the same approach in this paper, i.e., the execution speed and power consumption of a processor can be continuously tuned.

2.2. Problem definitions

With the above models, our combinatorial optimization problems to be solved in this paper are formally defined as follows.

Given n parallel tasks with precedence constraints $\prec$, 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* $\tilde{E}$ on m identical processors in a data center is to find the power supplies $p_1, p_2, \ldots, p_n$ (equivalently, the task execution speeds $s_1, s_2, \ldots, s_n$) and a non-preemptive schedule of the n tasks on the m processors, such that the schedule length is minimized and that the total energy consumed does not exceed $\tilde{E}$. This problem aims at achieving energy-efficient processing of large-scale parallel applications with the best possible performance.

Given n parallel tasks with precedence constraints $\prec$, 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* $\tilde{T}$ on m identical processors in a data center is to find the power supplies $p_1, p_2, \ldots, p_n$ (equivalently, the task execution speeds $s_1, s_2, \ldots, s_n$) and a non-preemptive schedule of the n tasks on the m processors, such that the total energy consumption is minimized and that the schedule length does not exceed $\tilde{T}$. This problem aims at achieving high-performance processing of large-scale parallel applications with the lowest possible energy consumption.

The above two problems are NP-hard even when the tasks are independent (i.e., $\prec = \emptyset$) and sequential (i.e., $\pi_i = 1$ for all $1 \le i \le n$) [40]. Thus, we will seek fast heuristic algorithms with near-optimal performance.

2.3. Lower bounds

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 parallel tasks. We define T^* to be the length of an optimal schedule, and E^* to be the minimum amount of energy consumed by an optimal schedule.

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,*

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

for the optimal schedule length.

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,*

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

for the minimum energy consumption.

The above lower bound theorems were proved for independent parallel tasks [41], and therefore, are also applicable to precedence constrained parallel tasks. The significance of these lower bounds is that they can be used to evaluate the performance of heuristic algorithms when their solutions are compared with optimal solutions.

3. Equal-speed algorithms

We propose to use the class of *equal-speed* (ES) algorithms for scheduling independent parallel tasks. In ES algorithms, all the tasks are supplied with the same power p and executed with the same speed s.

3.1. Energy-constrained scheduling

To solve the problem of minimizing schedule length with energy consumption constraint $\tilde{E}$, we notice that

$$\tilde{E} = \pi_1 r_1 p^{1-1/\alpha} + \pi_2 r_2 p^{1-1/\alpha} + \cdots + \pi_n r_n p^{1-1/\alpha} = W p^{1-1/\alpha},$$

which gives

$$p = \left(\frac{\tilde{E}}{W}\right)^{\alpha/(\alpha-1)},$$

and

$$s = p^{1/\alpha} = \left(\frac{\tilde{E}}{W}\right)^{1/(\alpha-1)},$$

and

$$t_i = \frac{r_i}{s} = r_i\left(\frac{W}{\tilde{E}}\right)^{1/(\alpha-1)}.$$

The n tasks with execution times $t_1, t_2, \ldots, t_n$ are scheduled by using an algorithm A for scheduling independent parallel tasks. We use ES-A to represent an equal-speed algorithm which uses an algorithm A to schedule the n tasks. For instance, if we use SIMPLE and GREEDY in [39], we have two equal-speed algorithms, namely, ES-SIMPLE and ES-GREEDY.

Given a list of tasks $1, 2, \ldots, n$, algorithm SIMPLE schedules the tasks in the given order as follows. Initially, tasks $1, 2, \ldots, j$ are scheduled for execution, where $\pi = \pi_1 + \pi_2 + \cdots + \pi_j \leq m$, but $\pi_1 + \pi_2 + \cdots + \pi_{j+1} > m$. In other words, we schedule as many tasks as possible for simultaneous execution; however, tasks beyond $j+1$ are not checked even though there may exist small tasks in $j+1, j+2, \ldots, n$, which can be scheduled for execution. Upon the completion of a task i, tasks $j+1, j+2, \ldots, k$ are scheduled for execution, where $\pi - \pi_i + \pi_{j+1} + \pi_{j+2} + \cdots + \pi_k \leq m$, but $\pi - \pi_i + \pi_{j+1} + \pi_{j+2} + \cdots + \pi_{k+1} > m$. Again, we schedule as many tasks as possible for simultaneous execution; but tasks beyond $k+1$ are not checked.

Algorithm GREEDY improves SIMPLE in the sense that initially and upon the completion of a task, each of the remaining tasks not scheduled yet is tested to see whether it can be scheduled for execution (i.e., whether there are enough processors for the task).

Notice that the subproblems of system partitioning and task scheduling may not be solved separately. In fact, in algorithms SIMPLE and GREEDY, the two subproblems are solved simultaneously, i.e., system partitioning is performed together

with task scheduling. Such a combined approach and the equal-speed method result in improved performance compared with our earlier algorithms in [45], where the subproblems of system partitioning and task scheduling are solved separately, and energy/time/power allocation is performed in a more sophisticated way.

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 multiple processors. The following theorem gives the performance ratio when the equal-speed algorithms are used to solve the problem of minimizing schedule length with energy consumption constraint.

Let $A(r_1, r_2, \ldots, r_n)$ represent the length of the schedule produced by algorithm A for n tasks with execution times $r_1, r_2, \ldots, r_n$.

Theorem 3. *By using equal-speed algorithm ES-A to solve the problem of minimizing schedule length with energy consumption constraint on multiple processors, the schedule length is*

$$T = A(r_1, r_2, \ldots, r_n)\left(\frac{W}{\tilde{E}}\right)^{1/(\alpha-1)}.$$

The performance ratio is

$$\beta \leq \frac{A(r_1, r_2, \ldots, r_n)}{W/m}.$$

Proof. We notice that for all $x \geq 0$, we have $A(t_1, t_2, \ldots, t_n) = xA(r_1, r_2, \ldots, r_n)$, if $t_i = xr_i$ for all $1 \leq i \leq n$. Hence, we get

$$T = A(t_1, t_2, \ldots, t_n) = A(r_1, r_2, \ldots, r_n)\left(\frac{W}{\tilde{E}}\right)^{1/(\alpha-1)}.$$

By Theorem 1, we obtain

$$\beta = \frac{T}{T^*} \leq \frac{A(r_1, r_2, \ldots, r_n)}{W/m}.$$

The theorem is proven. □

3.2. Time-constrained scheduling

To solve the problem of minimizing energy consumption with schedule length constraint $\tilde{T}$, we first use algorithm A to schedule n tasks with execution times $t_i = r_i W^{1/(\alpha-1)}$, assuming that only unit amount of energy is consumed, i.e., $E = 1$ and $p = 1/W^{\alpha/(\alpha-1)}$. Let T' be the resulted schedule length. We know that the t_i's and T' can be scaled down or up by a factor of $1/E^{1/(\alpha-1)}$ by consuming energy E. To satisfy the schedule length constraint $\tilde{T}$, we need

$$\frac{T'}{E^{1/(\alpha-1)}} = \tilde{T},$$

that is,

$$E = \left(\frac{T'}{\tilde{T}}\right)^{\alpha-1}.$$

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 multiple processors. The following theorem gives the performance ratio when the equal-speed algorithms are used to solve the problem of minimizing energy consumption with schedule length constraint.

Theorem 4. *By using equal-speed algorithm ES-A to solve the problem of minimizing energy consumption with schedule length constraint on multiple processors, the energy consumed is*

$$E = \left(\frac{A(r_1, r_2, \ldots, r_n)}{\tilde{T}}\right)^{\alpha-1} W.$$

The performance ratio is

$$\beta \leq \left(\frac{A(r_1, r_2, \ldots, r_n)}{W/m}\right)^{\alpha-1}.$$

Proof. We notice that $T' = A(t_1, t_2, \ldots, t_n) = A(r_1, r_2, \ldots, r_n)W^{1/(\alpha-1)}$. Hence, we get

$$E = \left(\frac{T'}{\tilde{T}}\right)^{\alpha-1} = \left(\frac{A(r_1, r_2, \ldots, r_n)W^{1/(\alpha-1)}}{\tilde{T}}\right)^{\alpha-1} = \left(\frac{A(r_1, r_2, \ldots, r_n)}{\tilde{T}}\right)^{\alpha-1} W.$$

By Theorem 2, we obtain

$$\beta = \frac{E}{E^*} \leq \left(\frac{A(r_1, r_2, \ldots, r_n)}{W/m}\right)^{\alpha-1}.$$

The theorem is proven. □

The above discussion also gives

$$\begin{aligned} p &= \left(\frac{E}{W}\right)^{\alpha/(\alpha-1)} \\ &= \left(\frac{T'}{\tilde{T}}\right)^{\alpha} \frac{1}{W^{\alpha/(\alpha-1)}} \\ &= \left(\frac{A(r_1, r_2, \ldots, r_n)W^{1/(\alpha-1)}}{\tilde{T}}\right)^{\alpha} \frac{1}{W^{\alpha/(\alpha-1)}} \\ &= \left(\frac{A(r_1, r_2, \ldots, r_n)}{\tilde{T}}\right)^{\alpha}. \end{aligned}$$

The processor speed s and task execution times t_i can be obtained easily.

4. Level-by-level scheduling algorithms

A set of n parallel tasks with precedence constraints can be represented by a partial order $\prec$ on the tasks, i.e., for two tasks i and j, if $i \prec j$, then task j cannot start its execution until task i finishes. It is clear that the n tasks and the partial order $\prec$ can be represented by a directed task graph, in which, there are n vertices for the n tasks and (i, j) is an arc if and only if $i \prec j$. Furthermore, such a task graph must be a *directed acyclic graph* (dag). An arc (i, j) is redundant if there exists k such that (i, k) and (k, j) are also arcs in the task graph. We assume that there is no redundant arc in the task graph.

A dag can be decomposed into levels, with v being the number of levels. Tasks with no predecessors (called initial tasks) constitute level 1. Generally, a task i is in level l if the number of nodes on the longest path from some initial task to task i is l, where $1 \leq l \leq v$. Note that all tasks in the same level are independent of each other, and hence, they can be scheduled by any of the algorithms (e.g., those from the last section) for scheduling independent tasks. Algorithm LL-ES-A, standing for *level-by-level* scheduling with algorithm ES-A, schedules the n tasks level by level in the order level 1, level 2, ..., level v. Tasks in level $l+1$ cannot start their execution until all tasks in level l are completed. For each level l, where $1 \leq l \leq v$, we use algorithm ES-A to generate its schedule.

It is clear that due to precedence constraints, a processor may be idle between execution of tasks. We assume that a processor consumes no dynamic power when it is idle. A processor still consumes some amount of power even when it is idle, which includes static power dissipation, short circuit power dissipation, and other leakage and wasted power. In [15], the authors considered static power consumption due to leakage current which is expected to increase significantly, and presented leakage-aware scheduling heuristics that determine the best trade-off between three techniques: dynamic voltage scaling, processor shutdown, and finding the optimal number of processors. However, we will mainly focus on dynamic power consumption.

4.1. Energy-constrained scheduling

To use a level-by-level scheduling algorithm to solve the problem of minimizing schedule length with energy consumption constraint $\tilde{E}$, we need to allocate the available energy $\tilde{E}$ to the v levels. We use $E_1, E_2, \ldots, E_v$ to represent an energy allocation to the v levels, where level l consumes energy E_l, and $E_1 + E_2 + \cdots + E_v = \tilde{E}$. Let n_l be the number of tasks in level l, and $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$ be the execution requirements of the n_l tasks in level l, and

$$W_l = \pi_{l,1} r_{l,1} + \pi_{l,2} r_{l,2} + \cdots + \pi_{l,n_l} r_{l,n_l}$$

be the total amount of work of the n_l tasks in level l, where $1 \leq l \leq v$.

Theorem 5 provides optimal energy allocation to the v levels for minimizing schedule length with energy consumption constraint in scheduling precedence constrained tasks by using level-by-level scheduling algorithms LL-ES-A.

Theorem 5. *For a given energy allocation $E_1, E_2, \ldots, E_v$ to the v levels, the level-by-level scheduling algorithm LL-ES-A produces schedule length*

$$T = A_1\left(\frac{W_1}{E_1}\right)^{1/(\alpha-1)} + A_2\left(\frac{W_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + A_v\left(\frac{W_v}{E_v}\right)^{1/(\alpha-1)},$$

where $A_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})$ is the length of the schedule produced by algorithm A for n_l tasks with execution times $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$. The energy allocation $E_1, E_2, \ldots, E_v$ which minimizes T is

$$E_l = \left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right)\tilde{E},$$

for all $1 \le l \le v$, and the minimized schedule length is

$$T = \frac{(A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha/(\alpha-1)}}{\tilde{E}^{1/(\alpha-1)}},$$

by using the above energy allocation.

Proof. From Theorem 3, we know that the schedule length for the n_l tasks in level l is

$$T_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})\left(\frac{W_l}{E_l}\right)^{1/(\alpha-1)} = A_l\left(\frac{W_l}{E_l}\right)^{1/(\alpha-1)},$$

by using algorithm ES-A. Since the level-by-level scheduling algorithm produces schedule length $T = T_1 + T_2 + \cdots + T_v$, we have

$$T = A_1\left(\frac{W_1}{E_1}\right)^{1/(\alpha-1)} + A_2\left(\frac{W_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + A_v\left(\frac{W_v}{E_v}\right)^{1/(\alpha-1)}.$$

To minimize T, we use the Lagrange multiplier system

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

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

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

that is,

$$A_l W_l^{1/(\alpha-1)}\left(-\frac{1}{\alpha-1}\right)\frac{1}{E_l^{1/(\alpha-1)+1}} = \lambda,$$

$1 \le l \le v$, we get

$$E_l = A_l^{1-1/\alpha} W_l^{1/\alpha}\left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

which implies that

$$\tilde{E} = (A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})\left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

and

$$E_l = \left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right)\tilde{E},$$

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

$$
\begin{aligned}
T &= \sum_{l=1}^{v} A_l \left(\frac{W_l}{E_l}\right)^{1/(\alpha-1)} \\
&= \sum_{l=1}^{v} \frac{A_l W_l^{1/(\alpha-1)}}{\left(\left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right)\tilde{E}\right)^{1/(\alpha-1)}} \\
&= \sum_{l=1}^{v} \frac{A_l^{1-1/\alpha} W_l^{1/\alpha} (A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{1/(\alpha-1)}}{\tilde{E}^{1/(\alpha-1)}} \\
&= \frac{(A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha/(\alpha-1)}}{\tilde{E}^{1/(\alpha-1)}}.
\end{aligned}
$$

The theorem is proven. □

Theorem 5 gives the power supply to the tasks in level l as

$$
\left(\frac{E_l}{W_l}\right)^{\alpha/(\alpha-1)} = \left(\left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right)\frac{\tilde{E}}{W_l}\right)^{\alpha/(\alpha-1)},
$$

for all $1 \le l \le v$.

4.2. Time-constrained scheduling

To use a level-by-level scheduling algorithm to solve the problem of minimizing energy consumption with schedule length constraint $\tilde{T}$, we need to allocate the time $\tilde{T}$ to the v levels. We use $T_1, T_2, \ldots, T_v$ to represent a time allocation to the v levels, where tasks in level l are executed within deadline T_l, and $T_1 + T_2 + \cdots + T_v = \tilde{T}$.

Theorem 6 provides optimal time allocation to the v levels for minimizing energy consumption with schedule length constraint in scheduling precedence constrained tasks by using level-by-level scheduling algorithms LL-ES-A.

Theorem 6. *For a given time allocation $T_1, T_2, \ldots, T_v$ to the v levels, the level-by-level scheduling algorithm LL-ES-A consumes energy*

$$
E = \left(\frac{A_1}{T_1}\right)^{\alpha-1} W_1 + \left(\frac{A_2}{T_2}\right)^{\alpha-1} W_2 + \cdots + \left(\frac{A_v}{T_v}\right)^{\alpha-1} W_v,
$$

where $A_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})$ is the length of the schedule produced by algorithm A for n_l tasks with execution times $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$. The time allocation $T_1, T_2, \ldots, T_v$ which minimizes E is

$$
T_l = \left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right)\tilde{T},
$$

for all $1 \le l \le v$, and the minimized energy consumption is

$$
E = \frac{(A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha}}{\tilde{T}^{\alpha-1}},
$$

by using the above time allocation.

Proof. From Theorem 4, we know that the energy consumed by the n_l tasks in level l is

$$
E_l = \left(\frac{A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})}{T_l}\right)^{\alpha-1} W_l = \left(\frac{A_l}{T_l}\right)^{\alpha-1} W_l,
$$

by using algorithm ES-A. Since the level-by-level scheduling algorithm consumes energy $E = E_1 + E_2 + \cdots + E_v$, we have

$$
E = \left(\frac{A_1}{T_1}\right)^{\alpha-1} W_1 + \left(\frac{A_2}{T_2}\right)^{\alpha-1} W_2 + \cdots + \left(\frac{A_v}{T_v}\right)^{\alpha-1} W_v.
$$

To minimize E, we use the Lagrange multiplier system

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

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

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

that is,

$$A_l^{\alpha-1} W_l \left(\frac{1-\alpha}{T_l^{\alpha}}\right) = \lambda,$$

$1 \le l \le v$, we get

$$T_l = A_l^{1-1/\alpha} W_l^{1/\alpha} \left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha},$$

which implies that

$$\tilde{T} = (A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}) \left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha},$$

and

$$T_l = \left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right) \tilde{T},$$

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

$$\begin{aligned} E &= \sum_{l=1}^{v} \left(\frac{A_l}{T_l}\right)^{\alpha-1} W_l \\ &= \sum_{l=1}^{v} \frac{A_l^{\alpha-1} W_l}{\left(\left(\frac{A_l^{1-1/\alpha} W_l^{1/\alpha}}{A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha}}\right) \tilde{T}\right)^{\alpha-1}} \\ &= \sum_{l=1}^{v} \frac{A_l^{1-1/\alpha} W_l^{1/\alpha} (A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha-1}}{\tilde{T}^{\alpha-1}} \\ &= \frac{(A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha}}{\tilde{T}^{\alpha-1}}. \end{aligned}$$

The theorem is proven. □

Theorem 6 gives the power supply to the tasks in level l as

$$\left(\frac{A_l}{T_l}\right)^{\alpha} = \left(\frac{A_l (A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})}{A_l^{1-1/\alpha} W_l^{1/\alpha} \tilde{T}}\right)^{\alpha},$$

for all $1 \le l \le v$.

5. Simulation results

Extensive simulations have been conducted to demonstrate the performance of equal-speed and level-by-level algorithms in scheduling independent and precedence constrained parallel tasks.

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

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

According to Theorem 5, we have

$$\mathrm{NSL} = \frac{(A_1^{1-1/\alpha} W_1^{1/\alpha} + A_2^{1-1/\alpha} W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} W_v^{1/\alpha})^{\alpha/(\alpha-1)}}{\frac{W^{\alpha/(\alpha-1)}}{m}},$$

which, surprisingly, is independent of $\tilde{E}$. NSL is an upper bound for the performance ratio $\beta = T/T^*$ for the problem of minimizing schedule length with energy consumption constraint on multiple processors. 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 $\bar{\beta} \leq \overline{\text{NSL}}$, i.e., the expected performance ratio is no greater 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

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

According to Theorem 6, we have

$$\text{NEC} = \frac{(A_1^{1-1/\alpha}W_1^{1/\alpha} + A_2^{1-1/\alpha}W_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha}W_v^{1/\alpha})^{\alpha}}{\frac{W^{\alpha}}{m^{\alpha-1}}},$$

which, surprisingly, is independent of $\tilde{T}$. NEC is an upper bound for the performance ratio $\beta = E/E^*$ for the problem of minimizing energy consumption with schedule length constraint on multiple processors. For the problem of minimizing energy consumption with schedule length constraint, we have $\bar{\beta} \leq \overline{\text{NEC}}$.

Notice that for a given task graph and a given algorithm ES-A, the expected normalized schedule length $\overline{\text{NSL}}$ and the expected normalized energy consumption $\overline{\text{NEC}}$ are determined by m, α, and the probability distributions of the π_i's and the r_i's. In our simulations, the number of processors is set as $m = 128$; 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 $b \geq 1$.

- Uniform distributions in the range $[1..u]$, i.e., $a_b = 1/u$ for all $1 \leq b \leq u$, 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 $1 \leq b \leq m$, 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 $1 \leq b \leq m$, 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.

The following task graphs are considered in our experiments.

- *Independent Tasks.* The task graph IT(n) for n independent tasks has $l = 1$ level.
- *Tree-Structured Computations.* Many computations are tree-structured, including backtracking search, branch-and-bound computations, game-tree evaluation, functional and logical programming, and various numeric computations. For simplicity, we consider CT(b, h), i.e., complete b-ary trees of height h (see Fig. 1 where $b = 2$ and $h = 4$). It is easy to see that there are $v = h + 1$ levels numbered as $0, 1, 2, \ldots, h$, and $n_l = b^l$ for $0 \leq l \leq h$, and $n = (b^{h+1} - 1)/(b - 1)$.
- *Partitioning Algorithms.* A partitioning algorithm PA(b, h) represents a divide-and-conquer computation with branching factor b and height (i.e., depth of recursion) h (see Fig. 2 where $b = 2$ and $h = 3$). The dag of PA(b, h) has $v = 2h + 1$ levels numbered as $0, 1, 2, \ldots, 2h$. A partitioning algorithm proceeds in three stages. In levels $0, 1, \ldots, h - 1$, each task

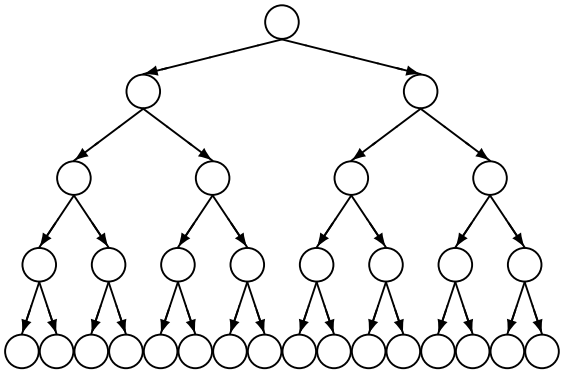

Fig. 1. CT(b, h): a complete binary tree with $b = 2$ and $h = 4$.

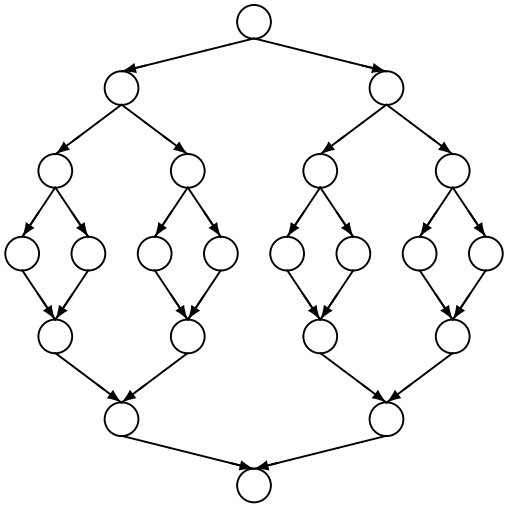

Fig. 2. PA(b, h): a partitioning algorithm with $b = 2$ and $h = 3$.

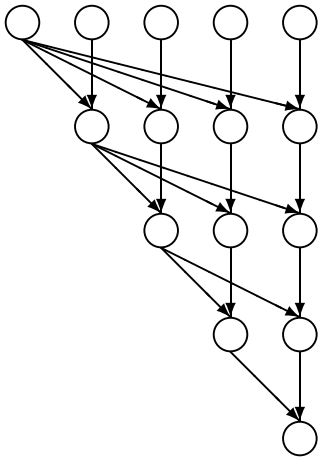

Fig. 3. LA(v): a linear algebra task graph with $v = 5$.

is divided into b subtasks. Then, in level h, subproblems of small sizes are solved directly. Finally, in levels $h + 1$, $h + 2, \ldots, 2h$, solutions to subproblems are combined to form the solution to the original problem. Clearly, $n_l = n_{2h-l} = b^l$, for all $0 \le l \le h - 1$, $n_h = b^h$, and $n = (b^{h+1} + b^h - 2)/(b - 1)$.

- *Linear Algebra Task Graphs.* A linear algebra task graph LA(v) with v levels (see Fig. 3 where $v = 5$) has $n_l = v - l + 1$ for $l = 1, 2, \ldots, v$, and $n = v(v + 1)/2$.

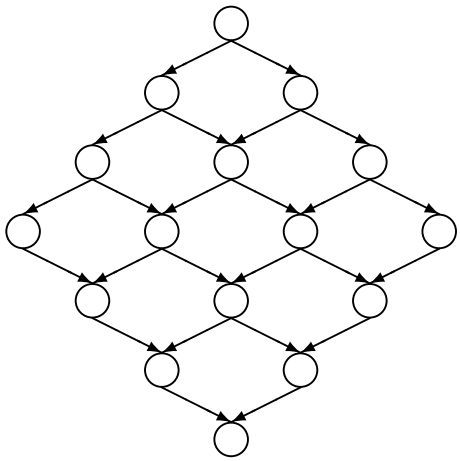

Fig. 4. DD(d): a diamond dag with $d = 4$.

Table 0
(a) Simulation data for expected NSL on IT(1000). (b) Simulation data for expected NEC on IT(1000).

$\bar{\pi}$	Uniform		Binomial		Geometric	
	SIMPLE	GREEDY	SIMPLE	GREEDY	SIMPLE	GREEDY
(a)						
10	1.0612151	1.0168790	1.0521286	1.0147309	1.0901721	1.0247102
20	1.1187971	1.0195745	1.0900334	1.0138343	1.1921851	1.0496759
30	1.1919770	1.0481047	1.1273770	1.0228665	1.2686441	1.0693486
40	1.2973743	1.0540485	1.1523803	1.0535529	1.3106863	1.0852705
50	1.3918785	1.1663052	1.2755402	1.2112754	1.3369496	1.0999473
60	1.3959948	1.1661416	1.1951052	1.0683883	1.3518497	1.1019147
(99% confidence interval ±0.493%)						
(b)						
10	1.1267452	1.0346511	1.1072562	1.0293253	1.1875484	1.0500715
20	1.2502904	1.0399032	1.1869774	1.0281415	1.4180250	1.0988909
30	1.4183074	1.0977834	1.2696579	1.0456720	1.6103616	1.1468834
40	1.6825777	1.1088284	1.3274379	1.1088664	1.7270726	1.1730450
50	1.9342936	1.3561545	1.6278985	1.4653180	1.7873708	1.2040632
60	1.9524615	1.3730117	1.4263815	1.1400377	1.8270598	1.2284763
(99% confidence interval ±0.928%)						

- *Diamond Dags.* A diamond dag DD(d) (see Fig. 4 where $d = 4$) contains $v = 2d - 1$ levels numbered as $1, 2, \ldots, 2d - 1$. It is clear that $n_l = n_{2d-l} = l$, for all $1 \le l \le d - 1$, $n_d = d$, and $n = d^2$.

Since each task graph has at least one parameter, we are actually dealing with classes of task graphs.

Our simulation data are shown in Tables 0–4. For each task graph in {IT(1000), CT(2, 12), PA(2, 12), LA(2000), DD(2000)}, and each algorithm LL-ES-A with $A \in$ {SIMPLE, GREEDY}, and each $\bar{\pi}$ in the range $10, 20, \ldots, 60$, and each probability distribution of task sizes, we generate *rep* sets of tasks, produce their schedules by using algorithm LL-ES-A, calculate their NSL (or NEC), report the average of NSL (or NEC) which is the experimental value of $\overline{\text{NSL}}$ (or $\overline{\text{NEC}}$). The number *rep* is large enough to ensure high quality experimental data. The 99% confidence interval of all the data in the same table is also given.

We have the following observations from our simulations.

- For algorithm LL-ES-SIMPLE, $\overline{\text{NSL}}$ is less than 1.40 and $\overline{\text{NEC}}$ is less than 1.96. Therefore, algorithm LL-ES-SIMPLE produces solutions reasonably close to optimum. The performance of algorithm LL-ES-SIMPLE is close to the performance of the algorithms in [41,45] with slight improvement in some cases and slight degradation in other cases.
- For algorithm LL-ES-GREEDY, $\overline{\text{NSL}}$ is less than 1.17 and $\overline{\text{NEC}}$ is less than 1.38. Therefore, algorithm LL-ES-GREEDY produces solutions very close to optimum. The performance of algorithm LL-ES-GREEDY is significantly better than the performance of the algorithms in [41,45] in all cases.
- The performance of algorithms LL-ES-SIMPLE and LL-ES-GREEDY in scheduling precedence constrained parallel tasks is very close to their performance in scheduling independent parallel tasks.

Table 1
(a) Simulation data for expected NSL on CT(2, 12). (b) Simulation data for expected NEC on CT(2, 12).

$\bar{\pi}$	Uniform		Binomial		Geometric	
	SIMPLE	GREEDY	SIMPLE	GREEDY	SIMPLE	GREEDY
(a)						
10	1.0668090	1.0201838	1.0577947	1.0193555	1.0957142	1.0270283
20	1.1205977	1.0185733	1.0923741	1.0146307	1.1927025	1.0432034
30	1.1929372	1.0419474	1.1290641	1.0198055	1.2706385	1.0606376
40	1.2985809	1.0457828	1.1529326	1.0507253	1.3130905	1.0748126
50	1.3928274	1.1545554	1.2761905	1.2081274	1.3369573	1.0851453
60	1.3956523	1.1603921	1.1958390	1.0688474	1.3506594	1.0923258
(99% confidence interval ±0.460%)						
(b)						
10	1.1383245	1.0416396	1.1179429	1.0386127	1.2023180	1.0533462
20	1.2550407	1.0377974	1.1935845	1.0293171	1.4196389	1.0859413
30	1.4229124	1.0822442	1.2732534	1.0408577	1.6095455	1.1248934
40	1.6878521	1.0892896	1.3312011	1.1039101	1.7231850	1.1525326
50	1.9349816	1.3261586	1.6285768	1.4606897	1.7855972	1.1821897
60	1.9481303	1.3432642	1.4307396	1.1421105	1.8318362	1.2003030
(99% confidence interval ±0.952%)						

Table 2
(a) Simulation data for expected NSL on PA(2, 12). (b) Simulation data for expected NEC on PA(2, 12).

$\bar{\pi}$	Uniform		Binomial		Geometric	
	SIMPLE	GREEDY	SIMPLE	GREEDY	SIMPLE	GREEDY
(a)						
10	1.0716128	1.0254429	1.0622476	1.0245758	1.1004858	1.0315472
20	1.1229537	1.0215426	1.0942444	1.0174901	1.1952533	1.0452092
30	1.1939384	1.0449567	1.1296327	1.0231406	1.2704328	1.0670118
40	1.2999920	1.0501849	1.1536161	1.0530590	1.3143386	1.0790531
50	1.3918059	1.1569191	1.2763570	1.2101363	1.3392503	1.0901296
60	1.3965619	1.1621468	1.1951861	1.0694355	1.3530281	1.0988777
(99% confidence interval ±0.321%)						
(b)						
10	1.1475745	1.0515120	1.1284554	1.0480413	1.2115691	1.0635553
20	1.2609131	1.0450170	1.1973206	1.0350568	1.4274128	1.0942839
30	1.4258802	1.0952942	1.2762781	1.0466918	1.6142826	1.1335437
40	1.6869286	1.1017721	1.3321073	1.1089859	1.7277448	1.1639895
50	1.9360513	1.3409519	1.6291149	1.4642838	1.7905343	1.1878522
60	1.9464194	1.3545830	1.4295539	1.1433611	1.8290595	1.2083101
(99% confidence interval ±0.682%)						

Table 3
(a) Simulation data for expected NSL on LA(2000). (b) Simulation data for expected NEC on LA(2000).

$\bar{\pi}$	Uniform		Binomial		Geometric	
	SIMPLE	GREEDY	SIMPLE	GREEDY	SIMPLE	GREEDY
(a)						
10	1.0612303	1.0159743	1.0519685	1.0140395	1.0906802	1.0238608
20	1.1182512	1.0186872	1.0898940	1.0134069	1.1897128	1.0457268
30	1.1918755	1.0455220	1.1271037	1.0215933	1.2678591	1.0666341
40	1.2978150	1.0485586	1.1524009	1.0525059	1.3123196	1.0822635
50	1.3908910	1.1609637	1.2757914	1.2101833	1.3372127	1.0938019
60	1.3948910	1.1666006	1.1947793	1.0682570	1.3517422	1.1020769
(99% confidence interval ±0.114%)						
(b)						
10	1.1260191	1.0324173	1.1066705	1.0285355	1.1895891	1.0477615
20	1.2503948	1.0377225	1.1876557	1.0270245	1.4163021	1.0933717
30	1.4202830	1.0936211	1.2704740	1.0435191	1.6088775	1.1375873
40	1.6849283	1.1002442	1.3284721	1.1075646	1.7226676	1.1700267
50	1.9350065	1.3458443	1.6276293	1.4645802	1.7878103	1.1961308
60	1.9450557	1.3607152	1.4284098	1.1410491	1.8270078	1.2144811
(99% confidence interval ±0.322%)						

Table 4
(a) Simulation data for expected NSL on DD(2000). (b) Simulation data for expected NEC on DD(2000).

$\bar{\pi}$	Uniform		Binomial		Geometric	
	SIMPLE	GREEDY	SIMPLE	GREEDY	SIMPLE	GREEDY
(a)						
10	1.0610775	1.0160462	1.0519582	1.0141385	1.0907664	1.0238278
20	1.1183243	1.0187398	1.0898882	1.0134304	1.1900367	1.0455208
30	1.1917453	1.0455270	1.1270523	1.0215835	1.2681809	1.0666328
40	1.2976583	1.0484570	1.1524144	1.0524424	1.3125313	1.0824322
50	1.3910242	1.1609358	1.2756748	1.2100847	1.3372368	1.0937529
60	1.3949963	1.1665262	1.1948174	1.0683092	1.3516736	1.1024780
(99% confidence interval ±0.096%)						
(b)						
10	1.1258153	1.0326963	1.1065136	1.0283041	1.1896335	1.0478935
20	1.2504657	1.0378320	1.1878762	1.0269091	1.4165509	1.0932355
30	1.4205594	1.0937435	1.2702350	1.0433919	1.6083204	1.1378991
40	1.6844089	1.1012971	1.3284540	1.1076607	1.7221262	1.1701462
50	1.9349758	1.3470838	1.6275125	1.4641654	1.7877041	1.1957474
60	1.9459369	1.3592738	1.4282224	1.1412452	1.8273985	1.2153072
(99% confidence interval ±0.178%)						

6. Concluding remarks

We have addressed scheduling independent and precedence constrained parallel tasks on multiple homogeneous processors in a data center with dynamically variable voltage and speed as combinatorial optimization problems. We considered the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on multiple processors. We noticed that energy-efficient scheduling of parallel tasks with precedence constraints has rarely been discussed before. Our investigation in this paper continues to make initial attempt to energy-efficient scheduling of parallel tasks with precedence constraints on multiple processors in a data center with dynamic voltage and speed. The approach in this paper results in significant performance improvement as compared with previous algorithms in scheduling independent parallel tasks and precedence constrained parallel tasks.

As mentioned in [42], there are three types of power-aware task scheduling algorithms, depending on the order of power supplying and task scheduling, i.e., *pre-power-determination algorithms* – power supplies are determined before task scheduling; *post-power-determination algorithms* – task scheduling is performed before power supplying; *hybrid algorithms* – task scheduling and power supplying are interleaved. It should be mentioned that the approach in [41,45] belongs to the category of post-power-determination algorithms, while the method in this paper belongs to the category of hybrid algorithms, which has been proven to exhibit significantly better performance than the post-power-determination algorithms in scheduling independent parallel tasks [41] and precedence constrained parallel tasks [45]. As a further research direction, it is worth to consider pre-power-determination algorithms for precedence constrained parallel tasks. Some studies have been conducted for independent parallel tasks [44].

Furthermore, our investigation can be extended to energy and time constrained scheduling of precedence constrained parallel tasks on multiple manycore processors in a cloud computing environment, multiple multiprocessor systems for parallel computing, and multiple clusters in a distributed grid computing environment, with the application of the above mentioned algorithmic techniques.

Acknowledgments

The author would like to thank the two anonymous reviewers for their suggestions to improve the manuscript.

References

[1] S. Albers, Energy-efficient algorithms, Commun. ACM 53 (5) (2010) 86–96.
[2] J. Augustine, S. Irani, C. Swamy, Optimal power-down strategies, SIAM J. Comput. 37 (5) (2008) 1499–1516.
[3] H. Aydin, R. Melhem, D. Mossé, P. Mejía-Alvarez, Power-aware scheduling for periodic real-time tasks, IEEE Trans. Comput. 53 (5) (2004) 584–600.
[4] E. Bampis, C. Dürr, F. Kacem, I. Milis, Speed scaling with power down scheduling for agreeable deadlines, Sustain. Comput. Inform. Syst. 2 (4) (2012) 184–189.
[5] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, J. ACM 54 (1) (2007), Article No. 3.
[6] P. Baptiste, M. Chrobak, C. Dürr, Polynomial-time algorithms for minimum energy scheduling, ACM Trans. Algorithms 8 (3) (2012), Article No. 26.
[7] J.A. Barnett, Dynamic task-level voltage scheduling optimizations, IEEE Trans. Comput. 54 (5) (2005) 508–520.
[8] A. Beloglazov, R. Buyya, Y.C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Adv. Comput. 82 (2011) 47–111.
[9] L. Benini, A. Bogliolo, G. De Micheli, A survey of design techniques for system-level dynamic power management, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 8 (3) (2000) 299–316.
[10] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M.Q. Dang, K. Pentikousis, Energy-efficient cloud computing, Comput. J. 53 (7) (2010) 1045–1051.

[11] D.P. Bunde, Power-aware scheduling for makespan and flow, in: Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures, 2006, pp. 190–196.
[12] H.-L. Chan, W.-T. Chan, T.-W. Lam, L.-K. Lee, K.-S. Mak, P.W.H. Wong, Energy efficient online deadline scheduling, in: Proceedings of the 18th ACM–SIAM Symposium on Discrete Algorithms, 2007, pp. 795–804.
[13] A.P. Chandrakasan, S. Sheng, R.W. Brodersen, Low-power CMOS digital design, IEEE J. Solid-State Circuits 27 (4) (1992) 473–484.
[14] S. Cho, R.G. Melhem, On the interplay of parallelization, program performance, and energy consumption, IEEE Trans. Parallel Distrib. Syst. 21 (3) (2010) 342–353.
[15] P. de Langen, B. Juurlink, Leakage-aware multiprocessor scheduling, J. Signal Process. Syst. 57 (1) (2009) 73–88.
[16] V. Devadas, H. Aydin, On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications, IEEE Trans. Comput. 61 (1) (2012) 31–44.
[17] D. Donofrio, L. Oliker, J. Shalf, M.F. Wehner, C. Rowen, J. Krueger, S. Kamil, M. Mohiyuddin, Energy-efficient computing for extreme-scale science, Computer 42 (11) (2009) 62–71.
[18] W.-c. Feng, K.W. Cameron, The green500 list: encouraging sustainable supercomputing, Computer 40 (12) (2007) 50–55.
[19] V.W. Freeh, D.K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B.L. Rountree, M.E. Femal, Analyzing the energy-time trade-off in high-performance computing applications, IEEE Trans. Parallel Distrib. Syst. 18 (6) (2007) 835–848.
[20] S.K. Garg, C.S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers, J. Parallel Distrib. Comput. 71 (6) (2011) 732–749.
[21] M.E.T. Gerards, Algorithmic power management – energy minimization under real-time constraints, Ph.D. thesis, University of Twente, Netherlands, 2014.
[22] M. Hamdaqa, L. Tahvildari, Cloud Computing Uncovered: A Research Landscape, Adv. Comput., vol. 86, Elsevier Inc., 2012, pp. 41–84.
[23] J.-J. Han, X. Wu, D. Zhu, H. Jin, L.T. Yang, J.-L. Gaudiot, Synchronization-aware energy management for VFI-based multicore real-time systems, IEEE Trans. Comput. 61 (12) (2012) 1682–1696.
[24] http://en.wikipedia.org/wiki/Dynamic_voltage_scaling.
[25] http://www.datacenterdynamics.com/focus/archive/2013/06/facebook-data-centers-energy-use-2012.
[26] http://www.greenbiz.com/blog/2011/07/27/4-reasons-why-cloud-computing-also-green-solution.
[27] http://www.top500.org/.
[28] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M.B. Srivastava, Power optimization of variable-voltage core-based systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 18 (12) (1999) 1702–1714.
[29] F. Hu, J.J. Evans, Power and environment aware control of Beowulf clusters, Clust. Comput. 12 (3) (2009) 299–308.
[30] C. Im, S. Ha, H. Kim, Dynamic voltage scheduling with buffers in low-power multimedia applications, ACM Trans. Embed. Comput. Syst. 3 (4) (2004) 686–705.
[31] Intel, Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor – White Paper, March 2004.
[32] S. Irani, S. Shukla, R. Gupta, Algorithms for power savings, ACM Trans. Algorithms 3 (4) (2007), Article No. 41.
[33] S.U. Khan, I. Ahmad, A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids, IEEE Trans. Parallel Distrib. Syst. 20 (3) (2009) 346–360.
[34] C.M. Krishna, Y.-H. Lee, Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems, IEEE Trans. Comput. 52 (12) (2003) 1586–1593.
[35] W.-C. Kwon, T. Kim, Optimal voltage allocation techniques for dynamically variable voltage processors, ACM Trans. Embed. Comput. Syst. 4 (1) (2005) 211–230.
[36] W.-K. Lee, S.-W. Lee, W.-O. Siew, Hybrid model for dynamic power management, IEEE Trans. Consum. Electron. 55 (2) (2009) 656–664.
[37] Y.C. Lee, A.Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1374–1381.
[38] Y.-H. Lee, C.M. Krishna, Voltage-clock scaling for low energy consumption in fixed-priority real-time systems, Real-Time Syst. 24 (3) (2003) 303–317.
[39] K. Li, An average-case analysis of online non-clairvoyant scheduling of independent parallel tasks, J. Parallel Distrib. Comput. 66 (5) (2006) 617–625.
[40] K. Li, Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1484–1497.
[41] K. Li, Energy efficient scheduling of parallel tasks on multiprocessor computers, J. Supercomput. 60 (2) (2012) 223–247.
[42] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. 61 (12) (2012) 1668–1681.
[43] K. Li, Power allocation and task scheduling on multiprocessor computers with energy and time constraints, in: A.Y. Zomaya, Y.C. Lee (Eds.), Energy-Efficient Distributed Computing Systems, John Wiley & Sons, 2012, pp. 1–37, Chapter 1.
[44] K. Li, Algorithms and analysis of energy-efficient scheduling of parallel tasks, in: I. Ahmad, S. Ranka (Eds.), Handbook of Energy-Aware and Green Computing, vol. 1, CRC Press/Taylor & Francis Group, 2012, pp. 331–360, Chapter 15.
[45] K. Li, Energy-efficient and high-performance processing of large-scale parallel applications in data centers, in: S.U. Khan, A.Y. Zomaya (Eds.), Data Centers, Springer, 2015, pp. 1–33, Chapter 1.
[46] K. Li, X. Tang, K. Li, Energy-efficient stochastic task scheduling on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 2867–2876.
[47] M. Li, B.J. Liu, F.F. Yao, Min-energy voltage allocation for tree-structured tasks, J. Comb. Optim. 11 (2006) 305–319.
[48] M. Li, A.C. Yao, F.F. Yao, Discrete and continuous min-energy schedules for variable voltage processors, Proc. Natl. Acad. Sci. USA 103 (11) (2006) 3983–3987.
[49] M. Li, F.F. Yao, An efficient algorithm for computing optimal discrete voltage schedules, SIAM J. Comput. 35 (3) (2006) 658–671.
[50] J.R. Lorch, A.J. Smith, PACE: a new approach to dynamic voltage scaling, IEEE Trans. Comput. 53 (7) (2004) 856–869.
[51] G. Lovász, F. Niedermeier, H. de Meer, Performance tradeoffs of energy-aware virtual machine consolidation, Clust. Comput. 16 (3) (2013) 481–496.
[52] R.N. Mahapatra, W. Zhao, An energy-efficient slack distribution technique for multimode distributed real-time embedded systems, IEEE Trans. Parallel Distrib. Syst. 16 (7) (2005) 650–662.
[53] J. Mei, K. Li, K. Li, Energy-aware task scheduling in heterogeneous computing environments, Clust. Comput. 17 (2) (2014) 537–550.
[54] B.C. Mochocki, X.S. Hu, G. Quan, A unified approach to variable voltage scheduling for nonideal DVS processors, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 23 (9) (2004) 1370–1377.
[55] V.A. Patil, V. Chaudhary, Rack aware scheduling in HPC data centers: an energy conservation strategy, Clust. Comput. 16 (3) (2013) 559–573.
[56] K. Pruhs, R. van Stee, P. Uthaisombut, Speed scaling of tasks with precedence constraints, in: T. Erlebach, G. Persinao (Eds.), Approximation and Online Algorithms, in: Lecture Notes in Computer Science, vol. 3879, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 307–319.
[57] G. Quan, X.S. Hu, Energy efficient DVS schedule for fixed-priority real-time systems, ACM Trans. Embed. Comput. Syst. 6 (4) (2007), Article No. 29.
[58] N.B. Rizvandi, J. Taheri, A.Y. Zomaya, Some observations on optimal frequency selection in DVFS-based energy consumption minimization, J. Parallel Distrib. Comput. 71 (8) (2011) 1154–1164.

[59] C. Rusu, R. Melhem, D. Mossé, Maximizing rewards for real-time applications with energy constraints, ACM Trans. Embed. Comput. Syst. 2 (4) (2003) 537–559.
[60] D. Shin, J. Kim, Power-aware scheduling of conditional task graphs in real-time multiprocessor systems, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2003, pp. 408–413.
[61] D. Shin, J. Kim, S. Lee, Intra-task voltage scheduling for low-energy hard real-time applications, IEEE Des. Test Comput. 18 (2) (2001) 20–30.
[62] M.R. Stan, K. Skadron, Guest editors' introduction: power-aware computing, IEEE Comput. 36 (12) (2003) 35–38.
[63] O.S. Unsal, I. Koren, System-level power-aware design techniques in real-time systems, Proc. IEEE 91 (7) (2003) 1055–1069.
[64] G.L. Valentini, W. Lassonde, S.U. Khan, N. Min-Allah, S.A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A.Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J.E. Pecero, D. Kliazovich, P. Bouvry, An overview of energy efficiency techniques in cluster computing systems, Clust. Comput. 16 (1) (2013) 3–15.
[65] V. Venkatachalam, M. Franz, Power reduction techniques for microprocessor systems, ACM Comput. Surv. 37 (3) (2005) 195–237.
[66] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, in: Proceedings of the 1st USENIX Symposium on Operating Systems Design and Implementation, 1994, pp. 13–23.
[67] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, R. Lauwereins, Energy-aware runtime scheduling for embedded-multiprocessor SOCs, IEEE Des. Test Comput. 18 (5) (2001) 46–58.
[68] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proceedings of the 36th IEEE Symposium on Foundations of Computer Science, 1995, pp. 374–382.
[69] H.-S. Yun, J. Kim, On energy-optimal voltage scheduling for fixed-priority hard real-time systems, ACM Trans. Embed. Comput. Syst. 2 (3) (2003) 393–430.
[70] B. Zhai, D. Blaauw, D. Sylvester, K. Flautner, Theoretical and practical limits of dynamic voltage scaling, in: Proceedings of the 41st Design Automation Conference, 2004, pp. 868–873.
[71] L. Zhang, K. Li, Y. Xu, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Inf. Sci. 319 (2015) 113–131, http://dx.doi.org/10.1016/j.ins.2015.02.023.
[72] L.M. Zhang, K. Li, D.C.-T. Lo, Y. Zhang, Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds, Sustain. Comput. Inform. Syst. 3 (2) (2013) 109–118.
[73] X. Zhong, C.-Z. Xu, Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee, IEEE Trans. Comput. 56 (3) (2007) 358–372.
[74] D. Zhu, R. Melhem, B.R. Childers, Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems, IEEE Trans. Parallel Distrib. Syst. 14 (7) (2003) 686–700.
[75] D. Zhu, D. Mossé, R. Melhem, Power-aware scheduling for AND/OR graphs in real-time systems, IEEE Trans. Parallel Distrib. Syst. 15 (9) (2004) 849–864.
[76] J. Zhuo, C. Chakrabarti, Energy-efficient dynamic task scheduling algorithms for DVS systems, ACM Trans. Embed. Comput. Syst. 7 (2) (2008), Article No. 17.
[77] S. Zhuravlev, J.C. Saez, S. Blagodurov, A. Fedorova, M. Prieto, Survey of energy-cognizant scheduling techniques, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1447–1464.
[78] Z. Zong, A. Manzanares, X. Ruan, X. Qin, EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters, IEEE Trans. Comput. 60 (3) (2011) 360–374.