Energy-Efficient Task Scheduling on Multiple Heterogeneous Computers: Algorithms, Analysis, and Performance Evaluation

Keqin Li, *Fellow, IEEE*

Abstract—The problems of energy-constrained and time-constrained task scheduling on multiple heterogeneous computers are investigated as combinatorial optimization problems. For a given set of independent tasks, our strategy is to find a schedule of the tasks first, and then find a power allocation to the tasks, where the power allocation is performed in such a way that the total task execution time or the total energy consumption is minimized. We are able to find an optimal partition of a given workload and use a modified list scheduling (MLS) algorithm to generate a partition of the set of tasks that is an approximation of the optimal workload partition. Our simulation results demonstrate that when compared with optimal solutions, the MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of independent tasks on multiple heterogeneous computers. For precedence constrained tasks represented by a directed acyclic graph (dag), our level-by-level modified list scheduling (LL-MLS) algorithm schedules tasks level by level (LL) and schedules tasks in the same level by using the MLS algorithm. We are able to solve the problems of optimal energy/time allocation to the levels and optimal workload partition for all levels in a given dag. Our simulation results demonstrate that when compared with optimal solutions, the LL-MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of precedence constrained tasks on multiple heterogeneous computers.

Index Terms—Energy-constrained scheduling, expected performance bound, modified list scheduling, multiple heterogeneous computers, time-constrained scheduling

1 Introduction

1.1 Motivation

Heterogeneous system architectures (HSA) [1] have attracted growing interest in modern high-performance computing to achieve higher performance/power ratio. In a recent (November 2015) Top500 listing, there are 104 supercomputing systems which adopt HSA. The Sugon Cluster W780I system equipped with Xeon E5-2640v3 8C 2.6 GHz and NVIDIA Tesla K80 (installed in the Institute of Modern Physics of Chinese Academy of Sciences) can achieve the power efficiency of 4.778 Gflops/Watt. In the area of cloud computing, there is a consensus that heterogeneous server architectures will dominate the future data centers [33]. A typical future heterogeneous data center contains specialized servers and accelerators including graphical processing units (GPUs), field-programmable gate arrays (FPGAs), and digital signal processors (DSPs); various storage systems such as network file system (NSF) and Hadoop distributed file system (HDFS); and flexible interconnects such as Gigabit Ethernet and InfiniBand [9]. A heterogeneous architecture is also able to provide low-power and high-throughput cores, along with application-specific accelerators for large-scale applications [13]. Notice that heterogeneity exists not only within a server [8], but also between the servers [28].

It is a challenge on how to effectively and efficiently manage heterogeneous processors in current and future supercomputing systems and cloud computing platforms. The problem becomes more difficult when processors are equipped with the capability of dynamic voltage and frequency scaling. Virtually all existing studies on energy-efficient task scheduling on multiprocessors or multiple computers assume that all the processors or computers have the same power consumption model, i.e., the power consumption is proportional to the execution speed raised to the power α, where α is the same for all processors or computers. However, it is well known that processors from different manufacturers and vendors have very different characteristics of performance (i.e., CPU/core speed) and power consumption [2]. Unfortunately, there has been little study on energy-efficient task scheduling on multiple heterogeneous computers which are characterized by different values of α. The motivation of this paper is to make some initial effort in this direction. Such investigation is certainly of theoretical interest and practical importance, since a large-scale data center may employ different servers manufactured by different vendors with different technologies, which result in different power consumption models.

1.2 Our Contributions

In this paper, we investigate the problems of energy-constrained and time-constrained task scheduling on multiple heterogeneous computers. We define these problems as combinatorial optimization problems. Given a set of tasks

• *The author is with the Department of Computer Science, State University of New York, New Paltz, NY 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 13 Feb. 2016; revised 3 Oct. 2016; accepted 30 Oct. 2016. Date of publication 11 Nov. 2016; date of current version 5 Jan. 2017. Recommended for acceptance by R.G. Melhem.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSUSC.2016.2623775

with an energy budget, the energy-constrained scheduling problem is to find a nonpreemptive schedule of the tasks, such that the total energy consumption does not exceed the given energy budget and that the total task execution time is minimized. Given a set of tasks with a time deadline, the time-constrained scheduling problem is to find a nonpreemptive schedule of the tasks, such that the total task execution time does not exceed the given time deadline and that the total energy consumption is minimized. These problems have been investigated for independent/precedence-constrained and sequential/parallel tasks on homogeneous computers [14], [15], [16], [18], [19], [20].

We find that for independent tasks, both problems contain two subproblems, i.e., power allocation and task scheduling. For a given set of tasks, our strategy is to find a schedule of the tasks first, and then find a power allocation to the tasks, where the power allocation is performed in such a way that the total task execution time or the total energy consumption is minimized (Theorems 1 and 2). Our investigation reveals the fact that both scheduling problems can be reduced to the problem of finding an optimal partition of the given set of tasks into disjoint subsets. Fortunately, we are able to find an optimal partition of a given workload (Theorems 3 and 4), and use a modified list scheduling (MLS) algorithm to generate a partition of the set of tasks that is an approximation of the optimal workload partition. Our simulation results demonstrate that when compared with optimal solutions, the MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of independent tasks on multiple heterogeneous computers.

For precedence constrained tasks represented by a directed acyclic graph (dag), there is an additional subproblem of dealing with task inter-dependency. Our strategy to handle precedence constraints is to schedule tasks level by level (LL) and to schedule tasks in the same level by using the MLS algorithm. Hence, our algorithm is called level-by-level modified list scheduling (LL-MLS) algorithm. The key issue in using the LL scheduling method is to find an optimal energy or time allocation to the levels. Fortunately, we are able to solve the problems of optimal energy/time allocation to the levels and optimal workload partition for all levels in a given dag (Theorems 5 and 6). Our simulation results demonstrate that when compared with optimal solutions, the LL-MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of precedence constrained tasks on multiple heterogeneous computers.

To the best of our knowledge, this is the first paper that investigates energy-efficient task scheduling on multiple heterogeneous computers in a systematic and analytic way.

1.3 Paper Outline

Due to space limitation, this section is moved to the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSUSC.2016.2623775.

2 Related Research

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. There exists an explosive body of literature on energy-efficient computing (see [3], [4], [32], [39] for comprehensive surveys).

2.1 Heterogeneous High-Performance Computing

Energy-efficient high-performance computing on heterogeneous systems has been studied extensively by numerous researchers. In [22], the authors were concerned with the problem of scheduling a bag-of-tasks application, made of a collection of independent stochastic tasks with normal distributions of task execution times, on a heterogeneous platform with deadline and energy consumption budget constraints. In [25], the authors proposed a new energy-aware scheduling algorithm with reduced task duplication, which takes the energy consumption as well as the makespan of an application into consideration. In [29], the authors developed a real-time and energy-efficient resource scheduling and optimization framework to achieve high energy efficiency and low response time in big data stream computing environments. In [30], the authors proposed an energy-efficient workflow task scheduling algorithm in order to obtain more energy reduction as well as maintain the quality of service by meeting the deadlines. In [35], the authors addressed the problem of energy-aware data allocation and task scheduling on a heterogeneous distributed shared-memory multiprocessor system for real-time applications. In [37], the authors devised a novel reliability maximization with energy constraint algorithm to effectively balance the tradeoff between high reliability and low energy consumption. In [38], the authors developed new green task scheduling algorithms for heterogeneous computers with changeable continuous and discrete speeds to reduce energy consumption as much as possible and finish all tasks before a deadline.

2.2 Heterogeneous Cloud Computing

Many researchers have investigated various issues in heterogeneous cloud systems, including resource allocation, task scheduling, performance guarantee, and energy saving. In [11], the authors outlined a principled approach to designing energy-efficient and heterogeneous data centers that are robust against data center workload variations. In [23], the authors emphasized that the operational cost of data centers is dominated by the cost on energy consumption, and modeled a data center as a cyber physical system to capture its thermal properties. In [24], the authors presented a green strategy model for heterogeneous cloud systems, and provided a solution for heterogeneous job-communicating tasks and heterogeneous virtual machines that make up the nodes of the cloud to guarantee the service-level agreement and to optimize energy savings. In [26], the authors had the objective to satisfy performance expectations of customers by heterogeneous dynamic dedicated server scheduling while considering heterogeneous servers and different priority classes of customers. In [27], the authors proposed an optimization strategy based on a mixed integer programming model for achieving improvement on power-efficiency while providing performance guarantee in the virtualized cluster. In [34], the authors studied the multi-resource allocation problem in cloud computing systems where the resource pool is

constructed from a large number of heterogeneous servers, representing different points in the configuration space of resources such as processing, memory, and storage. However, the above research did not include dynamic voltage and frequency scaling into consideration.

Due to space limitation, some part of this section is moved to the supplementary material, available online.

3 Power Allocation

The power consumption model adopted in this paper is the following, which is a standard model used by virtually all researchers in the field [3], [14], [15], [16], [18], [19], [20], [22], [25], [29], [30], [35], [37], [38], [39]. 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 $p = aCV^2f$, 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 $f \propto V^\phi$ with $0 < \phi \leq 1$, which implies that $V \propto f^{1/\phi}$, we know that power consumption is $p \propto f^\alpha$ and $p \propto s^\alpha$, where $\alpha = 1 + 2/\phi \geq 3$. Furthermore, the value of α can be as high as 7.667 for high supply voltage [36]. (Note: Although it has been known that power consumption can be highly dependent on types of applications [10], we assume that p is primarily determined by s, as in the existing literature.)

The main parameter which characterizes the power efficiency of a processor is α. In this paper, we consider task scheduling on m heterogeneous computers specified by $\alpha_1, \alpha_2, \ldots, \alpha_m$. Assume that we are given n independent or precedence constrained sequential tasks to be executed on m heterogeneous processors. Let r_i represent the execution requirement (i.e., the number of CPU cycles or the number of instructions) of task i, where $1 \leq i \leq n$. We use p_i to represent the power allocated to execute task i which is executed on processor k. For ease of discussion, we will assume that p_i is simply $s_i^{\alpha_k}$, where $s_i = p_i^{1/\alpha_k}$ is the execution speed of task i. The execution time of task i is $t_i = r_i/s_i = r_i/p_i^{1/\alpha_k}$. The energy consumed to execute task i is $e_i = p_i t_i = r_i p_i^{1-1/\alpha_k} = r_i s_i^{\alpha_k - 1}$.

3.1 Energy-Constrained Scheduling

In this section, we define and examine the energy-constrained scheduling problem.

Given a set of n tasks with execution requirements $r_1, r_2, \ldots, r_n$, a group of m heterogeneous computers characterized by $\alpha_1, \alpha_2, \ldots, \alpha_m$, and an energy constraint E, the *energy-constrained scheduling* problem is to find a power allocation $p_1, p_2, \ldots, p_n$ to the n tasks and a nonpreemptive schedule of the n tasks on the m computers, such that the total energy consumption of the n tasks does not exceed E and that the total execution time of the n tasks (i.e., the schedule length) is minimized.

A *schedule* of a set of n tasks on m computers is actually a *partition* of the set of n tasks into m disjoint subsets or groups, such that tasks in the kth group are executed on the kth computer. Let R_k denote the kth group as well as the total execution requirement of the tasks in the kth group, where $1 \leq k \leq m$. We use $R = r_1 + r_2 + \cdots + r_n = R_1 + R_2 + \cdots + R_m$ to represent the total execution requirement of all the n tasks. Assume that the given energy budget E is divided into m parts, i.e., $E_1, E_2, \ldots, E_m$, such that E_k is allocated to computer k, where $1 \leq k \leq m$.

The following theorem gives the optimal power allocation and the minimized schedule length for a given schedule.

Theorem 1. *For a given energy constraint E and a given schedule of a set of n tasks on m computers $R_1, R_2, \ldots, R_m$, the minimized schedule length T satisfies*

$$E = \frac{R_1^{\alpha_1}}{T^{\alpha_1 - 1}} + \frac{R_2^{\alpha_2}}{T^{\alpha_2 - 1}} + \cdots + \frac{R_m^{\alpha_m}}{T^{\alpha_m - 1}}.$$

The above minimized schedule length is achieved when $E_k = R_k^{\alpha_k}/T^{\alpha_k - 1}$, and all tasks in R_k are allocated with the same power $p_k = (R_k/T)^{\alpha_k}$ and executed with the same speed $s_k = R_k/T$, for all $1 \leq k \leq m$.

Proof. It is already known in [14] that given energy constraint E_k, the total execution time of the tasks in R_k on computer k is minimized when all tasks in R_k are supplied with the same power $p_k = (E_k/R_k)^{\alpha_k/(\alpha_k - 1)}$ and executed with the same speed $s_k = (E_k/R_k)^{1/(\alpha_k - 1)}$, and the minimized execution time is $T_k = R_k^{\alpha_k/(\alpha_k - 1)}/E_k^{1/(\alpha_k - 1)}$. It is clear that in order to minimize the schedule length T of the n tasks on the m computers, we need to have $T_1 = T_2 = \cdots = T_m = T$, i.e., all the m computers complete their assigned tasks at the same time. Hence, we get $T = R_k^{\alpha_k/(\alpha_k - 1)}/E_k^{1/(\alpha_k - 1)}$, which gives $E_k = R_k^{\alpha_k}/T^{\alpha_k - 1}$, for all $1 \leq k \leq m$. Since $E = E_1 + E_2 + \cdots + E_m$, we have E given in the theorem. It is easy to verify that $p_k = (R_k/T)^{\alpha_k}$ and $s_k = R_k/T$, for all $1 \leq k \leq m$. This proves the theorem. □

The significance of Theorem 1 is that we have reduced the energy-constrained scheduling problem on multiple heterogeneous computers to the problem of finding a schedule $R_1, R_2, \ldots, R_m$, such that T is minimized. This suggests an effective approach to solving the energy-constrained scheduling problem. The method consists of three steps. In the first step, we find $R_1^*, R_2^*, \ldots, R_m^*$, such that T is minimized (see Section 4.1 and Theorem 3). In the second step, we find a partition $R_1, R_2, \ldots, R_m$ of the n tasks into m disjoint groups, such that $R_1, R_2, \ldots, R_m$ are close to $R_1^*, R_2^*, \ldots, R_m^*$ (see Section 5.1). In the third step, we use Theorem 1 to find the optimal power allocation $p_1, p_2, \ldots, p_m$ and speed setting $s_1, s_2, \ldots, s_m$ for the given schedule $R_1, R_2, \ldots, R_m$.

3.2 Time-Constrained Scheduling

In this section, we define and examine the time-constrained scheduling problem.

Given a set of n tasks with execution requirements $r_1, r_2, \ldots, r_n$, a group of m heterogeneous computers characterized by $\alpha_1, \alpha_2, \ldots, \alpha_m$, and a time constraint T, the *time-constrained scheduling* problem is to find a power allocation $p_1, p_2, \ldots, p_n$ to the n tasks and a nonpreemptive schedule of the n tasks on the m computers, such that the total execution

time of the n tasks does not exceed T and that the total energy consumption of the n tasks is minimized.

The following theorem gives the optimal power allocation and the minimized energy consumption for a given schedule.

Theorem 2. *For a given a time constraint T and a given schedule of a set of n tasks on m computers $R_1, R_2, \ldots, R_m$, the minimized energy consumption E is*

$$E = \frac{R_1^{\alpha_1}}{T^{\alpha_1 - 1}} + \frac{R_2^{\alpha_2}}{T^{\alpha_2 - 1}} + \cdots + \frac{R_m^{\alpha_m}}{T^{\alpha_m - 1}}.$$

The above minimized energy consumption is achieved when all tasks in R_k are allocated with the same power $p_k = (R_k/T)^{\alpha_k}$ and executed with the same speed $s_k = R_k/T$, for all $1 \le k \le m$.

Proof. It is already known in [14] that given time constraint T, the total energy consumption of the tasks in R_k on computer k is minimized when all tasks in R_k are supplied with the same power $p_k = (R_k/T)^{\alpha_k}$ and executed with the same speed $s_k = R_k/T$, and the minimized energy consumption is $E_k = R_k^{\alpha_k}/T^{\alpha_k - 1}$, where $1 \le k \le m$. This implies that given a time constraint T and a partition $R_1, R_2, \ldots, R_m$, the minimized energy consumption of the n tasks is $E = E_1 + E_2 + \cdots + E_m$, i.e., the E given in the theorem. This proves the theorem. □

The significance of Theorem 2 is that we have reduced the time-constrained scheduling problem on multiple heterogeneous computers to the problem of finding a schedule $R_1, R_2, \ldots, R_m$, such that E is minimized. This suggests an effective approach to solving the time-constrained scheduling problem. The method consists of three steps. In the first step, we find $R_1^*, R_2^*, \ldots, R_m^*$, such that E is minimized (see Section 4.2 and Theorem 4). In the second step, we find a partition $R_1, R_2, \ldots, R_m$ of the n tasks into m disjoint groups, such that $R_1, R_2, \ldots, R_m$ are close to $R_1^*, R_2^*, \ldots, R_m^*$ (see Section 5.1). In the third step, we use Theorem 2 to find the optimal power allocation $p_1, p_2, \ldots, p_m$ and speed setting $s_1, s_2, \ldots, s_m$ for the given schedule $R_1, R_2, \ldots, R_m$.

4 Lower Bounds

4.1 A Lower Bound for Optimal Schedule Length

The main purpose of this section is to develop a numerical method to find $R_1^*, R_2^*, \ldots, R_m^*$, i.e., an optimal partition of a given workload R, assuming that $R_1, R_2, \ldots, R_m$ are continuous variables, such that T in Theorem 1 is minimized.

From Theorem 1, we know that the minimized schedule length T can be viewed as a function of $R_1, R_2, \ldots, R_m$. We are interested in the following optimization problem (i.e., *optimal workload partition*), namely, given m, $\alpha_1, \alpha_2, \ldots, \alpha_m$, R, and E, finding a partition $R_1, R_2, \ldots, R_m$ of the workload R, such that T is minimized, subject to the constraint that

$$F = R_1 + R_2 + \cdots + R_m = R.$$

Unfortunately, there is no closed-form solution. Our objective is to develop a numerical method to find $R_1^*, R_2^*, \ldots, R_m^*$ and the minimized T.

The significance of the study is two fold. First, the values of $R_1^*, R_2^*, \ldots, R_m^*$ can be used to guide us in finding an optimal schedule to solve the energy-constrained scheduling problem. Second, the minimized T can be used as a lower bound for the optimal schedule length T_{OPT}, such that our solutions can be compared with the optimal solutions.

Our main result of this section is the following theorem.

Theorem 3. *The optimal schedule length has a lower bound $T_{\text{OPT}} \ge T$, where the minimized T and the partition $R_1^*, R_2^*, \ldots, R_m^*$ that results in T can be obtained by solving the $m + 1$ equations, i.e., the constraint*

$$R_1 + R_2 + \cdots + R_m = R,$$

and m equations

$$T = \left(\alpha_k R_k^{\alpha_k - 1} \left(\sum_{j=1}^{m} \frac{R_j}{\alpha_j} \right) \frac{1}{E} \right)^{1/(\alpha_k - 1)},$$

for all $1 \le k \le m$.

Note: The lower bound is implicitly given by equations in the theorem. We will develop a numerical algorithm to solve these equations after the proof. The main idea of the proof is essentially to treat T as a function of $R_1, R_2, \ldots, R_m$. The optimal workload partition problem can be solved by minimizing T using a standard Lagrange multiplier system. The proof can be skipped if the reader is less interested in the details.

Proof. It is clear that the optimal schedule length may not reach the minimized T, since the partition $R_1^*, R_2^*, \ldots, R_m^*$ of R may not be realized by a partition of the n tasks, due to the limited values of $r_1, r_2, \ldots, r_n$. Hence, we have $T_{\text{OPT}} \ge T$.

To solve the optimization problem, we use a Lagrange multiplier system, i.e., $\nabla T = \phi \nabla F$, where ϕ is a Lagrange multiplier. The above equation implies that $\partial T / \partial R_k = \phi \partial F / \partial R_k$, for all $1 \le k \le m$.

To find $\partial T / \partial R_k$, let us recall the equation of T in Theorem 1, namely,

$$E = \frac{R_1^{\alpha_1}}{T^{\alpha_1 - 1}} + \frac{R_2^{\alpha_2}}{T^{\alpha_2 - 1}} + \cdots + \frac{R_m^{\alpha_m}}{T^{\alpha_m - 1}}.$$

We take a partial derivative of R_k on both sides of the above equation and get

$$\frac{\alpha_k R_k^{\alpha_k - 1}}{T^{\alpha_k - 1}} - \left(\sum_{j=1}^{m} \frac{(\alpha_j - 1) R_j^{\alpha_j}}{T^{\alpha_j}} \right) \frac{\partial T}{\partial R_k} = 0,$$

which gives rise to

$$\frac{\partial T}{\partial R_k} = \frac{\alpha_k R_k^{\alpha_k - 1}}{T^{\alpha_k - 1}} \left(\sum_{j=1}^{m} \frac{(\alpha_j - 1) R_j^{\alpha_j}}{T^{\alpha_j}} \right)^{-1},$$

for all $1 \le k \le m$. Since $\partial F / \partial R_k = 1$, we obtain

$$\frac{\alpha_k R_k^{\alpha_k - 1}}{T^{\alpha_k - 1}} \left(\sum_{j=1}^{m} \frac{(\alpha_j - 1) R_j^{\alpha_j}}{T^{\alpha_j}} \right)^{-1} = \phi,$$

for all $1 \le k \le m$.

The last equation can be rewritten as $\alpha_k R_k^{\alpha_k-1}/T^{\alpha_k-1} = \phi S$, where

$$S = \sum_{j=1}^{m} \frac{(\alpha_j - 1)R_j^{\alpha_j}}{T^{\alpha_j}}.$$

Hence, by Theorem 1, we have

$$E_k = \frac{R_k^{\alpha_k}}{T^{\alpha_k-1}} = \phi S \frac{R_k}{\alpha_k},$$

for all $1 \le k \le m$. The above equation implies that

$$E = E_1 + E_2 + \cdots + E_m = \phi S \sum_{j=1}^{m} \frac{R_j}{\alpha_j},$$

and

$$\phi S = E\left(\sum_{j=1}^{m} \frac{R_j}{\alpha_j}\right)^{-1},$$

and

$$E_k = \frac{R_k}{\alpha_k}\left(\sum_{j=1}^{m} \frac{R_j}{\alpha_j}\right)^{-1} E = \left(\frac{R_k/\alpha_k}{R_1/\alpha_1 + R_2/\alpha_2 + \cdots + R_m/\alpha_m}\right)E,$$

for all $1 \le k \le m$.

By Theorem 1, we have $T = \left(R_k^{\alpha_k}/E_k\right)^{1/(\alpha_k-1)}$. Substituting E_k, we get

$$T = \left(\alpha_k R_k^{\alpha_k-1}\left(\sum_{j=1}^{m} \frac{R_j}{\alpha_j}\right)\frac{1}{E}\right)^{1/(\alpha_k-1)},$$

for all $1 \le k \le m$. From the last m equations, plus the condition $R_1 + R_2 + \cdots + R_m = R$, we can find $R_1^*, R_2^*, \ldots, R_m^*$ and the resulting minimized T. □

It is clear that the sophistication of the system of nonlinear equations in Theorem 3 does not accommodate a closed-form solution. In the following, we develop a numerical method to find a solution to the $m + 1$ equations in Theorem 3.

First, we need a method to find $R_1, R_2, \ldots, R_m$ for a fixed T. To this end, we let $W = \sum_{k=1}^{m} R_k/\alpha_k$. Therefore, we obtain

$$R_k = T\left(\frac{E}{\alpha_k W}\right)^{1/(\alpha_k-1)},$$

for all $1 \le k \le m$. It is clear that W can be found by the classic bisection method [5]. The reason is that $R_1, R_2, \ldots, R_m$ all decrease with W. Hence, our numerical method can be described as follows. First, let $W = W_0$ be some arbitrary value. We keep reducing W (e.g., halving W) and increasing $R_1, R_2, \ldots, R_m$, until $W = W_1$ and $W_1 < \sum_{k=1}^{m} R_k/\alpha_k$. Next, we let $W = W_0$ and keep increasing W (e.g., doubling W) and decreasing $R_1, R_2, \ldots, R_m$, until $W = W_2$ and $W_2 > \sum_{k=1}^{m} R_k/\alpha_k$. Finally, we can search W in $[W_1, W_2]$, such that $W = \sum_{k=1}^{m} R_k/\alpha_k$. Once W is determined, $R_1, R_2, \ldots, R_m$ can be calculated in a straightforward manner. The detailed procedure of the above method is given in Algorithm 1 of the supplementary material, available online.

Second, we can also find the minimized T by using the bisection method. It is observed that the values in $R_1, R_2, \ldots, R_m$ all increase with T. Hence, our numerical method can be described as follows. First, let $T = T_0$ be some arbitrary value. We keep reducing T (e.g., halving T) and decreasing $R_1, R_2, \ldots, R_m$, until $R_1 + R_2 + \cdots + R_m < R$. Next, we let $T = T_0$ and keep increasing T (e.g., doubling T) and increasing $R_1, R_2, \ldots, R_m$, until $R_1 + R_2 + \cdots + R_m > R$. Finally, we can search T in $[T_1, T_2]$, such that $R_1 + R_2 + \cdots + R_m = R$. The final values of $R_1, R_2, \ldots, R_m$ are considered as the numerical solution of $R_1^*, R_2^*, \ldots, R_m^*$. The detailed procedure of the above method is given in Algorithm 2 of the supplementary material, available online.

4.2 A Lower Bound for Optimal Energy Consumption

The main purpose of this section is to develop a numerical method to find $R_1^*, R_2^*, \ldots, R_m^*$, i.e., an optimal partition of a given workload R, assuming that $R_1, R_2, \ldots, R_m$ are continuous variables, such that E in Theorem 2 is minimized.

From Theorem 2, we know that the minimized energy consumption E can be viewed as a function of $R_1, R_2, \ldots, R_m$. We are interested in the following optimization problem (i.e., *optimal workload partition*), namely, given m, $\alpha_1, \alpha_2, \ldots, \alpha_m$, R, and T, finding a partition $R_1, R_2, \ldots, R_m$ of the workload R, such that E is minimized, subject to the constraint that

$$F = R_1 + R_2 + \cdots + R_m = R.$$

Again, there is no closed-form solution. Our objective is to develop a numerical method to find $R_1^*, R_2^*, \ldots, R_m^*$ and the minimized E.

The significance of the study is two fold. First, the values of $R_1^*, R_2^*, \ldots, R_m^*$ can be used to guide us in finding an optimal schedule to solve the time-constrained scheduling problem. Second, the minimized E can be used as a lower bound for the optimal energy consumption E_{OPT}, such that our solutions can be compared with the optimal solutions.

Our main result of this section is the following theorem.

Theorem 4. *The optimal energy consumption has a lower bound $E_{\text{OPT}} \ge E$, where the minimized E and the partition $R_1^*, R_2^*, \ldots, R_m^*$ that results in E are*

$$E = T\sum_{k=1}^{m}\left(\frac{\phi}{\alpha_k}\right)^{\alpha_k/(\alpha_k-1)},$$

and

$$R_k^* = \left(\frac{\phi}{\alpha_k}\right)^{1/(\alpha_k-1)} T,$$

for all $1 \le k \le m$, and ϕ satisfies

$$\sum_{k=1}^{m}\left(\frac{\phi}{\alpha_k}\right)^{1/(\alpha_k-1)} = \frac{R}{T}.$$

Note: The lower bound is implicitly given by equations in the theorem. We will develop a numerical algorithm to solve these equations after the proof. The main idea of the proof is similar to that of Theorem 3 and can be skipped.

Proof. It is clear that the optimal energy consumption may not reach the minimized E, since the partition $R_1^*, R_2^*, \ldots, R_m^*$ of R may not be realized by a partition of the n tasks. Hence, we have $E_{\mathrm{OPT}} \geq E$.

To solve the optimization problem, we use a Lagrange multiplier system, i.e., $\nabla E = \phi \nabla F$, where ϕ is a Lagrange multiplier. The above equation implies that $\partial E/\partial R_k = \phi \partial F/\partial R_k$, for all $1 \leq k \leq m$. From Theorem 2, we get $\alpha_k R_k^{\alpha_k - 1}/T^{\alpha_k - 1} = \phi$, and consequently,

$$R_k = \left(\frac{\phi}{\alpha_k}\right)^{1/(\alpha_k - 1)} T,$$

for all $1 \leq k \leq m$. Since $R_1 + R_2 + \cdots + R_m = R$, we get

$$R = T \sum_{k=1}^{m} \left(\frac{\phi}{\alpha_k}\right)^{1/(\alpha_k - 1)}.$$

By Theorem 2, we have

$$E = \sum_{k=1}^{m} \frac{R_k^{\alpha_k}}{T^{\alpha_k - 1}} = T \sum_{k=1}^{m} \left(\frac{\phi}{\alpha_k}\right)^{\alpha_k/(\alpha_k - 1)}.$$

The theorem is proven. □

The equation for ϕ in Theorem 4 can be solved by using the bisection method, where we observe that the left-hand side of the equation is an increasing function of ϕ. Hence, our numerical method can be described as follows. Assume that

$$S = \sum_{k=1}^{m} \left(\frac{\phi}{\alpha_k}\right)^{1/(\alpha_k - 1)}.$$

First, let $\phi = \phi_0$ be some arbitrary value. We keep reducing ϕ (e.g., halving ϕ), until $S < R/T$. Next, we let $\phi = \phi_0$ and keep increasing ϕ (e.g., doubling ϕ), until $S > R/T$. Finally, we can search ϕ in $[\phi_1, \phi_2]$, such that $S = R/T$. Once ϕ is available, we can calculate $R_1^*, R_2^*, \ldots, R_m^*$ and the minimized energy consumption E. The detailed procedure of the above method is given in Algorithm 3 of the supplementary material, available online.

5 Scheduling Independent Tasks

5.1 The MLS Algorithm

We now describe our heuristic algorithms.

Let $R_1^*, R_2^*, \ldots, R_m^*$ denote the partition obtained in Theorem 3 which results in the minimized T. Our strategy to solve the energy-constrained scheduling problem is to find a partition $R_1, R_2, \ldots, R_m$ of the n tasks into m disjoint groups, such that $R_1, R_2, \ldots, R_m$ are close to the optimal workload partition $R_1^*, R_2^*, \ldots, R_m^*$.

Let $R_1^*, R_2^*, \ldots, R_m^*$ denote the partition obtained in Theorem 4 which results in the minimized E. Our strategy to solve the time-constrained scheduling problem is to find a partition $R_1, R_2, \ldots, R_m$ of the n tasks into m disjoint groups, such that $R_1, R_2, \ldots, R_m$ are close to the optimal workload partition $R_1^*, R_2^*, \ldots, R_m^*$.

For both energy-constrained scheduling and time-constrained scheduling, we are facing the same problem, namely, given $R_1^*, R_2^*, \ldots, R_m^*$, finding a partition of the n tasks into m disjoint groups, such that $R_1, R_2, \ldots, R_m$ are close to $R_1^*, R_2^*, \ldots, R_m^*$. Once a schedule $R_1, R_2, \ldots, R_m$ is determined, we can use Theorem 1 to find an optimal power allocation for energy-constrained scheduling, or Theorem 2 to find an optimal power allocation for time-constrained scheduling.

To generate a partition of the n tasks into m disjoint groups, we treat the task execution requirements $r_1, r_2, \ldots, r_n$ as task execution times. (Note: They are not the actual task execution times, but only the input to the MLS algorithm for the purpose of generating a partition.) The n tasks are assigned to the m computers by using the classic list scheduling algorithm [12] with some modifications. Our *modified list scheduling* algorithm works as follows.

- Initially, the n tasks are arranged into a list. The first m tasks are assigned to the m computers, one task per computer.
- Whenever a computer completes a task, the next task in the list is assigned to the computer.
- As soon as the total task execution requirement of a computer k exceeds R_k^*, the computer is blocked and does not accept more tasks.

The result of the above procedure is a task assignment to the m computers, i.e., a schedule of the n tasks on the m computers. The resulted partition $R_1, R_2, \ldots, R_m$ is considered as an approximation of $R_1^*, R_2^*, \ldots, R_m^*$.

5.2 Numerical Examples

In this section, we give examples of our numerical calculations.

In Table 1, we demonstrate numerical data of energy-constrained scheduling of independent tasks. We consider $m = 7$ heterogeneous computers with $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7) = (3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0)$. (Note: As commonly accepted, the value of α is at least 3 [7]. As mentioned earlier, the value of α can be as high as 7.667 for high supply voltage [36].) Let us assume that the total amount of task execution requirement is set as $R = 100$. For energy constraint $E = 200, 400, 600, 800, 1{,}000$, we show R_k^*, E_k, p_k, and s_k for all $1 \leq k \leq m$. (Note: The values of R and E are set in such a way that the execution speeds are in a reasonable range.) These data are calculated by using Theorem 3. The minimized T is also given. We observe that due to the heterogeneity of the α_k's, the workload R and the energy E are unevenly distributed among the computers, such that a computer with a smaller α_k is assigned more workload and allocated more energy. Furthermore, tasks assigned to a computer with a smaller α_k are supplied with more power and executed with faster speed. In fact, as E increases, the ratios R_1^*/R_m^*, E_1/E_m, p_1/p_m, and s_1/s_m all increase.

In Table 2, we demonstrate numerical data of time-constrained scheduling of independent tasks. The values of m, R, and the α_k's are the same as Table 1. For time constraint $T = 12.0, 10.5, 9.0, 7.5, 6.0$, we show R_k^*, E_k, p_k, and s_k for all $1 \leq k \leq m$. These data are calculated by using Theorem 4. The minimized E is also given. It is observed that as T decreases, the ratios R_1^*/R_m^*, E_1/E_m, p_1/p_m, and s_1/s_m all increase.

5.3 Performance Data

We now present simulation data to demonstrate the performance of our heuristic algorithms on $m = 7$ heterogeneous computers with the same α_k's in Section 5.2.

TABLE 1
Numerical Data of Energy-Constrained Scheduling of Independent Tasks

E	k	R_k^*	E_k	p_k	s_k	T
	1	18.79233	51.62622	4.55339	1.65747	
	2	15.97030	37.60589	3.31681	1.40857	
	3	14.42734	29.72604	2.62181	1.27248	
200	4	13.47787	24.68423	2.17713	1.18874	11.33798
	5	12.84683	21.17566	1.86767	1.13308	
	6	12.40420	18.58733	1.63939	1.09404	
	7	12.08114	16.59463	1.46363	1.06555	
	1	21.05451	113.76871	12.56077	2.32455	
	2	16.72241	77.45145	8.55112	1.84626	
	3	14.44063	58.52278	6.46128	1.59434	
400	4	13.06264	47.05626	5.19530	1.44220	9.05746
	5	12.15384	39.40420	4.35047	1.34186	
	6	11.51663	33.94390	3.74762	1.27151	
	7	11.04933	29.85270	3.29592	1.21991	
	1	22.44906	180.17983	22.73851	2.83305	
	2	17.13833	117.90436	14.87941	2.16284	
	3	14.41453	86.77005	10.95029	1.81910	
600	4	12.79567	68.46679	8.64044	1.61480	7.92399
	5	11.73840	56.52860	7.13385	1.48137	
	6	11.00140	48.16310	6.07813	1.38837	
	7	10.46260	41.98728	5.29875	1.32037	
	1	23.46903	249.41050	34.64410	3.25994	
	2	17.42106	158.68917	22.04255	2.41985	
	3	14.38067	114.61985	15.92115	1.99753	
800	4	12.59611	89.24105	12.39593	1.74965	7.19922
	5	11.44006	72.94557	10.13243	1.58907	
	6	10.63851	61.66783	8.56590	1.47773	
	7	10.05457	53.42603	7.42109	1.39662	
	1	24.27702	320.75766	48.02557	3.63489	
	2	17.63269	199.68864	29.89846	2.64006	
	3	14.34560	142.15488	21.28420	2.14790	
1,000	4	12.43579	109.53770	16.40058	1.86195	6.67889
	5	11.20696	88.84248	13.30198	1.67797	
	6	10.35890	74.65411	11.17762	1.55099	
	7	9.74305	64.36452	9.63700	1.45878	

TABLE 2
Numerical Data of Time-Constrained Scheduling of Independent Tasks

T	k	R_k^*	E_k	p_k	s_k	E
	1	18.23823	42.12946	3.51079	1.51985	
	2	15.77044	31.22483	2.60207	1.31420	
	3	14.41239	24.96896	2.08075	1.20103	
12.0	4	13.57551	20.90585	1.74215	1.13129	167.45616
	5	13.02027	18.04572	1.50381	1.08502	
	6	12.63234	15.91642	1.32637	1.05270	
	7	12.35082	14.26491	1.18874	1.02924	
	1	19.55369	67.81211	6.45830	1.86226	
	2	16.23462	48.25851	4.59605	1.54615	
	3	14.44007	37.55858	3.57701	1.37524	
10.5	4	13.34092	30.84417	2.93754	1.27056	253.92469
	5	12.61092	26.24078	2.49912	1.20104	
	6	12.09786	22.88474	2.17950	1.15218	
	7	11.72193	20.32580	1.93579	1.11637	
	1	21.12014	116.30682	12.92298	2.34668	
	2	16.74276	79.02943	8.78105	1.86031	
	3	14.43995	59.63970	6.62663	1.60444	
9.0	4	13.05025	47.91109	5.32345	1.45003	407.84453
	5	12.13410	40.09287	4.45476	1.34823	
	6	11.49187	34.51896	3.83544	1.27687	
	7	11.02093	30.34566	3.37174	1.22455	
	1	23.03194	217.20454	28.96061	3.07093	
	2	17.30206	139.85872	18.64783	2.30694	
	3	14.39664	101.82651	13.57687	1.91955	
7.5	4	12.68204	79.73265	10.63102	1.69094	707.81618
	5	11.56734	65.45207	8.72694	1.54231	
	6	10.79263	55.51678	7.40224	1.43902	
	7	10.22735	48.22491	6.42999	1.36365	
	1	25.44568	457.65669	76.27611	4.24095	
	2	17.92020	276.26237	46.04373	2.98670	
	3	14.28281	192.66399	32.11066	2.38047	
6.0	4	12.20084	146.29319	24.38220	2.03347	1,371.72046
	5	10.87478	117.35388	19.55898	1.81246	
	6	9.96611	97.77094	16.29516	1.66102	
	7	9.30959	83.71940	13.95323	1.55160	

The solutions produced by our algorithms will be compared with optimal solutions (actually, the lower bounds in Theorems 3 and 4). Our performance measures are defined as follows.

Let A be a heuristic algorithm for the energy-constrained scheduling problem. We use T_A to denote the length of the schedule produced by algorithm A. Then, the ratio T_A/T_{OPT} is called the *performance ratio* of algorithm A. If $T_A/T_{\mathrm{OPT}} \leq B$, then B is called a *performance bound*. Let T^* denote the minimized T in Theorem 3. It is clear that $B = T_A/T^*$ is a performance bound. Furthermore, if the task execution requirements are random variables, T_A, T^*, and B are all random variables. The expectation $\overline{B}$ is called the *expected performance bound*.

Let A be a heuristic algorithm for the time-constrained scheduling problem. We use E_A to denote the amount of energy consumed by algorithm A. Then, the ratio E_A/E_{OPT} is called the *performance ratio* of algorithm A. If $E_A/E_{\mathrm{OPT}} \leq C$, then C is called a *performance bound*. Let E^* denote the minimized E in Theorem 4. It is clear that $C = E_A/E^*$ is a performance bound. Furthermore, if the task execution requirements are random variables, E_A, E^*, and C are all random variables. The expectation $\overline{C}$ is called the *expected performance bound*.

In Table 3, we show simulation data for the expected performance bound $\overline{B}$ of the modified list scheduling algorithm for the energy-constrained scheduling problem. For each combination of $n = 50, 100, 150, \ldots, 500$ and $E = 100, 200, 300, 400, 500$, we generate 2,000 sets of n tasks, whose task execution requirements are random numbers uniformly

TABLE 3
Simulation Data of $\overline{B}$ for Independent Tasks (CI$= \pm 0.134\%$)

n	$E = 100$	$E = 200$	$E = 300$	$E = 400$	$E = 500$
50	1.04361	1.04947	1.05253	1.05580	1.05689
100	1.01326	1.01510	1.01600	1.01656	1.01668
150	1.00625	1.00765	1.00770	1.00780	1.00770
200	1.00366	1.00452	1.00452	1.00450	1.00441
250	1.00223	1.00296	1.00295	1.00287	1.00293
300	1.00150	1.00215	1.00213	1.00209	1.00202
350	1.00098	1.00155	1.00159	1.00153	1.00152
400	1.00046	1.00123	1.00122	1.00120	1.00119
450	1.00025	1.00097	1.00098	1.00095	1.00094
500	1.00038	1.00080	1.00079	1.00077	1.00077

TABLE 4
Simulation Data of $\overline{C}$ for Independent Tasks (CI$= \pm 0.441\%$)

n	$T = 4.0t$	$T = 3.5t$	$T = 3.0t$	$T = 2.5t$	$T = 2.0t$
50	1.04858	1.07863	1.11797	1.15142	1.18113
100	1.01890	1.03379	1.04336	1.04929	1.05416
150	1.01150	1.01874	1.02255	1.02490	1.02500
200	1.00829	1.01210	1.01428	1.01494	1.01410
250	1.00623	1.00871	1.00967	1.00950	1.00890
300	1.00489	1.00651	1.00680	1.00640	1.00624
350	1.00392	1.00503	1.00510	1.00482	1.00464
400	1.00334	1.00401	1.00402	1.00376	1.00359
450	1.00276	1.00333	1.00313	1.00289	1.00274
500	1.00238	1.00263	1.00251	1.00241	1.00223

distributed in $[0, 1]$. For each set of tasks, we calculate the lower bound T^* of $T_{\rm OPT}$ by using Theorem 3. We also simulate the MLS algorithm and find its schedule length $T_{\rm MLS}$. The performance bound is obtained as $B = T_{\rm MLS}/T^*$. The average value of the 2,000 values of B is then reported in Table 3. The 99 percent confidence interval of all the data in Table 3 is no more than ± 0.134 percent.

In Table 4, we show simulation data for the expected performance bound $\overline{C}$ of the modified list scheduling algorithm for the time-constrained scheduling problem. For each combination of $n = 50, 100, 150, \ldots, 500$ and $T = 4.0t, 3.5t, 3.0t, 2.5t, 2.0t$, where $t = n/50$, we generate 2,000 sets of n tasks, whose task execution requirements are random numbers uniformly distributed in $[0, 1]$. For each set of tasks, we calculate the lower bound E^* of $E_{\rm OPT}$ by using Theorem 4. We also simulate the MLS algorithm and find its energy consumption $E_{\rm MLS}$. The performance bound is obtained as $C = E_{\rm MLS}/E^*$. The average value of the 2,000 values of C is then reported in Table 4. The 99 percent confidence interval of all the data in Table 4 is no more than ± 0.441 percent.

In the following, we point out some observations.

From Table 3, we observe that $\overline{B}$ is very close to 1. For a given n, the expected performance bound $\overline{B}$ slightly increases as E increases, and then decreases as E further increases, primarily due to the lower bounds. For a given E, the expected performance bound $\overline{B}$ decreases as n increases.

From Table 4, we observe that $\overline{C}$ is very close to 1. For a given n, the expected performance bound $\overline{C}$ slightly increases as T decreases, and then decreases as T further decreases, primarily due to the lower bounds. For a given T, the expected performance bound $\overline{C}$ decreases as n increases.

Notice that all the data in Tables 3 and 4 are for expected performance bounds. The actual values of expected performance ratios are smaller and closer to the optimal. Our simulation results demonstrate that the MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of independent tasks on multiple heterogeneous computers.

6 Scheduling Precedence Constrained Tasks

6.1 The LL-MLS Algorithm

A set of precedence constrained tasks can be represented by a *directed acyclic graph*. The nodes in a dag are tasks and arcs in the dag are precedence constraints. A dag can be divided into v levels. Tasks with no any predecessor are in level 1. In general, a task is in level l if the longest path from a node in level 1 to the task contains l nodes. Let R_l denote the total execution requirement of tasks in level l, where $1 \le l \le v$.

Our strategy to schedule precedence constrained tasks is to schedule tasks in a dag *level by level*, that is, tasks in level l cannot be executed until all tasks in level $l-1$ are finished. Since tasks in each level l are independent of each other, they can be scheduled by using the MLS algorithm. Therefore, our algorithm is called *level-by-level modified list scheduling* algorithm.

Let E_l denote the amount of energy allocated to tasks in level l (or, the amount of energy consumed by tasks in level l), where $1 \le l \le v$, such that $E_1 + E_2 + \cdots + E_v = E$. Let T_l denote the execution time allocated to tasks in level l (or, the total execution time of tasks in level l), where $1 \le l \le v$, such that $T_1 + T_2 + \cdots + T_v = T$.

Our main concern for energy-constrained scheduling is the determination of $E_1, E_2, \ldots, E_v$, such that the total execution time of all tasks in a dag, i.e., $T = T_1 + T_2 + \cdots + T_v$, is minimized (see Section 6.2 and Theorem 5). Once an optimal energy allocation $E_1, E_2, \ldots, E_v$ is available, we can use the MLS algorithm in Section 5.1 to schedule the v levels separately and sequentially. Our main concern for time-constrained scheduling is the determination of $T_1, T_2, \ldots, T_v$, such that the total energy consumption of all tasks in a dag, i.e., $E = E_1 + E_2 + \cdots + E_v$, is minimized (see Section 6.3 and Theorem 6). Once an optimal time allocation $T_1, T_2, \ldots, T_v$ is available, we can use the MLS algorithm in Section 5.1 to schedule the v levels separately and sequentially. We will address these two issues in the next two sections.

6.2 Optimal Energy Allocation

According to the level-by-level scheduling method, the total execution time T of all tasks in a dag is simply the summation of the T_l's, i.e.,

$$T = T_1 + T_2 + \cdots + T_v,$$

where T_l is viewed as a function of E_l, and T is viewed as a function of $E_1, E_2, \ldots, E_v$. Hence, we can define the following optimization problem (i.e., *optimal energy allocation and workload partition*), namely, given m, $\alpha_1, \alpha_2, \ldots, \alpha_m$, R_1, $R_2, \ldots, R_v$, and E, finding $E_1, E_2, \ldots, E_v$, such that T is minimized, subject to the constraint that

$$F = E_1 + E_2 + \cdots + E_v = E.$$

Notice that each R_l should be further divided into $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ by using Theorem 3, such that T_l is minimized for a given E_l, where $1 \le l \le v$.

Note: The main purpose of the following mathematical derivation is to develop a numerical procedure to solve the above optimization problem. The reader can skip this part and go to Table 5.

To solve the above optimization problem, we use a Lagrange multiplier system, i.e., $\nabla T = \phi \nabla F$, where ϕ is a Lagrange multiplier. The above equation implies that

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

or, equivalently, $\partial T_l / \partial E_l = \phi$, for all $1 \le l \le v$. In the following, we develop a numerical procedure to solve the optimization problem.

TABLE 5
Numerical Data of Energy-Constrained Scheduling of Precedence Constrained Tasks

l	E_l	k	$R^*_{l,k}$	$E_{l,k}$	$p_{l,k}$	$s_{l,k}$	T_l
		1	1.87923	5.16262	4.55339	1.65747	
		2	1.59703	3.76059	3.31681	1.40857	
		3	1.44273	2.97260	2.62181	1.27248	
1	20.00000	4	1.34779	2.46842	2.17713	1.18874	1.13380
		5	1.28468	2.11757	1.86767	1.13308	
		6	1.24042	1.85873	1.63939	1.09404	
		7	1.20811	1.65946	1.46363	1.06555	
		1	3.75847	10.32524	4.55339	1.65747	
		2	3.19406	7.52118	3.31681	1.40857	
		3	2.88547	5.94521	2.62181	1.27248	
2	40.00000	4	2.69557	4.93685	2.17713	1.18874	2.26760
		5	2.56937	4.23513	1.86767	1.13308	
		6	2.48084	3.71747	1.63939	1.09404	
		7	2.41623	3.31893	1.46363	1.06555	
		1	5.63770	15.48787	4.55339	1.65747	
		2	4.79109	11.28177	3.31681	1.40857	
		3	4.32820	8.91781	2.62181	1.27248	
3	60.00000	4	4.04336	7.40527	2.17713	1.18874	3.40140
		5	3.85405	6.35270	1.86767	1.13308	
		6	3.72126	5.57620	1.63939	1.09404	
		7	3.62434	4.97839	1.46363	1.06555	
		1	7.51693	20.65049	4.55339	1.65747	
		2	6.38812	15.04236	3.31681	1.40857	
		3	5.77093	11.89042	2.62181	1.27248	
4	80.00000	4	5.39115	9.87369	2.17713	1.18874	4.53519
		5	5.13873	8.47026	1.86767	1.13308	
		6	4.96168	7.43493	1.63939	1.09404	
		7	4.83245	6.63785	1.46363	1.06555	

To find $\partial T_l/\partial E_l$, we recall from Theorem 3 that for a given E_l, there is an optimal partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ which results in the minimized T_l. The values of $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ and T_l can be obtained by solving the $m+1$ equations, i.e., the constraint

$$R_{l,1} + R_{l,2} + \cdots + R_{l,m} = R_l,$$

and m equations

$$T_l = \left(\alpha_k R_{l,k}^{\alpha_k - 1}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right)\frac{1}{E_l}\right)^{1/(\alpha_k-1)},$$

for all $1 \le k \le m$. From Table 1, we know that T_l as well as $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ are all functions of E_l. We take a partial derivative of E_l on both sides of the above equation and get

$$\begin{aligned}\frac{\partial T_l}{\partial E_l} &= \frac{1}{\alpha_k - 1}\cdot\frac{1}{T_l^{\alpha_k-2}}\\ &\cdot \alpha_k\left((\alpha_k-1)R_{l,k}^{\alpha_k-2}R'_{l,k}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right)\frac{1}{E_l}\right.\\ &\left.+R_{l,k}^{\alpha_k-1}\left(\sum_{j=1}^{m}\frac{R'_{l,j}}{\alpha_j}\right)\frac{1}{E_l} - R_{l,k}^{\alpha_k-1}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right)\frac{1}{E_l^2}\right),\end{aligned}$$

where $R'_{l,k} = \partial R_{l,k}/\partial E_l$, for all $1 \le k \le m$. The last equation can be rewritten as

$$\begin{aligned}&(\alpha_k-1)R'_{l,k}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right) + R_{l,k}\left(\sum_{j=1}^{m}\frac{R'_{l,j}}{\alpha_j}\right)\\ &= \phi T_l^{\alpha_k-2}\left(1-\frac{1}{\alpha_k}\right)\frac{E_l}{R_{l,k}^{\alpha_k-2}} + R_{l,k}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right)\frac{1}{E_l},\end{aligned}$$

for all $1 \le k \le m$.

For a given ϕ, we need to find E_l, where $1 \le l \le v$. For a given E_l, the values of $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ and T_l can be obtained from Theorem 3. The values of $R'_{l,1}, R'_{l,2}, \ldots, R'_{l,m}$ can be obtained by solving the above m equations, which form a system of linear equations,

$$\begin{aligned}&\sum_{j\ne k}\left(\frac{R_{l,k}}{\alpha_j}\right)R'_{l,j} + \left((\alpha_k-1)\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right) + \frac{R_{l,k}}{\alpha_k}\right)R'_{l,k}\\ &= \phi T_l^{\alpha_k-2}\left(1-\frac{1}{\alpha_k}\right)\frac{E_l}{R_{l,k}^{\alpha_k-2}} + R_{l,k}\left(\sum_{j=1}^{m}\frac{R_{l,j}}{\alpha_j}\right)\frac{1}{E_l},\end{aligned}$$

for all $1 \le k \le m$. The detailed procedure of the above method is given in Algorithm 4 of the supplementary material, available online. Hence, we only need to determine E_l, such that $R'_{l,1} + R'_{l,2} + \cdots + R'_{l,m} = 0$. It is noticed that $R'_{l,1} + R'_{l,2} + \cdots + R'_{l,m}$ is a decreasing function of E_l, and the value of E_l can be found by using the bisection method. The detailed procedure of the above method is given in Algorithm 5 of the supplementary material, available online.

Finally, to find $E_1, E_2, \ldots, E_v$, we notice that all the E_l's as well as $E_1 + E_2 + \cdots + E_v$ are increasing functions of ϕ. Therefore, we only need to determine ϕ, such that $E_1 + E_2 + \cdots + E_v = E$, by using the bisection method. Once $E_1, E_2, \ldots, E_v$ are available, we can calculate $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ and T_l, for all $1 \le l \le v$, by using Theorem 3, and T as a direct consequence. The detailed procedure of the above method is given in Algorithm 6 of the supplementary material, available online.

In Table 5, we demonstrate numerical data of energy-constrained scheduling of precedence constrained tasks. We consider $m = 7$ heterogeneous computers with the same α_k's in Section 5.2. Assume that a dag has $v = 4$ levels, with $R_l = 10l$, where $1 \le l \le v$, and $R = 100$. The energy constraint is $E = 200$. For all $1 \le l \le v$, we show E_l and T_l, as well as $R^*_{l,k}$, $E_{l,k}$, $p_{l,k}$, and $s_{l,k}$, for all $1 \le k \le m$. The following facts are observed, which are described by surprisingly simple relations, and are not obvious at all from the above derivation.

Theorem 5. *Assume that (1) $R^*_1, R^*_2, \ldots, R^*_m$ is an optimal partition of a workload R of independent tasks obtained from Theorem 3; (2) $E_1, E_2, \ldots, E_m$ is the corresponding optimal energy allocation of a given energy constraint E; (3) T is the resulting minimized schedule length; (4) $p_1, p_2, \ldots, p_m$ are the power settings of the m computers; (5) $s_1, s_2, \ldots, s_m$ are the speed settings of the m computers. For the optimal energy allocation and workload partition problem with $R_1, R_2, \ldots, R_v$ and E, where $R_1 + R_2 + \cdots + R_v = R$, we have the following facts.*

1) *E_l, $R^*_{l,k}$, $E_{l,k}$, and T_l are linearly proportional to R_l, i.e., they can be obtained from E, R^*_k, E_k, and T by scaling a factor of $R_l/(R_1 + R_2 + \cdots + R_v)$, for all $1 \le l \le v$.*
2) *$p_{1,k} = p_{2,k} = \cdots = p_{v,k} = p_k$.*
3) *$s_{1,k} = s_{2,k} = \cdots = s_{v,k} = s_k$.*

Proof. Notice that the minimized T' of the optimal energy allocation and workload partition problem cannot be shorter than the minimized T of the workload partition problem of the same set of tasks with all precedence constraints removed (i.e., tasks become independent of each other). The minimized T of the set of independent tasks is given by Theorem 3. If we can show that T' is the same as T, then T' will be obviously optimal, i.e., we get an optimal energy allocation and workload partition. Fortunately, this is indeed true, as stated in the theorem.

The key observation is the fact that the equations in Theorem 3 still hold if E, the R_k's, and T are scaled by a common factor. This explains Fact 1), i.e., if we set E_l, the $R^*_{l,k}$'s, the $E_{l,k}$'s, and T_l according to Fact 1), then they will satisfy the equations in Theorem 3, and become the input and output of the workload partition problem for level l, for all $1 \le l \le v$. This implies that $T' = T_1 + T_2 + \cdots + T_v = T$, and we do get a solution to the optimal energy allocation and workload partition problem.

Furthermore, since $p_{l,k} = (R_{l,k}/T_l)^{\alpha_k}$ and $s_{l,k} = R_{l,k}/T_l$, for all $1 \le l \le v$ and $1 \le k \le m$, Facts 2)-3) are direct consequences, since $R_{l,k}$ and T_l are all scaled by a common factor. □

6.3 Optimal Time Allocation

According to the level-by-level scheduling method, the total energy consumption E of all tasks in a dag is simply the summation of the E_l's, i.e., $E = E_1 + E_2 + \cdots + E_v$, where E_l is viewed as a function of T_l, and E is viewed as a function of $T_1, T_2, \ldots, T_v$. Hence, we can define the following optimization problem (i.e., *optimal time allocation and workload partition*), namely, given m, $\alpha_1, \alpha_2, \ldots, \alpha_m$, $R_1, R_2, \ldots, R_v$, and T, finding $T_1, T_2, \ldots, T_v$, such that E is minimized, subject to the constraint that

$$F = T_1 + T_2 + \cdots + T_v = T.$$

Notice that each R_l should be further divided into $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ by using Theorem 4, such that E_l is minimized for a given T_l, where $1 \le l \le v$.

Note: The main purpose of the following mathematical derivation is to develop a numerical procedure to solve the above optimization problem. The reader can skip this part and go to Table 6.

To solve the above optimization problem, we use a Lagrange multiplier system, i.e., $\nabla E = \phi \nabla F$, where ϕ is a Lagrange multiplier. The above equation implies that $\partial E/\partial T_l = \phi \partial F/\partial T_l$, or, equivalently, $\partial E_l/\partial T_l = \phi$, for all $1 \le l \le v$. In the following, we develop a numerical procedure to solve the optimization problem.

To find $\partial E_l/\partial T_l$, we recall from Theorem 4 that for a given T_l, there is an optimal partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ which results in the minimized E_l given by

$$E_l = T_l \sum_{k=1}^{m} \left(\frac{\phi_l}{\alpha_k}\right)^{\alpha_k/(\alpha_k-1)},$$

TABLE 6
Numerical Data of Time-Constrained Scheduling of Precedence Constrained Tasks

l	T_l	k	$R^*_{l,k}$	$E_{l,k}$	$p_{l,k}$	$s_{l,k}$	E_l
		1	1.82382	4.21295	3.51079	1.51985	
		2	1.57704	3.12248	2.60207	1.31420	
		3	1.44124	2.49690	2.08075	1.20103	
1	1.20000	4	1.35755	2.09059	1.74215	1.13129	16.74562
		5	1.30203	1.80457	1.50381	1.08502	
		6	1.26323	1.59164	1.32637	1.05270	
		7	1.23508	1.42649	1.18874	1.02924	
		1	3.64765	8.42589	3.51079	1.51985	
		2	3.15409	6.24497	2.60207	1.31420	
		3	2.88248	4.99379	2.08075	1.20103	
2	2.40000	4	2.71510	4.18117	1.74215	1.13129	33.49123
		5	2.60405	3.60914	1.50381	1.08502	
		6	2.52647	3.18328	1.32637	1.05270	
		7	2.47016	2.85298	1.18874	1.02924	
		1	5.47147	12.63884	3.51079	1.51985	
		2	4.73113	9.36745	2.60207	1.31420	
		3	4.32372	7.49069	2.08075	1.20103	
3	3.60000	4	4.07265	6.27176	1.74215	1.13129	50.23685
		5	3.90608	5.41372	1.50381	1.08502	
		6	3.78970	4.77493	1.32637	1.05270	
		7	3.70525	4.27947	1.18874	1.02924	
		1	7.29529	16.85178	3.51079	1.51985	
		2	6.30818	12.48993	2.60207	1.31420	
		3	5.76496	9.98758	2.08075	1.20103	
4	4.80000	4	5.43020	8.36234	1.74215	1.13129	66.98246
		5	5.20811	7.21829	1.50381	1.08502	
		6	5.05294	6.36657	1.32637	1.05270	
		7	4.94033	5.70596	1.18874	1.02924	

where ϕ_l is also a function of T_l. We take a partial derivative of T_l on both sides of the above equation and get

$$\frac{\partial E_l}{\partial T_l} = \sum_{k=1}^{m} \left(\frac{\phi_l}{\alpha_k}\right)^{\alpha_k/(\alpha_k-1)} + T_l \phi'_l \sum_{k=1}^{m} \left(\frac{1}{\alpha_k - 1} \left(\frac{\phi_l}{\alpha_k}\right)^{1/(\alpha_k-1)}\right),$$

where $\phi'_l = \partial \phi_l / T_l$. It remains to find ϕ'_l. Since ϕ_l satisfies

$$\sum_{k=1}^{m} \left(\frac{\phi_l}{\alpha_k}\right)^{1/(\alpha_k-1)} = \frac{R_l}{T_l},$$

we can take a partial derivative of T_l on both sides of the above equation and get

$$\phi'_l \sum_{k=1}^{m} \frac{1}{\alpha_k(\alpha_k - 1)} \left(\frac{\alpha_k}{\phi_l}\right)^{(\alpha_k-2)/(\alpha_k-1)} = -\frac{R_l}{T_l^2},$$

which implies that

$$\phi'_l = -\frac{R_l}{T_l^2} \left(\sum_{k=1}^{m} \frac{1}{\alpha_k(\alpha_k - 1)} \left(\frac{\alpha_k}{\phi_l}\right)^{(\alpha_k-2)/(\alpha_k-1)}\right)^{-1}.$$

Substituting ϕ'_l into the equation of $\partial E_l/\partial T_l$ and noticing that $\partial E_l/\partial T_l = \phi$, we get

TABLE 7
Simulation Data of $\overline{B}$ for Precedence Constrained Tasks (CI= $\pm 0.459\%$)

v	$E = 200$	$E = 400$	$E = 600$	$E = 800$	$E = 1{,}000$
2	1.16462	1.18053	1.20596	1.22700	1.24971
3	1.12846	1.13468	1.13272	1.13369	1.13539
4	1.09418	1.10565	1.10753	1.11145	1.11009
5	1.06875	1.08214	1.08684	1.08925	1.09172
6	1.05101	1.06395	1.06890	1.07236	1.07458
7	1.03842	1.05040	1.05540	1.05932	1.06155
8	1.02886	1.04020	1.04541	1.04851	1.05055
9	1.02151	1.03266	1.03740	1.04011	1.04268
10	1.01599	1.02661	1.03105	1.03427	1.03600
11	1.01198	1.02162	1.02599	1.02890	1.03079

TABLE 8
Simulation Data of $\overline{C}$ for Precedence Constrained Tasks (CI= $\pm 1.809\%$)

v	$T = 0.45t$	$T = 0.40t$	$T = 0.35t$	$T = 0.30t$	$T = 0.25t$
2	1.77451	1.94078	2.15927	2.48986	2.97784
3	1.40733	1.50228	1.63698	1.81861	2.05889
4	1.25637	1.32314	1.41607	1.53371	1.68714
5	1.18150	1.23322	1.30014	1.38803	1.49647
6	1.13428	1.17670	1.23373	1.29589	1.37592
7	1.10240	1.13920	1.18599	1.23841	1.29883
8	1.08240	1.11338	1.15292	1.19641	1.24686
9	1.06720	1.09421	1.12827	1.16466	1.20648
10	1.05637	1.08065	1.10980	1.14244	1.17549
11	1.04812	1.07067	1.09668	1.12244	1.15475

$$\phi = \sum_{k=1}^{m} \left(\frac{\phi_l}{\alpha_k}\right)^{\alpha_k/(\alpha_k-1)} - \frac{R_l}{T_l}\left(\sum_{k=1}^{m} \frac{1}{\alpha_k(\alpha_k-1)}\left(\frac{\alpha_k}{\phi_l}\right)^{(\alpha_k-2)/(\alpha_k-1)}\right)^{-1} \sum_{k=1}^{m}\left(\frac{1}{\alpha_k-1}\left(\frac{\phi_l}{\alpha_k}\right)^{1/(\alpha_k-1)}\right).$$

It is noticed that the right-hand side of the above equation is negative and an increasing function of T_l. Thus, for any given $\phi < 0$, we can find T_l by using the bisection method. The detailed procedure of the above method is given in Algorithm 7 of the supplementary material, available online.

To determine $T_1, T_2, \ldots, T_v$, we notice that all the T_l's as well as $T_1 + T_2 + \cdots + T_v$ are increasing functions of ϕ. Therefore, we only need to determine ϕ, such that $T_1 + T_2 + \cdots + T_v = T$. Once $T_1, T_2, \ldots, T_v$ are available, we can calculate $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ and E_l, for all $1 \le l \le v$, by using Theorem 4, and E as a direct consequence. The detailed procedure of the above method is given in Algorithm 8 of the supplementary material, available online.

In Table 6, we demonstrate numerical data of time-constrained scheduling of precedence constrained tasks. The values of m, the α_k's, v, and the R_l's are the same as Table 5. The time constraint is $T = 12$. For all $1 \le l \le v$, we show T_l and E_l, as well as $R^*_{l,k}$, $E_{l,k}$, $p_{l,k}$, and $s_{l,k}$, for all $1 \le k \le m$.

We observe the same facts as those in Table 5.

Theorem 6. *Assume that (1) $R^*_1, R^*_2, \ldots, R^*_m$ is an optimal partition of a workload R of independent tasks obtained from Theorem 4, where the time constraint is T; (2) $E_1, E_2, \ldots, E_m$ is the corresponding energy consumption of the m computers; (3) E is the resulting minimized energy consumption; (4) $p_1, p_2, \ldots, p_m$ are the power settings of the m computers; (5) $s_1, s_2, \ldots, s_m$ are the speed settings of the m computers. For the optimal time allocation and workload partition problem with $R_1, R_2, \ldots, R_v$ and T, where $R_1 + R_2 + \cdots + R_v = R$, we have the following facts.*

1) *T_l, $R^*_{l,k}$, $E_{l,k}$, and E_l are linearly proportional to R_l, i.e., they can be obtained from T, R^*_k, E_k, and E by scaling a factor of $R_l/(R_1 + R_2 + \cdots + R_v)$, for all $1 \le l \le v$.*
2) *$p_{1,k} = p_{2,k} = \cdots = p_{v,k} = p_k$.*
3) *$s_{1,k} = s_{2,k} = \cdots = s_{v,k} = s_k$.*

Proof. The proof follows the same argument as that of Theorem 5. The key observation is the fact that the equations in Theorem 4 still hold if E, the R_k's, and T are scaled by a common factor. □

6.4 Performance Data

In this section, we present simulation results to demonstrate the expected performance of the LL-MLS algorithm on $m = 7$ heterogeneous computers with the same α_k's in Section 5.2.

Again, the solutions produced by the LL-MLS algorithm are compared with optimal solutions. Notice that the optimal schedule length of tasks in a dag with a given energy budget cannot be shorter than the optimal schedule length of the same set of tasks with the same energy budget and with all precedence constraints removed. Similarly, the minimum energy consumption of tasks in a dag with a given time deadline cannot be less than the minimum energy consumption of the same set of tasks with the same time deadline and with all precedence constraints removed. Hence, the lower bounds in Section 4 are also applicable to precedence constrained tasks.

In Table 7, we show simulation data for the expected performance bound $\overline{B}$ of the LL-MLS algorithm for the energy-constrained scheduling problem. We consider a dag with v levels. The number of tasks in level l is $10l$, for all $1 \le l \le v$. For each combination of $v = 2, 3, 4, \ldots, 11$ and $E = 200, 400, 600, 800, 1{,}000$, we generate 2,000 sets of $n = 5v(v+1)$ tasks, whose task execution requirements are random numbers uniformly distributed in $[0, 1]$. For each set of tasks, we calculate the lower bound T^* of $T_{\rm OPT}$ by using Theorem 3. We also simulate the LL-MLS algorithm and find its schedule length $T_{\text{LL-MLS}}$. The performance bound is obtained as $B = T_{\text{LL-MLS}}/T^*$. The average value of the 2,000 values of B is then reported in Table 7. The 99 percent confidence interval of all the data in Table 7 is no more than ± 0.459 percent.

In Table 8, we show simulation data for the expected performance bound $\overline{C}$ of the LL-MLS algorithm for the time-constrained scheduling problem. We consider the same dag as that in Table 7. For each combination of $v = 2, 3, 4, \ldots, 11$ and $T = 0.45t, 0.40t, 0.35t, 0.30t, 0.25t$, where $t = v^2$, we generate 2,000 sets of n tasks, whose task execution requirements are random numbers uniformly distributed in $[0, 1]$. For each set of tasks, we calculate the lower bound E^* of $E_{\rm OPT}$ by using Theorem 4. We also simulate the LL-MLS algorithm and find its energy consumption $E_{\text{LL-MLS}}$. The performance bound is obtained as $C = E_{\text{LL-MLS}}/E^*$. The average value of the 2,000 values of C is then reported in

Table 8. The 99 percent confidence interval of all the data in Table 8 is no more than ± 1.809 percent.

From Table 7, we observe that $\overline{B}$ is very close to 1. For a given v, the expected performance bound $\overline{B}$ slightly increases as E increases, primarily due to the lower bounds. For a given E, the expected performance bound $\overline{B}$ decreases as v increases.

From Table 8, we observe that $\overline{C}$ is reasonably close to 1, except for small values of v. For a given v, the expected performance bound $\overline{C}$ increases slightly (except for small v) as T decreases, primarily due to the lower bounds. For a given T, the expected performance bound $\overline{C}$ decreases as v increases.

Notice that all the data in Tables 7 and 8 are for expected performance bounds. The actual values of expected performance ratios are smaller and closer to the optimal. Our simulation results demonstrate that the LL-MLS algorithm has excellent expected performance in solving the problems of energy-constrained and time-constrained scheduling of precedence constrained tasks on multiple heterogeneous computers.

7 Summary and Further Research

We have addressed the problems of energy-constrained and time-constrained scheduling of independent or precedence constrained tasks on multiple heterogeneous computers as combinatorial optimization problems. Four key techniques have been developed in this paper.

- First, we show how to find an optimal energy or time allocation to levels of a dag, such that the performance of the level-by-level scheduling method can be optimized (Theorems 5 and 6).
- Second, for independent tasks in the same level, we show how to find an optimal workload partition, such that the modified list scheduling algorithm can be employed to find an approximate task schedule (Theorems 3 and 4).
- Third, for a given schedule, we show how to find an optimal power allocation, such that the total execution time or the total energy consumption is minimized (Theorems 1 and 2).
- Fourth, we derive lower bounds for the optimal schedule length and the optimal energy consumption, so that solutions produced by our heuristic algorithms can be compared with optimal solutions (Theorems 3 and 4).

Our extensive simulation results demonstrate that by using the above techniques, the MLS algorithm has excellent and close-to-optimal expected performance in solving the problems of energy-constrained and time-constrained scheduling of independent tasks on multiple heterogeneous computers, and that the LL-MLS algorithm has excellent and close-to-optimal expected performance in solving the problems of energy-constrained and time-constrained scheduling of precedence constrained tasks on multiple heterogeneous computers. Our research in this paper reveals that energy-efficient task scheduling on heterogeneous computers can be studied analytically, just as what we did for homogeneous computers. To the best of our knowledge, there has been no such method and result in the existing literature. Our investigation in this paper has made significant initial effort and our advanced methods and deep results in this paper are the current state-of-the-art. Furthermore, this paper should inspire further research interest.

To conclude the paper, we would like to mention several further research directions. First, it has been observed that power consumption is dependent on specific types of applications [10]. It is a challenge to incorporate such dependency into energy-efficient task scheduling. Second, our study in this paper can be extended to more sophisticated power consumption models, such as discrete and bounded clock frequency and supply voltage and execution speed and power supply levels, as what we have done for homogeneous computers [20]. Third, in a way similar to that of most existing studies, our research in this paper is primarily analytical and experimental. It needs further effort to apply such analytical and simulation results to actual applications in realistic systems. Due to space limitation, further investigation in these directions is beyond the scope of the paper and deserves separate papers.

Acknowledgments

The author deeply appreciates four anonymous reviewers whose criticism and comments help to improve the presentation of the manuscript.

References

[1] [Online]. Available: http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa, Accessed on 9 Nov. 2016.
[2] [Online]. Available: http://www.spec.org/power_ssj2008/results/power_ssj2008.html, Accessed on 9 Nov. 2016.
[3] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, no. 5, pp. 86–96, 2010.
[4] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances Comput.*, vol. 82, pp. 47–111, 2011.
[5] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 2nd ed. Boston, MA, USA: Prindle, Weber & Schmidt, 1981.
[6] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.
[7] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[8] S. Crago, et al., "Heterogeneous cloud computing," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2011, pp. 378–385.
[9] S. P. Crago and J. P. Walters, "Heterogeneous cloud computing: The way forward," *IEEE Comput.*, vol. 48, no. 1, pp. 59–61, Jan. 2015.
[10] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking back and looking forward: Power, performance, and upheaval," *Commun. ACM*, vol. 55, no. 7, pp. 105–114, 2012.
[11] S. Garg, S. Sundaram, and H. D. Patel, "Robust heterogeneous data center design: A principled approach," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 3, pp. 28–30, Dec. 2011.
[12] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.*, vol. 2, pp. 416–429, 1969.
[13] R. Iyer, et al., "CogniServe: Heterogeneous server architecture for large-scale recognition," *IEEE Micro*, vol. 31, no. 3, pp. 20–31, May/Jun. 2011.
[14] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1484–1497, Nov. 2008.

[15] K. Li, "Energy efficient scheduling of parallel tasks on multiprocessor computers," *J. Supercomputing*, vol. 60, no. 2, pp. 223–247, 2012.
[16] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Trans. Comput.*, vol. 61, no. 12, pp. 1668–1681, Dec. 2012.
[17] K. Li, "Optimal power allocation among multiple heterogeneous servers in a data center," *Sustain. Comput.: Informat. Syst.*, vol. 2, no. 1, pp. 13–22, 2012.
[18] K. Li, "Energy-efficient and high-performance processing of large-scale parallel applications in data centers," in *Data Centers*, S. U. Khan and A. Y. Zomaya, Eds. Berlin, Germany: Springer, 2015, pp. 1–33.
[19] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci.*, vol. 82, pp. 174–190, 2016.
[20] K. Li, "Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels," *J. Parallel Distrib. Comput.*, vol. 95, pp. 15–28, 2016.
[21] K. Li, "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr./Jun. 2016.
[22] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.
[23] S. U. R. Malik, K. Bilal, S. U. Khan, B. Veeravalli, K. Li, and A. Y. Zomaya, "Modeling and analysis of the thermal properties exhibited by cyber physical data centers," *IEEE Syst. J.*, 2016, doi: 10.1109/JSYST.2015.2493565, in press, 2017.
[24] J. Mateo, J. Vilaplana, L. M. Plá, J. L. Lérida, and F. Solsona, "A green strategy for federated and heterogeneous clouds with communicating workloads," *Sci. World J.*, Nov. 11, 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4244950/
[25] J. Mei, K. Li, and K. Li, "Energy-aware task scheduling in heterogeneous computing environments," *Cluster Comput.*, vol. 17, no. 2, pp. 537–550, 2014.
[26] H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "h-DDSS: Heterogeneous dynamic dedicated servers scheduling in cloud computing," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 3475–3480.
[27] V. Petrucci, E. V. Carrera, O. Loques, J. C. B. Leite, and D. Mossé, "Optimized management of power and performance for virtualized heterogeneous server clusters," in *Proc. 11th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2011, pp. 23–32.
[28] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, Art. no. 7.
[29] D. Sun, G. Zhang, S. Yang, W. Zheng, S. U. Khan, and K. Li, "Re-Stream: Real-time and energy-efficient resource scheduling in big data stream computing environments," *Inf. Sci.*, vol. 319, pp. 92–112, 2015.
[30] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, 2016.
[31] Y. Tian, C. Lin, and K. Li, "Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing," *Cluster Comput.*, vol. 17, no. 3, pp. 943–955, 2014.
[32] G. L. Valentini, et al., "An overview of energy efficiency techniques in cluster computing systems," *Cluster Comput.*, vol. 16, no. 1, pp. 3–15, 2013.
[33] L. Van Doorn, "Heterogeneous server architectures will dominate the future data center," Open Server Summit, Santa Clara, CA, USA, Nov. 13, 2014. [Online]. Available: https://vimeo.com/119496605
[34] W. Wang, B. Liang, and B. Li, "Multi-Resource fair allocation in heterogeneous cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 10, pp. 2822–2835, Oct. 2015.
[35] Y. Wang, K. Li, H. Chen, L. He, and K. Li, "Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 134–148, Jun. 2014.
[36] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41st Des. Autom. Conf.*, 2004, pp. 868–873.
[37] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Inf. Sci.*, vol. 319, pp. 113–131, 2015.
[38] L. M. Zhang, K. Li, D. C.-T. Lo, and Y. Zhang, "Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds," *Sustain. Comput.: Informat. Syst.*, vol. 3, no. 2, pp. 109–118, 2013.
[39] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1447–1464, Jul. 2013.

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, and peer-to-peer file sharing systems. He has more than 460 publications. He is currently or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, and the *IEEE Transactions on Services Computing*. He is a fellow of the IEEE.