J. Parallel Distrib. Comput. 81–82 (2015) 36–46

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Scheduling for energy minimization on restricted parallel processors

Xibo Jin [a,d,*], Fa Zhang [b], Liya Fan [e], Ying Song [c], Zhiyong Liu [a,c]

[a] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[c] State Key Laboratory for Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[d] University of Chinese Academy of Sciences, Beijing, China
[e] IBM China Research Laboratory, Beijing, China

HIGHLIGHTS

- We propose an optimal scheduling algorithm for the case when all of the tasks have uniform computational work.
- We present a polynomial-time algorithm that achieves a bounded approximation factor when the tasks have arbitrary-size work.
- We evaluate the performance of the approximation algorithm by a set of simulations.

ARTICLE INFO

Article history:
Received 29 April 2014
Received in revised form
4 March 2015
Accepted 1 April 2015
Available online 17 April 2015

Keywords:
Energy-efficient scheduling
Restricted parallel processors
Speed scaling
Continuous speed model
Approximation algorithm

ABSTRACT

Scheduling for energy conservation has become a major concern in the field of information technology because of the need to reduce energy use and carbon dioxide emissions. Previous work has focused on the assumption that a task can be assigned to any processor. In contrast, we initially study the problem of task scheduling on restricted parallel processors. The restriction takes account of affinities between tasks and processors; that is, a task has its own eligible set of processors. We adopt the Speed Scaling (SS) method to save energy under an execution time constraint (on the makespan C_{max}), and the processors can run at arbitrary speeds in $[s_{min}, s_{max}]$. Our objective is to minimize the overall energy consumption. The energy-efficient scheduling problem, involving task assignment and speed scaling, is inherently complex as it is proved to be NP-complete for general tasks. We formulate the problem as an Integer Programming (IP) problem. Specifically, we devise a polynomial-time optimal scheduling algorithm for the case in which tasks have a uniform size. Our algorithm runs in $O(mn^3 \log n)$ time, where m is the number of processors and n is the number of tasks. We then present a polynomial-time algorithm that achieves a bounded approximation factor when the tasks have arbitrary-size work. Numerical results demonstrate that our algorithm could provide an energy-efficient solution to the problem of task scheduling on restricted parallel processors.

1. Introduction

Energy consumption has become an important issue for today's computational systems. Dynamic speed scaling is a popular approach to energy-efficient scheduling. It significantly reduces energy dissipation by dynamically changing the speeds of the processors. It is well known that speed and power are related by a cube-root rule. More precisely, a processor consumes power at a rate proportional to s^3 when it runs at a speed s [21,4]. Most research publications [27,16,15,8,1,23,12,3] have assumed a more general power function s^α, where $\alpha > 1$ is a constant power parameter. Note that the power is a convex function of the processor speed. Obviously, the energy consumption is the power integrated over time. Higher speeds allow faster execution, but at the same time result in higher energy consumption.

In the past few years, energy-efficient scheduling has received much attention for both single-processor and parallel-processor environments. In the algorithm community, the approaches used can generally be categorized into the following two classes with respect to reducing energy usage [15,1]:

1. *Dynamic speed scaling.* The processors lower their speeds as much as possible in such a way that they can still execute tasks

* Correspondence to: No.6 Kexueyuan South Road Zhongguancun, Haidian District Beijing, Beijing, Postcode: 100190, China.
E-mail addresses: jinxibo@ict.ac.cn (X. Jin), zhangfa@ict.ac.cn (F. Zhang), fanliya@cn.ibm.com (L. Fan), songying@ict.ac.cn (Y. Song), zyliu@ict.ac.cn (Z. Liu).

http://dx.doi.org/10.1016/j.jpdc.2015.04.001
0743-7315/

while fulfilling the time constraints on those tasks. The reason why energy is saved via this strategy is the convexity of the power function. The goal is to decide the processing speeds in a way that minimizes the total energy consumption and guarantees the prescribed deadline.

2. *Power-down management*. The processors are put into a power-saving state when they are idle. However, there is an energy cost of the transition back to the active state. In this strategy, one determines whether there exist idle periods that can outweigh the transition cost and decides when to wake processors from the power-saving mode in order to complete all tasks in time.

Our paper focuses on energy-efficient scheduling via the dynamic speed scaling strategy. In this policy, the goals of scheduling are either to minimize the total energy consumption or to trade off the conflicting objectives of energy and performance. The main difference is that the former goal reduces the total energy consumption as long as the time constraint is not violated, whereas the latter seeks the best point between the energy cost and some performance metrics (such as the makespan and flow time).

Intensive research, initiated by Yao et al. [27], has been done on saving energy by speed scaling. In previous work, it was assumed that a task can be assigned to any processor. But it is natural to consider restricted scheduling in modern computational systems. The reason is that systems have evolved over time, for example by the use of clusters of processors, so that the various processors in a system may differ from each other in their abilities. (For instance, processors may have different additional components or different memory capacities [19].) This means that a task can only be assigned to a processor that has the components required for that task. That is, there are different affinities between tasks and processors. In practice, certain tasks may have to be allocated to certain physical resources (such as graphics processing units) [24]. It has also been pointed out that the design of some processors is specialized for particular types of tasks, and therefore tasks should be assigned to the processor best suited for them [13]. Furthermore, when tasks and input data are considered, tasks need to be assigned to the processors that contain their input data (by means of Hadoop Data Locality-Aware, for instance [26]). In other words, some of the tasks can be assigned to a processor set A_i, and some of the tasks to a processor set A_j, but $A_i \neq A_j$, $A_i \cap A_j \neq \emptyset$. Another case in point is scheduling with processor restrictions aimed at minimizing the makespan. This case has been studied extensively; see [19] for an excellent survey. Therefore, it is important to study scheduling with processor restrictions for reasons of both practical and algorithmic requirements.

Our contribution: In this paper, we address the problem of task **S**cheduling with the objective of **E**nergy **M**inimization on **R**estricted **P**arallel **P**rocessors (SEMRPP). We assume that all tasks are ready at the beginning of the process and share a common deadline (a real-time constraint) [4,16,8,1]. We discuss a continuous speed setting where the processors can run at arbitrary speeds in $[s_{\min}, s_{\max}]$. Our main contributions can be summarized in the following three groups:

1. We propose an optimal scheduling algorithm for the case when all of the tasks have uniform computational work.
2. For the general case in which the tasks have nonuniform computational work, we prove that the minimization of energy is NP-complete in the strong sense. We give a $2^{\alpha-1}(2-1/p^{\alpha})$-approximation algorithm, where α is the power parameter and $p = \max_{\mathcal{M}_j} |\mathcal{M}_j|$, and where $\mathcal{M}_j$ is the eligible processing set for the task J_j.
3. The performance of the approximation algorithm is evaluated by a set of simulations after an analysis of the algorithm, and it is found that the simulation results are consistent with the proposed scheduling algorithm.

To the best of our knowledge, our work may be the first attempt to study energy consumption optimization with restricted parallel processors.

The remainder of this paper is organized as follows. Section 2 describes previous work on speed scaling. Section 3 provides a formal description of the model. Section 4 first discusses some preliminary results and formulates the problem as an integer programming problem. Then we devise a polynomial-time optimal scheduling algorithm in the case where the tasks have uniform size, and present a bounded-factor approximation algorithm for the general case in which the tasks have arbitrary-size work. Section 5 presents numerical results. Finally, we conclude the paper in Section 6.

2. Related work

Yao et al. [27] were the first to explore the problem of scheduling a set of tasks with the least amount of energy in a single-processor environment via speed scaling. They proposed an optimal offline greedy algorithm and two bounded online algorithms, named *Optimal Available* and *Average Rate*. Ishihara et al. [16] formulated the problem of energy minimization in dynamical voltage scheduling as an integer linear programming problem where all tasks were ready at the beginning and shared a common finishing time. They showed that in the optimal solution a processor runs at only two adjacent discrete speeds when it can use only a small number of discrete processor speeds.

Besides studying variants of the speed scaling problem on a single processor, researchers have also carried out studies on parallel-processor environments. Chen et al. [8] considered energy-efficient scheduling with and without task migration in a multiprocessor system. They proposed an approximation algorithm for different settings of the power characteristics where no task was allowed to migrate. When task migration was allowed and the migration cost was assumed to be negligible, they showed that there was an optimal real-time task-scheduling algorithm. Albers et al. [1] investigated the basic problem of scheduling a set of tasks in a multiprocessor setting with the objective of minimizing the total energy consumption. First, they studied the case in which all tasks have unit size, and proposed a polynomial-time algorithm for agreeable deadlines. They proved that this case is NP-hard for arbitrary release times and deadlines and gave an $\alpha^{\alpha}2^{4\alpha}$-approximation algorithm. For scheduling tasks with arbitrary processing size, they developed constant-factor approximation algorithms. Aupy et al. [4] studied the minimization of energy for a set of processors for which a task assignment had been given, and investigated different speed scaling models. Angel et al. [3] considered a multiprocessor migratory and preemptive scheduling problem with the objective of minimizing the energy consumption. They proposed an optimal algorithm in the case where the jobs have release dates, deadlines, and a power parameter $\alpha > 2$.

There are also some publications that describe research on performance with an energy bound. Pruhs et al. [23] discussed the problem of speed scaling to optimize the makespan under the constraint of an energy budget in a multiprocessor environment where the tasks had precedence constraints ($Pm|\text{prec}, \text{energy}|C_{\max}$, where m is the number of processors). They reduced the problem to $Qm|\text{prec}|C_{\max}$ and obtained a poly-log(m)-approximation algorithm assuming that the processors can change speed continuously over time. Greiner et al. [12] presented research on the trade-off between energy and delay; i.e., their objective was to minimize the sum of the energy cost and delay cost. They suggested a randomized algorithm $\mathcal{RA}$ for multiple processors: each task was assigned uniformly at random to a processor, and then the single-processor algorithm $\mathcal{A}$ was applied separately to each processor. They proved that the approximation