Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Energy efficient scheduling of parallelizable jobs ☆

Kyle Fox [a,*,1], Sungjin Im [b,2], Benjamin Moseley [c,3]

[a] The University of Texas at Dallas, Richardson, TX 75080, United States
[b] University of California at Merced, Merced, CA 95344, United States
[c] Carnegie Mellon University, Pittsburgh, PA 15213, United States

ARTICLE INFO

Article history:
Received 26 July 2017
Received in revised form 31 January 2018
Accepted 21 February 2018
Available online xxxx
Communicated by C. Kaklamanis

Keywords:
Scheduling
Online
Energy
Speed scaling
Parallelism
Speed-up curved

ABSTRACT

This paper considers scheduling parallelizable jobs in the non-clairvoyant speed scaling setting to minimize the objective of weighted flow time plus energy. Previously, strong lower bounds were shown on this model in the unweighted setting even when the algorithm is given a constant amount of resource augmentation over the optimal solution. However, these lower bounds were given only for certain families of algorithms that do not recognize the parallelizability of alive jobs. In this work, we circumvent previous lower bounds shown and give a *scalable* algorithm under the natural assumption that the algorithm can know the current parallelizability of a job. When a general power function is considered, this is also the first algorithm that has a constant competitive ratio for the problem using any amount of resource augmentation.

1. Introduction

Energy aware job scheduling has recently received a significant amount of attention in scheduling theory literature [8,9,12,5,10,30,16,38,15,19,32,37,14,7,39,4,18,20,3,31,48,22,33,49]. In a typical system the scheduler decides how resources (for example, machines or processors) are allocated to jobs over time. In an energy aware scheduling environment, the scheduler decides how to use resources in order to balance the energy consumed by the system with system performance. For example, the scheduler may turn off machines over time or the scheduler may dynamically scale the speed of a processor. The scheduler's goal is to minimize the energy consumption while optimizing the quality of service for the jobs (or the clients who submitted the jobs). These two objectives are naturally competing. By using less energy the scheduler will have less resources available to process the jobs.

An energy aware scheduling model considered in previous literature is the online speed scaling setting. In an *online* setting, the scheduler only becomes aware of a job once it arrives to the system, and the scheduler makes decisions over time without knowing the jobs that are to arrive in the future. Online models accurately capture most of the scheduling problems faced in the real world, because generally the scheduler will not be aware of a job until it arrives to the system.

☆ A preliminary version of this paper appeared in the proceeding of the 24th ACM-SIAM Symposium on Discrete Algorithms, 2013 [27].

* Corresponding author.
E-mail addresses: kyle.fox@utdallas.edu (K. Fox), sim3@ucmerced.edu (S. Im), moseleyb@andrew.cmu.edu (B. Moseley).

[1] K. Fox was supported in part by the Department of Energy Office of Science Graduate Fellowship Program (DOE SCGF), made possible in part by the American Recovery and Reinvestment Act of 2009, administered by ORISE-ORAU under contract no. DE-AC05-06OR23100.
[2] S. Im was supported in part by NSF grants CCF-1016684, 1409130 and 1617653.
[3] B. Moseley was partially supported by NSF grants CCF-1617724, CCF-1733873 and CCF-1725661, a Google Research Award, and a Yahoo Research Award.

https://doi.org/10.1016/j.tcs.2018.02.021
0304-3975/

The model can be made even more realistic by assuming the scheduler is *non-clairvoyant*; it does not know the processing time of the jobs. In the *speed scaling* model, the processors available to the scheduler can dynamically change their speed over time. This model captures the fact that many systems today have the ability to change the processor frequency over time such as the technology used in AMD's PowerNow! and Intel's SpeedStep. Such technology has become ever more important in mobile computing where battery power consumption is a high priority [42] and in large scale server farms that use a costly amount of energy [42].

Perhaps the most popular online scheduling objective without considering energy is minimizing the total flow time. The *flow time* of a job is the amount of time that passes between the job's release and its completion. By minimizing the total flow time, the server focuses on optimizing the average quality of service delivered to the jobs. In the online speed scaling setting, most previous literature considered minimizing the total flow time plus the energy consumed in the system. This has a natural interpretation; a system designer may decide that it is worth spending one unit of energy to save β units of flow time. In this case, the system designer would want to minimize the total flow time plus β multiplied by the energy. By scaling the units of time and energy, we may assume that $\beta = 1$ and consider minimizing the total flow time plus energy. Work on this objective was initiated by studying processors whose power consumption obeys a generalization of the cube root rule observed in standard CMOS based processors [42,13]. For example, if the processor runs at speed s then the power consumed is s^α for some constant $\alpha \geq 1$; the cubed root rule is observed when $\alpha = 3$. This model has been rigorously studied [9,12,5,16,37,14,7,39], until a generalization was introduced by Bansal et al. [8]. In the generalization, there is a convex function P where $P(s)$ is the power consumed when running the processor at speed s. Essential no assumptions are made on P except that it is convex and positive. This generalization is known as the arbitrary power function model. In either model, the total energy consumed by the system is the integral of power over all time.

1.1. Old and new scheduling models

In the online speed scaling setting the scheduler has two policies:

Job selection: The scheduler must decide which job should be scheduled at each moment in time.
Energy policy: The scheduler must decide how fast to run the processor(s) at each moment in time.

When considering scheduling jobs on a single machine with speed scaling, the general approach is to use a known algorithm that performs well on a single machine without speed scaling and couple the algorithm with a natural speed scaling policy introduced in [2]. Essentially, this speed scaling policy runs the machine at speed s such that $P(s)$ is proportional to the number of unsatisfied jobs. This policy creates a natural balance between the energy objective and the flow time objective, because the increase in the flow time objective at each moment in time is equal to the number of unsatisfied jobs. At each point in time, the two objectives will increase roughly at the same rate.

Recently, scheduling speed scaling jobs online on multiple processors or machines was addressed. When addressing a multiple processor setting there are two models which one can consider. The first is when each job can be assigned to at most one machine at each moment in time. This model naturally captures the settings faced in multiple machine environments such as server farms. It is known that when processors have fixed speeds and energy is not a part of the objective that no algorithm can be $O(1)$-competitive for the objective of total flow time, even if we do not assume non-clairvoyance [41,6]. This lower bound carries over to the speed scaling setting with arbitrary power functions. Due to this lower bound, most previous work has turned to using a *resource augmentation* analysis [35]. Here the algorithm is given extra resources over the adversary and then the competitive ratio is bounded. We say that an algorithm is *b-speed c-competitive* if the algorithm with b-speed achieves an objective value at most c times the optimal value with unit speed. In the fixed speed setting, we mean each of the processors runs b times faster than the optimal solution's processors. In the speed scaling setting, when the algorithm runs a processor at power $P(s)$ the algorithm can process jobs at speed $b \cdot s$. The goal of this form of analysis is to find a $(1+\varepsilon)$-speed $O(f(\varepsilon))$-competitive algorithm for any constant $\varepsilon > 0$ where f is some function of only ε. That is, an algorithm which is $O(1)$-competitive while using a minimum amount of extra resources over the adversary. Such an algorithm is called scalable. For problems with strong lower bounds on the competitive ratio, a scalable algorithm is the best result that one can show using worst case analysis.

The works of Lam et al. [40,39] consider the case where each machine is identical. When machines are not identical, the problem becomes more algorithmically challenging. Scalable clairvoyant and non-clairvoyant algorithms were found for heterogeneous machine settings [30,29].

Another possible parallel scheduling model is when jobs can be scheduled on more than one processor at each moment in time. That is, jobs can run in parallel on multiple processors. The challenge in this setting is that there can be different degrees of parallelizability, even within phases of a single job. That is, each job consists of a set of phases that are to be processed sequentially. According to the phase a job is in, the job may be considerably sped-up when assigned to multiple processors, might not get sped-up, or something in between. The phases of a job can lead to different amounts of parallelism because one phase may require a lot of computation that can be easily parallelized, while another phase may require I/O that cannot be sped-up with extra computational power.

This setting can be formalized into a model known as the arbitrary speed-up curves model, so named because each phase of a job will have its own arbitrary speed-up curve/function that specifies its parallizability. Scheduling in this model

Please cite this article in press as: K. Fox et al., Energy efficient scheduling of parallelizable jobs, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.02.021