Theoretical Computer Science 593 (2015) 79–87

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

New online algorithm for dynamic speed scaling with sleep state

Gunjan Kumar, Saswata Shannigrahi [*]

Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India

A R T I C L E   I N F O

Article history:
Received 8 December 2013
Received in revised form 15 May 2015
Accepted 28 May 2015
Available online 9 June 2015
Communicated by T. Erlebach

Keywords:
Energy efficient algorithm
Online algorithm
Scheduling algorithm

A B S T R A C T

We consider an energy-efficient scheduling problem where n jobs $J_1, J_2, \ldots, J_n$ need to be executed such that the total energy usage by these jobs is minimized while ensuring that each job is finished within its deadline. The processor executing these jobs can be in either *active* or *sleep* state at any point of time. The speed of the processor in an active state can be arbitrary. If the processor is running at a speed $s \geq 0$, the required power $P(s)$ is assumed to be equal to $s^\alpha + g$ where $\alpha > 1, g > 0$ are constants. On the other hand, the required power is zero when the processor is in the sleep state. However, $L > 0$ amount of wake-up energy is needed to wake-up the processor from the sleep state to the active state.

In this paper, we work in an online setting where a job is known only at its arrival time, along with its processing volume and deadline. In such a setting, the currently best-known algorithm by Han et al. (2010) [3] provides a competitive ratio $\max\{4, 2+\alpha^\alpha\}$ of energy usage. We present a new online algorithm SqOA which provides a competitive ratio $\max\{4, 2+(2-1/\alpha)^\alpha 2^{\alpha-1}\}$ of energy usage. For $\alpha \geq 2.34$, the competitive ratio of our algorithm is better than that of any other existing algorithm for this problem.

1. Introduction

Over the last few decades, the energy requirement for computing has increased exponentially. For example, the electricity cost imposes a significant strain on the budget of data centers where CPUs account for 50 to 60 percent of the energy consumption. The related problems include the heat generated, leading to a reduced reliability of hardware components. As a result, it has become an important challenge for algorithms researchers to design algorithms which are efficient with respect to the energy usage by the CPU.

Yao, Demers and Shenker [5] were the first ones to investigate the following scheduling problem where the objective is to minimize the total energy consumed by the processing of the jobs. The general setting of their problem, on which we also work, is explained below.

Assume that a processor can run at any arbitrary speed, and consider n jobs $J_1, \ldots, J_n$ that have to be processed by such a processor. Each job J_i has a release time r_i, a deadline d_i and a processing volume w_i. A job J_i must be executed in the time interval $[r_i, d_i]$. The processing volume w_i is the amount of work that must be completed to finish J_i. It is also assumed that preemption of jobs is allowed, i.e., the processing of a job may be interrupted and resumed later. If the processor is running at a speed s, the required power $P(s)$ is assumed to be equal to s^α where $\alpha > 1$ is a constant. The

[*] Corresponding author. Tel.: +91 361 2582375.
E-mail addresses: gunjan.kg1024@gmail.com (G. Kumar), saswata.sh@iitg.ernet.in (S. Shannigrahi).

http://dx.doi.org/10.1016/j.tcs.2015.05.045
0304-3975/© 2015 Elsevier B.V. All rights reserved.

energy consumed by the execution of the algorithm is the amount of power integrated over the execution period of the algorithm.

Under the above assumptions, Yao et al. [5] developed an $O(n^3)$-time algorithm to schedule n jobs such that the total energy consumption is minimized. However, this algorithm (called YDS algorithm) works *offline*, i.e., the release time, deadline and the processing volume of each job is known beforehand. In an online version of the problem, a job J_i is only known at the time of its arrival, when the deadline d_i and the processing volume w_i are also revealed. An online algorithm is called c-competitive if, for any job sequence, the total energy consumption of the algorithm is at most c times that of an optimal offline algorithm for the same set of jobs.

In the same paper [5] as above, Yao et al. devised two online algorithms for the energy-efficient scheduling problem described above. The first algorithm, called *Average Rate*, has a competitive ratio at most $2^{\alpha-1}\alpha^\alpha$, for any $\alpha \geq 2$. The competitive ratio of the second algorithm, called *Optimal Available (OA)*, is exactly α^α [2], and therefore the second algorithm is better than the first one in terms of competitiveness. Bansal et al. [2] provided an algorithm BKP which improved the competitive ratio to $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$. For small values of α, Bansal et al. [1] provided a better competitive algorithm *qOA* whose competitiveness is at most $\frac{4^\alpha}{2e^{\frac{1}{2}}\alpha^{\frac{1}{4}}}$.

Let us describe the ideas behind the two algorithms *Optimal Available (OA)* and *qOA* as these two algorithms are relevant to the work done in this paper. The algorithm OA can be described as follows: whenever a new job arrives, the algorithm computes an optimal schedule for the currently available unfinished jobs. The optimal schedule for the currently available unfinished jobs can be computed using the YDS algorithm mentioned earlier. The algorithm *qOA* slightly modifies the algorithm *OA*. It sets the speed of the processor to be $q \geq 1$ times the speed that the algorithm *OA* would run in the current state. Setting the value of q to be $2 - \frac{1}{\alpha}$, this algorithm achieves a competitive ratio of $\frac{4^\alpha}{2e^{\frac{1}{2}}\alpha^{\frac{1}{4}}}$.

1.1. Our contribution

Irani et al. [4] started the investigation of the scenario where a processor can be transitioned into a sleep state in which the energy consumption is 0. In the active state, the power consumption is given by the equation $P(s) = s^\alpha + g$, where $g > 0$. In the sleep state, the power consumption is 0. The transition from the active state to the sleep state is assumed not to consume any energy. However, the transition from the sleep state to the active state requires $L > 0$ amount of wake-up energy. Note that g was assumed to be 0 in the model considered by Yao et al. [5], and there was no concept of a sleep state in their model.

In the model considered by Irani et al. [4], one can note that the power consumption is strictly greater than 0 when the processor is in an active state. However, it is not always beneficial to send a processor to the sleep state when it has no work to do, because $L > 0$ amount of wake-up energy is needed to wake-up the processor from the sleep state to the active state. Accordingly, an online algorithm needs to decide when to send a processor to the sleep state, and when to wake-up the processor back to the active state, in addition to identifying a job to execute and determining the speed at every moment during the active state. Taking these trade-offs into account, Irani et al. [4] devised an online algorithm which is $(2^{2\alpha-2}\alpha^\alpha + 2^{\alpha-1} + 2)$-competitive for energy. Han et al. [3] developed an algorithm SOA which improves the competitive ratio to $\max\{4, 2 + \alpha^\alpha\}$.

In this paper, we combine the ideas used in the algorithms qOA [1] and SOA [3], and provide an algorithm SqOA with a competitive ratio of $\max\{4, 2 + (2 - 1/\alpha)^\alpha 2^{\alpha-1}\}$. Note that our algorithm has a better competitive ratio than SOA for $\alpha \geq 2.34$. In the next two sections, we prove Theorem 1. In Section 2, we describe the algorithm SqOA and prove some of its properties. In Section 3, we prove its competitiveness using these properties. In Section 4, we conclude with a mention about the difference of the techniques used in this paper with that of [1] and [3].

Theorem 1. *The algorithm SqOA achieves the competitive ratio* $\max\{4, 2 + (2 - 1/\alpha)^\alpha 2^{\alpha-1}\}$ *for any* $\alpha > 1$.

2. Algorithm SqOA: description and properties

Let us start with some definitions and notation that we will use in this paper. A processor is stated to be in a *working state* if it is currently running with a strictly positive speed. The processor is in an *idle state* if the processor is active but running with a speed 0. Finally, the processor is said to be in a *sleep state* if it is no longer active.

Throughout this paper, we use the notation used in Bansal et al. [1]. The current time is always denoted as t_0. For $t_0 \leq t' \leq t''$, $w_a(t', t'')$ denotes the total amount of unfinished work for SqOA at t_0 that has a deadline during $(t', t'']$. The quantity $w_o(t', t'')$ is defined similarly for the optimal algorithm OPT. The current speeds of SqOA and OPT are denoted by s_a and s_o, respectively.

Let $d(t', t'') = \max\{0, w_a(t', t'') - w_o(t', t'')\}$ be the excess unfinished work that SqOA has relative to OPT among the already released jobs with deadlines in the range $(t', t'']$. A sequence of critical times $t_0 < t_1 < t_2 < \ldots. < t_h$ is defined iteratively as follows: let t_1 be the latest time such that $d(t_0, t_1)/(t_1 - t_0)$ is maximized. Note that t_1 is no more than the latest deadline of any job released thus far. If t_i is earlier than the latest deadline, then let $t_{i+1} > t_i$ be the latest time, not later than the latest deadline, that maximizes $d(t_i, t_{i+1})/(t_{i+1} - t_i)$. We refer to the intervals $(t_i, t_{i+1}]$ as critical intervals. We define g_i as $g_i = d(t_i, t_{i+1})/(t_{i+1} - t_i)$. Note that $g_0, g_1, \ldots., g_{h-1}$ is a non-negative strictly decreasing sequence.