Sustainable Computing: Informatics and Systems 4 (2014) 151–159

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

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

Temperature aware online scheduling for throughput maximisation: The effect of the cooling factor

Martin Birks, Stanley P.Y. Fung*

Department of Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom

ARTICLE INFO

Article history:
Received 31 January 2013
Received in revised form 18 May 2014
Accepted 4 July 2014

Keywords:
Online algorithms
Scheduling
Competitive analysis
Temperature

ABSTRACT

We consider the problem of scheduling jobs in processors with temperature constraints. In our model, unit-length jobs arrive over time, each with a deadline and a heat contribution. The objective is to maximise the total number of completed jobs while keeping the processors within a given temperature threshold. Our main result is the analysis of a large class of 'reasonable' algorithms. We analyse the competitive ratio of these algorithms as a function of the cooling factor of the processors. Then we present a lower bound for the problem that shows that these algorithms are optimal in the case of a single processor. We give some other lower bounds for the multiple processors case. Then we perform some computational experiments to investigate the performance of the algorithms in an average case sense, bringing some interesting observations about the performance of the algorithms with respect to the cooling factor.

1. Introduction

1.1. Background

Advances in microprocessor technology have given a huge increase in processing power in computing devices. Moreover, the need for mobile and embedded devices means that these computing units are packed inside an even smaller space. These together give significant problems to thermal management in microprocessors. High temperatures are a problem for many reasons, such as negatively affecting system reliability, shortening processor lifespan, and incurring higher cooling costs. Some of these factors form a positive feedback loop (e.g. high temperature lead to higher 'leakage power', which in turn lead to even higher temperature) which makes them particularly difficult to handle. See e.g. [7,12] for a discussion of these problems. A lot of research work has been done to address these issues. A recent survey [12] gave an excellent and accessible account on the many aspects of thermal issues in microprocessor design and scheduling. While the problem can be tackled on a micro-architecture level, it has also become apparent that algorithmic techniques can be used to manage temperature and energy; see e.g. the survey in [11].

The temperature created by a processor is related to its power use as well as its cooling mechanism. Power usage is a convex function of processor speed (see e.g. [11]), so one way of controlling the temperature is to slow down the processor when the temperature gets too high; this is known as *dynamic voltage scaling* (DVS). Algorithms using DVS have been designed and analysed in [1]. These algorithms are competitive in minimising the maximum temperature. However, many devices have a *temperature threshold* that cannot be exceeded without causing problems or even permanent failure. It is therefore more useful in some cases to give a fixed temperature threshold that cannot be exceeded, and then maximise throughput subject to this threshold. This is the model we consider in this paper. Note that unlike DVS we consider that the speed of the processor is fixed, although as can be seen later we may need to insert idle time steps which in some sense is similar to reducing the speed of the processor. As for cooling, we follow the model in [1] and Fourier's Law, which states that the rate of cooling is proportional to the difference between the temperature of the processor and its environment, and that the environment's temperature is assumed to be constant.

Multicore processors have become increasingly popular over recent years. One of the main motivations of using multicore processors is that it is a useful tool in coping with the heat a processor generates (see e.g. [9] and references therein). A result of power usage being a convex function of processor speed is that it is more power (and therefore temperature) efficient to run several processors at a lower speed, than it is to run one processor at a high speed and still carry out the same amount of work.

* Corresponding author. Tel.: +44 1162523807.
E-mail addresses: mbirks@gmail.com (M. Birks), pyfung@mcs.le.ac.uk (S.P.Y. Fung).

http://dx.doi.org/10.1016/j.suscom.2014.07.006
2210-5379/

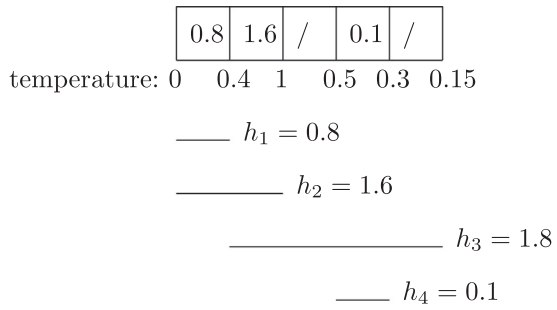

Fig. 1. An example execution with 4 jobs with heats $h_1, \ldots, h_4$. The line segments represent their availability (from release time to deadline). Here $R = 2$. Suppose in the first step job 1 is executed. Then in the next step job 3 is too hot to execute (its execution will result in a temperature of $(0.4 + 1.8)/2 = 1.1$), so job 2 is executed instead. Job 3 is still too hot to run in the next step so the processor idles. In fact it is still too hot for the next two steps as well.

1.2. Our model

We consider a simplified model with unit-length jobs, where each job j has a *release time* r_j, a *deadline* d_j and a *heat contribution* h_j. All release times and deadlines are integers. The jobs arrive *online* which means that the algorithm will not know anything about the job until it is released and so cannot use information regarding future jobs to make any scheduling decisions. Once a job arrives, all these parameters are known to the algorithm.

An algorithm has access to $m \geq 1$ identical processors. Time is divided into discrete unit-length time steps. At each time step $[t, t+1)$ the algorithm takes a decision for each processor deciding whether to schedule a job on that processor, and if so which job. (For simplicity, instead of saying $[t, t+1)$ we will just say the algorithm schedules a job at time t.)

Every processor has a separate temperature. For simplicity, we assume there is no heat transfer between different processors. It has been suggested that such lateral heat transfer is much smaller than vertical heat transfer (see e.g. [15,10] and the discussion therein), and such consideration would require information of the physical layout of the processing units. Thus the temperature of each processor is determined independently. If the temperature of a processor p at a time t is $\tau_{p,t}$, then the temperature at time $t+1$ on the same processor p is given by $\tau_{p,t+1} = (\tau_{p,t} + h_j)/R$, where j is the job scheduled at time t and $R > 1$ is the *cooling factor*, which is the factor that the temperature of a processor reduces by at the end of each time step. If no job is scheduled then assume $h_j = 0$. The temperature $\tau_{p,t}$ for any processor p and at any time t must not exceed the thermal threshold T. Without loss of generality we can assume that the initial temperature is 0 and that T is 1. See Fig. 1 for an example.

The objective is to compute a schedule that maximises the total number of completed jobs, while staying under the thermal threshold. Using standard notations this can be described as $P|\text{online-}r_i, h_i, p_i = 1|\sum U_i$ (where $U_i = 1$ if job i is completed on time and 0 otherwise). We measure the performance of online algorithms using *competitive analysis* [4]. An online algorithm is *c-competitive* if the number of jobs completed by the algorithm is at least $1/c$ of that obtained by an offline optimal algorithm, for any input instance.

Admittedly our thermal model as described above is highly simplified, but it can be seen as the limiting case of Fourier's law when the time steps are very small. In the systems literature there are much more detailed thermal modelling tools available e.g. HotSpot [14], but the more mathematical nature of our analysis requires a simplified model.

In a real system we of course have $R > 1$; in the hypothetical case of $R = 1$, heat were never dissipated, and the problem can be interpreted as an energy-limited scheduling problem where each job has a energy requirement and we want to schedule as many of them as possible subject to a fixed amount of energy available.

1.3. Previous results

There are so many work in the systems and processor design community to tackle the temperature management problem, that we cannot hope to survey them all here; see e.g. [7,15,12] and the references therein. They range from floorplanning during the design stage of the chip, to the dynamic scheduling of jobs by estimating their heat contribution to different parts of the CPU (e.g. register file or cache).

There are comparatively few results of analysing the problem in the framework of online competitive algorithms. Bansal et al. [1] considered the problem of minimising the maximum temperature and gave online competitive algorithms. For the unit-length throughput model that we consider in this paper, the main previous result was from [6], where it was shown that a large class of algorithms called *reasonable algorithms* (to be defined precisely in the next section) are 2-competitive. A matching lower bound on the competitive ratio was also given, showing that this class of algorithms are optimal. They also showed that the offline case is NP-hard. In the paper they only considered the single processor case and also only for $R = 2$.

1.4. Motivation of this paper

The primary motivation of modelling the problem using unit-length jobs is to represent the job slices given to the processor by the operating system [6]. As such the actual cooling factor R relates to the length of this time quantum and the 'RC constant' of the processing unit (a thermal property related to how quickly the unit cools). Different systems appear to have very different values for these parameters (see e.g. [13]), and it is therefore important that we can design and analyse algorithms for different values of R. For example, [13,5] reported that typical OS scheduler ticks Δt ranges from several to 10 ms, and the thermal time constant τ (vertical direction) ranges from tens to hundreds of milliseconds. Following the model in [15], these values translate into different values of R, e.g. if both Δt and $\tau = 10$ ms then $R = 2$ whereas if $\Delta t = 10$ ms and $\tau = 100$ ms then $R = 1.1$.

The use of multiple cores means that algorithms have more decisions to make when scheduling jobs, which can increase the scope for increasing the quality (or if the decisions are made poorly, then decreasing the quality) of the schedules produced. As one of the main motivations for using multiple processors was in an effort to reduce heat issues, it follows that it is important to maximise the extra capabilities that multicore systems provide. This means that the design and analysis of temperature aware algorithms for multicore systems is needed in order that these systems can reach their full potential.

1.5. Our results

In this paper we consider the case where the cooling factor can be any $R > 1$ and there can be any number of processors. We first show an upper bound for reasonable algorithms, for any $R > 1$ and any $m \geq 1$. We then give a lower bound that shows that reasonable algorithms are in fact optimal for all values of R when $m = 1$. For $m \geq 2$ we give some weaker lower bounds. The results show how the competitiveness depends on R: specifically, it increases as R gets smaller, and tends to infinity when R tends to 1.

Our results are based on an observation on the number of jobs that an optimal offline algorithm can execute that are too hot for the online algorithm after it has executed a hot job. We derive the relation between this number of jobs and the cooling factor, allowing