Multiprocessor Speed Scaling for Jobs with Arbitrary Sizes and Deadlines

Paul C. Bell * Prudence W.H. Wong *

1 Introduction

Energy consumption has become an important concern in the design of modern processors, not only for battery-operated mobile devices with single processors but also for server farms or laptops with multi-core processors. A popular technology to reduce energy usage is *dynamic speed scaling* (see e.g., [1, 2, 3, 6]) where the processor can vary its speed dynamically. The power consumption is modelled by s^α when the processor runs at speed s, where α is typically 3 [4]. Running a job slower saves energy, yet it takes longer to finish the job. The study of speed scaling was initiated by Yao et al. [6]. They studied deadline scheduling on a single processor in which jobs with arbitrary sizes and deadlines arrive online and the aim is to finish all jobs by their deadlines using the minimum amount of energy. They gave a simple online algorithm AVR which is $2^\alpha\alpha^\alpha$-competitive and they also proposed an online algorithm OA which was later shown by Bansal et al. to be α^α-competitive [3]. Algorithms with better competitive ratios were later proposed in [3, 2].

Albers et al. [1] have extended the study to the multiprocessor setting in the special cases of unit-size jobs or jobs with agreeable deadlines (jobs arriving earlier have earlier deadlines), and presented constant competitive algorithms for both cases. In the multiprocessor setting, in addition to determining processor speed, a job dispatching algorithm is required to assign jobs to processors. In this paper we study the generalized problems where jobs have arbitrary sizes and arbitrary deadlines. We first observe that the classified round robin algorithm (CRR) [1] does not scale well when jobs have arbitrary sizes and deadlines; the competitive ratio is at least $m^{\alpha-1}$ where m is the number of processors. We take one step forward to propose a non-migratory job dispatching algorithm, called DCRR, and show that DCRR coupled with AVR is $O(\log^\alpha P)$-competitive, where P is the ratio between the maximum and minimum job size; note that the competitive ratio is independent of m and holds even against an optimal migratory offline algorithm.

2 Model and DCRR

Model. We are to schedule a set of jobs onto m processors. We denote the release time, deadline and job size of a job J as $r(J)$, $d(J)$, and $w(J)$, respectively. The *span* of job J is $span(J) = d(J) - r(J)$ and the *density* $\delta(J) = \frac{w(J)}{d(J)-r(J)}$. Preemption of jobs is allowed without penalty. For any job set $\mathcal{J}$ and any algorithm A, we overload the symbol $A(\mathcal{J})$ to mean both the schedule of A on $\mathcal{J}$ and the energy required by the schedule. In [1], a relationship between an optimal schedule on a single

This research is partly supported by EPSRC Grant EP/E028276/1.

*Department of Computer Science, University of Liverpool, UK. {p.c.bell, pwong}@liverpool.ac.uk

processor and m processors is given (Lemma 1(a)). Furthermore, one can bound OPT by MIN($\mathcal{J}$) and AVR($\mathcal{J}$), where MIN($\mathcal{J}$) is the minimum energy to run each job of $\mathcal{J}$ independently of other jobs, i.e., MIN($\mathcal{J}$) $= \sum_{J\in\mathcal{J}}(\delta(J))^\alpha span(J)$; and AVR($\mathcal{J}$) is the energy of scheduling $\mathcal{J}$ by AVR on a single processor. Then Lemma 1(b) follows from Lemma 1(a) and the competitive ratio of AVR.

Lemma 1 ([1]). *(a) For any job set $\mathcal{J}$,* OPT($\mathcal{J}$) $\geq$ OPT1($\mathcal{J}$)$/m^{\alpha-1}$*, where* OPT *and* OPT1 *are the optimal algorithm on m processors and on a single processor, respectively. (b) For any job set $\mathcal{J}$, (i)* MIN($\mathcal{J}$) $\leq$ OPT($\mathcal{J}$)*; (ii)* AVR($\mathcal{J}$) $\leq \alpha^\alpha 2^{\alpha-1} m^{\alpha-1}$OPT($\mathcal{J}$)*.*

CRR. Let Δ be the maximum density of the jobs in $\mathcal{J}$. CRR classifies jobs with density Δ into density-class-0, and jobs with density in $[\Delta/2^k, \Delta/2^{k-1})$ into density-class-k, for $k > 0$. Jobs within each class are dispatched to processors in round-robin independently. For each processor, the speed is the sum of the densities of the unfinished jobs dispatched to that processor (i.e., AVR) and the processor processes these jobs by splitting the speed equally among them. In [1], it is shown that CRR is constant competitive for unit size jobs. Lemma 2 states that this is no longer true for arbitrary size jobs. Roughly speaking, the adversary releases two types of jobs of two different spans but with the same density. Let m be the number of processors. A total of m^2 jobs are released, all with tiny span except every m-th job which has an arbitrarily large span; the spans of the tiny span jobs do not overlap with each other. CRR will dispatch all large span jobs to the same processor while OPT can schedule each of the m large span jobs into separate processors.

Lemma 2. *For arbitrary size jobs, CRR has a competitive ratio of at least $m^{\alpha-1}$.*

DCRR. Apart from having density classes, DCRR also classifies jobs according to sizes. Let Γ be the maximum job size of a job set $\mathcal{J}$. Jobs with size in $(\Gamma/2^{h+1}, \Gamma/2^h]$ are classified into size-class-h, for $h \geq 0$ (note the difference from the definition of density-classes). We then define the set $C_{k,h}$ to be the set of jobs in density-class-k and size-class-h. For simplicity, we assume that Δ and Γ are known in advance (otherwise, the class definition can be modified slightly).

Algorithm 1 Algorithm DCRR

Job dispatching: Jobs of the same class $C_{k,h}$ are dispatched (upon their arrival) to the m processors using a round-robin strategy, i.e., the i-th job in $C_{k,h}$ is dispatched to processor-($i \mod m$), and different classes are handled independently.

Speed running: The speed of each processor is determined by AVR on the jobs dispatched and the speed is split equally among these jobs (note that this gives a feasible schedule).

To obtain a schedule for job set $\mathcal{J}$, we transform $\mathcal{J}$ to a *nice* job set $\mathcal{J}^*$ such that each job in $\mathcal{J}^*$ has density $\Delta/2^k$ and size $\Gamma/2^h$ for some $k, h \geq 0$. We transform each $J \in \mathcal{J}$ in $C_{k,h}$ to J^* (also in $C_{k,h}$) by setting $r(J^*) = r(J)$, $w(J^*) = \Gamma/2^h$, and $d(J^*) = r(J^*) + (\frac{\Gamma}{2^h} \cdot \frac{2^k}{\Delta})$, i.e., $\delta(J^*) = \Delta/2^k$. So job densities only decrease and sizes only increase for J^* and $span(J) \leq span(J^*) \leq 4 \times span(J)$.

Lemma 3. *For any job set $\mathcal{J}$ and its corresponding nice job set $\mathcal{J}^*$, (a)* 2^αOPT($\mathcal{J}$) $\geq$ OPT($\mathcal{J}^*$)*; (b)* DCRR($\mathcal{J}$) $\leq 2^{2\alpha}$DCRR($\mathcal{J}^*$)*.*

Analysis for nice job set $\mathcal{J}^*$. Let $\mathcal{J}_i^*$ be the subset of $\mathcal{J}^*$ that are dispatched to processor i by DCRR. Then DCRR($\mathcal{J}^*$) $= \sum_{1\leq i\leq m}$ AVR($\mathcal{J}_i^*$). The latter is bounded in Lemma 4. Then Theorem 5 follows from Lemmas 1(b), 3 and 4.

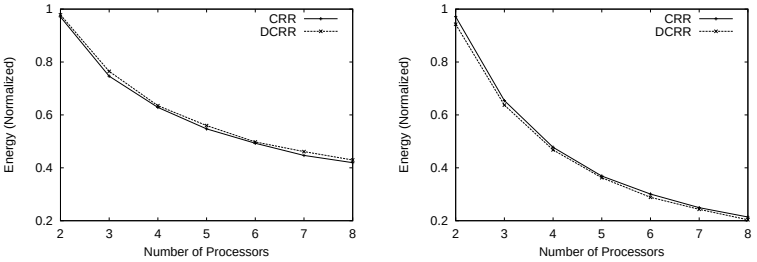
Lemma 4. *For any nice job set $\mathcal{J}^*$, we have* $\sum_{1\leq i\leq m}$ AVR($\mathcal{J}_i^*$) $\leq 2^{2\alpha}((\log^\alpha P)$ MIN($\mathcal{J}^*$) + AVR($\mathcal{J}^*$)$/m^{\alpha-1})$.

Proof. (Sketch) We adapt the proof of CRR in [1]. Let $s_i(t)$ denote the speed of AVR on processor i at time t. Note that $s_i(t)$ is the sum over all classes $C_{k,h}$ of the quantity $|C_{k,h}(t) \bigcap \mathcal{J}_i^*|\Delta/2^k$,

where $C_{k,h}(t)$ is the set of jobs in $C_{k,h}$ such that t is within the span of those jobs. The classes $C_{k,h}$ can be divided into two types: *type*-1 are those with only one job dispatched to processor i at time t, and *type*-2 are those with more than one such jobs. For each type-1 $C_{k,h}$, the contribution to $s_i(t)$ is $\Delta/2^k$; for each type-2 $C_{k,h}$, the contribution is $|C_{k,h}(t) \bigcap \mathcal{J}_i^*|\Delta/2^k$. As $\mathcal{J}^*$ is a nice job set (all jobs in class $C_{k,h}$ have the same size and same density), one can show that $|C_{k,h}(t) \bigcap \mathcal{J}_i^*| \leq \left\lceil \frac{|C_{k,h}(t)|}{m} \right\rceil$. Let $s(t)$ denote the speed of AVR for the whole job set $\mathcal{J}^*$. Then $s(t) = \sum_{k\geq 0} \sum_{h \geq 0} |C_{k,h}(t)|\Delta/2^k$. By arithmetic, one can then show that $s_i(t) \leq 4 \times \max\left\{(\log P)\frac{\Delta}{2^{k_1}}, \frac{s(t)}{m}\right\}$, where k_1 is the minimum k among type-1 $C_{k,h}(t)$. Integrating $(s_i(t))^\alpha$ over t and then summing over all $1 \leq i \leq m$, we then have $\sum_{1\leq i\leq m} \mathrm{AVR}(\mathcal{J}_i^*) \leq 2^{2\alpha}((\log^\alpha P)\,\mathrm{MIN}(\mathcal{J}^*) + \mathrm{AVR}(\mathcal{J}^*)/m^{\alpha-1})$. Details are omitted. □

Theorem 5. *For an arbitrary job set $\mathcal{J}$, the competitive ratio of algorithm DCRR is at most $2^{5\alpha}(\log^\alpha P + \alpha^\alpha 2^{\alpha-1})$, where P is the ratio between the maximum and minimum job size.*

Simulation results. We simulate CRR and DCRR on random job sets of size 400. The inter-arrival times and spans follow a Poisson distribution with average 0.1 and 5, respectively; and the jobs sizes are uniformly distributed in the range $[0.01, 20]$ (Figure 1(a)). We observe that CRR performs slightly better than DCRR. We also repeat the experiment but choose jobs randomly (with equal probability) from the above job set and another one with spans and sizes increased by a factor of 20 (to simulate the scenario of two jobs sets of different demand characteristics). In this case, DCRR performs slightly better.



(a) Single job set (b) Combined job sets

Figure 1: Simulation results (averaged over 100 instances).

Concluding remarks. We extend the study of energy efficient deadline scheduling on multiprocessor for jobs with arbitrary sizes and deadlines. One may consider how DCRR can be coupled with OA instead of AVR to improve the results. Another open question is to consider speed bounded processors [5] and derive algorithms that are competitive both in throughput and energy.

References

[1] S. Albers, F. Muller, and S. Schmelzer. Speed scaling on parallel processors. In *SPAA*, pages 289–298, 2007.

[2] N. Bansal, H. L. Chan, K. Pruhs, and D. Rogozhnikov-Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. In *ICALP*, 2009, to appear.

[3] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *JACM*, 54(1), 2007.

[4] D. Brooks, P. Bose, S. Schuster, H. M. Jacobson, P. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[5] H. L. Chan, W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak, and P. W. H. Wong. Energy efficient online deadline scheduling. In *SODA*, pages 795–804, 2007.

[6] F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382, 1995.