Deadline Scheduling and Power Management for Speed Bounded Processors

Xin Han[*] Tak-Wah Lam[†] Lap-Kei Lee[†] Isaac K.K. To[‡] Prudence W.H. Wong[‡]

1 Introduction

Energy consumption has become an important issue in the study of processor scheduling. Energy reduction can be achieved by allowing a processor to vary the speed dynamically (*dynamic speed scaling*) [2–4, 7, 10] or to enter a sleep state [1, 5, 8]. In the past, these two mechanisms are often studied separately. It is indeed natural to consider an integrated model in which a processor, when awake, can run at any speed $s > 0$, using power $P(s) = s^\alpha + \sigma$, where α is typically 3 [6] and $\sigma > 0$ is a constant.[1] To have zero energy usage, the processor can enter the sleep state, but wake-up requires $\omega > 0$ energy. Irani et al. [9] are the first to consider this integrated model; they studied deadline scheduling in the *infinite speed model* (processor speed can be scaled arbitrarily large). The aim is to minimize the energy for completing all jobs by their deadlines. In this paper, we extend this study to the more realistic *bounded speed model* [7] where a processor has a maximum speed T. With bounded speed, the system may be overloaded and cannot meet all job deadlines. The objective is to maximize the throughput (total size of jobs completed by their deadlines) while using the minimum amount of energy.

Adding a sleep state complicates the nature of speed scaling. To save energy in this model, a scheduler has to be proactive, sometimes forcing the processor to work faster (instead of slower) and to sleep more. In the infinite speed model, it is "safe" to let the processor to oversleep, and rely on extra speed to catch up with the deadlines. Yet this is no longer obvious for a speed bounded processor. In this paper, we present an online algorithm SOA that can exploit speed scaling and a sleep state more effectively. SOA can be considered as the sleep-aware version of the speed scaling algorithms OA [10] and OAT [7].

- In the infinite speed model, SOA (coupled with EDF) completes all jobs and is $(\alpha^\alpha + 2)$-competitive for energy, improving the ratio of Irani et al. [9] (which is $2^{\alpha-1}\alpha^\alpha + 2^{\alpha-1} + 2$).
- The major contribution of SOA is on the bounded speed model. SOA, capped at the maximum speed T, can support the job selection strategy Slow-D (proposed in [2]) to obtain an algorithm that is 4-competitive for throughput and $(\alpha^\alpha + \alpha^2 4^\alpha + 2)$-competitive for energy (note that the throughput ratio is optimal).

[*]Department of Mathematical Informatics, University of Tokyo, Japan. hanxin.mail@gmail.com

[†]Department of Computer Science, University of Hong Kong, Hong Kong. {twlam, lklee}@cs.hku.hk
T.W. Lam is partly supported by HKU Grant 7176104.

[‡]Department of Computer Science, University of Liverpool, UK. {isaacto, pwong}@liverpool.ac.uk
I.K.K. To and P.W.H. Wong are partly supported by EPSRC Grant EP/E028276/1.

[1]s^α is the dynamic power which is due to dynamic switching loss and increases with the processor speed; the static power σ is dissipated due to leakage current and is independent of processor speed.

2 The online algorithm SOA

Jobs are arriving online, with arbitrary sizes and deadlines. SOA dynamically determines the processor speed, as well as when to sleep and wake up. We first define some notations. It is useful to distinguish two types of awake state: *idle* state with zero speed, and *working* state with positive speed. At time t, let $w(t, t')$, for any $t' > t$, be the remaining work of jobs arriving at or before t and with deadlines in $(t, t']$. Define the density $\rho(t, t') = w(t, t')/(t' - t)$ and the highest density $\rho = \max_{t'>t} \rho(t, t')$.

SOA in the infinite speed model. Assuming no sleep state, a scheduler, to save energy, probably runs a job as slow as its deadline allows. When a sleep state is allowed, we want to let the processor sleep more. One possible way is to postpone job execution and work faster later. This idea is first used by PROCRASTINATOR [9], which relies on extra speed to catch up with the delayed jobs. In contrast to PROCRASTINATOR, SOA only mildly increases the speed to catch up with the delayed jobs. SOA prefers to extend an idle (or sleep) period when the highest density ρ is small. If ρ is small after the processor has idled for ω/σ time units, then the processor switches to the sleep state. Later when ρ is big enough, SOA is forced to set a high speed to avoid missing any deadline. On the other hand, in the working state, it is simple to determine the next transition. The processor keeps on working as long as there are unfinished jobs; otherwise switches to the idle state.

Below s_{crit} denotes the critical speed, defined as $(\sigma/(\alpha-1))^{1/\alpha}$.[2]

Algorithm 1 SOA, Sleep-aware Optimal Available

```
Consider any time $t$. Let $\rho$ be the highest density at time $t$.
In working state: If $\rho > 0$, keep working on the job with the earliest deadline (EDF) at speed $\max\{\rho, s_{\text{crit}}\}$; else (i.e., $\rho = 0$) switch to idle state.
In idle state: Let $t' \le t$ be the last time in the working state ($t' = 0$ if undefined). If $\rho \ge s_{\text{crit}}$, switch to working state; Else if $(t - t')\sigma = \omega$, switch to sleep state.
In sleep state: If $\rho \ge s_{\text{crit}}$, switch to working state.
```

Bounded speed model. We will still use SOA, but when SOA returns a speed higher than the speed bound T, we will just use T. In this case, we might not finish all jobs and so a job selection algorithm is required. We adapt the algorithm Slow-D [2], which works when there is no sleep state. Slow-D(SOA) simulates SOA running in the infinite speed model and adopts the speed computed in the simulation (but capped at T), as well as follows the state transition in the simulation. Details of Slow-D can be found in [2].

Theorem 1 (i) *In the infinite speed model,* SOA *(coupled with EDF) can complete all jobs.* (ii) *In the bounded speed model,* Slow-D(SOA) *is* 4*-competitive for throughput.*

2.1 Energy consumption of Slow-D(SOA)

In this section, we sketch the idea of the analysis of the energy of Slow-D(SOA). We denote by OPT the optimal offline algorithm that achieves maximum throughput. We distinguish three types of energy usage: the *working energy* used for processing jobs, the *idle energy* due to the static power σ during idle periods, and the *wake-up energy* which is ω times the number of wake-ups. As observed by Irani et al. [9], the idle energy and wake-up energy contribute only a small factor when compared with the energy used by the optimal offline algorithm.

[2]If a job J of size $w(J)$ is run to completion using speed s, the energy usage is $P(s)w(J)/s$, which is minimized when the speed s satisfies $P(s) = s \times P'(s)$, i.e., $s = (\sigma/(\alpha-1))^{1/\alpha}$. We call this speed the *critical speed* s_{crit}.

Lemma 2 *If the working energy of an algorithm using the same idle and sleep strategy as* SOA *is at most c times that of* OPT, *then its total energy is at most* $\max\{c+2, 4\}$ *times that of* OPT.

We then compare the working energy of Slow-D(SOA) and OPT. The analysis follows the framework of that of OA [7], but using different potential functions. The speed used by Slow-D(SOA) at any time t is the minimum of T and the speed of SOA in the infinite speed model. We denote this speed function as SOAT(t). We let A_{SOAT} and A_{OPT} be the working energy of SOAT and OPT; $A_{\text{SOAT}}(t)$ and $A_{\text{OPT}}(t)$ be the corresponding value up to time t. We give suitable potential functions $\phi(t)$ and $\beta(t)$, such that at a time t_{e} after all job deadlines, $\phi(t_{\text{e}}) = 0$ and $\beta(t_{\text{e}})$ is small; and the following inequality holds at any time t:

$$A_{\text{SOAT}}(t) + \phi(t) - \beta(t) \leq \alpha^{\alpha} A_{\text{OPT}}(t), \tag{1}$$

We can then apply the inequality at t_{e} to obtain $A_{\text{SOAT}} \leq \alpha^{\alpha} A_{\text{OPT}} + \beta(t_{\text{e}})$.

Note that OPT does not aim to complete all jobs. We say a job is *type-1* if OPT completes the job, and *type-0* otherwise. Roughly speaking, $\phi(t)$ captures the difference in progress of SOAT and OPT, while $\beta(t)$ captures the work SOAT would process for type-0 jobs. We can then prove the following lemma. Details are omitted.

Lemma 3 $\beta(t_{\text{e}}) \leq \alpha^2 4^{\alpha} A_{\text{OPT}}(t_{\text{e}})$.

Applying Inequality (1) and Lemma 3, we have $A_{\text{SOAT}} \leq (\alpha^{\alpha} + \alpha^2 4^{\alpha}) A_{\text{OPT}}$. Together with Lemma 2, Slow-D(SOA) is $\max\{\alpha^{\alpha} + \alpha^2 4^{\alpha} + 2, 4\}$-competitive, i.e., $(\alpha^{\alpha} + \alpha^2 4^{\alpha} + 2)$-competitive. If $T = \infty$, both SOA and OPT complete all jobs, i.e., there is no type-0 job, and $\beta(t_{\text{e}}) = 0$, thus SOA is $\max\{\alpha^{\alpha} + 2, 4\}$-competitive in the infinite speed model.

Theorem 4 (i) *In the bounded speed model,* Slow-D(SOA) *is* $(\alpha^{\alpha} + \alpha^2 4^{\alpha} + 2)$*-competitive for energy.* (ii) *In the infinite speed model,* SOA *is* $\max\{\alpha^{\alpha} + 2, 4\}$*-competitive for energy.*

References

[1] J. Augustine, S. Irani, and C. Swany. Optimal power-down strategies. In *Proc. FOCS*, pages 530–539, 2004.

[2] N. Bansal, H. L. Chan, T. W. Lam, and L. K. Lee. Scheduling for speed bounded processors. In *Proc. ICALP*, pages 409–420, 2008.

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

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

[5] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Tran. VLSI*, 8(3):299–316, 2000.

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

[7] 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 *Proc. SODA*, pages 795–804, 2007.

[8] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Tran. Embeded Comp. Sys.*, 2(3):325–346, 2003.

[9] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *TALG*, 3(4), 2007.

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