Brief Announcement: Reclaiming the Energy of a Schedule, Models and Algorithms

Guillaume Aupy, Anne Benoit, Fanny Dufossé, Yves Robert

Brief Announcement: Reclaiming the Energy of a Schedule, Models and Algorithms *

Guillaume Aupy, Anne Benoit, Fanny Dufossé and Yves Robert
LIP, École Normale Supérieure de Lyon, France
{Guillaume.Aupy|Anne.Benoit|Fanny.Dufosse|Yves.Robert}@ens-lyon.fr

ABSTRACT

We consider a task graph to be executed on a set of processors. We assume that the mapping is given, say by an ordered list of tasks to execute on each processor, and we aim at optimizing the energy consumption while enforcing a prescribed bound on the execution time. While it is not possible to change the allocation of a task, it is possible to change its speed. We study the complexity of the problem for different models: continuous speeds, discrete modes, distributed either arbitrarily or regularly, and VDD-hopping.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms

Performance, Theory, Algorithms.

Keywords

Energy models, complexity, bi-criteria optimization, algorithms, scheduling.

1. INTRODUCTION

The *energy consumption* of computational platforms has recently become a critical problem, both for economic and environmental reasons. Their power consumption is the sum of a static part (the cost for a processor to be turned on) and a dynamic part, which is a strictly convex function of the processor speed. More precisely, a processor running at speed s dissipates s^3 watts [4, 5] per time-unit, hence consumes $s^3 \times t$ joules when operated during t units of time.

Energy-aware scheduling aims at minimizing the energy consumed during the execution of the target application. Obviously, it makes sense only if it is coupled with some performance bound to achieve, otherwise, the optimal solution always is to run each processor at the slowest possible speed. In this paper, we investigate energy-aware scheduling strategies for executing a task graph on a set of processors. The main originality is that we assume that the mapping of the task graph is given, say by an ordered list of tasks to execute on each processor. There are many situations in which this problem is important, such as optimizing for legacy applications, or accounting for affinities between tasks and resources, or even when tasks are pre-allocated, for example for security reasons.

Optimization problem. Consider an application task graph $\mathcal{G} = (V, \mathcal{E})$, with $n = |V|$ tasks, $V = \{T_1, T_2, \ldots, T_n\}$: $\mathcal{E}$ denotes the precedence edges between tasks. Task T_i has a cost w_i for $1 \leq i \leq n$. We assume that the tasks in $\mathcal{G}$ have been allocated onto a parallel platform made up of identical processors. The *execution graph* generated by this allocation is $G = (V, E)$, with an augmented set of edges E: $\mathcal{E} \subseteq E$, and if T_1 and T_2 are executed successively, in this order, on the same processor, then $(T_1, T_2) \in E$. The goal is to minimize the energy consumed during the execution while enforcing a deadline D on the execution time. We formalize this MinEnergy(G,D) optimization problem in the simpler case where each task is executed at constant speed (valid for all models but the VDD-hopping one). Let d_i be the duration of the execution of task T_i, t_i its completion time, and s_i the speed at which it is executed.

$$\begin{array}{ll} \text{Minimize} & \sum_{i=1}^{n} s_i^3 \times d_i \\ \text{subject to} & \text{(i) } w_i = s_i \times d_i \text{ for each } T_i \in V \\ & \text{(ii) } t_i + d_j \leq t_j \text{ for each } (T_i, T_j) \in E \\ & \text{(iii) } t_i \leq D \text{ for each } T_i \in V \end{array} \quad (1)$$

We have $d_i = w_i/s_i$, hence a geometric problem in the non-negative variables t_i and $1/s_i$, with linear constraints and objective function rewritten as $\sum_{i=1}^{n} (1/s_i)^{-2} \times w_i$.

Energy models. In all models, when a processor operates at speed s during d time-units, the corresponding consumed energy is $s^3 \times d$, which is the dynamic part of the energy consumption [4, 5]. We do not take static energy into account, because all processors are up and alive during the whole execution. We now detail the possible speed values in each energy model, which should be added as a constraint in Equation (1).

Continuous: processors can have arbitrary speeds, from 0 to a maximum value s_{max}, and a processor can change its speed at any time during execution. This model is unrealistic but theoretically appealing [2].

Discrete: processors have a set of possible speed values, or modes, denoted as $s_1, ..., s_m$. There is no assumption on the range and distribution of these modes. The speed of a processor cannot change during the computation of a task, but it can change from task to task [7].

*The authors are with Université de Lyon, France. A. Benoit and Y. Robert are with the Institut Universitaire de France. This work was supported in part by the ANR *StochaGrid* and *RESCUE* projects.

SPAA'11, June 4–6, 2011, San Jose, California, USA.
ACM 978-1-4503-0743-7/11/06.

VDD-HOPPING: a processor can run at different speeds as in the previous model $(s_1, ..., s_m)$, but it can also change its speed during a computation. Any rational speed can be simulated [6]. The energy consumed during the execution of one task is the sum, on each time interval with constant speed s, of the energy consumed during this interval at speed s.

INCREMENTAL: we introduce a value δ that corresponds to the minimum permissible speed (i.e., voltage) increment. Possible speed values are obtained as $s = s_{min} + i \times \delta$, where i is an integer such that $0 \leq i \leq \frac{s_{max} - s_{min}}{\delta}$. Admissible speeds lie in the interval $[s_{min}, s_{max}]$. The different modes are spread regularly between $s_1 = s_{min}$ and $s_m = s_{max}$, instead of being arbitrarily chosen. This is intended as the modern counterpart of a potentiometer knob!

2. RESULTS

All proofs, algorithms, and related work can be found in the companion research report [1].

2.1 The CONTINUOUS model

THEOREM 1. *When G is a fork graph with $n+1$ tasks $T_0, T_1, \ldots, T_n$, where T_0 is the source, the optimal solution to* MINENERGY*(G,D) is to execute T_0 at speed*

$$s_0 = \frac{\left(\sum_{i=1}^{n} w_i^3\right)^{\frac{1}{3}} + w_0}{D} ,$$

and T_i (for $1 \leq i \leq n$) at speed

$$s_i = s_0 \times \frac{w_i}{\left(\sum_{i=1}^{n} w_i^3\right)^{\frac{1}{3}}} , \qquad \text{if} \quad s_0 \leq s_{max}.$$

Otherwise, T_0 should be executed at speed $s_0 = s_{max}$, and the other speeds are $s_i = \frac{w_i}{D'}$, with $D' = D - \frac{w_0}{s_{max}}$, if they do not exceed s_{max}, otherwise there is no solution.

THEOREM 2. MINENERGY*(G,D) can be solved in polynomial time when G is a tree, or a series-parallel graph (in the latter case, assuming $s_{max} = +\infty$).*

For arbitrary execution graphs, we have a geometric programming problem (see [3, Section 4.5]) for which efficient numerical schemes exist. However, as illustrated on simple fork graphs, the optimal speeds are not expected to be rational numbers but instead arbitrarily complex expressions (we have the cubic root of the sum of cubes for forks, and nested expressions of this form for trees). We do not know how to encode such numbers in polynomial size of the input (the rational task weights and the execution deadline). Still, we can always solve the problem numerically and get fixed-size numbers which are good approximations of the optimal values.

2.2 Discrete models

THEOREM 3. *With the* VDD-HOPPING *model,* MINENERGY*(G,D) can be solved in polynomial time (via linear programming).*

THEOREM 4. *With the* INCREMENTAL *model (and hence the* DISCRETE *model),* MINENERGY*(G,D) is NP-complete.*

THEOREM 5. *With the* INCREMENTAL *model, for any integer $K > 0$, the* MINENERGY*(G,D) problem can be approximated within a factor $(1 + \frac{\delta}{s_{min}})^2 \times (1 + \frac{1}{K})^2$, in a time polynomial in the size of the instance and in K.*

PROPOSITION 1.

- *For any integer $\delta > 0$, any instance of* MINENERGY*(G,D) with the* CONTINUOUS *model can be approximated within a factor $(1+\frac{\delta}{s_{min}})^2$ in the* INCREMENTAL *model with speed increment δ.*
- *For any integer $K > 0$, any instance of* MINENERGY*(G,D) with the* DISCRETE *model can be approximated within a factor $(1 + \frac{\alpha}{s_1})^2 \times (1 + \frac{1}{K})^2$, with $\alpha = \max_{1 \leq i < m}\{s_{i+1} - s_i\}$, in a time polynomial in the size of the instance and in K.*

3. CONCLUSION

We have assessed the tractability of a classical scheduling problem, with task preallocation, under various energy models. We have given several results related to CONTINUOUS speeds. However, while these are of conceptual importance, they cannot be achieved with physical devices, and we have analyzed several models enforcing a bounded number of achievable speeds, a.k.a. modes. In the classical DISCRETE model, admissible speeds can be irregularly distributed, which motivates the VDD-HOPPING approach that mixes two consecutive modes optimally. While computing optimal speeds is NP-hard with discrete modes, it has polynomial complexity when mixing speeds. Intuitively, the VDD-HOPPING approach allows for smoothing out the discrete nature of the modes. An alternate (and simpler in practice) solution to VDD-HOPPING is the INCREMENTAL model, where one sticks with unique speeds during task execution as in the DISCRETE model, but where consecutive modes are regularly spaced. Such a model can be made arbitrarily efficient, according to our approximation results. Altogether, this paper has laid the theoretical foundations for a comparative study of energy models.

4. REFERENCES

[1] G. Aupy, A. Benoit, F. Dufossé, and Y. Robert. Reclaiming the energy of a schedule: models and algorithms. Research report, INRIA, Lyon, France, Apr. 2011. Available at `http://graal.ens-lyon.fr/~abenoit`.

[2] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):1 – 39, 2007.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] A. P. Chandrakasan and A. Sinha. JouleTrack: A Web Based Tool for Software Energy Profiling. In *Design Automation Conference*, pages 220–225, 2001.

[5] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED*, pages 197–202. ACM Press, 1998.

[6] S. Miermont, P. Vivet, and M. Renaudin. A Power Supply Selector for Energy- and Area-Efficient Local Dynamic Voltage Scaling. In *Integrated Circuit and System Design*, LNCS 4644, pages 556–565, 2007.

[7] T. Okuma, H. Yasuura, and T. Ishihara. Software energy reduction techniques for variable-voltage processors. *IEEE Design Test of Computers*, 18(2):31–41, 2001.