Energy-aware scheduling: models and complexity results

Guillaume Aupy

Energy-aware scheduling: models and complexity results

Guillaume Aupy (1st year PhD student)
LIP, École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Advisors: Anne Benoit and Yves Robert
Email: {Guillaume.Aupy|Anne.Benoit|Yves.Robert}@ens-lyon.fr

***Abstract*—This paper presents several energy-aware scheduling algorithms whose design is optimized for different speed models. Dynamic Voltage and Frequency Scaling (DVFS) is a model frequently used to reduce the energy consumption of a schedule, but it has negative effect on reliability. While the reliability of a schedule can sometimes be neglected (battery powered systems such as cell-phones or personal computers), it becomes extremely important when considering massively parallel architectures (petascale, exascale).**

In this work, we consider the problem of minimizing the energy within a makespan constraint. Additionally, we consider two models, one that takes into account a reliability constraint, and one that does not. 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. Rather than using a local approach such as backfilling, we consider the problem as a whole and study the impact of several speed variation models on its complexity. To improve the reliability of a schedule while reducing the energy consumption, we allow for the re-execution of some tasks. We present several results in that framework, as well as future research plans.

I. Introduction

Energy-aware scheduling has proven an important issue in the past decade, both for economical and environmental reasons. This holds true for traditional computer systems, not even to speak of battery-powered systems. To help reduce energy dissipation, processors can run at different speeds. We call *dynamic voltage and frequency scaling* (DVFS) the common technique to decrease the energy consumption by changing the execution speed of the processor. 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, so that the execution of a given amount of work costs more power if a processor runs in a higher mode [8]. More precisely, a processor running at speed f dissipates f^3 watts [10] per time-unit, hence consumes $f^3 \times d$ joules when operated during d units of time. Faster speeds allow for a faster execution, but they also lead to a much higher (supra-linear) power consumption.

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. Hence we also consider the makespan of an execution, that is the total execution time.

Reliability and fault-tolerance have always been major concerns in computer science design. Zhu et al. [14] showed that energy management through DVFS has significant effects on reliability: for critical applications, the goal of saving energy by reducing execution speed must be carefully weighted with the goal of maintaining a certain level of reliability. Re-execution is a technique to increase the reliability of a process. It consists in re-executing each task that does not meet the reliability constraint on the same processor. A schedule specifies which tasks are re-executed, as well as the speed at which each task is executed (and possibly re-executed).

In this work we consider two types of problems: *bi-criteria* where one should minimize the energy consumption while matching a deadline bound, and *tri-criteria* where a constraint on reliability is added to the bi-criteria problem. For both problems, we consider different energy models. In the CONTINUOUS model, an execution speed can take any arbitrary real value; this model is appealing for theoretical work. On the contrary, the discrete models (discrete set of possible speeds) are closer to what exists and is actually implemented.

The paper is organized as follows. We start with a formal description of the framework and of the energy models in Section II. The next two sections constitute the heart of the work: in Section III, we provide theoretical results for the continuous speeds as well as a brief description of the heuristics that we implemented. In Section IV, we assess the complexity of the problem with all the discrete models, and provide some theoretical results. Finally we conclude and give research orientations in Section V.

II. Models

The application consists of n tasks $\{T_1, \cdots, T_n\}$ with dependence constraints, hence forming a directed acyclic task graph (DAG). For $1 \leq i \leq n$, task T_i has a weight w_i, that corresponds to the computation requirement of the task. The goal is to schedule the DAG on a platform of p identical processors. The traditional scheduling objective consists in minimizing the execution time, or makespan, to process the DAG. In order to do so, the DAG is mapped on the processors and an execution speed is assigned to each task of the DAG. Because the problem of finding a schedule that match the

makespan constraint is NP-complete, we consider that the DAG is already mapped on the processors. The schedule then consists in choosing the number of executions of each task (in case of re-execution), and the speeds at which these executions will happen. This makes sense in many situations, such as optimizing for legacy applications, or accounting for affinities between tasks and resources, or even when tasks are pre-allocated [12], for example for security reasons. Furthermore, our work can be coupled with classical list-scheduling heuristics that map the DAG on the platform.

We present novel theoretical work to minimize the energy consumption under the constraints of both a reliability threshold and a deadline bound under different speed models. These criteria are formally defined in this section. First we define the different speed models.

Speed models

The following speed models are relevant in different contexts:

- CONTINUOUS *model:* processors can have arbitrary speeds, from $f_{\min}$ to a maximum value $f_{\max}$, and a processor can change its speed at any time during execution. The CONTINUOUS model is used mainly for theoretical studies, which are then useful for relevant algorithms [5].
- DISCRETE *model:* processors have a set of possible speed values, denoted as $f_1, ..., f_m$. There is no assumption on the range and distribution of these speeds. The speed of a processor cannot change during the computation of a task, but it can change from task to task. This is the most commonly used model [9].
- VDD-HOPPING *model:* a processor can run at different speeds $f_1, ..., f_m$, as in the previous model, but it can also change its speed during a computation. The energy consumed during the execution of one task is the sum, on each time interval with constant speed f, of the energy consumed during this interval at speed f. It was shown that significant power can be saved by using two distinct voltages, and architectures using this principle have been developed [6].
- INCREMENTAL *model:* we introduce a value δ that corresponds to the minimum permissible speed increment [2]. That means that possible speed values are obtained as $f = f_{\min} + i \times \delta$, where i is an integer such that $0 \leq i \leq \frac{f_{\max} - f_{\min}}{\delta}$. Admissible speeds lie in the interval $[f_{\min}, f_{\max}]$. This new model aims at capturing a realistic version of the DISCRETE model, where the different modes are spread regularly between $f_1 = f_{\min}$ and $f_m = f_{\max}$, instead of being arbitrarily chosen. It is intended as the modern counterpart of a potentiometer knob.

The last three models are the so-called *discrete* models.

Optimization criteria

We consider three different optimization criteria: makespan, reliability, and energy.

a) Makespan: The makespan of a schedule is its total execution time. The execution time of a task T_i of weight w_i at speed f_i is $d_i = \frac{w_i}{f_i}$. The first task is scheduled at time 0, so that the makespan of a schedule is simply the maximum time at which one of the processors finishes its computations. We consider a deadline bound D, which is a constraint on the makespan. The makespan of a schedule should not be greater than this bound.

b) Reliability: Unfortunately, blindly applying DVFS for energy savings may cause significant degradation in system reliability. There are some systems where reliability may be irrelevant, however with the advent of supercomputers (petascale, exascale platforms), taking reliability into account when considering energy management becomes a necessity. It was shown that DVFS has a direct and negative effect on transient fault-rates [14]. In order to make up for the loss in reliability due to the energy efficiency, different models have been proposed for fault-tolerance.

- *Re-execution:* it consists in re-executing a task that does not meet the reliability constraint, see [14].
- *Replication:* this model, studied in [1], consists in executing the same task on p different processors simultaneously, in order to meet the reliability constraints.
- *Checkpointing:* this model, studied in [11], consists in "saving" the work done at some certain points of the work, hence reducing the amount of work lost when a failure occurs.

This work focuses on the re-execution model. The reliability of a task T_i executed at speed f can be defined as in [3]:

$$R_i(f) = 1 - \lambda_0 \, e^{d \frac{f_{\max} - f}{f_{\max} - f_{\min}}} \times \frac{w_i}{f}, \tag{1}$$

where $f_{\min} \leq f \leq f_{\max}$ is the processing speed, the exponent $d \geq 0$ is a constant, indicating the sensitivity of fault rates to DVFS, and λ_0 is the average fault rate corresponding to $f_{\max}$.

We want the reliability R_i of each task T_i to be greater than a given threshold, namely $R_i(f_{\texttt{rel}})$, hence enforcing a local constraint dependent on the task $R_i \geq R_i(f_{\texttt{rel}})$. If task T_i is executed only once at speed f, then the reliability of T_i is $R_i = R_i(f)$. Since the reliability increases with speed, we must have $f \geq f_{\texttt{rel}}$ to match the reliability constraint. If task T_i is re-executed (speeds $f^{(1)}$ and $f^{(2)}$), then the execution of T_i is successful if and only if both attempts do not fail, so that the reliability of T_i is $R_i = 1 - (1 - R_i(f^{(1)}))(1 - R_i(f^{(2)}))$, and this quantity should be at least equal to $R_i(f_{\texttt{rel}})$.

c) Energy: The goal is to the minimize the energy consumed during the execution. In all models, when a processor operates at speed f during t time-units, the corresponding consumed energy is $f^3 \times t$, which is the dynamic part of the energy consumption, following the classical models of the literature [4]. Note that we do not take static energy into account, because all processors are up and alive during the whole execution. The energy consumed by task T_i executed at speed f is $E_i = f^3 \frac{w_i}{f} = w_i f^2$.

When a task is scheduled to be re-executed at two different speeds $f^{(1)}$ and $f^{(2)}$, we always account for both executions, i.e., $E_i = w_i(f^{(1)2} + f^{(2)2})$, even when the first execution is

successful. In other words, we consider a worst-case execution scenario, and the deadline D must be matched even in the case where all tasks that are re-executed fail during their first execution. The total energy consumption is $E = \sum_{i=1}^{n} E_i$.

Optimization problems

Consider an application task graph $\mathcal{G} = (V, \mathcal{E})$, and p homogeneous processors. For each speed model, we define:

Definition 1. BI-CRIT. Given an application graph $\mathcal{G} = (V, \mathcal{E})$, mapped onto p homogeneous processors, BI-CRIT is the problem of deciding at which speed each task should be processed, in order to minimize the total energy consumption E, subject to the deadline bound D.

Definition 2. TRI-CRIT. Given an application graph $\mathcal{G} = (V, \mathcal{E})$, mapped onto p homogeneous processors, TRI-CRIT is the problem of deciding which tasks should be re-executed and at which speed each execution of a task should be processed, in order to minimize the total energy consumption E, subject to the deadline bound D and to the reliability constraints $R_i \geq R_i(f_{\mathtt{rel}})$ for each $T_i \in V$.

We point out that TRI-CRIT brings dramatic complications: in addition to choosing the speed of each task, as in BI-CRIT, we also need to decide which subset of tasks should be re-executed (and then choose both execution speeds). Few authors have tackled such a challenging problem.

III. THE CONTINUOUS MODEL

With the CONTINUOUS model, processor speeds can take any value between $f_{\min}$ and $f_{\max}$. We were able to show some theoretical results with or without reliability constraints.

The BI-CRIT *problem (see [2]):*

We provide optimal speed values for special execution graph structures (trees, series-parallel graphs), expressed as closed form algebraic formulas. These values may be irrational. We give an example for the graphs of type *Fork*:

Theorem (fork graphs)**.** *When G is a fork execution graph with $n+1$ tasks $T_0, T_1, \ldots, T_n$, the optimal solution to* BI-CRIT *is the following:*

- *the execution speed of the source (resp. sink) T_0 is*

$$f_0 = \frac{\left(\sum_{i=1}^{n} w_i^3\right)^{\frac{1}{3}} + w_0}{D} \ ;$$

- *for the other tasks T_i, $1 \leq i \leq n$, we have*

$$f_i = f_0 \times \frac{w_i}{\left(\sum_{i=1}^{n} w_i^3\right)^{\frac{1}{3}}} \text{ if } f_0 \leq f_{\max} \ .$$

Otherwise, T_0 should be executed at speed $f_0 = f_{\max}$, and the other speeds are $f_i = \frac{w_i}{D'}$, with $D' = D - \frac{w_0}{f_{\max}}$, if they do not exceed $f_{\max}$. Otherwise there is no solution.

If no speed exceeds $f_{\max}$, the corresponding energy consumption is

$$\mathbf{E}_{\textit{fork}}(G, D) = \frac{\left(\left(\sum_{i=1}^{n} w_i^3\right)^{\frac{1}{3}} + w_0\right)^3}{D^2} \ .$$

We formulate the problem for general DAGs as a geometric programming problem (see [7, Section 4.5]) for which efficient numerical schemes exist.

The TRI-CRIT *problem (see [3]):*

We show that this problem is NP-hard even in the simple case when there is only one processor and a set of tasks mapped on this processor (linear chain). However, we were able to find an optimal strategy for the case of a linear chain: first slow the execution of all tasks equally, then choose the tasks to be re-executed. Based on this strategy we were able to develop a first set of heuristics very efficient on linear-chain-like DAGs. We were also able to find a polynomial time algorithm to solve the problem for a fork. We point out that it is much more difficult to obtain closed-form formulas than for the BI-CRIT problem, even if all the tasks of the fork have the same weight. This polynomial-time algorithm is based on a totally different strategy than for linear chains: those highly parallelizable tasks should be preferred when allocating time slots for re-execution or deceleration. Based on this strategy, we were able to develop a second set of heuristics, very efficient on highly-parallelizable DAGs.

The heuristics that we developed are based on the failure probability, the task weights, and the processor speeds. They aim at minimizing the energy consumption while enforcing reliability and deadline constraints. After running our two sets of heuristics on a wide class of problem instances, we have identified two heuristics that are complementary, and that together are able to produce good results on most instances. Altogether, taking the best result out of those two heuristics always gives the best result over all simulations and is a good candidate for competitiveness. The constructive results are not theoretically proven to be approximation algorithms, though they are backed up by theoretic intuitions and experimental evaluation.

IV. DISCRETE MODELS

In this section, we present complexity results on the three energy models with a finite number of possible speeds.

The BI-CRIT *problem (see [2]):*

- *With the* VDD-HOPPING *model,* we show that this problem can be solved in polynomial time using a linear program.
- *With the* INCREMENTAL *model (and hence the* DISCRETE *model),* we show that this problem is NP-complete. However we were able to give polynomial time approximation algorithms (for instance, with the INCREMENTAL model, we can approximate the solution within a factor $(1 + \frac{\delta}{f_{\min}})^2 (1 + \frac{1}{K})^2$, in a time polynomial in the size of the instance and in K).

The TRI-CRIT *problem (see [3]):*

With reliability we focused on the VDD-HOPPING model which had a polynomial time solution without reliability. We first showed that only two different speeds are needed for the execution of a task under the VDD-HOPPING model (this is a well known result when studying BI-CRIT, which still holds

true with reliability). Then we showed that with the VDD-HOPPING model, TRI-CRIT is NP-complete (while BI-CRIT problem was in P).

Finally, we could easily adapt the heuristics for the CONTINUOUS model to the VDD-HOPPING model: for a solution given by a heuristic for the CONTINUOUS model, if a task should be executed at the continuous speed f, then we would execute it at the two closest discrete speeds that bound f, while matching the execution time and reliability for this task. There remains to quantify the performance loss incurred by the latter constraints.

V. CONCLUSION AND RESEARCH ORIENTATIONS

In this work, 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. In the classical DISCRETE model that arises from DVFS techniques, admissible speeds can be irregularly distributed, which motivates the VDD-HOPPING approach that mixes two consecutive speeds optimally. While the BI-CRIT problem is NP-hard with discrete speeds, it has polynomial complexity when mixing speeds. Intuitively, the VDD-HOPPING approach allows for smoothing out the discrete nature of the speeds. 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. Coupling this model with the classical reliability model used in [13], we have been able to formulate the TRI-CRIT problem: how to minimize the energy consumed given a deadline bound and a reliability constraint? The "antagonistic" relation between speed and reliability renders this tri-criteria problem much more challenging than the standard bi-criteria version. We have stated two variants of the problem, for processor speeds obeying either the CONTINUOUS or the VDD-HOPPING model. We have assessed the intractability of this tri-criteria problem, even in the case of a single processor. A very encouraging point is the fact that we were able to develop two very complementary heuristics for the CONTINUOUS TRI-CRIT problem, efficient on different sort of DAGs.

Future work involves several promising directions. On the theoretical side, it would be very interesting to prove a competitive ratio for the heuristic that takes the best out of our heuristics for the CONTINUOUS TRI-CRIT problem. However, this is quite a challenging work for arbitrary DAGs, and one may try to design approximation algorithms only for special graph structures, e.g. series-parallel graphs.

The previous heuristics are solving the problem where the mapping is given. However, as said earlier, they can provide efficient solutions for the general problem associated with a list-scheduling algorithm. When implementing those heuristics, we coupled them with a critical-path list-scheduling algorithm. It would be important to assess the impact of the list scheduling heuristic that precedes the energy-reduction heuristic. In other words, the classical critical-path list-scheduling heuristic, which is known to be efficient for deadline minimization, may well be superseded by another heuristic that trades-off execution time, energy and reliability when mapping ready tasks to processors. Such a study could open new avenues for the design of multi-criteria list-scheduling heuristics.

Finally, we point out that energy reduction and reliability will be even more important objectives with the advent of massively parallel platforms, made of a large number of clusters of multi-cores. More efficient solutions to the tri-criteria optimization problem (deadline, energy, reliability) could be achieved through combining replication with re-execution. A promising (and ambitious) research direction would be to search for the best trade-offs that can be achieved between these techniques that both increase reliability, but whose impact on execution time and energy consumption is very different.

REFERENCES

[1] I. Assayad, A. Girault, and H. Kalla. Tradeoff exploration between reliability power consumption and execution time. In *Proc. of Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, Washington, DC, USA, 2011. IEEE CS Press.

[2] G. Aupy, A. Benoit, F. Dufossé, and Y. Robert. Reclaiming the energy of a schedule: models and algorithms. Research Report 7598, INRIA, France, Apr. 2011. Available at http://gaupy.org/?paper. Short version appeared in SPAA'11.

[3] G. Aupy, A. Benoit, and Y. Robert. Energy-aware scheduling under reliability and makespan constraints. Research Report 7757, INRIA, France, Oct. 2011. Available at gaupy.org/?paper.

[4] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Proc. of Int. Parallel and Distributed Processing Symposium (IPDPS)*, pages 113–121. IEEE CS Press, 2003.

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

[6] E. Beigne, F. Clermidy, S. Miermont, Y. Thonnart, A. Valentian, and P. Vivet. A Localized Power Control mixing hopping and Super Cut-Off techniques within a GALS NoC. In *Proceedings of ICICDT 2008, the IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, pages 37–42, June 2008.

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

[8] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. In *Proc. of Int. Parallel and Distributed Processing Symposium (IPDPS)*, page 340, Los Alamitos, CA, USA, 2006. IEEE CS Press.

[9] Intel XScale technology. http://www.intel.com/design/intelxscale.

[10] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proc. of Int. Symposium on Low Power Electronics and Design (ISLPED)*, pages 197–202. ACM Press, 1998.

[11] R. Melhem, D. Mosse, and E. Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53:2004, 2003.

[12] V. J. Rayward-Smith, F. W. Burton, and G. J. Janacek. Scheduling parallel programs assuming preallocation. In P. Chrétienne, E. G. Coffman Jr., J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its Applications*. John Wiley and Sons, 1995.

[13] S. M. Shatz and J.-P. Wang. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Transactions on Reliability*, 38:16–27, 1989.

[14] D. Zhu, R. Melhem, and D. Mosse. The effects of energy management on reliability in real-time embedded systems. In *Proc. of IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pages 35–40, Washington, DC, USA, 2004. IEEE CS Press.