Energy-aware checkpointing of divisible tasks with soft or hard deadlines

Guillaume Aupy, Anne Benoit, Rami Melhem, Paul Renaud-Goud, Yves Robert

Energy-aware checkpointing of divisible tasks with soft or hard deadlines

Guillaume Aupy*, Anne Benoit*, Rami Melhem‡, Paul Renaud-Goud*§ and Yves Robert*¶
*LIP - ENS Lyon, France
‡University of Pittsburgh, USA
§LaBRI - Université de Bordeaux, France
¶University of Tennessee, Knoxville USA

***Abstract*—In this paper, we aim at minimizing the energy consumption when executing a divisible workload under a bound on the total execution time, while resilience is provided through checkpointing. We discuss several variants of this multi-criteria problem. Given the workload, we need to decide how many chunks to use, what are the sizes of these chunks, and at which speed each chunk is executed. Furthermore, since a failure may occur during the execution of a chunk, we also need to decide at which speed a chunk should be re-executed in the event of a failure. The goal is to minimize the expectation of the total energy consumption, while enforcing a deadline on the execution time, that should be met either in expectation (soft deadline), or in the worst case (hard deadline). For each problem instance, we propose either an exact solution, or a function that can be optimized numerically.**

I. Introduction

Divisible load scheduling has been extensively studied in the past years [6], [11]. For divisible applications, the computational workload can be divided into an arbitrary number of chunks, whose sizes can be freely chosen by the user. Such applications occur for instance in the processing of very large data files, e.g., signal processing, linear algebra computation, or DNA sequencing. Traditionally, the goal is to minimize the makespan of the application, i.e., the total execution time.

Nowadays, high performance computing is facing a major challenge with the increasing frequency of failures [10]. There is a need to use fault tolerance or resilience mechanisms to ensure the efficient progress and correct termination of the applications in the presence of failures. A well-established method to deal with failures is *checkpointing*: a checkpoint is taken at the end of the execution of each chunk. During the checkpoint, we check for the accuracy of the result; if the result is not correct, due to a transient failure (such as a memory error or software error), the chunk is re-executed. This model with transient failures is one of the most used in the literature, see for instance [19], [9].

978-1-4799-0623-9/13/$31.00 ©2013 IEEE

Furthermore, energy-awareness is now recognized as a first-class constraint in the design of new scheduling algorithms. To help reduce energy dissipation, current processors from AMD, and Intel allow the speed to be set dynamically, using a dynamic voltage and frequency scaling technique (DVFS). Indeed, a processor running at speed s dissipates s^3 watts per unit of time [5]. We therefore focus on two objective functions: execution time and energy consumption, while resilience is ensured through checkpointing. More precisely, we aim at minimizing energy consumption, including that of checkpointing and re-execution in case of failure, while enforcing a bound on execution time.

Given a workload W, we need to decide how many chunks to use, and of which sizes. Using more chunks leads to a higher checkpoint cost, but smaller chunks imply less computation loss (and less re-execution) when a failure occurs. We assume that a chunk can fail only once, i.e., we re-execute each chunk at most once. Indeed, the probability that a fault would strike during both the first execution and the re-execution is negligible. The accuracy of this assumption is discussed in the companion research report [3].

Due to the probabilistic nature of failure hits, it is natural to study the expectation $\mathbb{E}(E)$ of the energy consumption, because it represents the average cost over many executions. As for the bound D on execution time (the deadline), there are two relevant scenarios: either we enforce that this bound is a *soft deadline* to be met in expectation, or we enforce that this bound is a *hard deadline* to be met in the worst case. The former scenario corresponds to flexible environment where task deadlines can be viewed as average response times [7], while the latter scenario corresponds to real-time environments where task deadlines are always strictly enforced [15]. In both scenarios, we have to determine the number of chunks, their sizes, and the speed at which to execute (and possibly re-execute) every chunk.

Our first contribution is to formalize this important multi-objective problem. The general problem consists of finding n, the number of chunks, as well as the speeds for the execution and the re-execution of each chunk, both for soft and hard deadlines. We identify and discuss two important sub-cases that help tackling the most general problem instance: (i) a single chunk (the task is atomic); and (ii) re-execution speed is always identical to the first execution speed. The second contribution is a comprehensive study of all problem instances; for each instance, we propose either an exact solution, or a function that can be optimized numerically (using a computer-algebra software).

The rest of the paper is organized as follows. First we discuss related work in Section II. The model and the optimization problems are formalized in Section III. We first focus in Section IV on the simpler case of an atomic task, i.e., with a single chunk. The general problem with multiple chunks, where we need to decide for the number of chunks and their sizes, is discussed in Section V. Finally, we provide some concluding remarks and future research directions in Section VI.

II. Related work

Dynamic power management through voltage/frequency scaling [16] utilizes the slack in a given computation to reduce energy consumption while checkpointing.The authors of [8], [12] utilize that slack to improve the reliability of the computation. Hence, it is natural to explore the interplay of power management and fault tolerance [13], when both techniques result in delaying the completion time of tasks, thus resulting in a tradeoff between power consumption, reliability and performance. This tri-criteria optimization problem has been explored by many researchers, especially in real-time and embedded systems where the completion time of a task is as important as the reliability of its result.

The power/reliability/performance tradeoff has been explored from many different angles. In [17], an adaptive scheme is presented to place checkpoints based on the expected frequency of faults and is combined with dynamic speed scaling depending on the actual occurrence of faults. Similarly, in [13], the placement of checkpoints is chosen in a way that minimizes the total energy consumption assuming that the slack reserved for rollback recovery is used for speed scaling if faults do not occur. In [19], the effect of frequency scaling on the fault rate was considered and incorporated into the optimization problem. In [18], the study of the tri-criteria optimization was extended to the case of multiple tasks executing on the same processor. In [14], a constraint logic programming-based approach is presented to decide for the voltage levels, the start times of processes and the transmission times of messages, in such a way that transient faults are tolerated, timing constraints are satisfied and energy is minimized.

Recently, off-line scheduling heuristics that consider the three criteria were presented for systems where active replication, rather than fault recovery, is used to enhance reliability [1]. Selective re-execution of some tasks were considered in [4] to achieve a given level of reliability while minimizing energy, when tasks graphs are scheduled on multiprocessors with hard deadlines. Approximation algorithms for particular types of task graphs were presented to efficiently solve the same problem in [2].

In this work, we consider two types of deadlines that are commonly used for real-time tasks; hard and soft deadlines. In hard real-time systems [15], deadlines should be strictly met and any computation that does not meet its deadline is not useful to the system. These systems are built to cope with worst-case scenarios, especially in critical applications where catastrophic consequences may result from missing deadlines. Soft real-time systems [7] are more flexible and are designed to adapt to system changes that may prevent the meeting of the deadline. They are suited to novel applications such as multimedia and interactive systems. In these systems, it is desired to reduce the expected completion time rather than to meet hard deadlines.

III. Framework

Given a workload W, the problem is to divide W into a number of chunks and to decide at which speed each chunk is executed. In case of a transient failure during the execution of one chunk, this chunk is re-executed, possibly at a different speed.

Model. Consider first the case of a single chunk (or atomic task) of size W, denoted as SC. We execute this chunk on a processor that can run at several speeds. We assume continuous speeds, i.e., the speed of execution can take an arbitrary positive real value. The execution is subject to failure, and resilience is provided through the use of checkpointing. The overhead induced by checkpointing is twofold: execution time T_C, and energy consumption E_C.

We assume that failures strike with uniform distribution, hence the probability that a failure occurs during an execution is linearly proportional to the length of this execution. Consider the first execution of a task of size W executed at speed s: the execution time is $T_{\text{exec}} = W/s + T_C$, hence the failure probability is $P_{\text{fail}} = \lambda T_{\text{exec}} = \lambda(W/s + T_C)$, where λ is the instantaneous failure rate. If there is indeed a failure, we re-execute the task at speed σ (which may or may not differ from s); the re-execution time is then $T_{\text{reexec}} = W/\sigma + T_C$ so that the expected execution time is

$$\begin{aligned}\mathbb{E}(T) &= T_{\text{exec}} + P_{\text{fail}} T_{\text{reexec}} \\ &= (W/s + T_C) + \lambda(W/s + T_C)(W/\sigma + T_C) \ . \quad (1)\end{aligned}$$

Similarly, the worst-case execution time is

$$\begin{aligned}T_{wc} &= T_{\text{exec}} + T_{\text{reexec}} \\ &= (W/s + T_C) + (W/\sigma + T_C) \ . \quad (2)\end{aligned}$$

Remember that we assume success after re-execution, so we do not account for second and more re-executions. Along the same line, we could spare the checkpoint after re-executing the last task in a series of tasks, but this unduly complicates the analysis. In the companion research report [3], we show that this model with only a single re-execution is accurate up to second order terms when compared to the model with an arbitrary number of failures that follows an Exponential distribution of parameter λ.

What is the expected energy consumed during execution? The energy consumed during the first execution at speed s is $Ws^2 + E_C$, where E_C is the energy consumed during a checkpoint. The energy consumed during the second execution at speed σ is $W\sigma^2 + E_C$, and this execution takes place with probability $P_{\text{fail}} = \lambda T_{\text{exec}} = \lambda(W/s + T_C)$, as before. Hence the expectation of the energy consumed is

$$\mathbb{E}(E) = (Ws^2 + E_C) + \lambda\left(W/s + T_C\right)\left(W\sigma^2 + E_C\right) . \quad (3)$$

With multiple chunks (MC model), the execution times (worst case or expected) are the sum of the execution times for each chunk, and the expected energy is the sum of the expected energy for each chunk (by linearity of expectations). We point out that the failure model is coherent with respect to chunking. Indeed, assume that a divisible task of weight W is split into two chunks of weights w_1 and w_2 (where $w_1 + w_2 = W$). Then the probability of failure for the first chunk is $P^1_{\text{fail}} = \lambda(w_1/s + T_C)$ and that for the second chunk is $P^2_{\text{fail}} = \lambda(w_2/s + T_C)$. The probability of failure $P_{\text{fail}} = \lambda(W/s + T_C)$ with a single chunk differs from the probability of failure with two chunks only because of the extra checkpoint that is taken; if $T_C = 0$, they coincide exactly. If $T_C > 0$, there is an additional risk to use two chunks, because the execution lasts longer by a duration T_C. Of course this is the price to pay for a shorter re-execution time in case of failure: Equation (1) shows that the expected re-execution time is $P_{\text{fail}} T_{\text{reexec}}$, which is quadratic in W. There is a trade-off between having many small chunks (many T_C to pay, but small re-execution cost) and a few larger chunks (fewer T_C, but increased re-execution cost).

Optimization problems. Given a deadline D and a divisible task whose total computational load is W, the problem is to partition the task into n chunks of size w_i, where $\sum_{i=1}^n w_i = W$, and choose for each chunk an execution speed s_i and a re-execution speed σ_i in order to minimize the expected energy consumption:

$$\mathbb{E}(E) = \sum_{i=1}^{n} (w_i s_i^2 + E_C) + \lambda\left(\frac{w_i}{s_i} + T_C\right)\left(w_i \sigma_i^2 + E_C\right) ,$$

subject to the constraint that the deadline is met either in expectation or in the worst case:

ED (Expected deadline):
$\mathbb{E}(T) = \sum_{i=1}^n \left(\frac{w_i}{s_i} + T_C + \lambda\left(\frac{w_i}{s_i} + T_C\right)\left(\frac{w_i}{\sigma_i} + T_C\right)\right) \le D$

HD (Hard deadline):
$T_{wc} = \sum_{i=1}^n \left(\frac{w_i}{s_i} + T_C + \frac{w_i}{\sigma_i} + T_C\right) \le D$

The unknowns are the number of chunks n, the sizes of these chunks w_i, the speeds for the first execution s_i and for the second execution σ_i. We consider two variants of the problem, depending upon re-execution speeds:

- SS (Single speed): in this simpler variant, the re-execution speed is always the same as the speed chosen for the first execution. We then have to determine a single speed for each chunk: $\sigma_i = s_i$ for all i.
- MS (Multiple speeds): in this more general variant, the re-execution speed is freely chosen, and there are two different speeds to determine for each chunk.

We also consider the variant with a single chunk (SC), i.e., the task is atomic and we only need to decide for its execution speed (in the SS model), or for its execution and re-execution speeds (in the MS model). We start the study in Section IV with this simpler problem.

IV. With a single chunk

In this section, we consider the case of a single chunk, or equivalently of an atomic task: given a non-divisible workload W and a deadline D, find the values of s and σ that minimize

$$\mathbb{E}(E) = (Ws^2 + E_C) + \lambda\left(\frac{W}{s} + T_C\right)\left(W\sigma^2 + E_C\right)$$

subject to

$$\mathbb{E}(T) = \left(\frac{W}{s} + T_C\right) + \lambda\left(\frac{W}{s} + T_C\right)\left(\frac{W}{\sigma} + T_C\right) \le D$$

in the ED model, and subject to

$$\frac{W}{s} + T_C + \frac{W}{\sigma} + T_C \leq D$$

in the HD model. We first deal with the SS model, where we enforce $\sigma = s$, before moving on to the MS model.

A. Single speed model

In this section, we express $\mathbb{E}(E)$ as functions of the speed s. That is, $\mathbb{E}(E)(s) = (Ws^2 + E_C)(1 + \lambda(W/s + T_C))$. The following result is valid for both ED and HD models.

Lemma 1. *$\mathbb{E}(E)$ is convex on $\mathbb{R}_+^\star$. It admits a unique minimum $s^\star$ which can be computed numerically.*

Due to lack of space, the proof is available in the companion research report [3].

1) Expected deadline: In the SS ED model, we denote $\mathbb{E}(T)(s) = (W/s + T_C)(1 + \lambda(W/s + T_C))$ the constraint on the execution time.

Lemma 2. *For any D, if $T_C + \lambda T_C^2 \geq D$, then there is no solution. Otherwise, the constraint on the execution time can be rewritten as $s \in \left[W \frac{1 + 2\lambda T_C + \sqrt{4\lambda D + 1}}{2(D - T_C(1 + \lambda T_C))}, +\infty\right($.*

Proof: The function $s \mapsto \mathbb{E}(T)(s)$ is strictly decreasing and converges to $T_C + \lambda T_C^2$. Hence, if $T_C + \lambda T_C^2 \geq D$, then there is no solution. Else there exist a minimum speed s_0 such that, $\mathbb{E}(T)(s_0) = D$, and for all $s \geq s_0$, $\mathbb{E}(T)(s) \leq D$.

More precisely, $s_0 = W \frac{1 + 2\lambda T_C + \sqrt{4\lambda D + 1}}{2(D - T_C(1 + \lambda T_C))}$: since there is a unique solution to $\mathbb{E}(T)(s) = D$, we can solve this equation in order to find s_0. ■

To simplify the following results, we define

$$s_0 = W \frac{1 + 2\lambda T_C + \sqrt{4\lambda D + 1}}{2(D - T_C(1 + \lambda T_C))}. \tag{4}$$

Proposition 1. *In the* SS *model, it is possible to numerically compute the optimal solution for* SC *as follows:*
1) If $T_C + \lambda T_C^2 \geq D$, then there is no solution;
2) Else, the optimal speed is $\max(s_0, s^\star)$.

Proof: This is a corollary of Lemma 1: because $s \mapsto \mathbb{E}(T)(s)$ is convex on $\mathbb{R}_+^\star$, then its restriction to the interval $[s_0, +\infty($ is also convex and admits a unique minimum:

- if $s^\star < s_0$, then $\mathbb{E}(T)\ (s)$ is increasing on $[s_0, +\infty($, then the optimal solution is s_0
- else, clearly the minimum is reached when $s = s^\star$.

The optimal solution is then $\max(s_0, s^\star)$. ■

2) Hard deadline: In the HD model, the bound on the execution time can be written as $2\left(\frac{W}{s} + T_C\right) \leq D$

Lemma 3. *In the* SS HD *model, for any D, if $2T_C \geq D$, then there is no solution. Otherwise, the constraint on the execution time can be rewritten as $s \in \left[\frac{W}{\frac{D}{2} - T_C}; +\infty\right($*

Proof: The constraint on the execution time is now $2\left(\frac{W}{s} + T_C\right) \leq D$. ■

Proposition 2. *Let $s^\star$ be the solution indicated in Lemma 1. In the* SS HD *model if $2T_C \geq D$, then there is no solution. Otherwise, the minimum is reached when $s = \max\left(s^\star, \frac{W}{\frac{D}{2} - T_C}\right)$.*

Proof: The fact that there is no solution when $2T_C \geq D$ comes from Lemma 3. Otherwise, the result is obvious by convexity of the expected energy function. ■

B. Multiple speeds model

In this section, we consider the general MS model. We use the following notations:

$$\mathbb{E}(E)(s, \sigma) = (Ws^2 + E_C) + \lambda(W/s + T_C)(W\sigma^2 + E_C)$$

1) Expected deadline: The execution time in the MS ED model can be written as

$$\mathbb{E}(T)(s, \sigma) = (W/s + T_C) + \lambda(W/s + T_C)(W/\sigma + T_C)$$

We start by giving a useful property, namely that the deadline is always tight in the MS ED model:

Lemma 4. *In the* MS ED *model, in order to minimize the energy consumption, the deadline should be tight.*

Due to lack of space, the proof is available in the companion research report [3].

This lemma allows us to express σ as a function of s:

$$\sigma = \frac{\lambda W}{\frac{D}{\frac{W}{s} + T_C} - (1 + \lambda T_C)}.$$

Also we reduce the bi-criteria problem to the minimization problem of the single-variable function:

$$s \mapsto Ws^2 + E_C + \lambda\left(\frac{W}{s} + T_C\right) \times \left(W\left(\frac{\lambda W}{\frac{D}{\frac{W}{s} + T_C} - (1 + \lambda T_C)}\right)^2 + E_C\right) \tag{5}$$

which can be solved numerically.

2) Hard deadline: In this model we have similar results as with ED. The constraint on the execution time writes: $\frac{W}{s} + T_C + \frac{W}{\sigma} + T_C \le D$.

Lemma 5. *In the* MS ED *model, in order to minimize the energy consumption, the deadline should be tight.*

Again, the proof can be found in the companion research report [3].

This lemma allows us to express σ as a function of s:

$$\sigma = \frac{W}{(D - 2T_C)s - W} s$$

Finally, we reduce the bi-criteria problem to the minimization problem of the single-variable function:

$$s \mapsto W s^2 + E_C + \lambda \left(\frac{W}{s} + T_C \right) \times \left(W \left(\frac{W}{(D - 2T_C)s - W} s \right)^2 + E_C \right) \quad (6)$$

which can be solved numerically.

V. Several chunks

In this section, we deal with the general problem of a divisible task of size W that can be split into an arbitrary number of chunks. We divide the task into n chunks of size w_i such that $\sum_{i=1}^{n} w_i = W$. Each chunk is executed once at speed s_i, and re-executed (if necessary) at speed σ_i. The problem is to find the values of n, w_i, s_i and σ_i that minimize

$$\mathbb{E}(E) = \sum_i \left(w_i s_i^2 + E_C \right) + \lambda \sum_i \left(\frac{w_i}{s_i} + T_C \right) \left(w_i \sigma_i^2 + E_C \right)$$

subject to

$$\sum_i \left(\frac{w_i}{s_i} + T_C \right) + \lambda \sum_i \left(\frac{w_i}{s_i} + T_C \right) \left(\frac{w_i}{\sigma_i} + T_C \right) \le D$$

in the ED model, and subject to

$$\sum_i \left(\frac{w_i}{s_i} + T_C \right) + \sum_i \left(\frac{w_i}{\sigma_i} + T_C \right) \le D$$

in the HD model. We first deal with the SS model, where we enforce $\sigma_i = s_i$, before dealing with the MS model.

A. Single speed model

1) Expected deadline: In this section, we deal with the SS ED model and consider that for all i, $\sigma_i = s_i$. Then:

$$\mathbb{E}(T)(\cup_i (w_i, s_i, s_i)) = \sum_i \left(\frac{w_i}{s_i} + T_C \right) + \lambda \sum_i \left(\frac{w_i}{s_i} + T_C \right)^2$$

$$\mathbb{E}(E)(\cup_i (w_i, s_i, s_i)) = \sum_i \left(w_i s_i^2 + E_C \right) \left(1 + \lambda \left(\frac{w_i}{s_i} + T_C \right) \right)$$

Theorem 1. *In the optimal solution to the problem with the* SS ED *model, all n chunks are of equal size W/n and executed at the same speed s.*

Proof: Consider the optimal solution, and assume by contradiction that it includes two chunks w_1 and w_2, executed at speeds s_1 and s_2, where either $s_1 \neq s_2$, or $s_1 = s_2$ and $w_1 \neq w_2$. Let us assume without loss of generality that $\frac{w_1}{s_1} \ge \frac{w_2}{s_2}$.

We show that we can find a strictly better solution where both chunks have size $w = \frac{1}{2}(w_1 + w_2)$, and are executed at same speed s (to be defined later). The size and speed of the other chunks are kept the same. We will show that the execution time of the new solution is not larger than in the optimal solution, while its energy consumption is strictly smaller, hence leading to the contradiction.

We have seen that

$$\mathbb{E}(T)((w_1, s_1), (w_2, s_2)) = \frac{w_1}{s_1} + T_C + \frac{w_2}{s_2} + T_C + \lambda \left(\frac{w_1}{s_1} + T_C \right)^2 + \lambda \left(\frac{w_2}{s_2} + T_C \right)^2$$

$$\mathbb{E}(T)((w, s), (w, s)) = 2 \left(\frac{w}{s} + T_C \right) + 2\lambda \left(\frac{w}{s} + T_C \right)^2$$

Hence,

$$\mathbb{E}(T)((w_1, s_1), (w_2, s_2)) - \mathbb{E}(T)((w, s), (w, s)) = \left(\frac{w_1}{s_1} + \frac{w_2}{s_2} - \frac{2w}{s} \right) + \lambda \left(\left(\frac{w_1}{s_1} \right)^2 + \left(\frac{w_2}{s_2} \right)^2 - 2 \left(\frac{w}{s} \right)^2 \right)$$

Similarly, we know that:

$$\mathbb{E}(E)((w_1, s_1), (w_2, s_2)) = w_1 s_1^2 + E_C + w_2 s_2^2 + E_C + \lambda \left(\frac{w_1}{s_1} + T_C \right) \left(w_1 s_1^2 + E_C \right) + \lambda \left(\frac{w_2}{s_2} + T_C \right) \left(w_2 s_2^2 + E_C \right)$$

$$\mathbb{E}(E)((w, s), (w, s)) = 2 \left(w s^2 + E_C \right) + 2\lambda \left(\frac{w}{s} + T_C \right) \left(w s^2 + E_C \right)$$

and deduce

$$\mathbb{E}(E)((w_1, s_1), (w_2, s_2)) - \mathbb{E}(E)((w, s), (w, s)) = \left(w_1 s_1^2 + w_2 s_2^2 - 2 w s^2 \right) (1 + \lambda T_C) + \lambda E_C \left(\frac{w_1}{s_1} + \frac{w_2}{s_2} - \frac{2w}{s} \right) + \lambda \left(w_1^2 s_1 + w_2^2 s_2 - 2 w^2 s \right) \quad (7)$$

Let us now define $s_A = \frac{2w}{\frac{w_1}{s_1}+\frac{w_2}{s_2}} = \frac{w_1+w_2}{\frac{w_1}{s_1}+\frac{w_2}{s_2}}$ and $s_B = \frac{\sqrt{2}w}{\left(\left(\frac{w_1}{s_1}\right)^2+\left(\frac{w_2}{s_2}\right)^2\right)^{1/2}} = \frac{w_1+w_2}{\left(2\left(\frac{w_1}{s_1}\right)^2+2\left(\frac{w_2}{s_2}\right)^2\right)^{1/2}}$

We then fix $s = \max(s_A, s_B)$. Then, since $s \geq s_A$, we have $\frac{w_1}{s_1} + \frac{w_2}{s_2} - \frac{2w}{s} \geq 0$, and since $s \geq s_B$, we have $\left(\frac{w_1}{s_1}\right)^2 + \left(\frac{w_2}{s_2}\right)^2 - 2\left(\frac{w}{s}\right)^2 \geq 0$. This ensures that $\mathbb{E}(T)((w_1, s_1), (w_2, s_2)) - \mathbb{E}(T)((w, s), (w, s)) \geq 0$.

Note that

$$\begin{aligned}\frac{(w_1+w_2)^2}{s_B^2} &- \frac{(w_1+w_2)^2}{s_A^2}\\ &= 2\left(\frac{w_1}{s_1}\right)^2 + 2\left(\frac{w_2}{s_2}\right)^2 - \left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)^2\\ &= \left(\frac{w_1}{s_1}-\frac{w_2}{s_2}\right)^2 \geq 0\end{aligned}$$

This means that $s_A \geq s_B$, hence $s = s_A$. To prove that $\mathbb{E}(E)((w_1, s_1), (w_2, s_2)) - \mathbb{E}(E)((w, s), (w, s)) > 0$, we want to show that:

1) $w_1 s_1^2 + w_2 s_2^2 - 2ws^2 \geq 0$
2) $\frac{w_1}{s_1} + \frac{w_2}{s_2} - \frac{2w}{s} \geq 0$
3) $w_1^2 s_1 + w_2^2 s_2 - 2w^2 s \geq 0$
4) and that one of the previous inequalities is strict.

Note that by definition of $s = s_A$, the second inequality is true.

a) Let us first show that $w_1 s_1^2 + w_2 s_2^2 - 2ws_A^2 \geq 0$:

$$\begin{aligned}&\left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)^2\left(w_1 s_1^2 + w_2 s_2^2 - (w_1+w_2)\left(\frac{w_1+w_2}{\frac{w_1}{s_1}+\frac{w_2}{s_2}}\right)^2\right)\\ &= w_1^3 + w_1 w_2^2\left(\frac{s_1}{s_2}\right)^2 + 2\frac{w_1 w_2}{s_1 s_2} w_1 s_1^2 + w_2^3\\ &\quad + w_2 w_1^2\left(\frac{s_2}{s_1}\right)^2 + 2\frac{w_2 w_1}{s_1 s_2} w_2 s_2^2 - (w_1+w_2)^3\\ &= w_1 w_2^2 g\left(\frac{s_1}{s_2}\right) + w_1^2 w_2 g\left(\frac{s_2}{s_1}\right)\end{aligned}$$

where $g : u \mapsto u^2 + \frac{2}{u} - 3$. It is easy to show that g is nonnegative on $\mathbb{R}_+^\star$: indeed, $g'(u) = \frac{2}{u^2}(u^3 - 1)$ is negative in $[0, 1[$ and positive in $]1, \infty[$, and the unique minimum is $g(1) = 0$. We derive that $w_1 s_1^2 + w_2 s_2^2 - 2ws_A^2 \geq 0$.

b) Let us now show that $w_1^2 s_1 + w_2^2 s_2 - 2w^2 s \geq 0$: Remember that $2w = w_1 + w_2$.

$$\begin{aligned}&2\left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)\left(w_1^2 s_1 + w_2^2 s_2 - \frac{(w_1+w_2)^3}{2\left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)}\right)\\ &= 2w_1^3 + 2w_1^2 w_2 \frac{s_1}{s_2} + 2w_2^3 + 2w_1 w_2^2 \frac{s_2}{s_1} - (w_1+w_2)^3\\ &= w_1^3 + w_2^3 + w_1^2 w_2\left(2\frac{s_1}{s_2} - 3\right) + w_1 w_2^2\left(2\frac{s_2}{s_1} - 3\right)\end{aligned}$$

Remember that we assumed without loss of generality that $\frac{w_1}{s_1} \geq \frac{w_2}{s_2}$.

$$\begin{aligned}&2\left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)\left(w_1^2 s_1 + w_2^2 s_2 - \frac{(w_1+w_2)^3}{2\left(\frac{w_1}{s_1}+\frac{w_2}{s_2}\right)}\right)\\ &\geq w_2^3\left(\left(\frac{s_1}{s_2}\right)^3 + 1 + \left(\frac{s_1}{s_2}\right)^2\left(2\frac{s_1}{s_2} - 3\right) + \frac{s_1}{s_2}\left(2\frac{s_2}{s_1} - 3\right)\right)\\ &\geq 3w_2^3\left(\left(\frac{s_1}{s_2}\right)^3 - \left(\frac{s_1}{s_2}\right)^2 - \frac{s_1}{s_2} + 1\right)\\ &\geq 3w_2^3\left(\left(\frac{s_1}{s_2} - 1\right)^2\left(\frac{s_1}{s_2} + 1\right)\right) \geq 0\end{aligned}$$

Let us now conclude our study: if $\frac{s_1}{s_2} \neq 1$, then the energy consumption of the optimal solution is strictly greater than the one from our solution which is a contradiction. Hence we must have $s_1 = s_2$, and $w_1 \neq w_2$ (in fact, since we assumed that $\frac{w_1}{s_1} \geq \frac{w_2}{s_2}$, we must have $w_1 > w_2$). Then we can refine the previous analysis, and obtain that $w_1^2 s_1 + w_2^2 s_2 - 2w^2 s > 0$: again, the optimal energy consumption is strictly greater than in our solution; this is the final contradiction and concludes the proof. ■

Thanks to this result, we know that the problem with n chunks can be rewritten as follows: find s such that

$$\begin{aligned}n\left(\frac{W}{ns} + T_C\right) &+ n\lambda\left(\frac{W}{ns} + T_C\right)^2\\ &= \frac{W}{s} + nT_C + \frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right)^2 \leq D\end{aligned}$$

in order to minimize

$$\begin{aligned}n\left(\frac{W}{n}s^2 + E_C\right) &+ n\lambda\left(\frac{W}{ns} + T_C\right)\left(\frac{W}{n}s^2 + E_C\right)\\ &= \left(Ws^2 + nE_C\right)\left(1 + \frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right)\right)\end{aligned}$$

One can see that this reduces to the SC problem with the SS model (Section IV-A) up to the following parameter changes:

- $\lambda \leftarrow \frac{\lambda}{n}$
- $T_C \leftarrow nT_C$
- $E_C \leftarrow nE_C$

If the number of chunks n is given, we can express the minimum speed such that there is a solution with n chunks:

$$s_0(n) = W\frac{1 + 2\lambda T_C + \sqrt{4\frac{\lambda D}{n} + 1}}{2(D - nT_C(1 + \lambda T_C))}. \tag{8}$$

We can verify that when $D \leq nT_C(1 + \lambda n)$, there is no solution, hence obtaining an upper bound on n. Therefore, the two variables problem (with unknowns n and s) can be solved numerically.

2) Hard deadline: In the HD model, all results still hold, they are even easier to prove since we do not need to introduce a second speed.

Theorem 2. *In the optimal solution to the problem with the* SS HD *model, all n chunks are of equal size W/n and executed at the same speed s.*

Proof: The proof is similar to the one of Theorem 1, except we do not need to study the case where $s_B > s_A$. ■

B. Multiple speeds model

1) Expected deadline: In this section, we still deal with the problem of a divisible task of size W that we can split into an arbitrary number of chunks, but using the more general MS model. We start by proving that all re-execution speeds are equal:

Lemma 6. *In the* MS *model, all re-execution speeds are equal in the optimal solution: $\exists\sigma, \forall i, \sigma_i = \sigma$, and the deadline is tight.*

Due to lack of space, the proof is available in the companion research report [3].

We can now redefine

$$\mathbb{E}(T)(\cup_i(w_i, s_i, \sigma_i)) = T(\cup_i(w_i, s_i), \sigma)$$
$$\mathbb{E}(E)(\cup_i(w_i, s_i, \sigma_i)) = E(\cup_i(w_i, s_i), \sigma)$$

Theorem 3. *In the* MS *model, all chunks have the same size $w_i = \frac{W}{n}$, and are executed at the same speed s, in the optimal solution.*

Due to lack of space, the proof is available in the companion research report [3]. This proof uses the same reasoning as the proof of Theorem 1.

Thanks to this result, we know that the n chunks problem can be rewritten as follows: find s such that

- $\frac{W}{s} + nT_C + \frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right)\left(\frac{W}{\sigma} + nT_C\right) = D$
- in order to minimize $Ws^2 + nE_C + \frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right)\left(W\sigma^2 + nE_C\right)$

One can see that this reduces to the SC MS ED task problem where:

- $\lambda \leftarrow \frac{\lambda}{n}$
- $T_C \leftarrow nT_C$
- $E_C \leftarrow nE_C$

and allows us to write the problem to solve as a two-parameter function:

$$(n, s) \mapsto Ws^2 + nE_C \tag{9}$$
$$+\frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right)\left(W\left(\frac{\frac{\lambda}{n}W}{\frac{D}{\frac{W}{s}+nT_C} - (1+\lambda T_C)}\right)^2 + nE_C\right)$$

which can be minimized numerically.

2) Hard deadline: In this section, the constraint on the execution time can be written as:

$$\sum_i \left(\frac{w_i}{s_i} + T_C + \frac{w_i}{\sigma_i} + T_C\right) \leq D.$$

Lemma 7. *In the* MS HD *model with divisible chunk, the deadline should be tight.*

Lemma 8. *In the optimal solution, for all i, j, $\lambda\left(\frac{w_i}{s_i} + T_C\right)\sigma_i^3 = \lambda\left(\frac{w_j}{s_j} + T_C\right)\sigma_j^3$.*

The proofs for both lemmas are available in the companion research report [3].

Lemma 9. *If we enforce the condition that the execution speeds of the chunks are all equal, and that the re-execution speeds of the chunks are all equal, then all chunks should have same size in the optimal solution.*

Proof: This result is obvious since the problem can be reformulated as the minimization of $\alpha \sum w_i + \beta \sum w_i^2$ where neither α nor β depends on any w_i, under the constraints $\gamma \sum w_i + \zeta \leq D$, and $\sum w_i = W$. It is easy to see the result when there are only two chunks since there is only one variable, and the problem generalizes well in the case of n chunks. ■

We have not been able to prove a stronger result than Lemma 9. However we conjecture the following result:

Conjecture 1. In the optimal solution of MS HD, the re-execution speeds are identical, the deadline is tight. The re-execution speed is equal to $\sigma = \frac{W}{(D-2nT_C)s-W}s$. Furthermore the chunks should have the same size $\frac{W}{n}$ and should be executed at the same speed s.

This conjecture reduces the problem to the SC MS problem where

- $\lambda \leftarrow \frac{\lambda}{n}$
- $T_C \leftarrow nT_C$
- $E_C \leftarrow nE_C$

and allows us to write the problem to solve as a two-parameter function:

$$(n,s) \mapsto W s^2 + nE_C + \frac{\lambda}{n}\left(\frac{W}{s} + nT_C\right) \times \left(W\left(\frac{W}{(D-2nT_C)s - W}s\right)^2 + nE_C\right) \quad (10)$$

which can be solved numerically.

VI. Conclusion

In this work, we have studied the energy consumption of a divisible computational workload on volatile platforms. In particular, we have studied the expected energy consumption under different deadline constraints: a soft deadline (a deadline for the expected execution time), and a hard deadline (a deadline for the worst case execution time). We have been able to show mathematically, for all cases but one, that when using the multiple chunks model (MC), then (i) every chunk should be equally sized; (ii) every execution speed should be equal; and (iii) every re-execution speed should also be equal. This problem remains open in the multiple speeds hard deadline variant. Through a set of extensive simulations (see the extended version [3]), we have shown the following: (i) when the fault parameter λ is small, with expected deadline, the single chunk single speed model (SCSS) leads to almost optimal energy consumption. This is not true with hard deadlines, which accounts equally for execution and re-execution, thereby leading to higher energy consumption. Therefore, for the HD model (hard deadline) and for small values of λ, the model of choice should be single chunk multiple speeds, and that is not intuitive. When the fault parameter rate λ increases, using a single chunk is no longer energy-efficient, and one should focus on the MCMS model for both deadline types.

An interesting direction for future work is to extend this study to the case of an application workflow: instead of dealing with a single divisible task, we would deal with a DAG of tasks, that could be either divisible (checkpoints can take place anytime) or atomic (checkpoints can only take place at the end of the execution of some tasks). Again, we can envision both soft or hard constraints on the execution time, and we can keep the same model with a single re-execution per chunk/task, at the same speed or possibly at a different speed. Deriving complexity results and heuristics to solve this difficult problem is likely to be very challenging, but could have a dramatic impact to reduce the energy consumption of many scientific applications.

Acknowledgments: This work was supported in part by the ANR *RESCUE* project. A. Benoit and Y. Robert are with the Institut Universitaire de France.

References

[1] I. Assayad, A. Girault, and H. Kalla. Tradeoff exploration between reliability, power consumption, and execution time. In *Proc. of SAFECOMP, the 30th int. conf. on computer safety, reliability, and security*, pages 437–451, 2011.

[2] G. Aupy and A. Benoit. Approximation algorithms for energy, reliability and makespan optimization problems. Research report RR-8107, INRIA, France, Oct. 2012.

[3] G. Aupy, A. Benoit, R. Melhem, P. Renaud-Goud, and Y. Robert. Energy-aware checkpointing of divisible tasks with soft or hard deadlines. Research report RR-8238, INRIA, France, Feb. 2013.

[4] G. Aupy, A. Benoit, and Y. Robert. Energy-aware scheduling under reliability and makespan constraints. In *Proc. of HiPC, the 19th int. conf. on High Performance Computing*, 2012. Also available as INRIA Research Report RR-7757.

[5] 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, 2003.

[6] V. Bharadwaj, T. G. Robertazzi, and D. Ghose. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[7] G. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo. *Soft Real-Time Systems: Predictability vs. Efficiency*. Springer series in Computer Science, 2005.

[8] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, Feb. 1985.

[9] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M. J. Irwin. Soft errors issues in low-power caches. *IEEE Trans. Very Large Scale Integr. Syst.*, 13:1157–1166, October 2005.

[10] J. Dongarra, P. Beckman, P. Aerts, F. Cappello, T. Lippert, S. Matsuoka, P. Messina, T. Moore, R. Stevens, A. Trefethen, and M. Valero. The international exascale software project: a call to cooperative action by the global high-performance community. *Int. J. High Perform. Comput. Appl.*, 23(4):309–322, 2009.

[11] M. Drozdowski. Divisible load. In *Scheduling for Parallel Processing*, Computer Communications and Networks, pages 301–365. Springer, 2009.

[12] R. Geist, R. Reynolds, and J. Westall. Selection of a checkpoint interval in a critical-task environment. *IEEE Transactions on Reliability*, 37(4):395–400, Oct. 1988.

[13] R. Melhem, D. Mossé, and E. Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53(2):217–231, 2004.

[14] P. Pop, K. H. Poulsen, V. Izosimov, and P. Eles. Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems. In *Proc. of IEEE/ACM Int. Conf. on Hardware/software codesign and system synthesis (CODES+ISSS)*, pages 233–238, 2007.

[15] J. A. Stankovic, K. Ramamritham, and M. Spuri. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[16] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. *Operating Systems Design and Implementation*, pages 13–23, 1994.

[17] Y. Zhang and K. Chakrabarty. Energy-aware adaptive checkpointing in embedded real-time systems. In *Proc. of the Design, Automation and Test in Europe Conf. (DATE)*, pages 918–923. IEEE CS Press, 2003.

[18] D. Zhu and H. Aydin. Energy management for real-time embedded systems with reliability requirements. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pages 528–534, 2006.

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