Energy Optimal Activation of Processors for the Execution of a Single Task with Unknown Size

Jonatha Anselmi
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
Grenoble, 38000, France
jonatha.anselmi@inria.fr

Bruno Gaujal
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
Grenoble, 38000, France
bruno.gaujal@inria.fr

***Abstract*—A key objective in the management of modern computer systems consists in minimizing the electrical energy consumed by processing resources while satisfying certain target performance criteria. In this paper, we consider the execution of a single task with unknown size on top of a service system that offers a limited number of processing speeds, say N, and investigate the problem of finding a speed profile that minimizes the resulting energy consumption subject to a deadline constraint. Existing works mainly investigated this problem when speed profiles are continuous functions. In contrast, the novelty of our work is to consider *discontinuous* speed profiles, i.e., a case that arises naturally when the underlying computational platform offers a finite number of speeds.**

In our main result, we show that the computation of an optimal speed profile boils down to solving a convex optimization problem. Under mild assumptions, for such convex optimization we prove some structural results that yield the formulation of an extremely efficient solution algorithm. Specifically, we show that the optimal speed profile can be computed by solving $O(\log N)$ one-dimensional equations. Our results hold when the task size follows a known probability distribution function and the set of available speeds, if listed in increasing order, forms a sublinear concave sequence.

***Index Terms*—Energy minimization, GPU cluster, optimal speed, convex programming, parallel computing**

I. Introduction

To handle the demand induced by current deep learning technologies and applications, which affect several aspects of our daily life, AI companies and cloud providers are increasingly relying on large-scale GPU clusters. The electrical energy consumed by such systems has become a dominant issue because of its enormous financial and environmental impact [2]–[4]. In this context, a key objective consists in minimizing the energy consumed by processing resources such as CPUs or GPUs while satisfying some target user-perceived performance criteria. In this paper, we consider the execution of a single task on top of a processing system composed of N processors and investigate how to design a scheduler that dynamically activates the *optimal* number of processors at any point in time. Here, optimality refers to minimizing the overall electrical energy consumed by executing the task subject to a deadline constraint. In cloud and super computing, distributing a task over multiple processors is a common practice. Within machine learning tasks like TensorFlow, which are highly parallel, the number of processors used by a single task can vary by five orders of magnitude. This has been recently confirmed in a real trace from Google's Borg scheduler [12]. We also consider the realistic case where i) the processor speed-up function is sublinear and concave [14], and ii) the scheduler does not know in advance the exact size of the task, though we assume that it knows its probability distribution function. The N-processors assumption is a mere modeling abstraction as our framework only requires that the underlying computational platform offers a finite number of speeds. This case can be met even within a single CPU provided that it is compatible with Dynamic Voltage Frequency Scaling (DVFS) technologies.

This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02) and the LabEx PERSYVAL (ANR-11-LABX-0025-01).

A. Relevant Related Work

In the literature, the problem of minimizing the energy consumption induced by task executions while satisfying real-time constraints has been investigated in several contexts depending on the information available to the scheduler and the type of computing resources. Without being exhaustive, our goal here is to provide a high-level review of the state of the art highlighting the difference with respect to our work.

The work closest to our own is [10], which considers the execution of a single task as we do but when i) the speeds available to the scheduler form an interval of the real numbers and ii) the power consumption under speed s is of the form $P(s) = s^\alpha$ with $\alpha > 2$. Within these assumptions, the optimal speed profile admits an elegant "closed-form" expression. The "continuous speeds" assumption is justified by DVFS technologies as these allow the scheduler to change the current speed by changing the current CPU voltage. In practice however, processors may offer only a fixed, limited set of valid speeds, and in this case the solution in [10] does not extend to discrete speeds (as also mentioned in that reference). However, the following sub-optimal heuristic is proposed to handle in this case: "solve the continuous case and then use the closest discrete speed available to the scheduler". Unfortunately, it is not known how to bound the gap to the optimal solution, as also specified in [10, Section 8.4]. An additional difference of our work with respect to [10] is that we put milder assumptions on the structure of the power consumption as a function of the processing speed; our framework includes functions of the

form $P(s) = s^\alpha$ but also all increasing convex functions such that $P(0) = 0$.

Other relevant related scenarios consider the case where the given task is composed of a stream of sporadic jobs all with *known* sizes, arrival times and deadlines. Here, the problem of designing an algorithm for the optimal speed profile, or the number of active processors to use at any point in time, has been solved in a series of papers, with improving complexity [8], [9], [13]. In this online clairvoyant case, where the size and deadline of each job are revealed at the arrival time of the task, the problem can be formulated as a Markov Decision process and has been solved in [7] resp. [1] when the deadlines are hard resp. soft.

B. Contribution

Within the framework described above, our main contribution is to show that optimal speed profiles, or equivalently the number of processors to use at any point in time, are given by the solution of a convex optimization problem (see Theorem 1). This holds true when the task size follows an arbitrary probability distribution function and when the processor speed-up function is sublinear and concave. These speed-up function assumptions are common and justified because the process of distributing a single task over multiple processors usually implies some resource contention phenomena, and this slows down the computation [14]. When the distribution of the task size is continuous and power consumption function is strictly convex, we use this convex optimization problem to show that optimal speed profiles possess the following structural properties: i) there exists a unique optimal speed profile, ii) the optimal speed profile is increasing, which means that the task execution accelerates over time, and iii) the optimal speed profile uses a set of consecutive speeds. Property ii) is intuitive and also holds within the continuous-speed framework considered in [10]. Property iii) yields the formulation of an extremely efficient solution algorithm (see Algorithm 1). Specifically, we show that the optimal speed profile can be computed by solving $O(\log N)$ one-dimensional equations. The proposed algorithm is particularly useful to speed up the computation of an optimal profile in the context of machine learning tasks, as the number of processors used in this setting, and thus the number of available speeds, can vary by five orders of magnitude [12], e.g., $N \in \{1, \dots, 10^5\}$.

C. Organization

This paper is organized as follows. In Section II, we describe our framework and define our energy minimization problem. In Section III, we show that the considered energy minimization problem belongs to the framework of convex programming. In Section IV, we propose Algorithm 1 for the efficient computation of the optimal speed profile. Finally, in Section VI, we draw the conclusions of our work and discuss some generalizations of our results.

II. Energy Minimization Framework

The energy minimization framework that we investigate in the following is composed of the following components.

A. Processors

The underlying computational platform is composed of a finite number of identical processors, say N. Each processor operates with unitary speed, e.g., one work unit per second. By scaling work units, this assumption is not a loss of generality. When active resp. inactive, each processor consumes power $P_{\rm on}$, resp. $P_{\rm off}$. In other words, if a processor is active in a time interval $[t, t']$, then its energy consumption is $(t'-t)P_{\rm on}$.

B. Tasks

A task is a sequence of unitary operations of random size W (the total amount of *work*) that arrives at time zero. Only statistical information are known about the task size W. In particular, it is known its size probability distribution function $F(w) := \mathbb{P}(W \leq w)$, the probability that the task size is no more than w. Let also $F^c(w) := \mathbb{P}(W > w)$. We allow F to be discontinuous, though in some of our results we will require F to be continuous, which implies that W admits a density function. The continuity assumption will simplify the numerical computation of an optimal speed profile. We also assume that the support of W is $[W_{\min}, W_{\max}]$, where $W_{\min} \geq 0$ and $W_{\max} < \infty$ are the minimum and maximum task sizes, respectively. Finally, we impose the following *hard* real-time constraint: the task must be completed before a deadline D. This deadline must satisfy a feasibility constraint that will be discussed below.

C. Scheduler

The given task can be executed on multiple processors at the same time. The role of the scheduler is to allocate processors to the execution of the task at any point in time. The number of active processors may be changed at any time and we assume that these changes are immediate and do not incur any additional energy cost; for instance, ramp-up and/or speed change effects are neglected. The information available to the scheduler is:

- The deadline D (the task must be completed before D);
- The task size distribution (with support $[W_{\min}, W_{\max}]$);
- The effective speed s_n of execution when n processors are active, for all $n = 1, \dots, N$;
- The power dissipation $P(s_n)$ when n processors are active, for all $n = 1, \dots, N$;
- The current executed work w.

The whole process stops as soon as the task is complete and all processors are put to rest immediately. Of course, the scheduler does not know when this is going to happen since it does not know the actual size of the task W.

We denote by $s_n = dw/dt$ the overall processing speed when allocating n processors to the task. The function s_n is often called the speed-up function of the task and measures its scalability, or degree of parallelism. We assume that $(s_n)_n$ is non-decreasing and that $s_{n+1} - s_n$ is non-increasing. Thus, at higher speed, the work performed by unit of energy is lower, i.e., more energy is required to perform a fixed amount of work. These "diminishing returns" assumption is natural to model the overheads induced by parallel computations [14].

We assume that the (instantaneous) power consumption when operating at speed s_n, for $n = 1, \dots, N$, is

$$P(s_n) := nP_{\text{on}} + (N-n)P_{\text{idle}} = n(P_{\text{on}} - P_{\text{idle}}) + NP_{\text{idle}}, \quad (1)$$

where P_{on} is the (instantaneous) power consumption of a processor turned on and P_{idle} is the power consumption when the node is idle. In the RHS of (1), n is the inverse function of s_n. Since we have put mild assumptions on the sequence $(s_n)_n$, the power consumption function defined in (1), seen as a function of the processing speed, is in fact general. We let $P_{\text{idle}} \le P_{\text{on}}$, and thus $P(s_n)$ is increasing in n. We also assume $P(0) = 0$, which is equivalent to assume that the system is at complete rest when no processor is active.

Let also $P_\Delta = P_{\text{on}} - P_{\text{idle}}$ and $P_{\text{off}} = NP_{\text{idle}}$. Given that the speed-up sequence $(s_n)_n$ has non-increasing increments, $P(s)$ is convex as long as $s_2(P_\Delta + P_{\text{off}}) \le s_1(2P_\Delta + P_{\text{off}})$, with strict inequality holding as long as $P(s)$ is strictly convex. The latter inequality, which will be implicitly assumed in the following, holds true as long as P_{off} is not too large and/or the speed-up is low enough.

Since the scheduler does not know the size of the task in advance, it is natural to take decisions as a function of w (the work already executed) instead of as a function of time, as done in [10]. To see this, let us consider Figure 1, where speed s_1 is used for the first w_1 cycles, s_2 from w_1 to w_2 cycles and so forth. The first curve shows the cumulative amount of work

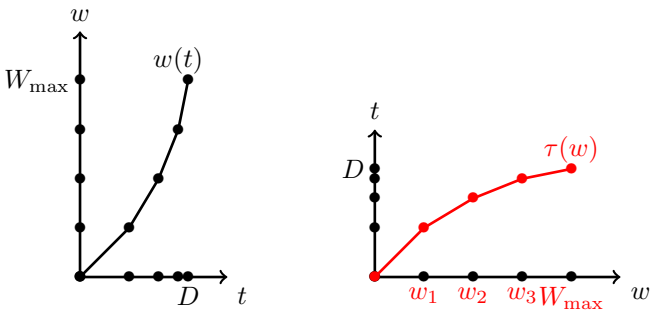

Figure 1. The function $w(t)$ is the cumulative work executed by time t under a given schedule (left) and $\tau(w)$ is the time needed to execute work w under the same schedule (right). The speed is the slope in the left figure and the inverse of the slope in the right one.

$w(t)$ executed by this schedule up to time t while the second curve shows $\tau(w)$, the time needed to execute w amount of work under the same schedule. The slope of the first curve is the current speed while the slope of the second one is the inverse of the current speed (as a function of w).

D. Objective Function

Let us define the following quantities:

- $w := w(t)$, the amount of work executed at time t;
- $s := s(w) \in \{s_1, \dots, s_N\}$, the speed used once exactly w work units have been executed, $w \in [0, W_{\max}]$;
- $Q(s)$, the power consumption *per unit of work* when operating at speed s.

We have

$$Q(s) := P(s)\frac{\mathrm{d}t}{\mathrm{d}w} = \frac{P(s)}{s}, \quad (2)$$

where derivatives are always intended as right derivatives. Since $P(0) = 0$ and P is convex, we notice that Q is non-decreasing.

Our objective is to find a speed profile $s : [0, W_{\max}] \to \{s_1, \dots, s_N\}$ that minimizes the mean energy consumption to execute the given job constrained by the deadline D. The mean energy consumption under speed profile s is defined by the following formula. By conditioning on the size ($W = w$) of the task, we get

$$\mathcal{E}(s) := \int_0^{W_{\max}} \left(\int_0^w Q(s(x))\,\mathrm{d}x \right) \mathrm{d}F(w) \quad (3)$$

$$= \int_0^{W_{\max}} Q(s(x)) \left(\int_x^{W_{\max}} \mathrm{d}F(w) \right) \mathrm{d}x \quad (4)$$

$$= \int_0^{W_{\max}} F^c(x)\, Q(s(x))\,\mathrm{d}x \quad (5)$$

where the second equality follows by changing the order of integration. Therefore, our objective is to solve the following infinite-dimensional optimization problem:

$$\min_{s:[0,W_{\max}]\to\{s_1,\dots,s_N\}} \int_0^{W_{\max}} Q(s(w))\, F^c(w)\,\mathrm{d}w \quad (6)$$

$$\text{subject to:} \quad \int_0^{W_{\max}} \frac{\mathrm{d}w}{s(w)} \le D. \quad (7)$$

Note that the constraint in (7) simply states that the job has to be completed before the deadline D. To see this, let $\tau(w)$ denote the time to execute the first w epochs, as in Figure 1. Then,

$$\frac{\mathrm{d}\tau(w)}{\mathrm{d}w} = \frac{1}{s(w)} \quad (8)$$

and integrating both sides and using that $\tau(0) = 0$ and that $\tau(W_{\max}) \le D$, we obtain (7).

The constraint (7) can be replaced by an equality without any loss of generality because a schedule that finishes before time t can always be delayed and start at time $t_0 > 0$ to reach D exactly; see Figure 2. This shows that such a schedule

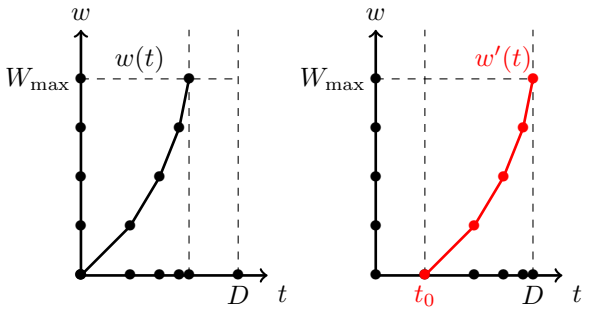

Figure 2. Shifting the starting time to reach D.

is suboptimal, as an optimal schedule will always start with active processors from time zero.

The problem (6)-(7) is feasible if

$$D \geq \frac{W_{\max}}{s_N}. \tag{9}$$

Indeed, when the scheduler uses its maximal speed from the start, it must be able to complete the largest possible job before the deadline D. This condition is assumed to be true in the rest of the paper. As a side remark, notice that if the processor has no slack ($Ds_N = W_{\max}$), then the solution is trivial: *use all processors from the start*, as long as the probability $\mathbb{P}(W \geq W_{\max} - \epsilon) > 0$, for all $\epsilon > 0$. Dually, if D is so large that using a single processor is enough to always satisfy the constraint ($D > W_{\max}/s_1$), then the solution is also trivial: *use a single processor from the start*. Thus, we also assume that $D \leq W_{\max}/s_1$ to make sure the constraint (7) is satisfied as an equality under the optimal solution.

III. A Convex Programming Solution

Since the scheduler has a finite set of possibilities at each w, its decisions split $[0, W_{\max}]$ into a set of intervals such that in each interval, the speed is constant. In general, the same number n of processors can be used in several intervals, but we will show in Proposition 2 that the optimal schedule uses an increasing number of processors with w. Therefore, an optimal schedule can be seen as a set of $N-1$ decisions points $x_0 = 0 \leq x_1 \leq x_2 \leq \cdots \leq x_{N-1} \leq x_N = W_{\max}$ such that n processors are used in the interval $[x_{n-1}, x_n]$. With this property, the optimization of interest in (6)-(7) can be transformed into the following optimization problem:

$$\min_{x_i,\, i=1,\ldots,N-1} \quad \sum_{i=1}^{N} Q(s_i) \int_{x_{i-1}}^{x_i} F^c(w)\mathrm{d}w \tag{10a}$$

$$\text{subject to:} \quad \sum_{i=1}^{N} \frac{x_i - x_{i-1}}{s_i} \leq D \tag{10b}$$

$$x_i \leq x_{i+1}, \qquad \forall i = 0, \ldots, N-1. \tag{10c}$$

where $x_0 = 0$ and $x_N = W_{\max}$. This optimization yields speed profiles where at most $N-1$ changes of speed are allowed. Specifically, the i-th change is performed when the amount of executed work reaches x_i, and within the interval $[x_{i-1}, x_i)$, the speed is s_i. Thus, if $x_{i-1} = x_i$, then speed s_i is never used.

Proposition 1. *The optimization problem* (10) *is convex, with a differentiable objective function when F is continuous. In addition, if F is strictly increasing, then it is strictly convex.*

Proof. Let $\bar{F} := \int F^c(w)\mathrm{d}w$. Note that the objective in (10a) can be written as

$$Q(s_N)\bar{F}(x_N) - Q(s_1)\bar{F}(x_0) + \sum_{i=1}^{N-1} (Q(s_i) - Q(s_{i+1}))\bar{F}(x_i)$$

Here, we note that $\bar{F}(w)$ is concave because $F^c(w)$ is positive non-increasing. Within the assumptions on the s_n's, $Q(s_i) < Q(s_{i+1})$, and given that the sum of convex functions is convex, the optimization (10a)-(10c) is convex. In addition, if F is strictly increasing, then $\bar{F}(w)$ is strictly concave and the optimization (10a)-(10c) is strictly convex. Finally, the differentiability of the objective function follows by the fact that F is continuous. □

Therefore, an optimizer of (10) can be computed efficiently by using existing algorithms from optimization theory such as the interior point method, which are guaranteed to converge to an optimal solution; see, e.g., [5].

The next theorem, our main result, connects the optimization (10) with the original optimization (6)-(7). It shows that any speed profile induced by the former also solves the latter.

Theorem 1. *Let s^* be a speed profile such that*

$$s^*(w) = s_i \quad \textit{if and only if} \quad w \in [x_{i-1}^*, x_i^*) \tag{11}$$

where x^* solves (10)*, for all $i = 1, \ldots, N$. Then, $\mathcal{E}(s^*)$ is an optimal schedule.*

Proof. First, we show in the following proposition that we can restrict to non-decreasing speed profiles, i.e., $s^*(w)$ is non-decreasing.

Proposition 2. *Any optimizer s of* (6)-(7) *is non-decreasing.*

Proof. We show that we can always improve a profile that is not non-decreasing. This means that, if we denote by s a feasible speed profile which is not non-decreasing, we can always find another feasible speed profile s' different from s such that $\mathcal{E}(s') \leq \mathcal{E}(s)$.

Let s be a feasible speed profile such that $s(w) = s_k$ when $w \in [w_a, w_b)$ and $s(w) = s_h$ when $w \in [w_b, w_c)$, for some w_a, w_b and w_c such that $0 \leq w_a < w_b < w_c \leq W_{\max}$ and for some $k \in \{2, \ldots, N\}$ and $h < k$. Now, consider a new feasible speed profile s' that is identical to s outside the interval $[w_a, w_c)$ and such that $s'(w) = s_h$ when $w \in [w_a, w_b')$ and $s'(w) = s_k$ when $w \in [w_b', w_c)$, for some w_b'. Note that there exists a unique choice for w_b' and necessarily $w_b' < w_b$; see Figure III.

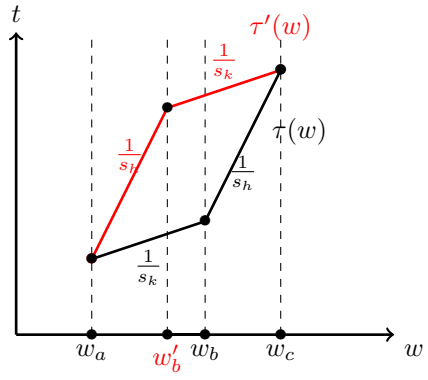

Figure 3. Execution time functions $\tau(w)$ and $\tau'(w)$ corresponding to schedules s and s' respectively.

Let

$$\mathcal{E}(s) := \int_0^W Q(s(w))\,\mathrm{d}w \tag{12}$$

be the random variable denoting the energy consumption under speed profile s for a task of size W. We have the following cases

- If $W < w_a$, then $\mathcal{E}(s) = \mathcal{E}(s')$;
- If $W \in [w_a, w_b')$, then $Q(s_k) > Q(s_h)$ implies

$$\begin{aligned}\mathcal{E}(s) &= \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_k)(W - w_a)\\ &> \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_h)(W - w_a) = \mathcal{E}(s')\end{aligned}$$

- If $W \in [w_b', w_b)$, then $Q(s_k) > Q(s_h)$ implies

$$\begin{aligned}\mathcal{E}(s) &= \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_k)(W - w_a)\\ &> \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_h)(w_b' - w_a)\\ &\quad + Q(s_k)(W - w_b') = \mathcal{E}(s').\end{aligned}$$

- If $W \in [w_b, w_c)$, then

$$\begin{aligned}\mathcal{E}(s) &= \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_k)(w_b' - w_a)\\ &\quad + Q(s_k)(w_b - w_b') + Q(s_h)(W - w_b)\\ &> \int_0^{w_a} Q(s(w))\,\mathrm{d}w + Q(s_h)(w_b' - w_a)\\ &\quad + Q(s_k)(w_b - w_b') + Q(s_k)(W - w_b)\\ &= \mathcal{E}(s')\end{aligned}$$

  where the inequality follows because $w_b' - w_a = w_c - w_b > W - w_b$.
- If $W > w_c$, then $\mathcal{E}(s) = \mathcal{E}(s')$ because the time to execute the first w_c epochs is identical within s and s', and because $s(w) = s'(w)$ when $w \geq w_c$.

Since $\mathcal{E}(s) \geq \mathcal{E}(s')$, then s cannot optimize (6)-(7). □

The main consequence of Proposition 2 is that we can restrict to speed profiles that change speed at most $N-1$ times. Thus, let $x_i \in [0, W_{\max}]$ for all $i = 1, \ldots, N-1$ be the points of the task size where the speed is allowed to change, with $x_i \leq x_{i+1}$. Let also $x_0 := 0$ and $x_N := W_{\max}$. The interpretation is that the work in the interval $[x_{i-1}, x_i)$ is performed with speed s_i, for all $i = 1, \ldots, N$. Within this setting, $s(w)$ is a cadlag piece-wise constant function and

$$\mathcal{E}(s) = \sum_{i=1}^{N} Q(s_i) \int_{x_{i-1}}^{x_i} F^c(w)\,\mathrm{d}w \tag{13}$$

and the constraint (7) can be written as

$$\sum_{i=1}^{N} \frac{x_i - x_{i-1}}{s_i} = D. \tag{14}$$

Therefore, the optimization in (6)-(7) boils down to the optimization problem (10). As mentioned before, in (10b), equality has been replaced by "$\leq$" as this does not change the set of optimizers. □

IV. Efficient Computation

Theorem 1 shows that solving (6)-(7) amounts to solving the finite dimensional ($N-1$ variables) convex optimization problem in (10), and this can be done using existing algorithms from optimization theory. We notice that if the task size distribution F is continuous, then the objective function is differentiable and an optimizer can be computed by using, e.g., interior point methods [5]. However, if the task size distribution F is discontinuous, then the objective function is *not* differentiable. In this case, one can always use a linear or non-linear interpolation to obtain a continuous version of F that approximates the given F arbitrarily well. Otherwise, one can rely on existing algorithms from derivative-free optimization theory [6].

A. Structural Properties and Algorithm

We now investigate structural properties possessed by an optimal speed profile when the objective is strictly convex and regular. These will lead to the design of a low complexity algorithm for the computation of the optimal solution.

The first result says that only *consecutive* speeds are used. Specifically, if the job starts under speed, e.g., s_2 and then moves to speed s_3, either it runs with speed s_3 until it terminates or it moves to s_4, but there is no possibility to move directly from s_3 to any s_i with $i \geq 5$.

Proposition 3. *Assume that F is continuous and P is strictly convex. Let s^* be an optimal schedule. Then, s^* uses a consecutive set of speeds.*

Proof. We assume that $x_{i-2}^* < x_{i-1}^* = x_i^* < x_{i+1}$ for some $i \in \{2, \ldots, N\}$, i.e., speed s_i is never used while speeds s_{i-1} and s_{i+1} are both used, and show that we can improve the objective function. Let x coincide with x^* except on coordinates i and $i-1$ where we let $x_{i-1} = x_i^* - \varepsilon$ and $x_i = x_i^* + q(\varepsilon)$. Here, $\varepsilon > 0$ is such that $x_i^* - \varepsilon > x_{i-2}^*$, $x_i^* + q(\varepsilon) < x_{i+1}^*$ and another condition to be set later (see (15)), and $q(\varepsilon)$ ensures that times needed to complete $x_i^* + q(\varepsilon)$ units of work under the speed profiles induced by x and x^* are identical. A simple computation yields (see Figure 4).

$$q(\varepsilon) = \varepsilon \left(\frac{s_i}{s_{i-1}} \frac{s_{i+1} - s_{i-1}}{s_{i+1} - s_i} - 1 \right).$$

Outside the interval $[x_i^* - \varepsilon, x_i^* + q(\varepsilon)]$, the objective function (10a) is identical in both x and x^*. When restricted to $[x_i^* - \varepsilon, x_i^* + q(\varepsilon)]$, the objective (10a) under x resp. x^* is

$$\begin{aligned}O_x &:= Q(s_i) \int_{x_i^* - \varepsilon}^{x_i^* + q(\varepsilon)} F^c(w)\mathrm{d}w\\ O_{x^*} &:= Q(s_{i-1}) \int_{x_i^* - \varepsilon}^{x_i^*} F^c(w)\mathrm{d}w\\ &\quad + Q(s_{i+1}) \int_{x_i^*}^{x_i^* + q(\varepsilon)} F^c(w)\mathrm{d}w.\end{aligned}$$

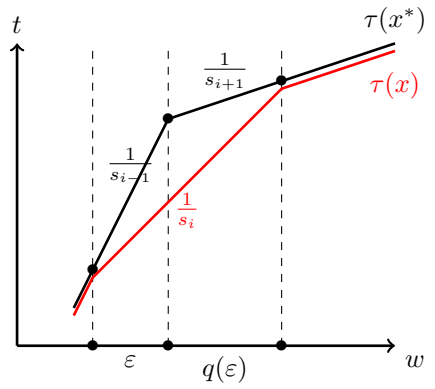

Figure 4. Construction of x and $q(\varepsilon)$, starting from x^* and ε.

By continuity of F^c,

$$\int_{x_i^*-\varepsilon}^{x_i^*+q(\varepsilon)} F^c(w)\mathrm{d}w = (\varepsilon+q(\varepsilon))F^c(x_i^*) + r_0(\varepsilon)$$
$$\int_{x_i^*-\varepsilon}^{x_i^*} F^c(w)\mathrm{d}w = \varepsilon F^c(x_i^*) + r_1(\varepsilon)$$
$$\int_{x_i^*}^{x_i^*+q(\varepsilon)} F^c(w)\mathrm{d}w = q(\varepsilon)F^c(x_i^*) + r_2(\varepsilon),$$

where $r_1(\varepsilon)$, $r_2(\varepsilon)$ and $r_0(\varepsilon)$ are all negligible w.r.t. ε, meaning $r_a(\varepsilon)/\varepsilon$ goes to 0 when ε goes to 0, for $a \in \{0,1,2\}$.

Using the value of $q(\varepsilon)$ and setting $r(\varepsilon) := r_0(\varepsilon) - r_1(\varepsilon) - r_2(\varepsilon)$,

$$\begin{aligned}
&\frac{1}{F^c(x_i^*)}(O_x - O_{x^*} - r(\varepsilon)) \\
&= \varepsilon(Q(s_i) - Q(s_{i-1})) - q(\varepsilon)(Q(s_{i+1}) - Q(s_i)), \\
&= \varepsilon\left(\frac{P(s_i)}{s_{i-1}}\left(1 + \frac{s_i - s_{i-1}}{s_{i+1} - s_i}\right) - \frac{P(s_{i-1})}{s_{i-1}}\right. \\
&\quad \left. - \frac{P(s_{i+1})}{s_{i-1}}\frac{s_i - s_{i-1}}{s_{i+1} - s_i}\right) \\
&= \frac{\varepsilon}{s_{i-1}(s_{i+1} - s_i)}\Big((s_{i+1} - s_{i-1})P(s_i) \\
&\quad - (s_{i+1} - s_i)\,P(s_{i-1}) - (s_i - s_{i-1})\,P(s_{i+1})\Big).
\end{aligned}$$

Since P is strictly convex, $(s_{i+1} - s_{i-1})\,P(s_i) - (s_{i+1} - s_i)\,P(s_{i-1}) - (s_i - s_{i-1}P(s_{i+1})) < 0$. Therefore, $O_x - O_{x^*} = -\varepsilon\gamma + r(\varepsilon)$, where $\gamma > 0$ does not depend on ε.

As mentioned before, $r(\varepsilon)/\varepsilon$ goes to 0 when ε goes to 0. Therefore, as soon as ε is small enough so that

$$r_0(\varepsilon) - r_1(\varepsilon) - r_2(\varepsilon) < \varepsilon\gamma, \tag{15}$$

then $O_x - O_{x^*} < 0$. □

Proposition 4. *Assume that F is continuous and P is strictly convex. Then, the optimal schedule always uses the maximal speed v_N.*

Proof. By contradiction, assume that the last speed used in the optimal schedule is s_{N-1}, over the interval $[x_{N-1}, W_{\max}]$. Now, let us consider an alternative schedule that only changes the speed in $[x_{N-1}, W_{\max}]$ and uses speed s_{N-2} over $[x_{N-1}, z_1]$, speed s_{N-1} over $[z_1, z_2]$ and speed s_N over $[z_2, W_{\max}]$ and achieves the same total work. This is always possible if z_1 is close enough to x_{N-1}, since $s_{N-2} < s_{N-1} < s_N$. Using the fact that (by the deadline constraint)

$$\frac{z_1 - x_{N-1}}{s_{N-2}} + \frac{z_2 - z_1}{s_{N-1}} + \frac{W_{\max} - z_2}{s_N} = \frac{W_{\max} - x_{N-1}}{s_{N-1}},$$

$z_2 = W_{\max} - \frac{s_N(s_{N-1}-s_{N-2})}{s_{N-2}(s_N - s_{N-1})}(z_1 - x_{N-1})$.

Now, the average cost over the interval $[x_{N-1}, W_{\max}]$ of the new schedule is

$$\begin{aligned}
&Q(s_{N-2})\int_{x_{N-1}}^{z_1} F^c(w)\mathrm{d}w + Q(s_{N-1})\int_{z_1}^{z_2} F^c(w)\mathrm{d}w \\
&\qquad + Q(s_N)\int_{z_2}^{W_{\max}} F^c(w)\mathrm{d}w.
\end{aligned}$$

Since F^c is continuous, then the average cost is differentiable. Its derivative d_1 with respect to z_1 is

$$\begin{aligned}
d_1 &= F^c(z_1)(Q(v_{N-2}) - Q(v_{N-1}) \\
&\quad + F^c(z_2)(Q(s_N) - Q(v_{N-1})\frac{s_N(s_{N-1} - s_{N-2})}{s_{N-2}(s_N - s_{N-1})}.
\end{aligned}$$

When $z_1 = x_{N-1}$, $d_1 \le 0$ because $F^c(W) = 0$, and $d_1 < 0$ when P is strictly convex, because this implies that Q is strictly increasing. In turn, this implies that the average cost is not minimal at $z_1 = x_{N-1}$ and that the new schedule is better than the original one for some $z_1 > x_{N-1}$, also implying that speed s_N is used by the optimal schedule. □

Now, given $m \in \{1, \dots, N\}$, let us construct the speed profile $v := v_m : [0, W_{\max}] \to \{s_m, \dots, s_N\}$ as follows:

$$v(w) = s_i \quad \text{if and only if} \quad w \in [y_{i-1}, y_i) \tag{16}$$

for all $i = m, \dots, N$ where $y_0 = \cdots = y_{m-1} = 0$, $y_N = W_{\max}$ and the speed change vector point $(y_m, \dots, y_{N-1}) \in [0, W_{\max}]^{N-m}$ satisfies

$$y_i = (F^c)^{-1}\left(\frac{\Gamma_m}{\Gamma_i}F^c(y_m)\right), \quad i = m+1, \dots, N-1 \tag{17}$$

where $(F^c)^{-1}(y) := \sup\{z : F^c(z) = y\}$,

$$\Gamma_i := \frac{s_i P(s_{i+1}) - s_{i+1}P(s_i)}{s_{i+1} - s_i} \tag{18}$$

and

$$\sum_{i=m}^{N} \frac{y_i - y_{i-1}}{s_i} = D. \tag{19}$$

The next proposition gives a property on v.

Proposition 5. *Assume that F is continuous and P is strictly convex. Then, there exists a unique optimizer that solves (6)-(7). In addition, if such optimizer uses all the speeds $s_1, \dots, s_N$, then it is given by v with $m = 1$.*

Proof. Uniqueness is trivial because (6)-(7) is a strictly convex optimization problem. The proof is a direct application of the KKT conditions to our optimization problem. By introducing the multipliers $\nu \in \mathbb{R}_+^M$ and $\lambda \in \mathbb{R}_+$, for all $i = 1, \dots, N-1$, the Lagrangian function associated to (6)-(7) is

$$\begin{aligned} L(x, \lambda, \nu) := & \sum_{i=1}^{N} Q(s_i) \int_{x_{i-1}}^{x_i} F^c(w) \mathrm{d}w \\ & + \lambda \left(\sum_{i=1}^{N} \frac{x_i - x_{i-1}}{s_i} - D \right) \\ & + \sum_{i=0}^{N-1} \nu_i (x_i - x_{i+1}). \end{aligned}$$

Now, we obtain $\frac{\partial L}{\partial x_i} = 0$ if and only if

$$(Q(s_i) - Q(s_{i+1}))F^c(x_i) + \lambda \left(\frac{1}{s_i} - \frac{1}{s_{i+1}} \right) + \nu_i - \nu_{i-1} = 0.$$

Assuming that all speeds are used in the optimal solution, necessarily $\nu_i = 0$ for all i by complementary slackness and the unique optimizer must satisfy

$$(Q(s_{i+1}) - Q(s_i))F^c(x_i) = \lambda \left(\frac{1}{s_i} - \frac{1}{s_{i+1}} \right). \tag{20}$$

We obtain that the optimizer of (10a)-(10b)-(10c) is given by (17) and (19), as desired. □

By combining the previous propositions, we can design an algorithm that computes the optimal schedule *even when some speeds are not used in the optimal solution*; see Algorithm 1. Specifically, we have the following result.

Algorithm 1: Dichotomy over the set of speeds for the computation of the optimal schedule.

Input: The set of speeds $s_1, \dots, s_N$, the power function P, the deadline D and the probability distribution of the task size F.

Output: The optimal subset of speeds $s_U, \dots, s_N$, and schedule $y_U, \dots y_N$.

1 $U := N; L := 1;$
2 **while** $U > L$ **do**
3 $\quad m := \lfloor (U + L)/2 \rfloor;$
4 $\quad$ Solve (19) for y_m using speeds $s_m, \dots, s_N$, where $y_{m+1}, \dots, y_N$ are given by (17);
5 $\quad$ **if** $y_m \leq 0$ **then**
6 $\quad\quad L := m;$
7 $\quad$ **else**
8 $\quad\quad U := m;$
9 $\quad$ **end**
10 **end**

Theorem 2. *Assume that F is continuous and P is strictly convex. Then, Algorithm 1 computes the (unique) optimal schedule. The complexity of this algorithm is $\log_2(N)$ times the complexity of solving the one-dimensional equation* (19)*.*

Proof. One technical difficulty with (17) is that it is not defined when $y_1 < 0$ or $y_1 > W_{\max}$. To deal with this, let us extend F^c by any continuous strictly decreasing function on $\mathbb{R}$ and let us consider the solution of (17)-(19) using speeds $s_m, \dots, s_N$. Now, let us notice that since Γ_i are strictly increasing in i and $(F^c)^{-1}(x) \geq \{z : F^c(z) \leq x\}$, then if $0 < y_m < W_{\max}$, then $0 < y_m < y_{m+1} < \cdots < y_N < W_{\max}$. Therefore, if $0 < y_m < W_{\max}$, then the optimal schedule with speeds $s_m, \dots, s_N$ is the solution $y_m, \dots, y_N$. In this case, adding lower speeds can lead to a better schedule in terms of energy. In Algorithm 1, this case is treated in Lines 7–8: the lowest speed is lowered from y_m to $y_{m'}$ with $m' := \lfloor (L+m)/2 \rfloor$. If $y_m > W_{\max}$ then $W_{\max}/s_m < D$. This implies that using the current smallest speed y_m is enough to complete the task before its deadline and this is obviously the optimal solution with the current set of speeds. In this case as well, adding lower speeds can also lead to a better schedule in terms of energy. In Algorithm 1, this case is also treated in Lines 7–8. Finally, if $y_m \leq 0$, there does not exist an optimal solution where the speeds $s_m \dots s_N$ are all used. Since speed s_N is always used in the optimal solution (Proposition 4), one must remove the low speeds to get a feasible schedule and change m to $m' = \lfloor (U + m)/2 \rfloor$; Lines 5–6 in Algorithm 1. □

Let us comment on the computational complexity of Algorithm 1. First, it is clear that it halts after no more than $\log_2 N$ iterations. Then, for each iteration, a one-dimensional equation needs to be solved, i.e., (19). Rearranging terms and using (17), this equation can be rewritten as

$$\begin{aligned} D &= \frac{W_{\max}}{s_N} + \sum_{i=m}^{N-1} y_i S_i \\ &= \frac{W_{\max}}{s_N} + \sum_{i=m}^{N-1} (F^c)^{-1} \left(\frac{\Gamma_m}{\Gamma_i} F^c(y_m) \right) S_i \end{aligned} \tag{21}$$

where $S_i := \frac{1}{s_i} - \frac{1}{s_{i+1}}$. We notice that (21) admits a unique solution and that F is differentiable almost everywhere because it is increasing. If derivatives can be computed, one can solve (21) by using (efficient) standard root finding algorithms such as the Newton–Raphson or the secant methods.

B. Explicit Calculation of Speed Change Points

Within specific choices of the task size distribution F, an expression for the solution of (17)-(19), and thus of an optimal speed profile, can be constructed explicitly. Clearly, this reduces the computational overhead of an optimal speed profile. For illustration purposes, let us consider the case where F takes the form

$$F(w) = 1 - K \left(1 - \frac{w}{W_{\max}} \right)^q, \quad q \in \mathbb{R}$$

where $K := \left(1 - \frac{W_{\min}}{W_{\max}} \right)^{-q}$ is a normalizing constant. If $q = 1$, then W is uniformly distributed over $[W_{\min}, W_{\max}]$. Then,

$$(F^c)^{-1}(y) = W_{\max} - W_{\max} \left(\frac{y}{K} \right)^{1/q}$$

and substituting in (17), we obtain, for $i = m+1, \ldots, N-1$

$$y_i = W_{\max} - W_{\max} \left(\frac{1}{K} \frac{\Gamma_m}{\Gamma_i} F^c(y_m) \right)^{1/q}$$
$$= W_{\max} - W_{\max} \left(\frac{\Gamma_m}{\Gamma_i} \right)^{1/q} \left(1 - \frac{y_m}{W_{\max}} \right).$$

Thus, the dependence among the y_i's is linear, and an explicit formula for y_m can be obtained by using (19), which in this case boils down to a linear equation with one unknown. Using (21)

$$D = W_{\max} \sum_{i=m}^{N-1} \left(1 - \left(\frac{\Gamma_m}{\Gamma_i} \right)^{1/q} \left(1 - \frac{y_m}{W_{\max}} \right) \right) S_i + \frac{W_{\max}}{s_N}$$

which gives

$$y_m = W_{\max} - W_{\max} \frac{\frac{1}{s_N} - \frac{D}{W_{\max}} + \sum_{i=m}^{N-1} S_i}{\sum_{i=m}^{N-1} S_i \left(\frac{\Gamma_1}{\Gamma_i} \right)^{1/q}}.$$

V. Generalizations

We discuss some possible generalizations of our model and results, though a detailed analysis is left as future work:

- *Delayed Processor Activation*. It may be the case that the activation of a processor requires some setup time. Thus, if the decision of activating a processor is taken at time t, then that processor can actually be used from time $t + \delta$, where we may suppose that the setup time δ is deterministic. In this case, a possible solution would consist in considering the speed profile s^* proposed in Theorem 1 and activating a processor at time $t_i^* - \delta$, where t_i^* is the time corresponding to the task size decision point x_i^*, i.e., $t_i^* = \sum_{j=1}^{i} \frac{1}{s_j}(x_j^* - x_{j-1}^*)$. This is not optimal because the processor consumes energy in $[t_i^* - \delta, t_i^*)$ but we claim that this heuristic should work reasonably well.
- *Containers*. Within applications that run on top of Kubernetes, processors (or servers) run in "containers". In this setting, the activation of a single processor may trigger the activation of a number of other processors, which may remain idle but ready to use. In Proposition 3, we have shown that the optimal speed profile s^* uses a consecutive set of speeds. Thus, upon activation of a new container, only one extra processor should be used if the idle processors in the container do not consume extra energy. While this latter assumption holds true in some scenarios, e.g., [11], it may not hold in general and in this case it may be convenient to activate more than one processors at the same time.

VI. Concluding Remarks

We have examined the execution of a single task of unknown size on top of a multi-processor system and investigated the problem of dynamically activating the number of processors that minimizes the overall electrical energy consumed by executing the task subject to a hard deadline constraint. We have shown that this problem can be formulated as a convex optimization problem and we have identified key structural properties that allowed us to design an extremely efficient algorithm for the computation of the optimal dynamic allocation of processors.

Our work is in close relationship with [10]; see Section I-A. In this reference, we stress that the speeds available to the scheduler form an interval of the real numbers, while we consider a *finite* number of speeds, i.e., N. Under appropriate conditions, it would be interesting to investigate whether the proposed optimal speed profile converges to the optimal speed profile proposed in [10] in the limit where $N \to \infty$, and if convergence occurs, then at what rate. Also, our results facilitate the design of heuristics to handle the case where a stream of tasks of unknown duration join the system of over time. This case is much more difficult to deal with: the hard deadline constraint makes non-trivial even just the problem of finding a feasible schedule. We leave these natural continuations of our work for future research.

References

[1] Jonatha Anselmi, Bruno Gaujal, and Louis Sébastien Rebuffi. Optimal Speed Profile of a DVFS Processor under Soft Deadlines. *Performance Evaluation*, 152, December 2021.

[2] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[3] Josep Ll. Berral, Íñigo Goiri, Ramón Nou, Ferran Julià, Jordi Guitart, Ricard Gavaldà, and Jordi Torres. Towards energy-aware scheduling in data centers using machine learning. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, page 215–224, New York, NY, USA, 2010. Association for Computing Machinery.

[4] Damien Borgetto, Henri Casanova, Georges Da Costa, and Jean-Marc Pierson. Energy-aware service allocation. *Future Gener. Comput. Syst.*, 28(5):769–779, may 2012.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[6] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, PA, USA, 2009.

[7] Bruno Gaujal, Alain Girault, and Stéphan Plassart. Dynamic Speed Scaling Minimizing Expected Energy Consumption for Real-Time Tasks. *Journal of Scheduling*, pages 1–25, July 2020.

[8] Bruno Gaujal, Alain Girault, and Stéphan Plassart. A Pseudo-Linear Time Algorithm for the Optimal Discrete Speed Minimizing Energy Consumption. *Discrete Event Dynamic Systems*, 31:163–184, 2021.

[9] Minming Li, Frances F. Yao, and Hao Yuan. An $O(n^2)$ algorithm for computing optimal continuous voltage schedules. In *TAMC'17*, volume 10185 of *LNCS*, pages 389–400, Bern, Switzerland, April 2017.

[10] Jacob R. Lorch and Alan Jay Smith. Improving dynamic voltage scaling algorithms with PACE. In *ACM SIGMETRICS 2001 Conference*, pages 50–61, 2001.

[11] N. Mahmoudi and H. Khazaei. Performance modeling of serverless computing platforms. *IEEE Transactions on Cloud Computing*, pages 1–1, 2020.

[12] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: The next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery.

[13] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of IEEE Annual Foundations of Computer Science*, pages 374–382, 1995.

[14] Xusheng Zhan, Yungang Bao, Christian Bienia, and Kai Li. Parsec3.0: A multicore benchmark suite with network stacks and splash-2x. *SIGARCH Comput. Archit. News*, 44(5):1–16, feb 2017.