INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Energy-aware scheduling of flow applications on master-worker platforms

Jean-François Pineau — Yves Robert — Frédéric Vivien

N° 6706

October 2008

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6706--FR+ENG

Energy-aware scheduling of flow applications on master-worker platforms

Jean-François Pineau , Yves Robert , Frédéric Vivien

Thème NUM — Systèmes numériques
Projet GRAAL

Rapport de recherche n° 6706 — October 2008 — 30 pages

Abstract: In this report, we consider the problem of scheduling an application composed of independent tasks on a fully heterogeneous master-worker platform with communication costs. We introduce a bi-criteria approach aiming at maximizing the throughput of the application while minimizing the energy consumed by participating resources. Assuming arbitrary super-linear power consumption laws, we investigate different models for energy consumption, with and without start-up overheads. Building upon closed-form expressions for the uniprocessor case, we are able to derive optimal or asymptotically optimal solutions for both models.

Key-words: Scheduling, energy, master-worker platforms, communication

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 —Télécopie +33 4 76 61 52 52

Ordonnancement énergétiquement efficace d'un flot de tâches sur une plate-forme maître-esclaves

Résumé : Dans ce rapport, nous étudions l'ordonnancement d'une application composée de tâches indépendantes qui doivent être exécutées sur une plate-forme maître-esclaves hétérogène où le coût des communications ne peut être négligé. Nous proposons une approche bi-critère visant à maximiser le débit de l'application tout en minimisant l'énergie dissipée par les ressources de calcul utilisées. En supposant que les lois de puissance électrique consommée sont super-liméaires, nous considérons différents modèles de consommation énergétique, avec ou sans coût de démarrage. À partir de formes clauses pour le cas avec un seul processeur nous construisons une solution asymptotiquement optimale pour les deux modèles.

Mots-clés : Ordonancement, énergie, plates-formes maître-esclaves, communication

1 Introduction

The Earth Simulator requires about 12 megawatts of peak power, and Petaflop systems may require 100 MW of power, nearly the output of a small power plant (300 MW). At $100 per MegaWatt.Hour, peak operation of a petaflop machine may thus cost $10,000 per hour [12]. And these estimates ignore the additional cost of dedicated cooling. Power consumption is also a critical factor because most of the power consumed is released by processors as heat. Current estimations state that cooling solutions are rising at $1 to $3 per watt of heat dissipated [23]. This is just one of the many economical reasons why energy-aware scheduling has proved to be an important issue in the past decade, even without considering battery-powered systems such as laptop and embedded systems.

Many important scheduling problems involve large collections of identical tasks [7, 1]. In this paper, we consider a single bag-of-tasks application which is launched on a heterogeneous platform. We suppose that all processors have a discrete number of speeds (or modes) of computation: the quicker the speed, the less efficient energetically-speaking. Our aim is to maximize the throughput, i.e., the fractional number of tasks processed per time-unit, while minimizing the energy consumed. Unfortunately, the goals of low power consumption and efficient scheduling are contradictory. Indeed, throughput can be maximized by using more energy to speed up processors, while energy can be minimized by reducing the speeds of the processors, hence the total throughput.

Altogether, power-aware scheduling truly is a bi-criteria optimization problem. A common approach to such problems is to fix a threshold for one objective and to minimize the other. This leads to two interesting questions. If we fix energy, we get the *laptop problem*, which asks "What is the best schedule achievable using a particular energy budget, before battery becomes critically low?". Fixing schedule quality gives the *server problem*, which asks "What is the least energy required to achieve a desired level of performance?".

The particularity of this work is to consider a fully heterogeneous master-worker platform, and to take communication costs into account. Here is the summary of our main results:

• We use arbitrary super-linear power consumption laws rather than restricting to relations of the form $P_d = s^\alpha$ where P_d is the power dissipation, s the processor speed, and α some constant greater than 1.

• Under an ideal power-consumption model, we derive an optimal polynomial algorithm to solve either bi-criteria problem (maximize throughput within a power

consumption threshold, or minimize energy consumption while guaranteeing a required throughput).

• Under a refined power-consumption model with start-up overheads, we derive a polynomial algorithm which is asymptotically optimal.

These results constitute a major step with regards to state-of-the-art scheduling techniques on heterogeneous master-worker platforms. The paper is organized as follows. We first present the framework and different power consumption models in Section 2. We study the bi-criteria scheduling problem under the ideal power consumption model in Section 3, and under the more realistic model with overheads in Section 4. Section 5 is devoted to an overview of related work. Finally, we state some concluding remarks in Section 6.

2 Framework

We outline in this section the model for the target applications and platforms, as well as the characteristics of the consumption model. Next we formally state the bi-criteria optimization problem.

2.1 Application and platform model

We consider a bag-of-tasks application $\mathcal{A}$, composed of a large number of independent, same-size tasks, to be deployed on a heterogeneous master-worker platform. We let ω be the amount of computation (expressed in *flops*) required to process a task, and δ be the volume of data (expressed in *bytes*) to be communicated for each task. We do not consider return messages, instead we assume that task results are stored on the workers. This simplifying hypothesis could be alleviated by considering the cost of longer messages (append the return message for a given task to the incoming message of the next one).

The master-worker platform, also called star network, or single-level tree in the literature, is composed of a master P_{master}, the root of the tree, and p workers P_u ($1 \leq u \leq p$). Without loss of generality, we assume that the master has no processing capability. Otherwise, we can simulate the computations of the master by adding an extra worker paying no communication cost. The link between P_{master} and P_u has a bandwidth b_u. We assume a linear cost model, hence it takes a time δ/b_u to send a task to processor P_u. We suppose that the master can send/receive data to/from all workers at a given time-step according to the *bounded multi-port* model [13, 14]. There is a limit on the amount of data that the master can send per time-unit,

denoted as BW. In other words, the total amount of data sent by the master to all workers each time-unit cannot exceed BW. Intuitively, the bound BW corresponds to the bandwidth capacity of the master's network card; the flow of data out of the card can be either directed to a single link or split among several links, hence the multi-port hypothesis. The bounded multi-port model fully accounts for the heterogeneity of the platform, as each link has a different bandwidth.

2.2 Energy model

For processors based on CMOS technology, power consumption is dominated by the dynamic power dissipation P_d, which is given as a function of the operating frequency, $P_d = C_{\text{eff}} \cdot V^2 \cdot s$, where C_{eff} is the average switched capacitance per cycle, V is the operating voltage, and s is the operating frequency. Among the main system-level energy-saving techniques, Dynamic Voltage Scaling (DVS) plays a very important role. DVS works on a very simple principle: decrease the supply voltage to the CPU so as to consume less power. But there is a minimum voltage required to drive the microprocessor at the desired frequency. So DVS reduces the power consumption by changing the clock frequency and voltage settings. For this reason, DVS is also called *frequency-scaling* or *speed scaling* [15]. Most authors use the expression $P_d = s^\alpha$, where $\alpha > 1$. We adopt a more general approach, as we only assume that power consumption is a *super-linear* function (i.e., a strictly increasing and convex function) of the processor speed. We denote by $\mathfrak{P}_u$ the power consumption per time unit of processor P_u.

We deal with a discrete voltage-scaling model. The computational speed of worker P_u has to be picked among a limited number of m_u modes. We denote the computational speeds $s_{u,i}$, meaning that the processor P_u running in the ith mode (noted $P_{u,i}$) takes $X/s_{u,i}$ time-units to execute X floating point operations (hence the time required to process one task of $\mathcal{A}$ of size ω on $P_{u,i}$ is $\omega/s_{u,i}$). The power consumption per time-unit of $P_{u,i}$ is denoted by $\mathfrak{P}_{u,i}$. We will suppose that processing speeds are listed in increasing order on each processor ($s_{u,1} \leq s_{u,2} \leq \cdots \leq s_{u,m_u}$). Modes are exclusive: one processor can only run at a single mode at any given time.

There exist many ways to refine the previous model in order to get realistic settings. Under a fluid model, switching among the modes does not cost any penalty. In real life, it costs a penalty depending on the modes. There are two kinds of overhead to consider when changing the processor speed: the time overhead and the power overhead. However, most authors suppose that the time overhead is negligible, since processors can still execute instruction during transitions [6], and time overhead

is linear in processor speed. We may also wonder what happens when the utilization of a processor tends to zero. There also exist two policies: either (i) we assume that an idle processor does not consume any power, so the power consumption is super-linear from 0 to the power consumption at frequency $s_{u,1}$; or (ii) we state that once a processor is on, it will always be above a minimal power consumption defined by its idle frequency, or speed, $s_{u,1}$. We can have any combination of the previous models.

In addition, there are different problems when dealing with consumption overhead. First of all, we have to specify when the consumption overhead is paid, as one can have an overhead only when turning on the worker, when turning it off, or for each transition of mode; a processor turned on can consume even when idle.

Under the latter (more realistic) models, power consumption now depends on the length of the interval during which the processor is turned on (we pay the overhead only once during this interval). We introduce a new notation to express power consumption as a function of the length t of the execution interval:

$$\mathfrak{P}_{u,i}(t) = \mathfrak{P}_{u,i}^{(1)} \cdot t + \mathfrak{P}_{u}^{(2)} \tag{1}$$

where $\mathfrak{P}_{u}^{(2)}$ is the energy overhead to turn processor P_u on.

To summarize, we consider two models: an **ideal model** simply characterized by $\mathfrak{P}_{u,i}$, the power consumption per time-unit of P_u running in mode i, and a **model with start-up overheads**, where power consumption is given by Equation 1 for each processor.

2.3 Objective function

As stated above, our goal is bi-criteria scheduling. The first objective is to minimize the power consumption, and the second objective is the maximization of the throughput. We denote by $\rho_{u,i}$ the throughput of worker $P_{u,i}$ for application $\mathcal{A}$, i.e., the average number of tasks of $\mathcal{A}$ that $P_{u,i}$ executes each time-unit. There is a limit to the number of tasks that each mode of one processor can perform per time-unit. First of all, as $P_{u,i}$ runs at speed $s_{u,i}$, it cannot execute more than $s_{u,i}/\omega$ tasks per time-unit. Second, as all modes of P_u are exclusive, if $P_{u,i}$ is at its maximal throughput, no other mode can be requested. So their is a strong relationship between the throughput of one mode and the maximum throughput available for all remaining modes. As $\frac{\rho_{u,i}\ \omega}{s_{u,i}}$ represents the fraction of time spent under mode $m_{u,i}$

per time-unit, this constraint can be expressed by:

$$\forall\ u \in [1..p],\ \sum_{i=1}^{m_u} \frac{\rho_{u,i}\ \omega}{s_{u,i}} \leq 1.$$

Under the ideal model, and for the simplicity of proofs, we can add an additional idle mode $P_{u,0}$ whose speed is $s_{u,0} = 0$. The power consumption per time-unit of $P_{u,i}$, when fully used, is $\mathfrak{P}_{u,i}$ ($\mathfrak{P}_{u,0} = 0$). Its power consumption per time-unit with a throughput of $\rho_{u,i}$ is then $\frac{\rho_{u,i}\ \omega}{s_{u,i}}\mathfrak{P}_{u,i}$

We denote by ρ_u the throughput of worker P_u, i.e., the sum of the throughput of each mode of P_u (except the throughput of the idle mode), so the total throughput of the platform is denoted by:

$$\rho = \sum_{u=1}^{p} \rho_u = \sum_{u=1}^{p} \sum_{i=1}^{m_u} \rho_{u,i}.$$

We define problem MinPower (ρ) as the problem of minimizing the power consumption $\mathfrak{P} = \sum_{u=1}^{p} \mathfrak{P}_u$ while achieving a throughput ρ. Similarly, MaxThroughput ($\mathfrak{P}$) is the problem of maximizing the throughput while not exceeding the power consumption $\mathfrak{P}$. In Section 3 we first deal with an ideal model without power nor timing overhead (a processor can be turned off without any cost). We extend this work to a more realistic model in Section 4.

3 Ideal model

Both bi-criteria problems (maximizing the throughput given an upper bound on power consumption and minimizing the power consumption given a lower bound on throughput) have been studied at the processor level, using particular power consumption laws such as $P_d = s^\alpha$ [2, 4, 5]. However, we are able to solve these problems optimally using the sole assumption that the power consumption is super-linear. Furthermore, we also solve these problems at the platform level, that is, for a heterogeneous set of processors.

A key step is to establish closed-form formulas linking the power consumption and the throughput of a single processor:

Proposition 1. *The optimal power consumption to achieve a throughput of $\rho > 0$ is*

$$\mathfrak{P}_u(\rho) = \max_{0 \leq i < m_u} \left\{ (\omega\rho - s_{u,i}) \frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \right\},$$

and is obtained using two consecutive modes, P_{u,i_0} and P_{u,i_0+1}, such that $\frac{s_{u,i_0}}{\omega} < \rho \leq \frac{s_{u,i_0+1}}{\omega}$.

Proof. The minimization of the power consumption is bounded by two types of constraints: i) The first constraint states that the processor has to ensure a given throughput, ii) The second constraint states that the processing capacity of $P_{u,i}$ cannot be exceeded, and that the different modes are exclusive. So our optimization problem is :

$$\begin{cases} \text{MINIMIZE } \mathfrak{P}_u = \displaystyle\sum_{i=1}^{m_u} \frac{\rho_{u,i}\ \omega}{s_{u,i}} \mathfrak{P}_{u,i} \text{ SUBJECT TO} \\ \displaystyle\sum_{i=1}^{m_u} \rho_{u,i} = \rho \\ \displaystyle\sum_{i=1}^{m_u} \frac{\rho_{u,i}}{s_{u,i}} \omega \ \leq \ 1 \end{cases} \tag{2}$$

A first remark is that the throughput that the processor has to achieve must be lower than its maximum throughput ($\rho \leq \frac{s_{u,m_u}}{\omega}$), otherwise the system has no solution. Linear program (2) can easily be solved over the rationals, and the throughput of the modes of the processor depend on the total throughput that has to be achieved. If $0 < \rho \leq \frac{s_{u,m_u}}{\omega}$, we denote by i_0 the unique mode of P_u such as $\frac{s_{u,i_0}}{\omega} < \rho \leq \frac{s_{u,i_0+1}}{\omega}$. Then, we define $\tilde{\mathcal{S}}$ by the following scheduling:

$$\tilde{\rho}_{u,i_0} = \frac{s_{u,i_0}(s_{u,i_0+1} - \omega\rho_u)}{\omega(s_{u,i_0+1} - s_{u,i_0})} \quad \tilde{\rho}_{u,i_0+1} = \frac{s_{u,i_0+1}(\omega\rho - s_{u,i_0})}{\omega(s_{u,i_0+1} - s_{u,i_0})} \quad \tilde{\rho}_{u,i} = 0 \text{ if } i \notin \{i_0, i_0{+}1\}.$$

First, one can note that $\tilde{\mathcal{S}}$ is feasible, and respects all constraints of Linear program (2).

$$\begin{aligned} \sum_{i=1}^{m_u} \tilde{\rho}_{u,i} &= \frac{s_{u,i_0}(s_{u,i_0+1} - \omega\rho_u) + s_{u,i_0+1}(\omega\rho - s_{u,i_0})}{\omega(s_{u,i_0+1} - s_{u,i_0})} = \rho; \\ \sum_{i=1}^{m_u} \frac{\tilde{\rho}_{u,i}\omega}{s_{u,i}} &= \frac{(s_{u,i_0+1} - \omega\rho_u)}{s_{u,i_0+1} - s_{u,i_0}} + \frac{(\omega\rho - s_{u,i_0})}{s_{u,i_0+1} - s_{u,i_0}} = 1. \end{aligned}$$

Let $\mathcal{S}'$ be an optimal solution, $\mathcal{S}' = \{\rho'_{u,1}, \cdots, \rho'_{u,m_u}\}$. As $\mathcal{S}'$ is a solution, it respects all the constraints of Linear program (2). So:

$$\sum_{i=1}^{m_u} \rho'_{u,i} = \rho \quad \text{and} \quad \sum_{i=1}^{m_u} \frac{\rho'_{u,i}\omega}{s_{u,i}} \leq 1.$$

Let $i_{\min}$ be the slowest mode used by $\mathcal{S}'$, and $i_{\max}$ the fastest. Then we can distinguish three cases:

- **If $\boldsymbol{i_{\min} > i_0}$ or $\boldsymbol{i_{\min} = i_0}$ and $\boldsymbol{\rho'_{u,i_0} < \tilde{\rho}_{u,i_0}}$:** In both cases, $\rho'_{u,i_0} < \tilde{\rho}_{u,i_0}$, so there exists $\epsilon > 0$, such that $\rho'_{u,i_0} = \tilde{\rho}_{u,i_0} - \epsilon$. Then we can look at the power consumption of $\mathcal{S}'$:

$$\begin{aligned}
\sum_{i=1}^{m_u} \rho'_{u,i}\mathfrak{K}_{u,i} &\geq \rho'_{u,i_0}\mathfrak{K}_{u,i_0} + \left(\sum_{i=i_0+1}^{m_u} \rho'_{u,i}\right)\mathfrak{K}_{u,i_0+1} \\
&= \rho'_{u,i_0}\mathfrak{K}_{u,i_0} + (\rho - \rho'_{u,i_0})\mathfrak{K}_{u,i_0+1} \\
&= (\tilde{\rho}_{u,i_0} - \epsilon)\mathfrak{K}_{u,i_0} + (\rho - \tilde{\rho}_{u,i_0} + \epsilon)\mathfrak{K}_{u,i_0+1} \\
&= \tilde{\rho}_{u,i_0}\mathfrak{K}_{u,i_0} + \tilde{\rho}_{u,i_0+1}\mathfrak{K}_{u,i_0+1} + \epsilon\left(\mathfrak{K}_{u,i_0+1} - \mathfrak{K}_{u,i_0}\right) \\
&\geq \tilde{\rho}_{u,i_0}\mathfrak{K}_{u,i_0} + \tilde{\rho}_{u,i_0+1}\mathfrak{K}_{u,i_0+1}.
\end{aligned}$$

  And so our solution does not consume more power, and is thus also optimal.

- **If $\boldsymbol{i_{\max} < i_0 + 1}$ or $\boldsymbol{i_{\max} = i_0 + 1}$ and $\boldsymbol{\rho'_{u,i_0+1} < \tilde{\rho}_{u,i_0+1}}$:** In both cases, $\rho'_{u,i_0+1} < \tilde{\rho}_{u,i_0+1}$, so there exists $\epsilon > 0$, such that $\rho'_{u,i_0+1} = \tilde{\rho}_{u,i_0+1} - \epsilon$. Then, we have:

$$\sum_{i=1}^{i_0} \rho'_{u,i} = \rho - \rho'_{u,i_0+1} \geq \rho - \tilde{\rho}_{u,i_0+1} + \epsilon = \tilde{\rho}_{u,i_0} + \epsilon$$

  And
$$\begin{aligned}
\sum_{i=1}^{i_{\max}} \frac{\rho'_{u,i}\omega}{s_{u,i}} &= \sum_{i=1}^{i_0} \frac{\rho'_{u,i}\omega}{s_{u,i}} + \frac{\rho'_{u,i_0+1}\omega}{s_{u,i_0+1}} \\
&\geq \frac{\omega \sum_{i=1}^{i_0} \rho'_{u,i}}{s_{u,i_0}} + \frac{\rho'_{u,i_0+1}\omega}{s_{u,i_0+1}} \\
&\geq \omega\frac{\tilde{\rho}_{u,i_0} + \epsilon}{s_{u,i_0}} + \omega\frac{\tilde{\rho}_{u,i_0+1} - \epsilon}{s_{u,i_0+1}} \geq 1 + \omega\epsilon\left(\frac{1}{s_{u,i_0}} - \frac{1}{s_{u,i_0+1}}\right) > 1.
\end{aligned}$$

  which is in contradiction with the second constraint.

- **Otherwise** we know that either $i_{\min} < i_0$, so $\rho'_{u,i_{\min}} \geq \tilde{\rho}_{u,i_{\min}} = 0$, or $i_{\min} = i_0$ and $\rho'_{u,i_{\min}} \geq \tilde{\rho}_{u,i_{\min}}$. In both cases $\rho'_{u,i_{\min}} \geq \tilde{\rho}_{u,i_{\min}}$, and, for the same reasons, $\rho'_{u,i_{\max}} \geq \tilde{\rho}_{u,i_{\max}}$. We also know that (at least) one virtual processor among P_{u,i_0} and P_{u,i_0+1} has a throughput in $\mathcal{S}'$ strictly smaller than in $\tilde{\mathcal{S}}$ (otherwise the power consumption of $\mathcal{S}'$ is greater). Let call that processor P_α. The idea of the proof is to give an amount $\epsilon_{\min}$ of the work of $P_{i_{\min}}$ to P_α. As P_α is faster than $P_{i_{\min}}$, it takes less time to P_α to process $\epsilon_{\min}$ than to $P_{i_{\min}}$. During the spared time, P_α has time to do an amount $\epsilon_{\max}$ of the work of $P_{i_{\max}}$. Basically, $\epsilon_{\min}$ and $\epsilon_{\max}$ are defined such as the throughput in the new scheduling $\mathcal{S}''$ of either $P_{i_{\min}}$, or $P_{i_{\max}}$ is set to its throughput in $\tilde{\mathcal{S}}$:

$$\begin{cases} \rho''_{u,i_{\min}} &= \rho'_{u,i_{\min}} - \epsilon_{\min} \\ \rho''_{u,\alpha} &= \rho'_{u,\alpha} + \epsilon_{\min} + \epsilon_{\max} \\ \rho''_{u,i_{\max}} &= \rho'_{u,i_{\max}} - \epsilon_{\max} \\ \rho''_{u,i} &= \rho'_{u,i} \quad \text{otherwise.} \end{cases}$$

$$\epsilon_{\min} = \min\left\{\rho'_{u,i_{\min}} - \tilde{\rho}_{u,i_{\min}}; \frac{(\rho'_{u,i_{\max}} - \tilde{\rho}_{u,i_{\max}})}{\lambda}\right\}, \epsilon_{\max} = \epsilon_{\min}\lambda, \lambda = \frac{s_{u,i_{\max}}(s_{u,\alpha} - s_{u,i_{\min}})}{s_{u,i_{\min}}(s_{u,i_{\max}} - s_{u,\alpha})}.$$

λ gives the relation between the amount of work taken from $P_{i_{\min}}$ and the amount of work of $P_{i_{\max}}$ that can be performed by P_α during its spared time.

$\mathcal{S}''$ still respects the given constraints:

$$\begin{aligned}
\sum_{i=1}^{m_u} \rho''_{u,i} &= \sum_{i=1}^{m_u} \rho'_{u,i} = \rho; \\
\sum_{i=1}^{m_u} \frac{\rho''_{u,i}}{s_{u,i}} &= \left(\sum_{\substack{i=1 \\ i \neq i_{\min},\alpha,i_{\max}}}^{m_u} \frac{\rho'_{u,i}}{s_{u,i}} \right) + \frac{\rho''_{u,i_{\min}}}{s_{u,i_{\min}}} + \frac{\rho''_{u,\alpha}}{s_{u,\alpha}} + \frac{\rho''_{u,i_{\max}}}{s_{u,i_{\max}}} \\
&= \left(\sum_{i=1}^{m_u} \frac{\rho'_{u,i}}{s_{u,i}} \right) + \frac{\epsilon_{\min} + \epsilon_{\max}}{s_{u,\alpha}} - \frac{\epsilon_{\min}}{s_{u,i_{\min}}} - \frac{\epsilon_{\max}}{s_{u,i_{\max}}} \\
&= \left(\sum_{i=1}^{m_u} \frac{\rho'_{u,i}}{s_{u,i}} \right) + \epsilon_{\min} \left(\frac{1+\lambda}{s_{u,\alpha}} - \frac{1}{s_{u,i_{\min}}} - \frac{\lambda}{s_{u,i_{\max}}} \right) \\
&= \left(\sum_{i=1}^{m_u} \frac{\rho'_{u,i}}{s_{u,i}} \right) + \\
&\quad \frac{\epsilon_{\min}}{s_{u,i_{\min}} s_{u,\alpha}} \left(s_{u,i_{\min}} + \frac{s_{u,i_{\max}}(s_{u,\alpha} - s_{u,i_{\min}})}{(s_{u,i_{\max}} - s_{u,\alpha})} - s_{u,\alpha} - \frac{s_{u,\alpha}(s_{u,\alpha} - s_{u,i_{\min}})}{(s_{u,i_{\max}} - s_{u,\alpha})} \right) \\
&= \sum_{i=1}^{m_u} \frac{\rho'_{u,i}}{s_{u,i}} \leq \frac{1}{\omega}.
\end{aligned}$$

And the power consumed by the new solution is not greater than the original optimal one:

$$\begin{aligned}
\sum_{i=1}^{m_u} \rho'_{u,i} \mathfrak{K}_{u,i} - \sum_{i=1}^{m_u} \rho''_{u,i} \mathfrak{K}_{u,i} &= \epsilon_{\min} \mathfrak{K}_{u,i_{\min}} + \epsilon_{\max} \mathfrak{K}_{u,i_{\max}} - (\epsilon_{\min} + \epsilon_{\max}) \mathfrak{K}_{u,\alpha} \\
&= \epsilon_{\min} (\mathfrak{K}_{u,i_{\min}} - \mathfrak{K}_{u,\alpha}) + \lambda \epsilon_{\min} (\mathfrak{K}_{u,i_{\max}} - \mathfrak{K}_{u,\alpha}) \\
&= \epsilon_{\min} (s_{u,\alpha} - s_{u,i_{\min}}) \left(\frac{\mathfrak{K}_{u,i_{\min}} - \mathfrak{K}_{u,\alpha}}{s_{u,\alpha} - s_{u,i_{\min}}} + \frac{s_{u,i_{\max}}}{s_{u,i_{\min}}} \frac{\mathfrak{K}_{u,i_{\max}} - \mathfrak{K}_{u,\alpha}}{s_{u,i_{\max}} - s_{u,\alpha}} \right) \\
&\geq \epsilon_{\min} (s_{u,\alpha} - s_{u,i_{\min}}) \left(\frac{\mathfrak{K}_{u,i_{\max}} - \mathfrak{K}_{u,\alpha}}{s_{u,i_{\max}} - s_{u,\alpha}} - \frac{\mathfrak{K}_{u,\alpha} - \mathfrak{K}_{u,i_{\min}}}{s_{u,\alpha} - s_{u,i_{\min}}} \right) \\
&\geq 0 \quad \text{because of the convexity of } \mathfrak{K}.
\end{aligned}$$

At each iteration, we set the throughput of either $i_{\min}$ or $i_{\max}$ to its throughput in $\tilde{\mathcal{S}}$, so the number of virtual processors which have different throughputs in $\mathcal{S}''$ and $\tilde{\mathcal{S}}$ is strictly decreasing. At the end, either one of the two other cases

is reached so $\tilde{\mathcal{S}}$ does not consume more power than $\mathcal{S}''$, or $\tilde{\mathcal{S}} = \mathcal{S}''$. Overall, our scheduling is optimal.

Then we consider the power consumption of $\tilde{\mathcal{S}}$:

$$\begin{aligned}
\mathfrak{P}_u(\rho) &= \tilde{\rho}_{i,i_0}\mathfrak{K}_{u,i_0} + \tilde{\rho}_{i,i_0+1}\mathfrak{K}_{u,i_0+1} \\
&= \frac{s_{u,i_0}(s_{u,i_0+1} - \omega\rho)}{\omega(s_{u,i_0+1} - s_{u,i_0})}\mathfrak{K}_{u,i_0} + \frac{s_{u,i_0+1}(\omega\rho - s_{u,i_0})}{\omega(s_{u,i_0+1} - s_{u,i_0})}\mathfrak{K}_{u,i_0+1} \\
&= \rho\frac{s_{u,i_0+1}\mathfrak{K}_{u,i_0+1} - s_{u,i_0}\mathfrak{K}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} - \frac{s_{u,i_0}s_{u,i_0+1}\left(\mathfrak{K}_{u,i_0+1} - \mathfrak{K}_{u,i_0}\right)}{\omega(s_{u,i_0+1} - s_{u,i_0})} \\
&= \omega\rho\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} - \frac{s_{u,i_0}\mathfrak{P}_{u,i_0+1} - s_{u,i_0+1}\mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} \\
&= \omega\rho\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} - s_{u,i_0}\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \mathfrak{P}_{u,i_0}\frac{s_{u,i_0+1} - s_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} \\
&= (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \mathfrak{P}_{u,i_0}
\end{aligned}$$

As $\mathfrak{P}$ is super-linear, we have, if $j < k$:

$$\frac{\mathfrak{P}_{u,k} - \mathfrak{P}_{u,j}}{s_{u,k} - s_{u,j}} \geq \frac{\mathfrak{P}_{u,j+1} - \mathfrak{P}_{u,j}}{s_{u,j+1} - s_{u,j}} \quad \Rightarrow \quad \mathfrak{P}_{u,k} \geq (s_{u,k} - s_{u,j})\frac{\mathfrak{P}_{u,j+1} - \mathfrak{P}_{u,j}}{s_{u,j+1} - s_{u,j}} + \mathfrak{P}_{u,j}$$

and, if $j > k$:

$$\frac{\mathfrak{P}_{u,j} - \mathfrak{P}_{u,k}}{s_{u,j} - s_{u,k}} \leq \frac{\mathfrak{P}_{u,j+1} - \mathfrak{P}_{u,j}}{s_{u,j+1} - s_{u,j}} \quad \Rightarrow \quad \mathfrak{P}_{u,k} \geq \mathfrak{P}_{u,j} - (s_{u,j} - s_{u,k})\frac{\mathfrak{P}_{u,j+1} - \mathfrak{P}_{u,j}}{s_{u,j+1} - s_{u,j}}$$

As $s_{u,i_0} \leq \omega\rho_u \leq s_{u,i_0+1}$ and $\mathfrak{P}$ is super-linear, we have, for all if $s_{u,i_0} > s_{u,i}$:

$$\begin{aligned}
(\omega\rho - s_{u,i})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} &= (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \\
&\quad \left((s_{u,i_0} - s_{u,i})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i}\right) \\
&\leq (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i_0} \\
&\leq (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \mathfrak{P}_{u,i_0} = \mathfrak{P}_u(\rho)
\end{aligned}$$

And, if $s_{u,i_0+1} \leq s_{u,i}$, so we have:

$$\begin{aligned}
(\omega\rho - s_{u,i})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} &\leq (\omega\rho - s_{u,i_0+1})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \\
&\quad \left(\mathfrak{P}_{u,i} - (s_{u,i} - s_{u,i_0+1})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}}\right) \\
&\leq (\omega\rho - s_{u,i_0+1})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i_0+1} \\
&\leq (\omega\rho - s_{u,i_0+1})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \mathfrak{P}_{u,i_0+1} \\
&\quad \textit{(* because } (\omega\rho - s_{u,i_0+1}) < 0 \textit{ *)} \\
&\leq (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \\
&\quad \left(\mathfrak{P}_{u,i_0+1} - (s_{u,i_0+1} - s_{u,i_0})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}}\right) \\
&\leq (\omega\rho - s_{u,i_0})\frac{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}}{s_{u,i_0+1} - s_{u,i_0}} + \mathfrak{P}_{u,i_0} = \mathfrak{P}_u(\rho)
\end{aligned}$$

Then i_0 is the mode that maximizes the formula:

$$(\omega\rho - s_{u,i})\frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i}$$

□

The following result shows how to solve the converse problem, namely maximizing the throughout subject to a prescribed bound on power consumption. The proof is similar to that of Proposition 1.

Proposition 2. *The maximum achievable throughput according to the power consumption limit* $\mathfrak{P}$ *is*

$$\rho_u(\mathfrak{P}) = \min\left\{\frac{s_{u,m_u}}{\omega}; \max_{1\leq i\leq m_u}\left\{\frac{\mathfrak{P}(s_{u,i+1} - s_{u,i}) + s_{u,i}\mathfrak{P}_{u,i+1} - s_{u,i+1}\mathfrak{P}_{u,i}}{\omega(\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i})}\right\}\right\},$$

and is obtained using two consecutive modes, P_{u,i_0} *and* P_{u,i_0+1}*, such that:* $\mathfrak{P}_{u,i_0} < \mathfrak{P} \leq \mathfrak{P}_{u,i_0+1}$.

Proof. We define a solution $\tilde{\mathcal{S}}$ as follows:

$$\tilde{\rho}_{u,i_0} = \frac{s_{u,i_0}(\mathfrak{P}_{u,i_0+1} - \mathfrak{P})}{\omega(\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0})} \qquad \tilde{\rho}_{u,i_0+1} = \frac{s_{u,i_0+1}(\mathfrak{P} - \mathfrak{P}_{u,i_0})}{\omega(\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0})} \qquad \tilde{\rho}_{u,i} = 0 \text{ if } i \notin \{i_0, i_0{+}1\}$$

We first show that $\tilde{\boldsymbol{\mathcal{S}}}$ **is feasible**:

$$\begin{aligned}
\sum_{i=1}^{m_u} \tilde{\rho}_{u,i} \frac{\mathfrak{P}_{u,i}\omega}{s_{u,i}} &= \frac{\mathfrak{P}_{u,i_0}(\mathfrak{P}_{u,i_0+1} - \mathfrak{P}) + \mathfrak{P}_{u,i_0+1}(\mathfrak{P} - \mathfrak{P}_{u,i_0})}{\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0}} = \mathfrak{P}; \\
\sum_{i=1}^{m_u} \frac{\tilde{\rho}_{u,i}\omega}{s_{u,i}} &= \frac{(\mathfrak{P}_{u,i_0+1} - \mathfrak{P})}{(\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0})} + \frac{(\mathfrak{P} - \mathfrak{P}_{u,i_0})}{(\mathfrak{P}_{u,i_0+1} - \mathfrak{P}_{u,i_0})} = 1.
\end{aligned}$$

Let $\mathcal{S}'$ be an optimal solution, $\mathcal{S}' = \{\rho'_{u,1}, \cdots, \rho'_{u,m_u}\}$. As $\mathcal{S}'$ is a solution of the linear program, it respects all the constraints. So:

$$\sum_{i=1}^{m_u} \rho'_{u,i}\mathfrak{K}_{u,i} \leq \mathfrak{P} \quad \text{and} \quad \sum_{i=1}^{m_u} \frac{\rho'_{u,i}\omega}{s_{u,i}} \leq 1.$$

Let $i_{\min}$ be the slowest mode used by $\mathcal{S}'$, and $i_{\max}$ the fastest. Then we can distinguish two cases:

- **If $\boldsymbol{i_{\min} > i_0}$ or $\boldsymbol{i_{\min} = i_0}$ and $\boldsymbol{\rho'_{u,i_0} = \tilde{\rho}_{u,i_0} - \epsilon_0 (\epsilon_0 > 0)}$:** (in both cases, $\rho'_{u,i_0} = \tilde{\rho}_{u,i_0} - \epsilon$) then we have:

$$\begin{aligned}
\left(\sum_{i=i_0+1}^{m_u} \rho'_{u,i}\right) \mathfrak{K}_{u,i_0+1} &\leq \sum_{i=i_0+1}^{m_u} \rho'_{u,i}\mathfrak{K}_{u,i} \\
&\leq \mathfrak{P} - \rho'_{u,i_0}\mathfrak{K}_{u,i_0} = \mathfrak{P} - (\tilde{\rho}_{u,i_0} - \epsilon)\mathfrak{K}_{u,i_0} \\
&\leq (\mathfrak{P} - \tilde{\rho}_{u,i_0}\mathfrak{K}_{u,i_0}) + \epsilon\mathfrak{K}_{u,i_0} = \tilde{\rho}_{u,i_0+1}\mathfrak{K}_{u,i_0+1} + \epsilon\mathfrak{K}_{u,i_0} \\
&\leq \mathfrak{K}_{u,i_0+1}\left(\tilde{\rho}_{u,i_0+1} + \epsilon\frac{\mathfrak{K}_{u,i_0}}{\mathfrak{K}_{u,i_0+1}}\right) \\
\Rightarrow \quad \sum_{i=i_0}^{m_u} \rho'_{u,i} &\leq \rho'_{u,i_0} + \sum_{i=i_0+1}^{m_u} \rho'_{u,i} \\
&\leq (\tilde{\rho}_{u,i_0} - \epsilon) + \left(\tilde{\rho}_{u,i_0+1} + \epsilon\frac{\mathfrak{K}_{u,i_0}}{\mathfrak{K}_{u,i_0+1}}\right) \\
&\leq \sum_{i=i_0}^{m_u} \tilde{\rho}_{u,i} - \epsilon\left(1 - \frac{\mathfrak{K}_{u,i_0}}{\mathfrak{K}_{u,i_0+1}}\right) \leq \sum_{i=i_0}^{m_u} \tilde{\rho}_{u,i}
\end{aligned}$$

And so our solution does not have a smaller throughput, and is thus also optimal.

- **If $i_{\max} < i_0 + 1$ or $i_{\max} = i_0 + 1$ and $\rho'_{u,i_0+1} = \tilde{\rho}_{u,i_0+1} - \epsilon_1 (\epsilon_1 > 0)$:** (in both cases, $\rho'_{u,i_0+1} = \tilde{\rho}_{u,i_0+1} - \epsilon$) then, we have:

$$\begin{aligned}
\frac{\omega \sum_{i=1}^{i_0} \rho'_{u,i}}{s_{u,i_0}} &\leq \sum_{i=1}^{i_0} \frac{\rho'_{u,i}\omega}{s_{u,i}} = \sum_{i=1}^{i_{\max}} \frac{\rho'_{u,i}\omega}{s_{u,i}} - \frac{\rho'_{u,i_0+1}\omega}{s_{u,i_0+1}} \\
&\leq 1 - \frac{\rho'_{u,i_0+1}\omega}{s_{u,i_0+1}} = \left(1 - \frac{\tilde{\rho}_{u,i_0+1}\omega}{s_{u,i_0+1}}\right) + \frac{\epsilon\omega}{s_{u,i_0+1}} \\
&\leq \frac{\omega\tilde{\rho}_{u,i_0}}{s_{u,i_0}} + \frac{\epsilon\omega}{s_{u,i_0+1}}
\end{aligned}$$

So the throughput of $\mathcal{S}'$ is:

$$\begin{aligned}
\sum_{i=1}^{i_{\max}} \rho'_{u,i} &\leq \sum_{i=1}^{i_0} \rho'_{u,i} + \rho'_{u,i_0+1} \\
&\leq \left(\tilde{\rho}_{u,i_0} + \epsilon\frac{s_{u,i_0}}{s_{u,i_0+1}}\right) + (\tilde{\rho}_{u,i_0} - \epsilon) \\
&\leq \sum_{i=1}^{m_u} \tilde{\rho}_{u,i} - \epsilon\left(1 - \frac{s_{u,i_0}}{s_{u,i_0+1}}\right) \leq \sum_{i=1}^{m_u} \tilde{\rho}_{u,i}
\end{aligned}$$

And so our solution does not have a smaller throughput, and is thus also optimal.

- **Otherwise** we use the same new scheduling $\mathcal{S}''$ than is the previous section:

$$\begin{cases}
\rho''_{u,i_{\min}} &= \rho'_{u,i_{\min}} - \epsilon_{\min} \\
\rho''_{u,\alpha} &= \rho'_{u,\alpha} + \epsilon_{\min} + \epsilon_{\max} \\
\rho''_{u,i_{\max}} &= \rho'_{u,i_{\max}} - \epsilon_{\max} \\
\rho''_{u,i} &= \rho'_{u,i} \quad \text{otherwise}
\end{cases}$$

with $\epsilon_{\min} = \min\left\{\rho'_{u,i_{\min}}; \frac{\rho'_{u,i_{\max}}}{\lambda}\right\}$, $\epsilon_{\max} = \epsilon_{\min}\lambda$, and $\lambda = \frac{s_{u,i_{\max}}(s_{u,\alpha} - s_{u,i_{\min}})}{s_{u,i_{\min}}(s_{u,i_{\max}} - s_{u,\alpha})}$.

From the previous section, we know that $\mathcal{S}''$ does not consume more power than $\mathcal{S}'$, and so still respects the given constraints. And the throughput achieved is

the same than $\mathcal{S}'$. By iterating this construction, we can extract an optimal scheduling where $i_{\min} = \alpha$ (each iteration sets the throughput of either $i_{\min}$ or $i_{\max}$ to zero).

We then conclude using arguments similar to the one used in the proof of Proposition 1. □

To the best of our knowledge, these uni-processor formulas, linking the throughput to the power consumption, are new, even for standard laws. They will prove to be very useful when dealing with multi-processor problems.

3.1 Minimizing power consumption

Thanks to Propositions 1 and 2, we do not need to specify the throughput for each frequency on any given processor. We only have to fix a throughput for each processor to know how to achieve the minimum power consumption on that processor. Furthermore, the bounded multi-port hypothesis is easy to take into account: either the outgoing capacity of the master is able to ensure the given throughput (BW $\geq \rho$), or the system as no solution. Overall, we have the following linear program (Equation (3)). This linear program is defined by three types of constraints:

- The first constraint states that the system has to ensure the given throughput
- The second set of constraints states that the processing capacity of a processor P_u as well as the bandwidth of the link from P_{master} to P_u are not exceeded
- The last constraint links the power consumption of one processor according to its throughput

$$\begin{cases} \text{Minimize } \mathfrak{P} = \displaystyle\sum_{u=1}^{p} \mathfrak{P}_u \text{ subject to} \\ \displaystyle\sum_{u=1}^{p} \rho_u = \rho \\ \forall u,\ \rho_u \ \leq\ \min\left\{\dfrac{s_{u,m_u}}{\omega}; \dfrac{b_u}{\delta}\right\} \\ \forall\ u,\ \forall\ 1 \leq i \leq m_u,\ \mathfrak{P}_u \geq (\omega\rho_u - s_{u,i})\dfrac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \end{cases} \tag{3}$$

For each value $\mathfrak{P}_u$ used in the objective function (recall that $\mathfrak{P}_u$ is the power consumption per time unit of P_u), we have m_u equations (see Proposition 1). When looking at the constraints, we observe that the problem can be optimally solved using a greedy strategy. We first sort processors in an increasing order according to their power consumption ratio. This power consumption ratio depends on the different modes of the processors, and the same processor will appear a number of times equal to its number of modes. Formally, we sort in non decreasing order the quantities $\left\{\frac{\mathfrak{P}_{u,i+1}-\mathfrak{P}_{u,i}}{s_{u,i+1}-s_{u,i}}\right\}$. The next step is to select the cheapest mode of the processors so that the system can achieve the required throughput, given that each processor throughput is limited by its maximal frequency and the bandwidth of the link between itself and the master. Altogether, we obtain Algorithm 1.

Algorithm 1: Greedy algorithm minimizing power consumption under a given throughput

Data: throughput ρ that has to be achieved
for $u = 1$ **to** p **do**
  $\mathcal{T}[u] \leftarrow 0$; /* *throughput of processor* P_u */
$\Phi \leftarrow 0$; /* *total throughput of the system* */
$\mathcal{L} \leftarrow$ sorted list of the P_{u_k,i_k} such that $\forall\ j$, $\frac{\mathfrak{P}_{u_j,1+i_j}-\mathfrak{P}_{u_j,i_j}}{s_{u_j,1+i_j}-s_{u_j,i_j}} \leq \frac{\mathfrak{P}_{u_{j+1},1+i_{j+1}}-\mathfrak{P}_{u_{j+1},i_{j+1}}}{s_{u_{j+1},1+i_{j+1}}-s_{u_{j+1},i_{j+1}}}$;
while $\Phi < \rho$ **do**
  $P_{u_k,i_k} \leftarrow \text{next}(\mathcal{L})$; /* *selection of next cheapest mode* */
  $\rho' \leftarrow \mathcal{T}[u_k]$; /* *previous throughput of* P_{u_k} *(at mode* $i_k - 1$*)* */
  $\mathcal{T}[u_k] \leftarrow \min\left\{\frac{s_{u_k,i_k}}{\omega}; \frac{b_{u_k}}{\delta}; \rho' + (\rho - \Phi)\right\}$; /* *new throughput of* P_{u_k} *(at mode* i_k*)* */
  if $\mathcal{T}[u_k] = \frac{b_{u_k}}{\delta}$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} \backslash \{P_{u_k,j}\}$; /* *no need to look at faster modes for* P_{u_k} */
  $\Phi \leftarrow \Phi + \mathcal{T}[u_k] - \rho'$;

One can detail more precisely the line labeled /* *new throughput* */ that gives the new throughput of P_{u_k} at mode i_k. This throughput is bounded by the maximum throughput at this speed, by the maximum communication throughput, and also by the previous throughput (ρ') plus the remaining throughput that has to be achieved ($\rho - \Phi$). We point out that, if the last selected mode is $P_{u_{k_0},i_{k_0}}$, Algorithm 1 will

1. fully use each processor having at least one mode consuming strictly less than $P_{u_{k_0},i_{k_0}}$, and this either at the throughput of the bandwidth if reached (this throughput is achieved according to Proposition 1), or at the largest single fastest mode that consumes strictly less than $P_{u_{k_0},i_{k_0}}$ or at the same mode than $P_{u_{k_0},i_{k_0}}$;
2. either not use at all or fully use at its first non-trivial mode any processor whose first non-trivial mode consumes exactly the same than $P_{u_{k_0},i_{k_0}}$;
3. not use at all any processor whose first non-trivial mode consumes strictly more than the mode $P_{u_{k_0},i_{k_0}}$;
4. use $P_{u_{k_0},i_{k_0}}$ at the minimum throughput so the system achieves a throughput of ρ (according to Proposition 1).

Theorem 1. *Algorithm 1 optimally solves problem* MinPower *(ρ) (see linear program (3)).*

Proof. Let $\tilde{\mathcal{S}} = \{\tilde{\rho}_u\}$ be the throughput of each processor given by Algorithm 1, and $\mathcal{S} = \{\rho_u\}$ be an optimal solution of the problem, different from our solution. We know that there exists at least one processor whose throughput in $\mathcal{S}$ is strictly lower that its throughput in $\tilde{\mathcal{S}}$, otherwise the power consumed by $\mathcal{S}$ would be greater than the one of $\tilde{\mathcal{S}}$. Let P_m be one of these processors. Of course, the remaining work of P_m in $\tilde{\mathcal{S}}$ has to be performed by (at least) one other processor, and thus at least one processor has a throughput strictly greater in $\mathcal{S}$ than in $\tilde{\mathcal{S}}$ (otherwise, $\mathcal{S}$ could not achieve a total throughput of ρ). Let P_M be one of these processors.

The idea is then to transfer a portion of work from P_M to P_m. This amount of work ϵ equals to the minimum of the additional throughput needed by P_m to achieve a throughput $\tilde{\rho}_m$, and of the excess of throughput of P_M when compared to $\tilde{\mathcal{S}}$:

$$\epsilon = \min\{\tilde{\rho}_m - \rho_m; \rho_M - \tilde{\rho}_M\}.$$

What do we know about P_M in $\tilde{\mathcal{S}}$? We know for sure that Algorithm 1 required from it a throughput $\tilde{\rho}_M$ (which may be equal to 0). That means, according to the selection process of Algorithm 1, that: 1) either P_M is saturated by its bandwidth, but in that case, $\tilde{\rho}_M \geq \rho_M$, which contradicts the definition of P_M, or 2) P_M is saturated at a given mode $P_{M,i}$, and the next mode $P_{M,i+1}$ has a power consumption ratio greater than, or equal to, any other selected processor, P_m included, or 3) P_M is

not saturated, but in that case it is the last selected mode by Algorithm 1 and so has a power consumption ratio greater than, or equal to, any other selected processor, P_m included. Overall, the power consumption ratio of P_M is greater than, or equal to, the one of P_m.

Let $\mathcal{S}'$ be the scheduling where:

$$\rho'_m = \rho_m + \epsilon; \qquad \rho'_M = \rho_M - \epsilon; \qquad \rho'_{i,j} = \rho_{i,j} \text{ otherwise.}$$

Then, the power consumed by $\mathcal{S}'$ is

$$\sum_{u=1}^{p} \mathfrak{P}'_u = \left(\sum_{\substack{u=1 \\ u \neq m,M}}^{p} \mathfrak{P}_u \right) + \mathfrak{P}'_m + \mathfrak{P}'_M.$$

$$\begin{aligned} \mathfrak{P}'_m &= \max_i \left\{ (\omega\rho'_m - s_{m,i}) \frac{\mathfrak{P}_{m,i+1} - \mathfrak{P}_{m,i}}{s_{m,i+1} - s_{m,i}} + \mathfrak{P}_{m,i} \right\} \\ &= \rho'_m \left(\omega \frac{\mathfrak{P}_{m,i_m+1} - \mathfrak{P}_{m,i_m}}{s_{m,i_m+1} - s_{m,i_m}} \right) + \left(\mathfrak{P}_{m,i_m} - s_{m,i_m} \frac{\mathfrak{P}_{m,i_m+1} - \mathfrak{P}_{m,i_m}}{s_{m,i_m+1} - s_{m,i_m}} \right) \\ &= (\rho_m + \epsilon)\lambda_{m_1} + \lambda_{m_2} = \mathfrak{P}_m + \epsilon\lambda_{m_1} \\ \text{and } \mathfrak{P}'_M &= (\rho_M - \epsilon)\lambda_{M_1} + \lambda_{M_2} = \mathfrak{P}_M - \epsilon\lambda_{M_1}. \end{aligned}$$

We also know that $\frac{\mathfrak{P}_{m,i_m+1} - \mathfrak{P}_{m,i_m}}{s_{m,i_m+1} - s_{m,i_m}} \leq \frac{\mathfrak{P}_{M,i_{M+1}} - \mathfrak{P}_{M,i_M}}{s_{M,i_{M+1}} - s_{M,i_M}}$, because of the Greedy selection, so $\lambda_{m_1} - \lambda_{M_1} \leq 0$, and :

$$\sum_{u=1}^{p} \mathfrak{P}'_u \leq \left(\sum_{\substack{u=1 \\ u \neq m,M}}^{p} \mathfrak{P}_u \right) + \mathfrak{P}_m + \mathfrak{P}_M = \sum_{u=1}^{p} \mathfrak{P}_u.$$

We can iterate these steps as long as $\mathcal{S}$ is different of $\tilde{\mathcal{S}}$, hence proving the optimality of our scheduling. □

3.2 Maximizing the throughput

Maximizing the throughput is a very similar problem. We only need to adapt Algorithm 1 so that the objective function considered during the selection process is replaced by the power consumption:

$$\mathcal{T}[u_k] \leftarrow \min \left\{ \mathfrak{P}_{u_k,i_k}; \left(\omega \frac{b_{u_k}}{\delta} - s_{u_k,i_k} \right) \frac{\mathfrak{P}_{u_k,i_k+1} - \mathfrak{P}_{u_k,i_k}}{s_{u_k,i_k+1} - s_{u_k,i_k}} + \mathfrak{P}_{u_k,i_k}; \mathfrak{P}' + (\mathfrak{P} - \Psi) \right\}.$$

where Ψ is the current power consumption (we iterate while $\Psi \leq \mathfrak{P}$). The proof that this modified algorithm **optimally solves** problem MaxThroughput ($\mathfrak{P}$) is very similar to that of Algorithm 1 and can be found in [20].

4 Model with start-up overheads

When we move to more realistic models, the problem gets much more complicated. In this section, we still look at the problem of minimizing the power consumption of the system with a throughput bound, but now we suppose that there is a power consumption overhead when turning a processor on. We denote this problem MinPowerOverhead (ρ). First we need to modify the closed-form formula given by Proposition 1, in order to determine the power consumption of processor P_u when running at throughput ρ_u during t time-units. The new formula is then:

$$\begin{aligned}
\mathfrak{P}_u(t,\rho_u) &= \max_{0\leq i<m_u}\left\{(\omega\rho_u - s_{u,i})\frac{\mathfrak{P}_{u,i+1}(t)-\mathfrak{P}_{u,i}(t)}{s_{u,i+1}-s_{u,i}} + \mathfrak{P}_{u,i}(t)\right\}\\
&= \max_{0\leq i<m_u}\left\{(\omega\rho_u - s_{u,i})\frac{\mathfrak{P}^{(1)}_{u,i+1}-\mathfrak{P}^{(1)}_{u,i}}{s_{u,i+1}-s_{u,i}}\cdot t + \mathfrak{P}^{(1)}_{u,i}\cdot t\right\} + \mathfrak{P}^{(2)}_u\\
&= \mathfrak{P}^{(1)}_u(\rho_u)\cdot t + \mathfrak{P}^{(2)}_u.
\end{aligned}$$

The overhead is payed only once, and the throughput ρ_u is still obtained by using the same two modes P_{u,i_0} and P_{u,i_0+1} as in Proposition 1. We first run the mode P_{u,i_0} during $\frac{t(s_{u,i_0+1}-\rho_u\omega)}{s_{u,i_0+1}-s_{u,i_0}}$ time-units, then the mode P_{u,i_0+1} during $\frac{t(\rho_u\omega-s_{u,i_0})}{s_{u,i_0+1}-s_{u,i_0}}$ time-units (these values are obtained from the fraction of time the mode are used per time-unit). We can now prove the following dominance property about optimal schedules:

Proposition 3. *There exists an optimal schedule in which all processors, except possibly one, are used at a maximum throughput, i.e., either the throughput dictated by their bandwidth, or the throughput achieved by one of their execution modes.*

Proof. Let $\mathcal{S}$ be an optimal schedule without that property. We study $\mathcal{S}$ during an interval of arbitrary length, say t time-units. As we have no control on the behavior of $\mathcal{S}$, every processor can be turned on and off arbitrarily many times. Let $\Delta_u(t)$ be the communication volume received by P_u during the t time-units, and $\Omega_u(t)$ the computational volume performed during this interval. Both volumes are not necessary equal, as we chose an arbitrary time interval. We now compare $\mathcal{S}$ and $\mathcal{S}'$, with $\mathcal{S}'$ being the schedule identical to $\mathcal{S}$ outside of the considered interval and

which, during that interval, sends tasks to each processor P_u at rate $\frac{\Delta_u(t)}{t}$, and where each processor P_u computes with a throughput of $\frac{\Omega_u(t)}{t}$. We need to check that P_u does not starve, i.e., that it always has in memory some task ready to be executed, in order to ensure the computational throughput. The most constrained problem occurs when the total communication throughput is lower than the total computational throughput. We suppose that the memory contains, at time $t = 0$, $\mathcal{M}_0$ (which may be equal to zero) tasks:

- **Communications under $\mathcal{S}'$ are feasible:** Under $\mathcal{S}$, each processor received a volume of tasks equals to $\Delta_u(t)$ during t time-units, so its bandwidth throughput was greater than or equal to $\frac{\Delta_u(t)}{t}$, which means that $\mathcal{S}'$ also respects the bandwidth constraints. For the master's point of view, the total volume of communication during the t time-units under $\mathcal{S}$ is $\sum_{u=1}^{p} \Delta_u(t)$, so we had: $\sum_{u=1}^{p} \Delta_u(t) \leq t \cdot \text{BW}$. Consequently, $\sum_{u=1}^{p} \frac{\Delta_u(t)}{t}$ and $\mathcal{S}'$ respects the bounded capacity of the master.

- **Computations under $\mathcal{S}'$ are feasible:** As $\mathcal{S}$ is feasible, we have $\Omega_u(t) \leq \mathcal{M}_0 + \Delta_u(t)$. According to Proposition 1, we know that, under $\mathcal{S}'$, the processor needs only two consecutive modes to perform its computational throughput, P_{u,i_0} and P_{u,i_0+1}, $\frac{s_{u,i_0}}{\omega} \leq \frac{\Omega_u(t)}{t} \leq \frac{s_{u,i_0+1}}{\omega}$ (i_0 might be equal to zero). We run at the slowest mode first, in order to minimize the power consumption and to be sure to have enough tasks in memory to run at the second mode later and to obtain a feasible schedule. When using the mode P_{u,i_0} during $t_1 = \frac{t\left(s_{u,i_0+1} - \frac{\Omega_u(t)}{t}\omega\right)}{s_{u,i_0+1} - s_{u,i_0}}$ time-units, and the mode P_{u,i_0+1} during $t_2 = \frac{t\left(\frac{\Omega_u(t)}{t}\omega - s_{u,i_0}\right)}{s_{u,i_0+1} - s_{u,i_0}}$ time-units, we obtain a feasible solution. Indeed, either the fastest computation rate is smaller than the communication throughput, and so the number of stored tasks increases with time, or $\frac{\Delta_u(t)}{t} \leq \frac{s_{u,i_0+1}}{\omega}$. In that case, after t_1 time-units, the processor has in memory a fraction of tasks equal to: $\mathcal{M}_0 + \left(\frac{\Delta_u(t)}{t} - \frac{s_{u,i_0}}{\omega}\right) t_1$. Then, if we look at the memory $\mathcal{M}$ of the processor during the computation under the mode P_{u,i_0+1} after t' time-units

$(t' \leq t_2)$, we have:

$$\begin{aligned}
\mathcal{M} &= \mathcal{M}_0 + \left(\frac{\Delta_u(t)}{t} - \frac{s_{u,i_0}}{\omega}\right) t_1 - \left(\frac{s_{u,i_0+1}}{\omega} - \frac{\Delta_u(t)}{t}\right) t' \geq \mathcal{M}_0 + \left(\frac{\Delta_u(t)}{t} - \frac{s_{u,i_0}}{\omega}\right) t_1 - \left(\frac{s_{u,i_0+1}}{\omega} - \frac{\Delta_u(t)}{t}\right) t_2 \\
&= \mathcal{M}_0 + \frac{\Delta_u(t)}{t}(t_1+t_2) - \left(\frac{t_1 s_{u,i_0}}{\omega} + \frac{t_2 s_{u,i_0+1}}{\omega}\right) \\
&= \mathcal{M}_0 + \Delta_u(t)\,(t_1+t_2) - t\left(\frac{s_{u,i_0}\left(s_{u,i_0+1} - \frac{\Omega_u(t)}{t}\omega\right)}{\omega(s_{u,i_0+1} - s_{u,i_0})} + \frac{s_{u,i_0+1}\left(\frac{\Omega_u(t)}{t}\omega - s_{u,i_0}\right)}{\omega(s_{u,i_0+1} - s_{u,i_0})}\right) \\
&= \mathcal{M}_0 + \Delta_u(t) - t\frac{\Omega_u(t)}{t} \geq \Omega_u(t) - \Omega_u(t) = 0
\end{aligned}$$

So the processor memory always contains some tasks, and then $\mathcal{S}'$ is feasible.

- **$\mathcal{S}'$ does not consume more power than $\mathcal{S}$:** We only pay a power overhead each time a processor is turned on, and $\mathcal{S}'$ turned on only once each processor used by $\mathcal{S}$. Furthermore, the average throughput of each processor is the same under $\mathcal{S}'$ than under $\mathcal{S}$. Overall, the power consumption of $\mathcal{S}'$ is not greater than that of $\mathcal{S}$.

Consider now the throughput of each worker under $\mathcal{S}'$. If $\mathcal{S}'$ does not have the desired property, then there exist (at least) two processors P_m and P_M that are not running at a maximum throughput (i.e., dictated by one of the modes or by the bandwidth). We know that these throughputs can be achieved using only two modes P_{M,i_M} and P_{M,i_M+1} for P_M (P_{M,i_M} may have a throughput of zero), and P_{m,i_m}, P_{m,i_m+1} for P_m. Suppose that P_M consumes more power at its throughput than P_m at its own one. This means that:

$$\frac{\mathfrak{P}_{m,i_m+1}(t) - \mathfrak{P}_{m,i_m}(t)}{s_{m,i_m+1} - s_{m,i_m}} \leq \frac{\mathfrak{P}_{M,i_{M+1}}(t) - \mathfrak{P}_{M,i_M}(t)}{s_{M,i_{M+1}} - s_{M,i_M}}$$

We now construct a new schedule $\mathcal{S}''$ from $\mathcal{S}'$, with $\mathcal{S}''$ equal to:

$$\rho''_m = \rho'_m + \epsilon \quad \rho''_M = \rho'_M - \epsilon \quad \text{and} \quad \rho''_u = \rho'_u \text{ otherwise,}$$

and $\epsilon = \min\left\{\frac{b_u}{\delta} - \rho'_m; \frac{s_{m,i_m+1}}{\omega} - \rho'_m; \rho'_M - \frac{s_{M,i_M}}{\omega}\right\}$. Then, if we compare the power consumed by $\mathcal{S}''$ and $\mathcal{S}'$:

$$\sum_{u=1}^{p} \mathfrak{P}''_u(t) = \left(\sum_{\substack{u=1 \\ u\neq m,M}}^{p} \mathfrak{P}'_u(t)\right) + \mathfrak{P}''_m(t) + \mathfrak{P}''_M(t).$$

As a reminder, we saw in the proof of Theorem 1 that:

$$\mathfrak{P}''_m(t) = \mathfrak{P}'_m(t) + \epsilon\lambda_{m_1} \qquad \text{and} \quad \mathfrak{P}''_M(t) = \mathfrak{P}'_M(t) - \epsilon\lambda_{M_1},$$

with

$$\lambda_{m_1} = \omega \frac{\mathfrak{P}_{m,i_m+1}(t) - \mathfrak{P}_{m,i_m}(t)}{s_{m,i_m+1} - s_{m,i_m}} \quad \text{and} \quad \lambda_{M_1} = \omega \frac{\mathfrak{P}_{M,i_{M+1}}(t) - \mathfrak{P}_{M,i_M}(t)}{s_{M,i_{M+1}} - s_{M,i_M}}.$$

So $\epsilon(\lambda_{m_1} - \lambda_{M_1}) \leq 0$, and:

$$\sum_{u=1}^{p} \mathfrak{P}''_u(t) \leq \left(\sum_{\substack{u=1 \\ u \neq m,M}}^{p} \mathfrak{P}'_u(t) \right) + \mathfrak{P}'_m(t) + \mathfrak{P}'_M(t) = \sum_{u=1}^{p} \mathfrak{P}'_u(t).$$

Then $\mathcal{S}''$ achieves the same throughput as $\mathcal{S}'$, and does not consume more power than $\mathcal{S}'$. As the number of processors that are not at a maximum throughput is strictly smaller in $\mathcal{S}''$ than in $\mathcal{S}'$, we can iterate the process until at most one processor is unsaturated. □

Unfortunately, Proposition 3 does not help design an optimal algorithm. However, we prove that a modified version of the previous algorithm remains asymptotically optimal. The general principle of the approach is as follows: instead of looking at the power consumption per time-unit, we look at the energy consumed during d time-units, where d will be defined later. Let α_u be the throughput of P_u during d time-units. Thus, the throughput of each processor per time-unit is $\rho_u = \frac{\alpha_u}{d}$. As all processors are not necessarily enrolled, let $\mathcal{U}$ be the set of selected processors' index. The constraint on the energy consumption can be written:

$$\forall\ u,\ \forall\ 1 \leq i \leq m_u,\ \mathfrak{P}_u \cdot d \geq \left((\omega\rho_u - s_{u,i}) \frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \right) \cdot d + \mathfrak{P}_u^{(2)},$$

or,

$$\forall\ u,\ \forall\ 1 \leq i \leq m_u,\ \mathfrak{P}_u - \frac{\mathfrak{P}_u^{(2)}}{d} \geq (\omega\rho_u - s_{u,i}) \frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i}$$

The linear program is then:

$$\begin{cases} \text{MINIMIZE } \mathfrak{P} = \sum_{u \in \mathcal{U}} \mathfrak{P}_u \text{ SUBJECT TO} \\ \sum_{u=1}^{p} \rho_u = \rho \\ \forall u,\ \rho_u \ \leq\ \min\left\{ \frac{s_{u,m_u}}{\omega}; \frac{b_u}{\delta} \right\} \\ \forall\ u \in \mathcal{U},\ \forall\ 1 \leq i \leq m_u,\ \mathfrak{P}_u - \frac{\mathfrak{P}_u^{(2)}}{d} \geq (\omega\rho_u - s_{u,i}) \frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \end{cases} \tag{4}$$

However, this linear program cannot be solved unless we know $\mathcal{U}$. So we need to add some constraints. In the meantime, we make a tiny substitution into the objective function, in order to simplify one constraint:

$$\begin{cases} \text{MINIMIZE } \mathfrak{P} = \sum_{u=1}^{p} \left(\mathfrak{P}_u + \frac{\mathfrak{P}_u^{(2)}}{d} \right) \text{ SUBJECT TO} \\ \sum_{u=1}^{p} \rho_u = \rho \\ \forall u,\ \rho_u \ \leq\ \min\left\{ \frac{s_{u,m_u}}{\omega}; \frac{b_u}{\delta} \right\} \\ \forall\ u,\ \forall\ 1 \leq i \leq m_u,\ \mathfrak{P}_u \geq (\omega\rho_u - s_{u,i}) \frac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \end{cases} \tag{5}$$

The inequalities are stronger than previously, so every solution of (5) is a solution of (4). Of course, optimal solutions for (5) are most certainly not optimal for the initial problem (4). However, the larger d, the closer the constraints are from each other. Furthermore, Algorithm 1 builds optimal solutions for (5). So, the expectation is that when d becomes large, solutions built by Algorithm 1 becomes good approximate solutions for (5). Indeed we derive the following result:

Theorem 2. *Algorithm 1 is asymptotically optimal for problem* MINPOWEROVERHEAD *(ρ)(see linear program (4)).*

Proof. If the application $\mathcal{A}$ is composed of $\mathcal{B}$ tasks, the optimal scheduling time will be $\mathcal{T} = \frac{\mathcal{B}}{\rho}$, where ρ is the throughput bound. We note $\mathfrak{P}_{opt}$ the optimal power

consumption that would be obtained in the ideal model, $\mathfrak{P}^*$ the optimal power consumption that can be achieve under the model with start-up overheads, and $\mathfrak{P}$ the power consumption given by Algorithm 1.

As the model with start-up overheads is more constrained than the fluid model, the minimum power consumption under this model is greater than under the fluid model, so we have $\mathfrak{P}_{opt} \leq \mathfrak{P}^* \leq \mathfrak{P}$. Also, one can remark that the power consumption of the solution given by Algorithm 1 is a function of the time interval, as the start-up overheads are paid only once each d time-units. Thus, during t time-units:

$$\mathfrak{P}(t) \leq \mathfrak{P}_{opt} \cdot t + \left\lceil \frac{t}{d} \right\rceil \sum_{u=1}^{p} \mathfrak{P}_u^{(2)} = \mathfrak{P}_{opt} \cdot t + \left\lceil \frac{t}{d} \right\rceil \cdot p \cdot \max_{u=1}^{p} \left\{ \mathfrak{P}_u^{(2)} \right\}.$$

If we fix $d = \sqrt{\mathcal{T}}$, we have

$$\mathfrak{P}(\mathcal{T}) \leq \mathfrak{P}^* \cdot \mathcal{T} + \left(1 + \sqrt{\mathcal{T}}\right) \cdot p \cdot \max_{u=1}^{p} \left\{ \mathfrak{P}_u^{(2)} \right\}. \tag{6}$$

Then, when comparing $\mathfrak{P}$ and $\mathfrak{P}^*$ during the scheduling of the $\mathcal{B}$ tasks of application $\mathcal{A}$, we obtain:

$$\begin{aligned} \frac{\mathfrak{P}(\mathcal{T})}{\mathfrak{P}^*(\mathcal{T})} &= 1 + \left(\frac{1}{\mathcal{T}} + \frac{1}{\sqrt{\mathcal{T}}}\right) \frac{p \cdot \max_{u=1}^{p} \left\{ \mathfrak{P}_u^{(2)} \right\}}{\mathfrak{P}^*} \\ &= 1 + \mathcal{O}\left(\frac{1}{\sqrt{\mathcal{T}}}\right). \end{aligned}$$

which achieves the proof of optimality of Algorithm 1. □

5 Related Work

Several papers have been targeting the minimization of power consumption. Most of them suppose they can switch to arbitrary speed values. Here is a brief overview:

- **Unit time tasks.** Bunder in [5] focuses on the problem of offline scheduling unit time tasks with release dates, while minimizing the makespan or the total flow time on one processor. He chooses to have a continuous range of speeds for the processors. He extends his work from one processor to multi-processors, but unlike this paper, does not take any communication time into account. His approach corresponds to scheduling on multi-core processors. He also proves

the NP-completeness of the problem of minimizing the makespan on multiprocessors with jobs of different amount of work. Authors in [2] concentrate on minimizing the total flow time of unit time jobs with release dates on one processor. After proving that no online algorithm can achieve a constant competitive ratio if job have arbitrary sizes, they exhibit a constant competitive online algorithm and solve the offline problem in polynomial time. Contrarily to [5] where tasks are gathered into blocks and scheduled with increasing speed in order to minimize the makespan, here the authors prove that the speed of the blocks need to be decreasing in order to minimize both total flow time and the energy consumption.

- **Communication-aware.** In [24], the authors are interested about scheduling task graphs with data dependences while minimizing the energy consumption of both the processors and the inter-processor communication devices. They demonstrate that in the context of multiprocessor systems, the inter-processor communications were an important source of consumption, and their algorithm reduces up to 80% the communications. However, as they focus on multiprocessor problems, they only consider the energy consumption of the communications, and they suppose that the communication times are negligible compared to the computation times.

- **Discrete voltage case.** In [18], the authors deal with the problem of scheduling tasks on a single processor with discrete voltages. They also look at the model where the energy consumption is related to the task, and describe how to split the voltage for each task. They extend their work in [19] to online problems. In [26], the authors add the constraint that the voltage can only be changed at each cycle of every task, in order to limit the number of transitions and thus the energy overhead. They find that under this model, the minimal number of frequency transitions in order to minimize the energy may be greater than two.

- **Task-related consumption.** [3] addresses the problem of periodic independent real-time tasks on one processor, the period being a deadline to all tasks. The particularity of this work is that they suppose the energy consumption is related to the *task* that is executed on the processor. They exhibit a polynomial algorithm to find the optimal speed of each task, and they prove that EDF can be used to obtain a feasible schedule with these optimal speed values.

- **Deadlines.** Many papers are trying to minimize the energy consumed by the platform given a set of deadlines for all tasks on the system. In [21], the authors focus on the problem where tasks arrive according to some release dates. They show that during any elementary time interval defined by some release dates and deadlines of applications, the optimal voltage is constant, and they determine this voltage, as well as the minimum constant speed for each job. [4] improves the best known competitive ratio to minimize the energy while respecting all deadlines. [8] works with an overloaded processor (which means that no algorithm can finish all the jobs) and try to maximize the throughput. Their online algorithm is $O(1)$ competitive for both throughput maximization and energy minimization. [10] has a similar approach by allowing task rejection, and proves the NP-hardness of the studied problem.

- **Slack sharing.** In [27, 22], the authors investigate dynamic scheduling. They consider the problem of scheduling DAGs before deadlines, using a semi-clairvoyant model. For each task, the only information available is the worst-case execution time. Their algorithm operates in two steps: first a greedy static algorithm schedules the tasks on the processors according to their worst-case execution times and the deadline, and reduces the processors speed so that each processor meets the deadline. Then, if a task ends sooner than according to the static algorithm, a dynamic slack sharing algorithm uses the extra-time to reduce the speed of computations for the following tasks. The authors investigate the problem with time overhead and voltage overhead when changing processor speeds, and adapt their algorithm accordingly. However, they do not take communications into account.

- **Heterogenous multiprocessor systems.** Authors in [11] study the problem of scheduling real-time tasks on two heterogenous processors. They provide a FPTAS to derive a solution very close to the optimal energy consumption with a reasonable complexity. In [17], the authors propose a greedy algorithm based on affinity to assign frame-based real-time tasks, and then they re-assign them in pseudo-polynomial time when any processing speed can be assigned for a processor. Authors of [25] propose an algorithm based on integer linear programming to minimize the energy consumption without guarantees on the schedulability of a derived solution for systems with discrete voltage. Some authors also explored the search of approximation algorithms for the minimization of allocation cost of processors under energy constraints [9, 16].

6 Conclusion

In this paper, we have studied the problem of scheduling a single application with power consumption constraints, on a heterogeneous master-worker platform. We derived new closed-form relations between the throughput and the power consumption at the processor level. These formulas enabled us to develop an optimal bi-criteria algorithm under the ideal power consumption model.

Moving to a more realistic model with start-up overheads, we were able to prove that our approach provides an asymptotically optimal solution. We hope that our results will provide a sound theoretical basis for forthcoming studies.

As future work, it would be interesting to address sophisticated models with frequency switching costs, which we expect to lead to NP-hard optimization problems, and then look for some approximation algorithms.

Acknowledgment This work was supported in part by the ANR StochaGrid project.

References

[1] M. Adler, Y. Gong, and A. L. Rosenberg. Optimal sharing of bags of tasks in heterogeneous clusters. In *15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA'03)*, pages 1–10. ACM Press, 2003.

[2] Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms*, 3(4):49, 2007.

[3] Hakan Aydin, Rami Melhem, Daniel Mosse, and Pedro Mejia-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proceedings of the IEEE EuroMicro Conférence on Real-Time Systems*, pages 225–232, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[4] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 520–529, 17-19 Oct. 2004.

[5] David P. Bunde. Power-aware scheduling for makespan and flow. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 190–196, New York, NY, USA, 2006. ACM.

[6] T. Burd. *Energy-Efficient Processor System Design*. PhD thesis, Berkeley, 2001. Available at the url http://bwrc.eecs.berkeley.edu/Publications/2001/THESES/energ_eff_process-sys_des/BurdPhD.pdf.

[7] H. Casanova and F. Berman. *Grid Computing: Making The Global Infrastructure a Reality*, chapter Parameter Sweeps on the Grid with APST. John Wiley, 2003. Hey, A. and Berman, F. and Fox, G., editors.

[8] Ho-Leung Chan, Wun-Tat Chan, Tak-Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence W. H. Wong. Energy efficient online deadline scheduling. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 795–804, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[9] Jian-Jia Chen and Tei-Wei Kuo. Allocation cost minimization for periodic hard real-time tasks in energy-constrained dvs systems. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 255–260, New York, NY, USA, 2006. ACM.

[10] Jian-Jia Chen, Tei-Wei Kuo, Chia-Lin Yang, and Ku-Jei King. Energy-efficient real-time task scheduling with task rejection. In *DATE '07: Proceedings of the conference on Design, automation and test in Europe*, pages 1629–1634, San Jose, CA, USA, 2007. EDA Consortium.

[11] Jian-Jia Chen and Lothar Thiele. Energy-efficient task partition for periodic real-time tasks on platforms with dual processing elements. In *International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE Computer Society Press, 2008.

[12] Rong Ge, Xizhou Feng, and Kirk W. Cameron. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 34, Washington, DC, USA, 2005. IEEE Computer Society.

[13] B. Hong and V.K. Prasanna. Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press, 2004.

[14] Bo Hong and Viktor K. Prasanna. Adaptive allocation of independent tasks to maximize throughput. *IEEE Trans. Parallel Distributed Systems*, 18(10):1420–1435, 2007.

[15] 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. *Parallel and Distributed Processing Symposium, International*, 0:340, 2006.

[16] Heng-Ruey Hsu, Jian-Jia Chen, and Tei-Wei Kuo. Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 1061–1066, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[17] Tai-Yi Huang, Yu-Che Tsai, and E.T.-H. Chu. A near-optimal solution for the heterogeneous multi-processor single-level voltage setup problem. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10, March 2007.

[18] Tohru Ishihara and Hiroto Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 1998. ACM.

[19] Takanori Okuma, Tohru Ishihara, and Hiroto Yasuura. Real-time task scheduling for a variable voltage processor. In *ISSS '99: Proceedings of the 12th international symposium on System synthesis*, page 24, Washington, DC, USA, 1999. IEEE Computer Society.

[20] Jean-François Pineau. *Communication-aware scheduling on heterogeneous master-worker platforms*. PhD thesis, ENS Lyon, 2008. Available at `http://graal.ens-lyon.fr/~jfpineau`.

[21] Gang Quan and Xiaobo Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *Design Automation Conference*, pages 828–833, 2001.

[22] Cosmin Rusu, Rami Melhem, and Daniel Mossé. Multi-version scheduling in rechargeable energy-aware real-time systems. *J. Embedded Comput.*, 1(2):271–283, 2005.

[23] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.*, 1(1):94–125, 2004.

[24] Girish Varatkar and Radu Marculescu. Communication-aware task scheduling and voltage selection for total systems energy minimization. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 510, Washington, DC, USA, 2003. IEEE Computer Society.

[25] Yang Yu and V.K. Prasanna. Power-aware resource allocation for independent tasks in heterogeneous real-time systems. *International Conference on Parallel and Distributed Systems*, pages 341–348, Dec. 2002.

[26] Yumin Zhang, Xiaobo Sharon Hu, and Danny Z. Chen. Energy minimization of real-time tasks on variable voltage processors with transition energy overhead. In *ASPDAC: Proceedings of the 2003 conference on Asia South Pacific design automation*, pages 65–70, New York, NY, USA, 2003. ACM.

[27] Dakai Zhu, Rami Melhem, and Bruce R. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(7):686–700, 2003.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399