Energy-Aware Scheduling of Flow Applications on Master-Worker Platforms

Jean-François Pineau[5], Yves Robert[2,3,4], and Frédéric Vivien[1,3,4]

[1] INRIA
[2] ENS Lyon
[3] Université de Lyon
[4] LIP laboratory, ENS Lyon–CNRS–INRIA–UCBL, Lyon, France
[5] LIRMM laboratory, UMR 5506, CNRS–Université Montpellier 2, France

Abstract. 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 derive optimal or asymptotically optimal solutions for both models.

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 [1]. And these estimates ignore the additional cost of dedicated cooling. Current estimates state that cooling costs $1 to $3 per watt of heat dissipated [2]. This is just one of the many economical reasons why energy-aware scheduling is an important issue, even without considering battery-powered systems such as laptop and embedded systems.

Many important scheduling problems involve large collections of identical tasks [3,4]. 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

H. Sips, D. Epema, and H.-X. Lin (Eds.): Euro-Par 2009, LNCS 5704, pp. 281–292, 2009.

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 contribution 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:

- 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.

This 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. This simplifying hypothesis could be alleviated by considering 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: 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 [5,6]. There is a limit on the total amount of data that

the master can send per time-unit, denoted as 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. We also assume a computation model called *synchronous start* computation: the computation of a task on a worker can start at the same time as the reception of the task starts, provided that the computation rate is no greater than the communication rate (the communication must complete before the computation). This models the fact that, in several applications, only the first bytes of data are needed to start executing a task. In addition, the theoretical results of this paper are more easily expressed under this model, which provides an upper bound on the achievable performance. However, results in [7] show that proofs written under that model can be extended to more realistic models (one-port communication and atomic computation).

2.2 Energy Model

Among the main system-level energy-saving techniques, Dynamic Voltage Scaling (DVS) works on a very simple principle: decrease the supply voltage (and so the clock frequency) to the CPU so as to consume less power. For this reason, DVS is also called *frequency-scaling* or *speed scaling* [8]. We suppose 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 processor P_u running in the ith mode (noted $P_{u,i}$) needs $\omega/s_{u,i}$ time-units to execute one task of $\mathcal{A}$. We suppose that processing speeds are listed in increasing order ($s_{u,1} \leq s_{u,2} \leq \cdots \leq s_{u,m_u}$), and modes are exclusive: one processor can only run in a single mode at any given time.

Rather than assuming a relation 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, 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 $\mathfrak{P}_{u,i}$ the power consumption per time unit of processor $P_{u,i}$. We focus on two power consumption models. Under the **ideal model**, switching among the modes does not cost any penalty, and an idle processor does not consume any power. Consequently, for each processor P_u, the power consumption is super-linear from 0 to the power consumption at frequency $s_{u,1}$. Under the **model with start-up overheads**, once a processor is on, its power consumption is non-null and at least that of its idle frequency, or speed, $s_{u,1}$. If the cost of turning on and off a processor is null, this model is meaningless. This is why we need to add a start-up overhead. Under this more realistic model, power consumption now depends on the duration of the interval during which the processor is turned on (the overhead is only paid 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

Our goal is bi-criteria scheduling: the first objective is to minimize the power consumption, and the second to maximize 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 $P_{u,i}$ executes per 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 P_u can only be at one mode at a time, and $\frac{\rho_{u,i}\ \omega}{s_{u,i}}$ 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 $s_{u,0} = 0$. As 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 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 deal with the ideal model . 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 $P_d = s^\alpha$ [9,10,11]. However, we solve these problems optimally using the sole assumption that the power consumption is super-linear. Furthermore, we 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 power consumption and throughput on a single processor:

Proposition 1. *For any processor P_u, 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 $\frac{s_{u,i_0}}{\omega} < \rho \leq \frac{s_{u,i_0+1}}{\omega}$.

The following result shows how to solve the converse problem, namely maximizing the throughout subject to a prescribed bound on power consumption.

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: $\mathfrak{P}_{u,i_0} < \mathfrak{P} \leq \mathfrak{P}_{u,i_0+1}$.

Due to lack of space, see [7] for the proofs.

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 (2)). 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.

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

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 $\left\{ \dfrac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} \right\}$. 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 as $P_{u_{k_0},i_{k_0}}$;
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* (2)*).*

Due to lack of space, see [7] for the proof.

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 [7].

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:

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

The overhead is paid 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 modes are used per time-unit; see [7] for more details). We can now prove the following dominance property about optimal schedules (see [7] for the proof):

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.*

Unfortunately, Proposition 3 does not help design an optimal algorithm. However, 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 later defined. 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 the selected processors' indexes. The constraint on the energy consumption can be written:

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

$$\text{or,}\qquad \forall\ u,\ \forall\ 1\le i\le m_u,\ \mathfrak{P}_u - \frac{\mathfrak{P}^{(2)}_u}{d} \ge (\omega\rho_u - s_{u,i})\frac{\mathfrak{P}_{u,i+1}-\mathfrak{P}_{u,i}}{s_{u,i+1}-s_{u,i}} + \mathfrak{P}_{u,i}.$$

The linear program is then:

$$\begin{cases}\text{Minimize } \mathfrak{P} = \displaystyle\sum_{u\in\mathcal{U}}\mathfrak{P}_u \text{ subject to}\\ \displaystyle\sum_{u=1}^{p}\rho_u = \rho\\ \forall u,\ \rho_u \le \min\left\{\frac{s_{u,m_u}}{\omega};\frac{b_u}{\delta}\right\}\\ \forall\ u\in\mathcal{U},\ \forall\ 1\le i\le m_u,\ \mathfrak{P}_u - \dfrac{\mathfrak{P}^{(2)}_u}{d} \ge (\omega\rho_u - s_{u,i})\dfrac{\mathfrak{P}_{u,i+1}-\mathfrak{P}_{u,i}}{s_{u,i+1}-s_{u,i}} + \mathfrak{P}_{u,i}\end{cases} \tag{3}$$

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 $\left(\mathfrak{P}'_u = \mathfrak{P}_u - \frac{\mathfrak{P}_u^{(2)}}{d}\right)$, in order to simplify the last constraint (the first two constraints remain unchanged):

$$\begin{cases} \text{MINIMIZE } \mathfrak{P} = \displaystyle\sum_{u=1}^{p} \left(\mathfrak{P}'_u + \frac{\mathfrak{P}_u^{(2)}}{d} \right) \text{ SUBJECT TO} \\ \dots \\ \forall\ u,\ \forall\ 1 \leq i \leq m_u, \mathfrak{P}'_u \geq (\omega\rho_u - s_{u,i}) \dfrac{\mathfrak{P}_{u,i+1} - \mathfrak{P}_{u,i}}{s_{u,i+1} - s_{u,i}} + \mathfrak{P}_{u,i} \end{cases} \tag{4}$$

The inequalities are stronger than previously, so every solution of (4) is a solution of (3). Of course, optimal solutions for (4) are most certainly not optimal for the initial problem (3). However, the larger d, the closer the constraints are from each other. Furthermore, Algorithm 1 builds optimal solutions for (4). So, the expectation is that when d becomes large, solutions built by Algorithm 1 becomes good approximate solutions for (4). Indeed we derive the following result (see [7] for the proof):

Theorem 2. *Algorithm 1 is asymptotically optimal for problem* MinPower-Overhead *(ρ)(see linear program* (3)*).*

5 Related Work

Several papers have been targeting the minimization of power consumption. Most of them suppose they can switch to arbitrary speed values.

- **Unit time tasks.** Bunde in [11] 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. He also proves the NP-completeness of the problem of minimizing the makespan on multi-processors with jobs of different amount of work. Authors in [9] 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 [11] 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 [12], the authors are interested about scheduling task graphs with data dependencies 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 [13], 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 [14] to online problems. In [15], 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.** [16] addresses the problem of periodic independent real-time tasks on one processor, the period being a deadline to all tasks. In this work the energy consumption is related to the *task* that is executed on the processor. A polynomial algorithm is exhibited to find the optimal speed of each task, and it is shown that EDF can be used to obtain a feasible schedule with these optimal speed values.
- **Deadlines.** In [17], 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. [10] improves the best known competitive ratio to minimize the energy while respecting all deadlines. [18] 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. [19] has a similar approach by allowing task rejection, and proves the NP-hardness of the studied problem.
- **Slack sharing.** In [20,21], 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. However, they do not take communications into account.
- **Heterogeneous multiprocessor systems.** Authors in [22] study the problem of scheduling real-time tasks on two heterogeneous processors. They provide a FPTAS to derive a solution very close to the optimal energy consumption with a reasonable complexity. In [23], 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 [24] 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 [25,26].

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. Ge, R., Feng, X., Cameron, K.W.: Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC 2005). IEEE CS, Los Alamitos (2005)
2. Skadron, K., Stan, M.R., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D.: Temperature-aware microarchitecture: Modeling and implementation. ACM Transactions on Architecture and Code Optimization 1(1), 94–125 (2004)
3. Casanova, H., Berman, F.: Parameter Sweeps on the Grid with APST. In: Hey, A., Berman, F., Fox, G. (eds.) Grid Computing: Making The Global Infrastructure a Reality. John Wiley, Chichester (2003)
4. Adler, M., Gong, Y., Rosenberg, A.L.: Optimal sharing of bags of tasks in heterogeneous clusters. In: Proceedings of SPAA, pp. 1–10. ACM Press, New York (2003)
5. Hong, B., Prasanna, V.: Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In: Proceedings of IPDPS. IEEE CS, Los Alamitos (2004)
6. Hong, B., Prasanna, V.K.: Adaptive allocation of independent tasks to maximize throughput. IEEE TPDS 18(10), 1420–1435 (2007)
7. Pineau, J.F.: Communication-aware scheduling on heterogeneous master-worker platforms. PhD thesis, ENS Lyon (2008)

8. Hotta, Y., Sato, M., Kimura, H., Matsuoka, S., Boku, T., Takahashi, D.: Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster. In: Proceedings of IPDPS. IEEE CS, Los Alamitos (2006)
9. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4) (2007)
10. Bansal, N., Kimbrel, T., Pruhs, K.: Dynamic speed scaling to manage energy and temperature. In: Foundations of Computer Science (FoCS), pp. 520–529 (2004)
11. Bunde, D.P.: Power-aware scheduling for makespan and flow. In: Proceedings of SPAA, pp. 190–196. ACM Press, New York (2006)
12. Varatkar, G., Marculescu, R.: Communication-aware task scheduling and voltage selection for total systems energy minimization. In: International Conference on Computer-Aided Design (ICCAD). IEEE CS, Los Alamitos (2003)
13. Ishihara, T., Yasuura, H.: Voltage scheduling problem for dynamically variable voltage processors. In: Proceedings of ISLPED, pp. 197–202. ACM Press, New York (1998)
14. Okuma, T., Ishihara, T., Yasuura, H.: Real-time task scheduling for a variable voltage processor. In: Proceedings of ISSS. IEEE CS, Los Alamitos (1999)
15. Zhang, Y., Hu, X.S., Chen, D.Z.: Energy minimization of real-time tasks on variable voltage processors with transition energy overhead. In: Asia South Pacific Design Automation Conference (ASPDAC), pp. 65–70. ACM Press, New York (2003)
16. Aydin, H., Melhem, R., Mosse, D., Mejia-Alvarez, P.: Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In: Proceedings of EMRTS, pp. 225–232. IEEE CS, Los Alamitos (2001)
17. Quan, G., Hu, X.: Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In: Design Automation Conference, pp. 828–833 (2001)
18. Chan, H.L., Chan, W.T., Lam, T.W., Lee, L.K., Mak, K.S., Wong, P.W.H.: Energy efficient online deadline scheduling. In: Proceedings of SODA, pp. 795–804. SIAM, Philadelphia (2007)
19. Chen, J.J., Kuo, T.W., Yang, C.L., King, K.J.: Energy-efficient real-time task scheduling with task rejection. In: Proceedings of DATE, European Design and Automation Association, pp. 1629–1634 (2007)
20. Zhu, D., Melhem, R., Childers, B.R.: Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE TPDS 14(7), 686–700 (2003)
21. Rusu, C., Melhem, R., Mossé, D.: Multi-version scheduling in rechargeable energy-aware real-time systems. Journal of Embedded Computing 1(2), 271–283 (2005)
22. Chen, J.-J., Thiele, L.: Energy-efficient task partition for periodic real-time tasks on platforms with dual processing elements. In: Proceedings of ICPADS. IEEE CS, Los Alamitos (2008)
23. Huang, T.Y., Tsai, Y.C., Chu, E.H.: A near-optimal solution for the heterogeneous multi-processor single-level voltage setup problem. In: Proceedings of IPDPS (2007)
24. Yu, Y., Prasanna, V.: Power-aware resource allocation for independent tasks in heterogeneous real-time systems. In: Proceedings of ICPADS, pp. 341–348 (2002)
25. Chen, J.J., Kuo, T.W.: Allocation cost minimization for periodic hard real-time tasks in energy-constrained dvs systems. In: International Conference on Computer-Aided Design (ICCAD), pp. 255–260. ACM, New York (2006)
26. Hsu, H.-R., Chen, J.-J., Kuo, T.-W.: Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In: Proceedings of DATE, pp. 1061–1066. European Design and Automation Association (2006)