Scheduling Divisible Workloads on Heterogeneous Platforms under Bounded Multi-Port Model

Olivier Beaumont, Nicolas Bonichon and Lionel Eyraud-Dubois
Université de Bordeaux,
INRIA Bordeaux - Sud-Ouest,
France
{Olivier.Beaumont,Nicolas.Bonichon,Lionel.Eyraud-Dubois}@labri.fr

Abstract

In this paper, we discuss complexity issues for scheduling divisible workloads on heterogeneous systems under the bounded multi-port model. To our best knowledge, this paper is the first attempt to consider divisible load scheduling under a realistic communication model, where the master node can communicate simultaneously to several slaves, provided that bandwidth constraints are not exceeded. In this paper, we concentrate on one round distribution schemes, where a given node starts its processing only once all data has been received. Our main contributions are (i) the proof that processors start working immediately after receiving their work (ii) the study of the optimal schedule in the case of 2 processors and (iii) the proof that scheduling divisible load under the bounded multi-port model is NP-complete. This last result strongly differs from divisible load literature and represents the first NP-completeness result when latencies are not taken into account.

Keywords: *scheduling, divisible tasks, one-round algorithms, bounded multi-port model.*

1 Introduction

Scheduling computational tasks on a given set of processors is a key issue for high-performance computing. In this paper, we restrict our attention to the processing of independent tasks whose size (and number) are a parameter of the scheduling algorithm. This corresponds to the divisible load model which has been widely studied in the last several years, and popularized by the landmark book written by Bharadwaj, Ghose, Mani and Robertazzi [5]. A divisible job is a job that can be arbitrarily split in a linear fashion among any number of processors. This corresponds to a perfectly parallel job: any sub-task can itself be processed in parallel, and on any number of processors. The applications of the divisible load model encompass a large spectrum of scientific problems, including among others Kalman filtering [15], image processing [13], video and multimedia broadcasting [1, 2], database searching [8, 6], and the processing of large distributed files [16] (see [5] for more examples and [4] for a recent survey).

On the practical side, the divisible load model provides a simple yet realistic framework to study the mapping of independent tasks on heterogeneous platforms. The granularity of the tasks can be arbitrarily chosen by the user, thereby providing a lot of flexibility in the implementation tradeoffs. On the theoretical side, the success of the divisible load model is mostly due to its analytical tractability. Optimal algorithms and closed-form formulas exist for the simplest instances of the divisible load problem. This is in sharp contrast with the theory of task graph scheduling, which abounds in NP-completeness theorems [10, 9] and in inapproximability results [7, 3].

In this paper, the target computing platform is a heterogeneous master/worker platform, with p worker processes running on p processors labeled $P_1, P_2, \ldots, P_p$. The master P_0 sends out chunks to workers over a network: we can think of a star-shaped network, with the master in the center.

In this paper, we concentrate on the bounded multi-port model. In this model, the master is associated to an outgoing bandwidth B_0 and each slave has an incoming bandwidth b_i, $1 \leq i \leq p$. The master can serve any number of slaves simultaneously, each using a bandwidth $b'_i \leq b_i$ provided that its outgoing bandwidth is not exceeded, i.e. $\sum_i b'_i \leq B_0$. This corresponds to modern network infrastructure, where each communication is associated to a TCP connection, and QoS mechanism enable a prescribed sharing of the bandwidth [12]. This model has been advocated by Hong et al [11] for independent task distribution on heterogeneous platforms. This model strongly differs from the traditional model in divisible load scheduling, where

connections are made in exclusive mode: the master can communicate with a single worker at any time-step. Even though the latter may be easier to implement, the former enables a much better use of communication resources, that are usually the bottleneck for the master-slave implementation of divisible load processing.

In the literature, several models have been considered. The master processor can distribute the chunks to the workers in a single round, (also called installment in [5]), or send the chunks to the workers in multiple rounds: the communications will be shorter (less latency) and pipelined, and the workers will be able to compute the current chunk while receiving data for the next one. In both cases, a slave processor can start processing tasks only once it has received the whole data. In this paper, we concentrate on the single round case. Indeed, our goal is to study the complexity of divisible load scheduling onto heterogeneous platforms under the bounded multi-port model and to compare them with the results obtained under the single port model. More specifically, the main contribution of this paper is to prove that considering more realistic communication models makes things harder, since we prove that scheduling divisible load under the bounded multi-port model in one round is NP-complete. In the case of multi-rounds, latencies have to be introduced (otherwise the optimal solution consists in an infinite number of 0 length rounds), and it has already been proved that divisible load scheduling is NP-complete when latencies are taken into account (both for single and multiple rounds cases) in [14] and [17]. Clearly, these results extend to the bounded multi-port model. Indeed, in the case where slave incoming bandwidths are larger than the master outgoing bandwidth, i.e. $\forall i,\ \ b_i \geq B_0$, bounded multi-port model and 1-port model are equivalent.

The rest of the paper is organized as follows. We begin with models for computation and communication costs in Section 2. Next, we prove a set of useful lemmas on the shape of the solution under bounded multi-port model in Section 3. The optimal solution in the case of two slave processors is described in Section 4. In section 5, we prove the main results of the paper, stating that scheduling divisible load under bounded multi-port model is NP-complete. Finally, we state some concluding remarks in Section 7.

2 Models

Let us denote by B_0 be the output bandwidth of the master processor and by $\{P_i\}_{i=1..p}$ the set of slave processors. Let b_i be the incoming bandwidth of P_i and w_i the time needed by P_i to compute a unit-task.

Let us denote by $b'_i(t)$ the actual bandwidth used by the communication between the master processor and P_i at time t. Due to bandwidth constraints, note that $\forall t,\ \ b'_i(t) \leq b_i$ (bandwidth limitation for slave P_i, we assume w.l.o.g that $\forall i,\ \ b_i \leq B_0$) and $\forall t,\ \ \sum_i b'_i(t) \leq B_0$ (bandwidth limitation at master node). We will denote by q_i the fractional number of tasks processed by P_i and by t_i the time when processor P_i stops communicating (i.e. $t_i = \max\{t,\ b'_i(t) > 0\}$). Since a processor starts processing only once all data has been received: $t_i + w_i.q_i \leq 1$.

Finally, we consider the ordering of the slave processors $P_{i_1}, \ldots, P_{i_n}$ by increasing values of t_i so that $t_{i_1} \leq \ldots \leq t_{i_n}$.

These notations are depicted in Figure 1, using the normal form proved in Section 3. It is worth noting that in this figure, the overall number of processed tasks is exactly the surface paved by the rectangles corresponding to the communications between the master and the slave processors.

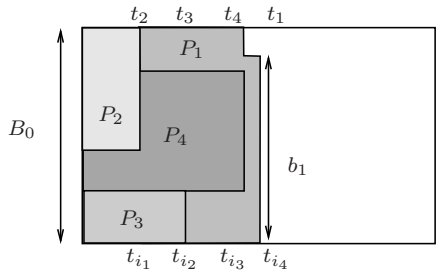

Figure 1. Illustration of the notations.

Throughout this paper, our goal is to maximize the number of tasks that can be processed under bounded multi-port model within 1 time unit. More specifically, we define BOUNDEDMPDIVISIBLE as follows.

Definition 1. BOUNDEDMPDIVISIBLE *Given* B_0, n, b_i, $1 \leq i \leq n$ *and* w_i, $1 \leq i \leq n$ *and a bound* K, *find a bandwidth allocation* $b'_i(t)$ *such that* $\forall t, b'_i(t) \leq b_i$, $\sum_i b'_i(t) \leq B_0$, $t_i + w_i.q_i \leq 1$ *and* $\sum_i q_i \geq K$, *where* $q_i = \int_0^1 \sum_i b'_i(t)dt$ *and* $t_i = \max\{t,\ b'_i(t) > 0\}$.

This problem is trivially equivalent to the problem where the goal is to minimize the necessary time to process a given number of tasks.

An example instance is depicted on Figure 2, together with different solutions. All of these solutions are obtained by a reasonable *priority* scheme, which consists of the following:

- Select a permutation of the processors
- For each processor in this order, serve tasks to this processor, using all the available bandwidth, so that the computation of these tasks finishes exactly at time 1

We can see that none of the intuitive orderings yield an optimal solution. In the 1-port model, serving the processors

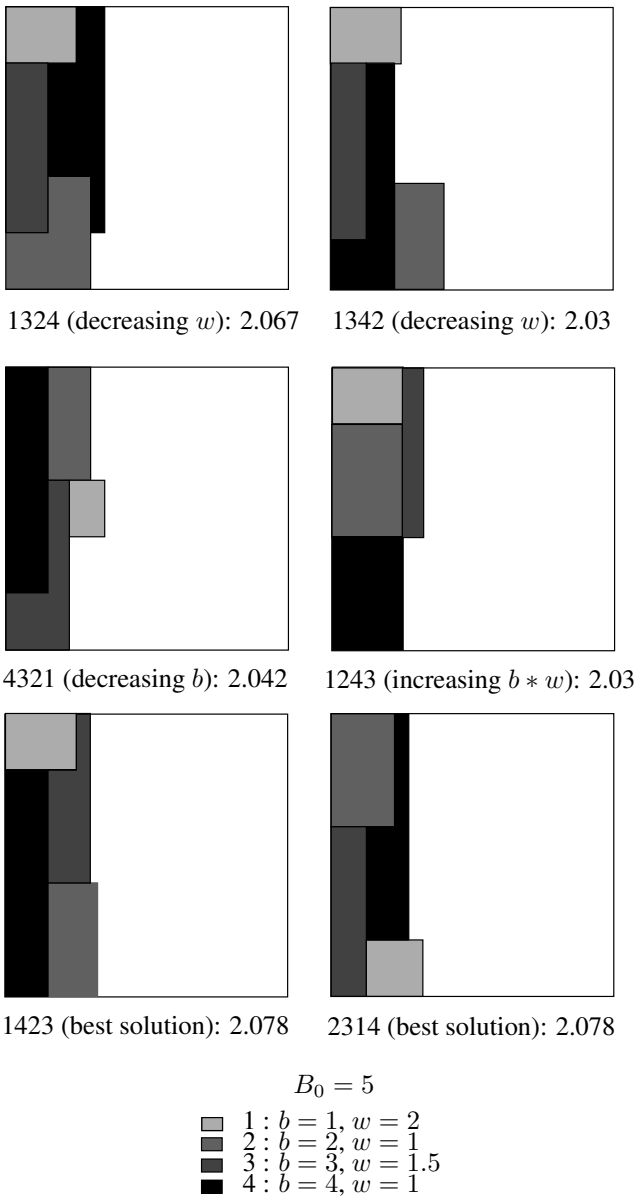

Figure 2. An exemple instance and several solutions.

in the order of their communication speeds is known to be the best possible choice. This instance shows that this property does not hold true under multi-port model. Serving the fastest processing nodes first yields to an even worse schedule; we can even see that the way of breaking the ties matters in a non-trivial way. Indeed, once processors 0 and 2 are served, serving processor 1 first yield a better solution, even though it communicates slowly than processor 3, for the same processing speed.

However, it is worth noting that the worst permutation yields to a total number of processed tasks of 2.03, while the best solution yields 2.078. This small difference indicates that it might be easy to provide good quality schedules in reasonable time. This property will be discussed in more details in Section 6.

3 Useful lemmas

In this Section, we prove a set of lemmas that will be useful to derive optimal solutions. Indeed, given the bounded multi-port model constraints, the bandwidth allocated to a processor may vary continuously over time, thus making the solution very difficult to build and to analyze. We prove (Lemma 1) that all slave processors are involved in the computation and that they start processing tasks immediately after receiving data and until the time bound 1 (Lemma 2). We also prove that the optimal schedule can be put in a special form, where the bandwidth allocated to a given slave only changes when a communication between the master processor and a (possibly different) slave ends (Lemma 3).

Lemma 1. *In an optimal schedule, all processors take part in the computations.*

Proof. Let us suppose that there exist an optimal solution in which processor P_i does not process any task ($t_i = 0$) and let P_{i_p} denote the processor that communicates at last. Since $q_{i_p} > 0$, $w_{i_p} > 0$ and P_{i_p} starts its processing after t_{i_p}, then $t_{i_p} < 1$. After time t_{i_p} the communication medium is free and we can allocate bandwidth b_i to P_i during ϵ_i time units, provided that

$$t_{i_p} + \epsilon_i + \epsilon_i b_i w_i \leq 1,$$

i.e. provided that P_i has enough time to process received tasks before the deadline 1. Clearly, P_i can receive and process as much as $\frac{b_i(1-t_{i_p})}{1+b_i w_i} > 0$ tasks without changing the rest of the schedule, what contradicts the optimality of the initial schedule. Therefore, in an optimal schedule, all processors take part in the computation. □

Lemma 2. *In an optimal schedule, slaves start processing tasks immediately after the end of the communication with the master and stop processing at time 1, i.e. there is no idle*

time between the end of the communication and the deadline.

Proof. Let us show by induction on k that in every optimal schedule, every processor P_{i_j}, $p-k \leq j \leq p$ is processing from $t = t_{i_j}$ to $t = 1$.

Assume that P_{i_p} stops computing at $t = 1 - \epsilon$ (or starts computing at $t = t_{i_p} + \epsilon$). Since P_{i_p} is the last one communicating, we can make it communicate longer without interfering with the other processors. Hence we can build a better schedule. This is in contradiction with the fact that the original schedule is optimal. Therefore, in an optimal solution, the processor P_{i_p} is processing during $]t_{i_p}, 1]$. Assume that every processor P_{i_j}, $p-k \leq j \leq p$ is processing from $t = t_{i_j}$ to $t = 1$. Assume also that $P_{i_{p-k-1}}$ stops computing at $t = 1-\epsilon$ (or starts computing at $t = t_{i_{p-k-1}} + \epsilon$). We can build an equivalent schedule by making $P_{i_{p-k-1}}$ communicates longer, treating some tasks originally treated by (at least) one processor P_{i_j} with $p-k \leq j \leq p$ (see Figure 3). In this new schedule, processor P_{i_j} ends communicating at the same instant t_{i_j} and has fewer tasks to treat.

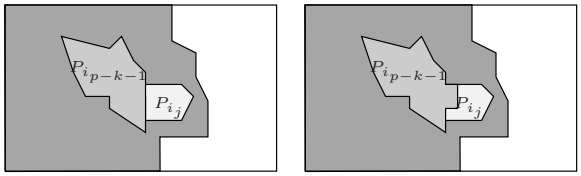

Figure 3. Transformation of a non optimal solution

Hence it does not have enough tasks to compute from $t = t_{i_j}$ to $t = 1$. By the induction hypothesis, the new schedule is not optimal. Since both schedules process the same number of tasks, the original schedule is neither optimal. Therefore, in every optimal solution, every processor P_{i_j} , $p-(k+1) \leq j \leq p$ is processing from $t = t_{i_j}$ to $t = 1$. This concludes the induction. □

Lemma 3. *There exists an optimal schedule such that during each time slot $]t_{i_k}, t_{i_{k+1}}]$ the bandwidth used by any processor is constant.*

Proof. Let us consider a schedule S_1 and a time interval $]t_{i_k}, t_{i_{k+1}}]$ such that $t_{i_k} \neq t_{i_{k+1}}$. We can build a new schedule S_2 such that, during the time interval $]t_{i_k}, t_{i_{k+1}}]$, each processor receives at any time step $t_{i_k} < t \leq t_{i_{k+1}}$ the average value of the bandwidth allocated to it in S_1 during the time interval $]t_{i_k}, t_{i_{k+1}}]$. Clearly, in S_2, during this time interval, each processor receives the same amount of task than in S_1, and the bandwidth constraints are still satisfied, since the average value is smaller that the maximal value. Moreover, in S_1, no processor stops receiving tasks in this time interval, so that the time steps $t_1, t_2, \ldots, t_p$ are the same in S_1 and S_2. Hence the new schedule S_2 computes the same amount of task as S_1. By performing the same transformation to each non-empty interval, we therefore build a new schedule in normal form, that processes exactly the same amount of tasks than S_1. □

Definition 2. *A schedule is said to be in* normal form *if*

1. *all processors are involved in the processing of tasks,*
2. *all slaves start processing tasks immediately after the end of the communication with the master (at time t_i) and stop processing at time 1,*
3. *during each time slot $]t_{i_k}, t_{i_{k+1}}]$ the bandwidth used by any processor is constant.*

Theorem 1. *Let us consider a platform made of a master processor with outgoing bandwidth B_0 and n slave processors $P_1, \ldots, P_n$. Let us denote by t_i the date when slave P_i ends its communication with the master and let us consider the ordering of the slave processors $P_{i_1}, \ldots, P_{i_n}$ by increasing values of t_i (i.e. $t_{i_1} \leq \ldots \leq t_{i_n}$). Then, we can restrict the search of optimal solutions within the valid schedules in normal form.*

Proof. The proof of the theorem comes directly from Lemmas 1, 2 and 3. □

4 Case of Two Slaves

Let us consider the case of 2 slave processors P_1 and P_2. We denote by b_1 and b_2 their respective incoming bandwidth and by B_0 the outgoing bandwidth of the master processor. We also denote by w_i the time it takes to P_i to process a unit-task. Let us consider a schedule in normal form, as depicted in Figure 4.

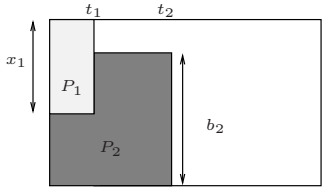

Figure 4. The general pattern with 2 processors.

Theorem 2. *The possible optimal schedules with two slaves are the normal form schedules depicted in Figure 5, according to the respective values of B_0, b_1, b_2, w_1 and w_2.*

1. **If** $b_2 < B_0 \frac{w_1}{w_1+w_2}$ **and** $b_1 < B_0 \frac{w_2}{w_1+w_2}$, *then* P_1 *and* P_2 *both communicate with respective bandwidth* b_1 *and* b_2 *until each of them receives enough tasks to compute until* $t = 1$. *The overall number of processed tasks is given by* $q = \frac{(b_1+b_2)+b_1b_2(w_1+w_2)}{(1+b_1w_1)(1+b_2w_2)}$.

2. **If** $b_2 < B_0 \frac{w_1}{w_1+w_2}$ **and** $b_1 \geq B_0 \frac{w_2}{w_1+w_2}$, *then* P_2 *communicates with bandwidth* b_2 *until it has enough tasks to compute until* $t = 1$. P_1 *communicates with bandwidth* $B_0 - b_2$ *with the master until it has enough tasks to compute until* $t = 1$ *(note that* $t_1 < t_2$*). The overall number of processed tasks is given by* $q = \frac{B_0+(w_1+w_2)(B_0b_2-{b_2}^2)}{(1+(B_0-b_2)w_1)(1+b_2w_2)}$.

3. **If** $b_2 \geq B_0 \frac{w_1}{w_1+w_2}$ **and** $b_1 < B_0 \frac{w_2}{w_1+w_2}$, *then* P_1 *communicates with bandwidth* b_1 *until it has enough tasks to compute until* $t = 1$. P_2 *communicates with bandwidth* $B_0 - b_1$ *with the master until it has enough tasks to compute until* $t = 1$ *(note that* $t_1 > t_2$*). The overall number of processed tasks is given by* $q = \frac{B_0+(w_1+w_2)(B_0b_1-{b_1}^2)}{(1+(B_0-b_1)w_2)(1+b_1w_1)}$.

4. **If** $b_2 \geq B_0 \frac{w_1}{w_1+w_2}$ **and** $b_1 \geq B_0 \frac{w_2}{w_1+w_2}$ *There are two different optimal schedules: In the first one,* P_1 *communicates with bandwidth* b_1 *until it has enough data to compute until* $t = 1$*, and* P_2 *first communicates with bandwidth* $B_0 - b_1$ *until* t_1 *and then with bandwidth* b_2 *until it has enough data to compute until* $t = 1$. *In the second one,* P_2 *communicates with bandwidth* b_2 *and* P_1 *first communicates with bandwidth* $B_0 - b_2$ *until* t_2 *and then with bandwidth* b_2*. In both cases, the overall number of processed tasks is given by* $q = \frac{B_0+w_1b_2b_1+b_2b_1w_2}{(1+b_1w_1)(1+b_2w_2)}$.

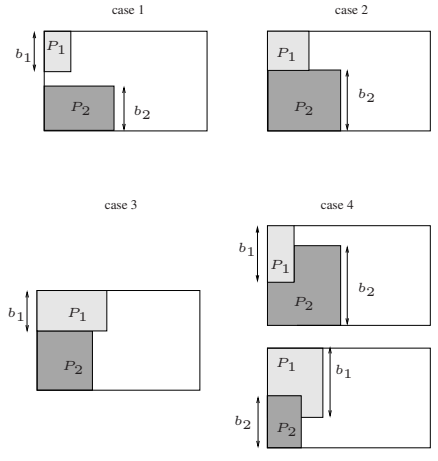

Figure 5. The different optimal configurations with two slaves.

Proof. Case 1: Observe that $b_1 + b_2 < B_0$ and therefore the processors can independently communicate with their maximum possible bandwidth.

Assume now that $b_1 + b_2 \geq B_0$. Let us first also assume that $t_1 \leq t_2$ and let x_1 be the bandwidth used by P_1 (see Figure 4).

To determine t_1 and t_2 we simply write that the number of tasks received between time step 0 and t_i is equal to the number of tasks processed during $[t_i, 1]$. Therefore, we get $x_1t_1 = \frac{1-t_1}{w_1}$ and $(B_0-x_1)t_1+b_2(t_2-t_1) = \frac{1-t_2}{w_2}$. Observe that the bandwidth constraints are satisfied if and only if $(B_0 - b_2) \leq x_1 \leq b_1$. Thus,

$$t_1 = \frac{1}{1 + x_1w_1},$$

$$t_2 = \frac{1 + (b_2 + x_1 - B_0)t_1w_2}{1 + b_2w_2}$$

$$\text{and } t_2 - t_1 = \frac{x_1(w_1 + w_2) - B_0w_2}{(1 + x_1w_1)(1 + b_2w_2)}$$

The fact that $t_1 \leq t_2$ is equivalent to the following condition $x_1 \geq B_0 \frac{w_2}{w_1+w_2}$. Hence, this pattern can occur only if $b_1 \geq B_0 \frac{w_2}{w_1+w_2}$. Let us consider the quantity of processed tasks $q = q_1 + q_2$ as a function of x_1.

$$q(x_1) = B_0t_1 + b_2(t_2 - t_1),$$

$$q(x_1) = \frac{B_0 + w_1b_2x_1 + b_2x_1w_2}{(1 + x_1w_1)(1 + b_2w_2)},$$

$$q'(x_1) = \frac{w_1b_2 + b_2w_2 - w_1B_0}{(1 + x_1w_1)^2(1 + b_2w_2)}$$

and therefore

$$q'(x_1) \geq 0 \Leftrightarrow b_2 \geq B_0 \frac{w_1}{w_1 + w_2}.$$

Thus, if $b_1 \geq B_0 \frac{w_2}{w_1+w_2}$ and $b_2 \geq B_0 \frac{w_1}{w_1+w_2}$ (case 4), one optimal schedule is obtained when $x_1 = b_1$ and the overall quantity of processed tasks is given by

$$q = \frac{B_0 + w_1b_2b_1 + b_2b_1w_2}{(1 + b_1w_1)(1 + b_2w_2)}.$$

Observe that if we switch the roles of P_1 and P_2, then q remains the same, what provides the second optimal schedule.

Finally, if $b_1 \geq B_0 \frac{w_2}{w_1+w_2}$ and $b_2 < B_0 \frac{w_1}{w_1+w_2}$ (case 2), the optimal schedule is obtained when $x_1 = B_0 - b_2$ and the overall number of processed tasks is then given by

$$q = \frac{B_0 + (w_1 + w_2)(B_0b_2 - {b_2}^2)}{(1 + (B_0 - b_2)w_1)(1 + b_2w_2)}.$$

At last, note that case 3 is symmetric with case 2. □

5 NP-Completeness of BOUNDEDMPDIVISIBLE

Theorem 3. BOUNDEDMPDIVISIBLE *is NP-complete.*

Proof. If we restrict the search of the optimal schedule to schedules in normal form, then BOUNDEDMPDIVISIBLE clearly belongs to NP. Let prove that BOUNDEDMPDIVISIBLE is NP-hard with a reduction from 2-PARTITION [10]. Let $I = (a_i)_{i \leq n}$ be an instance of 2-PARTITION, such that $\sum a_i = 2A$. A solution to this instance is a partition of the a_i's into two groups G_1 and G_2 such that $\sum_{i \in G_1} a_i = \sum_{i \in G_2} a_i = A$. We build an instance I' of BOUNDEDMPDIVISIBLE with n processors, with $b_i = a_i$ and $w_i = 1/b_i$ for all i, and $B_0 = A$. We will prove in the following that I has a solution if and only if there is a solution to I' that performs at least $3B_0/4$ tasks.

Given the choice of the w_i's, it is easy to see that any processor that starts receiving tasks at its maximum bandwidth at a given time t will end its communication at time $\frac{t+1}{2}$. If I has a solution, then it is possible to schedule all communications for the first group of processors from 0 to $1/2$, using all the available bandwidth B_0, and the communications for all remaining processors from time $1/2$ to time $3/4$. Since the total bandwidth of the master is kept fully busy from time 0 to time $3/4$, this schedule will perform exactly $3B_0/4$ tasks. Figure 6 (a) shows such a schedule.

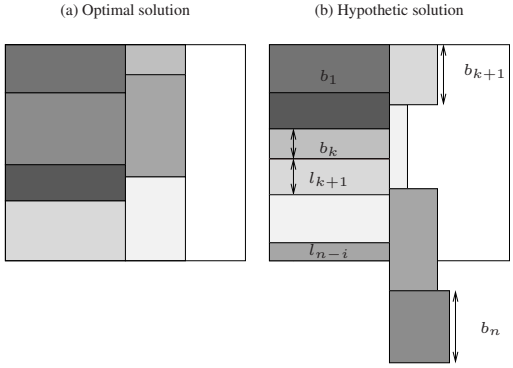

Figure 6. 2 schedules with the same number of processed tasks.

Reciprocally, let us prove that any schedule which does not follow this pattern performs strictly less than $3B_0/4$ tasks. Lemma 2 implies that in an optimal solution, no processor can finish its communication before time $1/2$. For all i, let l_i be the bandwidth allocated to processor i between time 0 and $1/2$. We will assume w.l.o.g. that there exists a index k such that $l_i = b_i$ if and only if $i \leq k$. The first k processors thus end their communication at time $1/2$, and the remaining ones can still communicate. An upper bound on the number q'_i of tasks processor i can receive after time $1/2$ corresponds to allocating its full bandwidth after time $1/2$. In that case, its communication ends at time $t_i = 1/2 + \delta_i$ where

$$\left(\frac{l_i}{2} + b_i\delta_i\right)\frac{1}{b_i} + 1/2 + \delta_i = 1, \text{i.e. } \delta_i = \frac{1}{4}\left(1 - \frac{li}{bi}\right).$$

Thus, $\forall i > k$, $q'_i \leq \delta_i b_i$. The total number of tasks q^* is then upper bounded by

$$q^* = \frac{B_0}{2} + \sum_{i=k+1}^{n} b_i\delta_i$$

since $\sum_{i=1}^{k} b_i + \sum_{i=k+1}^{n} li = B_0$

$$\begin{aligned} q^* &= \frac{B_0}{2} + \frac{1}{4}\left(\sum_{i=1}^{n} b_i - B_0\right) \\ &= \frac{3B_0}{4} \end{aligned}$$

However, it is not possible that all $n-k$ processors get their maximum bandwidth simultaneously. Indeed, this would require a total bandwidth of $\sum_{i=k+1}^{n} b_i$, what is strictly larger than B_0 since the first k processors do not use exactly B_0 in the first interval (see Figure 6 (b)). Hence, at least one of the inequalities $q'_i \leq \delta_i b_i$ is strict, and consequently we have $q < q^*$. □

6 Remarks and Open Problems

There are still many open questions to be solved in the context of divisible load under the bounded multi-port model. Even if the main result of this paper is negative (i.e. scheduling divisible load is NP-Complete), we strongly believe that this model deserves more attention. First of all, it is more realistic than the usual 1-port model. Indeed, it is unrealistic to consider that a server with high outgoing bandwidth will be kept inactive when communicating with a DSL node with small incoming bandwidth. In order to use communication resources efficiently, communications must take place in parallel. Second, this model may also be of interest in the case of dynamic platforms. Indeed, even if the optimal schedule is easier to derive under the 1-port model, since it is enough to sort the nodes by non decreasing communication times, making wrong decisions (based on wrong estimates of the communication cost) may lead to arbitrarily poor results. On the other hand, as noted in Section 2, the overall number of processed tasks only slightly depend on communication ordering under the bounded multiport model, at least for all the examples we have studied. This leaves the door open for the design of clever approximation algorithms that may be able to provide efficient solutions even if resource performances change over time.

7 Conclusion

In this paper, we consider complexity issues for scheduling divisible workloads on heterogeneous systems under

the bounded multi-port model. We prove that finding the optimal schedule for 1-round distribution schemes is NP-Complete, what strongly differs from divisible load literature and represents the first NP-completeness result when latencies are not taken into account. We also derive a general normal form for optimal schedules and extensively study the case of 2 slave processors, what leaves the door open for clever approximation algorithms for divisible loads under the bounded multi-port model.

References

[1] D. Altilar and Y. Paker. An optimal scheduling algorithm for parallel video processing. In *IEEE Int. Conference on Multimedia Computing and Systems*. IEEE Computer Society Press, 1998.

[2] D. Altilar and Y. Paker. Optimal scheduling algorithms for communication constrained parallel processing. In *Euro-Par 2002*, LNCS 2400, pages 197–206. Springer Verlag, 2002.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, Berlin, Germany, 1999.

[4] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang. Scheduling divisible loads on star and tree networks: results and open problems. *IEEE Transactions on Parallel and Distributed Systems*, 16(3):207–218, 2005.

[5] V. Bharadwaj, D. Ghose, V. Mani, and T. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, 1996.

[6] J. Blazewicz, M. Drozdowski, and M. Markiewicz. Divisible task scheduling - concept and verification. *Parallel Computing*, 25:87–98, 1999.

[7] P. Chrétienne, E. G. C. Jr., J. K. Lenstra, and Z. Liu, editors. *Scheduling Theory and its Applications*. John Wiley and Sons, 1995.

[8] M. Drozdowski. *Selected problems of scheduling tasks in multiprocessor computing systems*. PhD thesis, Instytut Informatyki Politechnika Poznanska, Poznan, 1997.

[9] H. El-Rewini, T. G. Lewis, and H. H. Ali. *Task scheduling in parallel and distributed systems*. Prentice Hall, 1994.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1991.

[11] B. Hong and V. Prasanna. Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. *International Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004.

[12] B. Hubert et al. Linux Advanced Routing & Traffic Control. *http://lartc.org/lartc.pdf*, 2002.

[13] C. Lee and M. Hamdi. Parallel image processing applications on a network of workstations. *Parallel Computing*, 21:137–160, 1995.

[14] A. Legrand, Y. Yang, and H. Casanova. Np-completeness of the divisible load scheduling problem on heterogeneous star platforms with affine costs. *Research Report CS2005-0818, GRAIL Project, University of California at San Diego, march*, 2005.

[15] J. Sohn, T. Robertazzi, and S. Luryi. Optimizing computing costs using divisible load analysis. *IEEE Transactions on parallel and distributed systems*, 9(3):225–234, Mar. 1998.

[16] R. Wang, A. Krishnamurthy, R. Martin, T. Anderson, and D. Culler. Modeling communication pipeline latency. In *Measurement and Modeling of Computer Systems (SIGMETRICS'98)*, pages 22–32. ACM Press, 1998.

[17] Y. Yang, H. Casanova, M. Drozdowski, M. Lawenda, and A. Legrand. On the complexity of multi-round divisible load scheduling. Research report RR-6096, INRIA, 2007.

8 Biographies

Olivier Beaumont received his PhD degree from the University of Rennes in 1999. From 1999 to 2001, he was assistant professor at Ecole Normale Supérieure de Lyon. Then, he was appointed at ENSEIRB in Bordeaux. In 2004, he defended his "habilitation à diriger les recherches". His research interests focus on the design of parallel and distributed algorithms, overlay networks on large scale heterogeneous platforms and combinatorial optimization. **Nicolas**

Bonichon received his PhD degree from the Université Bordeaux 1 in 2002. He has been holding a position as assistant professor in Université Bordeaux 1 since 2004. His research interests include distributed algorithms, compact data structure, graph drawing and enumerative combinatorics. **Lionel**

Eyraud-Dubois received his PhD degree from the University Joseph Fourier of Grenoble in 2006. He joined the INRIA as a permanent researcher in 2007. His research interests include scheduling, combinatorics and distributed computing.