An Adaptive Scheduling Method for Grid Computing

Salah-Salim Boutammine, Daniel Millot, and Christian Parrot

GET / INT
Département Informatique
91011 Évry, France
{Salah-Salim.Boutammine, Daniel.Millot, Christian.Parrot}@int-evry.fr

Abstract. This paper presents an adaptive scheduling method, which can be used for parallel applications whose total workload is unknown a priori. This method can deal with the unpredictable execution conditions commonly encountered on grids. To address this scheduling problem, parameters which quantify the dynamic nature of the execution conditions had to be defined. The proposed scheduling method is based on an on-line algorithm so as to be adaptable to the varying execution conditions, but avoids the idle periods inherent to this on-line algorithm.

Keywords: scheduling, parallel application, grid, master-worker, on-line, multi-round, heterogeneity, dynamicity.

1 Introduction

In this paper, we present an adaptive method for scheduling parallel applications, that can be used in the dynamic context of grids and when some of the information traditionally used by scheduling algorithms is lacking. This method is based on an on-line algorithm from Drozdowski [1]. We assume that a set of grid resources has been identified and tackle the problem of distributing optimally the tasks of a parallel application on this set of resources, so that the application terminates as soon as possible. Precisely, we consider applications that process a finite –but a priori unknown– amount of data independently. The total workload of the application is supposed arbitrarily divisible in any number of chunks where each chunk consists of some amount of data. The same computation is performed on each chunk, producing its own result without any communication. Such applications are suitable for the master-worker programming model, with the master distributing chunks to the workers, then collecting the corresponding results from them. Clearly, for such a parallelization to be useful, the processing cost for a chunk by a worker must dominate the corresponding communication costs between master and worker in a certain sense that will be stated later on, when appropriate notations have been introduced (see inequality (2)).

It has to be noted that although we consider so called divisible load, the DLT (Divisible Load Theory [2, 3, 4, 5]) cannot be straightforwardly applied in our case, as we suppose that the total workload of the application is not known a priori. For this reason, we shall be bound to use on-line algorithms to address our scheduling problem.

We adopt a one-port communication model [6] without contention, which means that for a fixed node neither two emissions nor two receptions can overlap each other, whereas one emission can overlap one reception, and computation can overlap communication. Furthermore, in the context of grid computing, computation and communication latencies must both be considered.

This paper is organized as follows. Section 2 defines precisely the scheduling problem we consider. Section 3 describes the on-line algorithm which our scheduling method is based on and introduces some notations. Section 4 presents the new method itself. It first gives an overview of the approach then successively states the conditions for the method to succeed, details the various computations of the proposed scheduling algorithm and finally compares it with the initial on-line algorithm. Section 5 concludes the paper and outlines future work.

2 The Scheduling Problem

We consider a master-worker model for which the data to be processed are continuously received by the master in an input buffer until the final item is obtained. It is only when the master acquires this last item that the total workload of the application happens to be known. We want to minimize the makespan of the application on a set of grid resources. As this problem is NP-complete when latencies are considered [7], it can only be heuristically dealt with.

Execution parameters on a grid, such as available computing power or network bandwidth, vary both in space (heterogeneity) and time (dynamicity). We assume that we know all past values of these parameters and are unaware of the future ones. Because of the one-port communication model, the workers cannot start their work simultaneously: the master has to finish the emission of some chunk to one worker before being able to begin to send a chunk to another one.

In this paper, we do not consider the fundamental problem of choosing the nodes to be used and the order in which they are served. To terminate the execution of the application as soon as possible, the computation should start as soon as possible on all the selected worker nodes, which should then be sent small initial amounts of work in order to quickly start their computation.

When each worker has received a first chunk, the execution enters the steady-state phase [8]. The main characteristic of this phase is that the total workload is still unknown. If the choice of the computing resources is optimal (i.e. optimal nodes are chosen in optimal proportion), then keeping the selected nodes active minimizes the makespan. When the master gets the final data item to be processed, the steady-state phase ends and the clean-up phase begins.

From this time instant, the problem of scheduling the remaining load is suitable for DLT, as now the total workload is known: namely the amount of data still present in the master input buffer. So, according to the optimality principle, we can try and minimize the makespan by synchronizing the termination of the computation of all the workers. This, being possible only if the master does not overload any worker too much during the steady-state phase, which would cause a discrepancy too large for late workers to catch up, thus preventing a synchonous termination of all workers. In the following, we focus on scheduling during the steady-state phase.

3 Drozdowski's On-Line Scheduling Method

Our scheduling method is based the On-Line method presented by Drozdowski in [1], denoted "OL" thereafter. OL proceeds incrementally, computing the size $\alpha_{i,j}$ of the chunk to be sent to a worker N_i for each new round j, in order to try and maintain a constant duration τ for the different rounds and thus avoid contention at the master.

OL determines $\alpha_{i,j}$ so as to make the distribution asymptotically periodic with period τ, an arbitrarily fixed value, for all the workers. For worker N_i, let $\sigma_{i,j-1}$ be the elapsed time between the begining of the emission of the chunk of its $(j-1)^{th}$ round and the end of the reception of the result corresponding to this chunk. OL determines the value of $\alpha_{i,j}$ as follows:

$$\alpha_{i,j} = \alpha_{i,j-1} \cdot \frac{\tau}{\sigma_{i,j-1}}. \tag{1}$$

That is it allocates comparatively bigger (resp. smaller) chunks to workers with higher (resp. lower) performance. Hence, this method can take the heterogeneous nature of computing and communication resources into account, without explicit knowledge of execution parameters (as equality (1) shows); as Drozdowski states, "the application itself is a good benchmark" [1] (actually the best one).

Lemma 6.1 in [1] shows that, in a static context, with affine cost models for communication, the way $\alpha_{i,j}$ is computed using equation (1) ensures the convergence of $\sigma_{i,j}$ to τ when j increases indefinitely.

Being an estimation of the asymptotic period used for task distribution, τ is also an upper-bound on the discrepancy between workers. Being able to control this bound makes it possible to minimize the makespan during the clean-up phase.

The following notations are used throughout the rest of the paper:

- N number of workers,
- γ_i start-up time for a computation by worker N_i,
- $w_{i,j}$ computation cost for a chunk of size 1 of the j^{th} round by worker N_i,
- β_i (resp. β'_i) start-up time for a communication from the master to N_i (resp. from N_i to the master),
- $c_{i,j}$ (resp.$c'_{i,j}$) transfer cost for a data (resp. result) chunk of size 1 of the j^{th} round from the master to worker N_i (resp. from N_i to the master).

It should be noted that, unlike previous work [1, 9], this paper introduces computation start-up times in order to be more realistic when considering grids. As suggested in section 2, the values of the execution parameters of any worker N_i — here $w_{i,j}$, $c_{i,j}$ and $c'_{i,j}$ — depend on the round. We assume that costs are roundwise affine in the size of chunks. Hence, for a chunk of strictly positive size α (i.e. $\alpha \in \mathbb{R}^{+*}$) of the j^{th} round, we define the cost of:

- sending the chunk to worker $\mathrm{N_i}$ $\qquad \alpha \cdot c_{i,j} + \beta_i,$
- processing the chunk on worker $\mathrm{N_i}$ $\qquad \alpha \cdot w_{i,j} + \gamma_i,$
- receiving the corresponding result from worker $\mathrm{N_i}$ $\qquad \alpha \cdot c'_{i,j} + \beta'_i.$

We indicated in section 1 that the processing cost for a chunk should dominate its communication costs in a certain sense. We choose to formulate this assumption as:

$$\forall \alpha \in \mathbb{R}^{+*},\ \gamma_i + \alpha \cdot \min_{j \in \mathbb{N}^*} w_{i,j} \geq \left(\alpha \cdot \max_{j \in \mathbb{N}^*} c_{i,j} + \beta_i \right) + \left(\alpha \cdot \max_{j \in \mathbb{N}^*} c'_{i,j} + \beta'_i \right) \quad (2)$$

$for\ i = 1, N.$

Equation (2) ensures that sending chunks of any size α to a worker N_i and receiving the corresponding results cost less than processing these chunks.

The problem with OL is that computation never overlaps communication in any worker node, as the emission of the chunk of the next round is at best triggered by the return of the result of the previous one, with no possible anticipation.

4 The OLMR Method

4.1 Overview of the Method

Our method is based on OL, but avoids idle time with respect to computing. When the total load is important compared to the available bandwidth between master and workers, the workload should be delivered in multiple rounds [10, 11, 12]. Therefore we will have each worker receive its share of the load through multiple rounds, hence the name On-Line Multi-Round method [9], denoted "OLMR" thereafter. OLMR divides the chunk sent to N_i for each round j into two subchunks "I" and "II" of respective sizes $\overline{\alpha}_{i,j}$ and $\alpha_{i,j} - \overline{\alpha}_{i,j}$. Dividing the chunks in two parts is enough in order to apply the principle, and the division allows the computation to overlap the communications as can be seen in figure Fig.1. In order to compute $\alpha_{i,j}$, we use a value of $\sigma_{i,j-1}$ derived from the measurement of the elapsed time (including both communications and computation) for subchunk I of the previous round: $\overline{\sigma}_{i,j-1}$. We will show that, thanks to this anticipation (compared to OL) in the computation of $\alpha_{i,j}$, we can avoid the inter-round starvation.

Figure Fig.2 gives the OLMR scheduling algorithm. The OLMR scheduler computes $\alpha_{i,j}$ in the same way as the OL scheduler does, and the values of $\sigma_{i,j-1}$ and $\overline{\alpha}_{i,j}$ as detailed later in the next subsections.

Unfortunately, while attempting to deal with the inter-round starvations inherent to OL, there is a risk of creating intra-round starvation between subchunks I and II (see on figure Fig.3 the idle period Δ). We explain below how to prevent both risks.

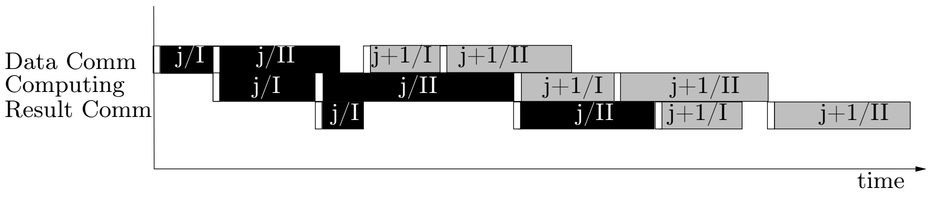

Fig. 1. Overlapping between communication and computation with OLMR

```
while (the last data item has not been acquired) do
    if (Reception from $N_i$ of the result of subchunk $I$ of its $(j-1)^{th}$ round) then
        • Get $\overline{\sigma}_{i,j-1}$, $\omega_{i,j-1}$, $c'_{i,j-1}$ (and $\gamma_i$ for the first result from $N_i$)
        • Compute $\sigma_{i,j-1}$ .............................................. (cf. (8))
        • Compute $\alpha_{i,j}$ ................................................. (cf. (1))
        • Compute $\overline{\alpha}_{i,j}$ ................................................. (cf. (7))
        • Send a subchunk of size $\overline{\alpha}_{i,j}$ to $N_i$ as subchunk $I$ of its $j^{th}$ round
        • Send a subchunk of size $(\alpha_{i,j} - \overline{\alpha}_{i,j})$ to $N_i$ as subchunk $II$ of its $j^{th}$ round
    end if
end while
```

Fig. 2. OLMR scheduler

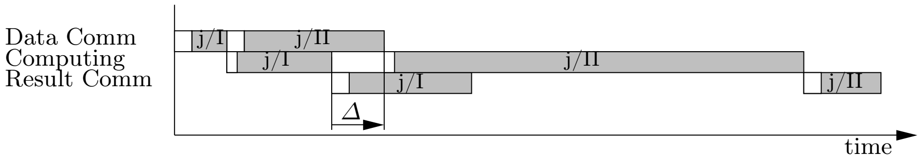

Fig. 3. Example of intra-round starvation with OLMR

As we assume that (2) holds, intra-round starvation can be avoided if $\overline{\alpha}_{i,j}$ is large enough for the processing of subchunk I to overlap the sending of subchunk II of size $\alpha_{i,j} - \overline{\alpha}_{i,j}$. There is no intra-round starvation if and only if

$$\overline{\alpha}_{i,j} \geq \frac{\beta_i - \gamma_i + \alpha_{i,j} \cdot c_{i,j}}{w_{i,j} + c_{i,j}}. \tag{3}$$

Inter-round starvation between the j^{th} and $(j+1)^{th}$ rounds of N_i could occur if subchunk I happens to be too large compared to subchunk II (see figure FIG.4). Let $\nu_{i,j}$ be some real number dominating $\overline{\alpha}_{i,j+1}$: $\nu_{i,j} \geq \overline{\alpha}_{i,j+1}$. Figure FIG.4 shows that, when N_i is given a subchunk I of size $\nu_{i,j}$ for its $(j+1)^{th}$ round, there is no inter-round starvation if and only if

$$\overline{\alpha}_{i,j} \leq \frac{\alpha_{i,j} \cdot w_{i,j} - \nu_{i,j} \cdot c_{i,j+1} + \gamma_i - (\beta'_i + \beta_i)}{c'_{i,j} + w_{i,j}}. \tag{4}$$

If inequality (4) holds, then the necessary constraint $\overline{\alpha}_{i,j} < \alpha_{i,j}$ holds too, as soon as $(\beta'_i + \beta_i) > \gamma_i$.

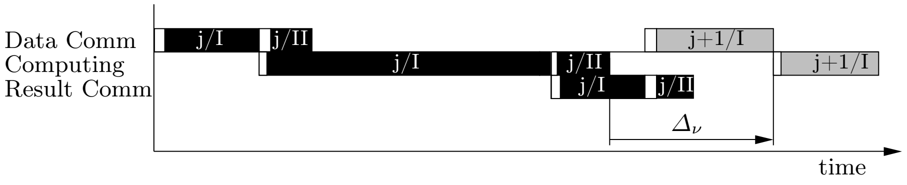

Fig. 4. Example of inter-round starvation with OLMR

Relying on inequations (3) and (4), we can choose $\overline{\alpha}_{i,j}$ so as to avoid idle periods of N_i. Finally, nothing remains but to determine $\sigma_{i,j-1}$ and $\overline{\alpha}_{i,j}$.

4.2 Determining $\overline{\alpha}_{i,j}$

In order to fix the value of $\overline{\alpha}_{i,j}$ according to constraint (4), we need a value for $\nu_{i,j}$. We can decide such a value by extrapolating an upper bound for $\overline{\alpha}_{i,j+1}$ from the values of $\overline{\alpha}_{i,k}$ for the previous rounds, $(\overline{\alpha}_{i,k})_{k=1,j}$. So long as inequalities (3) and (4) hold, an inaccuracy in the value of $\nu_{i,j}$ does not have any dramatic consequence on the course of the method. That is, if inequalities (3) and (4) are compatible, then starvation risks can be avoided.

As the amount of data processed during the steady-state phase is finite, there necessarily exists a real number λ_i ($\lambda_i \geq 1$) for each N_i such that:

$$\overline{\alpha}_{i,j+1} \leq \lambda_i \cdot \overline{\alpha}_{i,j} \qquad\qquad \forall j \in \mathbb{N}^*.$$

λ_i characterizes the amplitude of the fluctuations of $\overline{\alpha}_{i,k}$ between two successive rounds. If λ_i can be estimated (see Remarks 2 and 3 for hints), then we have an upper-bound $\nu_{i,j}$ for $\overline{\alpha}_{i,j+1}$:

$$\nu_{i,j} = \lambda_i \cdot \overline{\alpha}_{i,j}. \tag{5}$$

The following Theorem proposes a way to set the value of $\overline{\alpha}_{i,j}$ so that constraints (3) and (4) are both satisfied (see [9] for similar proof).

Theorem 1. *Given $\alpha_{i,j}$, if γ_i, $w_{i,j}$, $c_{i,j}$, β_i, $c'_{i,j}$ and β'_i satisfy (2) and*

$$(\alpha_{i,j} - (\lambda_i + 1)) \cdot w_{i,j} + (\lambda_i + 1) \cdot \gamma_i \geq (\lambda_i \cdot \alpha_{i,j} + (\lambda_i + 1)) \cdot c_{i,j} + (\lambda_i + 1) \cdot \beta_i \tag{6}$$

for i=1,N.
Then, taking

$$\overline{\alpha}_{i,j} = \frac{\alpha_{i,j}}{\lambda_i + 1}, \tag{7}$$

constraints (3) and (4) are satisfied. Therefore, the workers will compute without any idle period during the steady-state phase.

Remark 1. *Parameters τ and λ_i are characteristic of the evolution of the execution parameters. On the one hand, τ characterizes their speed of evolution. Practically, it is the period that should be used for reconsidering their value. On the other hand, λ_i measures the amplitude of their variations on such a period. The obvious dependence between τ and λ_i can take on the most varied forms. For* *instance, we can have rapid variations (small τ) with little consequence on the scheduling of the application (λ_i close to 1), or on the contrary slow variations (large τ) with important consequences on the scheduling (λ_i far from 1).*

Remark 2. *The knowledge of $\nu_{i,j}$ is implicitly the result of some extrapolation of the values $(\overline{\alpha}_{i,k})_{k=1,j}$ to get an upper-bound of $\overline{\alpha}_{i,j+1}$. If the variations are slight, one can use the quasi-stationary approximation of $\overline{\alpha}_{i,j+1}$ by $\overline{\alpha}_{i,j}$. In this case, we have $\overline{\alpha}_{i,j+1} = \overline{\alpha}_{i,j}$. Then we only have to apply Theorem 1 with $\lambda_i = 1$. More generally, considering a polynomial interpolation of degree (p - 1) for the value of $\overline{\alpha}_{i,j+1}$, we have $\overline{\alpha}_{i,j+1} = p \cdot \overline{\alpha}_{i,j} - \sum_{k=1}^{p-1} \overline{\alpha}_{i,k}$. In this case, it suffices to apply Theorem 1 with $\lambda_i = p$.*

Remark 3. *The satisfaction of the hypotheses of Theorem 1 guarantees the absence of idle time for the workers but requires the knowledge of $(\lambda_i)_{i=1,N}$. Nevertheless, OLMR may still be used when these values (which characterize the dynamicity of execution parameters) are not known. Starting with arbitrary values (e.g. $\lambda_i = 1$ corresponding to a stability assumption) the scheduler could, if necessary, adjust λ_i values at any round according to information provided by the workers. Actually an inappropriate value of λ_i used for some round will lead to an intra- or inter-round starvation observable by the corresponding worker. The scheduler could then adjust this value for the next round, according to the type of starvation observed by the worker.*

Remark 4. *Although different, hypotheses (2) and (6) both make the assumption that processing should dominate communications. Recall that hypothesis (2) ensures an efficient usage of the master-worker paradigm.*

4.3 Determining $\sigma_{i,j-1}$

In order to determine the size of the chunk to be sent for the next round without waiting for the result of the currently processed chunk, it suffices to replace the measured value $\sigma_{i,j-1}$ in expression (1) by some computed value derived from $\overline{\sigma}_{i,j-1}$. But we only know the values of the execution parameters for the data whose result have been received by the master. We choose to get these parameters just after the master has reveived the result for subchunck I of round $j-1$ (see tag "Snapshot" on FIG.5). It is another extrapolation problem. In order to solve it, we assume that the time taken by N_i to process some amount of data during its $(j-1)^{th}$ round is the same for both subchunks I and II.

With the help of figure FIG.5, and omitting the cost of the scheduling algorithm itself, we have

$$\sigma_{i,j-1} = \overline{\sigma}_{i,j-1} + A + B - C,$$
$$\sigma_{i,j-1} = \overline{\sigma}_{i,j-1} + (\alpha_{i,j-1} - \overline{\alpha}_{i,j-1}) \cdot \omega_{i,j-1} + (\alpha_{i,j-1} - 2 \cdot \overline{\alpha}_{i,j-1}) \cdot c'_{i,j-1} + \gamma_i. \quad (8)$$

Remark 5. *The values of $\omega_{i,j-1}$, γ_i and $c'_{i,j-1}$ can be estimated easily by the master with the help of N_i. There is no need to know the value of either the communication start-up times β_i and β'_i or that of $c_{i,j-1}$ in order to compute $\sigma_{i,j-1}$ by means of equation (8).*

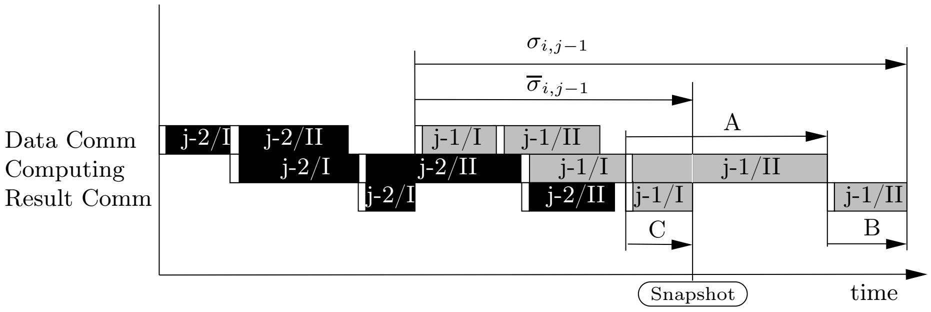

Fig. 5. From the measurement of $\overline{\sigma}_{i,j-1}$ to the computation of $\sigma_{i,j-1}$

4.4 Comparing OL and OLMR

In this section, we compare OL and OLMR, and quantify the benefit of using OLMR compared to OL. We study their behaviour in identical settings: a static context.

Using OLMR requires that the hypotheses of Theorem 1 be satisfied. Lemma 6.1 in [1] sets the context of OL as static. Let us denote c_i, w_i and c'_i the value of $c_{i,j}$, $w_{i,j}$ and $c'_{i,j}$ for any round j as they do not depend on the round, due to the static nature of the execution environment. Under these conditions, both methods send chunks of the same size $\alpha_{i,j}$ to N_i for any round j; for the same value of τ. So processing a workload of size M by both methods requires the same number of rounds δ_M. The gain G_M of OLMR over OL when processing this workload can thus be estimated as :

$$G_M = (\delta_M - 1) \cdot (\beta_i + \beta'_i) - \delta_M \cdot \gamma_i + (M - \overline{\alpha}_{i,1}) \cdot c_i + (M - (\alpha_{i,\delta_M} - \overline{\alpha}_{i,\delta_M})) \cdot c'_i.$$

This gain is the direct consequence of overlapping computation and communications (see figure Fig.6).

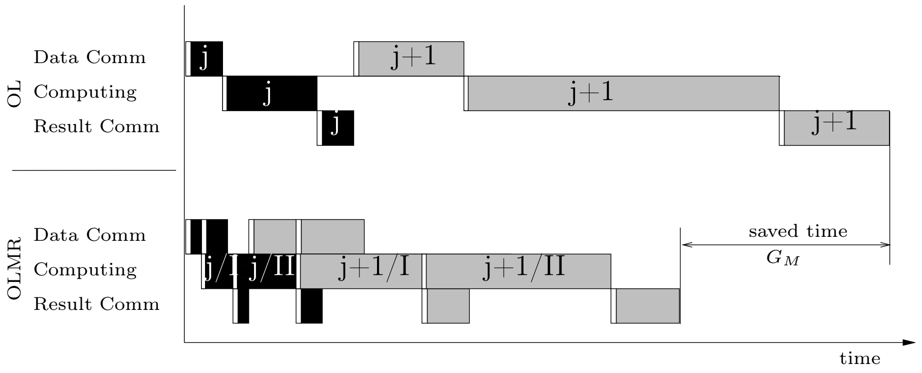

Fig. 6. Comparison between OL and OLMR

Given hypotheses of Theorem 1 and an optimal choice of resources, competitive analysis [13] of OLMR method with an off-line method is not necessary; due to the full use of the computing resources.

5 Conclusion

In this paper, we have considered a scheduling problem that we think is realistic when executing parallel applications on shared resources such as those of a grid. To the best of our knowledge, this scheduling problem has not received much attention up to now. We have presented an adaptive scheduling method, OLMR, to optimize the workload distribution, which can deal with the heterogeneity and dynamicity of the grid if our modelisation hypotheses are realistic; it can also be used when the information that scheduling algorithms traditionally need is lacking. Sufficient conditions have been stated for full usage of the computing resources by means of avoiding idle time.

In order to design the OLMR method, we had to consider the characterization of the dynamicity of the execution conditions. This led us to define $N+1$ parameters: τ and $(\lambda_i)_{i=1,N}$ (see Remark 1). But the improvement made by OLMR to the on-line method presented in [1] has been quantified in a static execution context only.

This approach of scheduling is susceptible to numerous developments, either tending to confirm the results of this paper or aiming at enlarging the potentialities of the OLMR method. First of all, it is useful to check experimentally that, under the hypotheses of our model, the method gives the expected results. For that, we are currently developping simulation programs, using the SimGrid toolkit [14] in order to study OLMR behavior in various conditions and make comparisons with other methods. Furthermore, OLMR could be adapted in different ways: in this paper, τ and λ_i have implicitly been considered as constant throughout all the rounds, but this hypothesis restricts the degree of approximation (order one) of the dynamicity that the scheduler takes into account. From one round to the next, the value of τ could be adapted in order to take further account of the evolution of heterogeneity and dynamicity that would be noticed.

References

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

[2] V. Bharadwaj, D. Ghose, V. Mani, and T.G Robertazzi. Scheduling divisible loads in parallel and distributed systems. *IEEE Computing Society Press*, 1996.

[3] T.G. Robertazzi, J. Sohn, and S. Luryi. Load sharing controller for optimizing monetary cost, March 30 1999. US patent # 5,889,989.

[4] T.G. Robertazzi. Ten reasons to use divisible load theory. *IEEE Computer*, 36(5)(63-68), 2003.

[5] K. van der Raadt and H. Casanova Y. Yang. Practical divisible load scheduling on grid platforms with apst-dv. In *Proceeding of the 19th International Parallel and Distributed Processing Symposium (IPDPS'05)*, volume 1, page 29b, IEEE Computing Society Press, April 2005.

[6] O. Beaumont. Nouvelles méthodes pour l'ordonnancement sur plates-formes hétérogènes. Habilitation à diriger des recherches, Université de Bordeaux 1 (France), December 2004.

[7] A. Legrand, Y. Yang, and H. Casanova. Np-completeness of the divisible load scheduling problem on heterogeneous star platforms with affine costs. Technical Report CS2005-0818, UCSD/CSE, March 2005.

[8] L. Marchal, Y. Yang, H. Casanova, and Y. Robert. Steady-state scheduling of multiple divisible load applications on wide-area distributed computing platforms. *Int. Journal of High Performance Computing Applications*, 2006, to appear.

[9] S. Boutammine, D. Millot, and C. Parrot. A runtime scheduling method for dynamic and heterogeneous platforms. In *Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)*, IEEE Computing Society Press, 2006.

[10] V. Bharadwaj, D. Ghose, and V. Mani. Multi-installment load distribution in tree networks with delays. *IEEE Transactions on Aerospace and Electronic Systems*, 31(2):555–567, 1995.

[11] Y. Yang and H. Casanova. *UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads.* IEEE Computing Society Press, April 2003.

[12] O. Beaumont, A. Legrand, and Y. Robert. Optimal algorithms for scheduling divisible workloads on heterogeneous systems. Technical Report 4595, INRIA, Le Chesnay(France), October 2002.

[13] K. Pruhs, J. Sgall, and E. Torng. *Online scheduling, Handbook of Scheduling: Algorithms, Models, and Performance Analysis.* J. Leung, ed., CRC Press, 2004.

[14] H. Casanova, A. Legrand, and L. Marchal. Scheduling distributed applications: the simgrid simulation framework. In *Proceedings of the 3th International Symposium on Cluster Computing and the Grid (CCGrid03)*, IEEE Computing Society Press, 2003.