First experimental assessments of the adaptivity of the scheduling with AS4DR

Daniel Millot, Christian Parrot

First experimental assessments of the adaptivity of the scheduling with AS4DR

Daniel Millot and Christian Parrot
Telecom sudParis
Institut Mines-Telecom
France
{*Daniel.Millot, Christian.Parrot*}*@mines-telecom.fr*

***Abstract*—The AS4DR (Adaptive Scheduling for Distributed Resources) scheduling method experimented in this paper aims at maximizing the CPU use efficiency when executing divisible load applications on heterogeneous distributed memory platforms. AS4DR adapts the scheduling to: the unawareness of the total workload, both the unspecification and the variation over time of the execution parameters (available communication speed, available computing speed, etc.). This paper presents the first experimental assessments of the adaptivity of the scheduling with this method.**

***Keywords*-parallel application; multi-round divisible load scheduling; heterogeneous platform; adaptive scheduling; unspecified distributed memory platform;**

I. Introduction

This paper addresses the problem of maximizing the CPU utilization with useful work when scheduling a divisible load over a set of heterogeneous distributed resources, according to a master-worker model.

On the one hand, we suppose that the master receives a continuous input stream of data to be processed by the workers (such as some video stream, for instance). So the size of the total workload happens to be known only when the last item is acquired by the master, and as it is unknown when scheduling starts, the master must proceed to an iterated distribution as the workload flows in. Hence the resulting algorithm is necessarily multi-round, where we call "round" a sequence of consecutive actions leading the master to feed all the workers with chunks once and collect the corresponding results from the workers.

On the other hand, we suppose that the execution platform is a heterogeneous distributed memory platform whose communication and computation resources have inaccurately specified characterics: available communication speeds, available computation speeds, latencies, etc., liable to vary over time and called execution parameters in the sequel.

In this paper, we assume that all the available workers are used, in a predefined order. From now on, for a given scheduling, we call CPU-efficiency the ratio of the time spent in useful computation over the corresponding elapsed time. Our goal is to maximize the CPU-efficiency. We suppose that the duration of communicating or processing a chunk is affine according to the chunk size, and we assume that computation can overlap communication. Besides, we consider a 1-port bi-directional communication model, which allows a communication from master to worker to overlap a communication from worker to master; a risk of contention may then appear when workers compete to access the master.

The next section is a reminder of the operating principle of the AS4DR scheduler. Section III gives hints on related works whereas section IV presents experimental results. Finally we conclude.

II. Presentation of the AS4DR method

A. Operating principle

The ultimate goal of the AS4DR method is to automatically adapt the scheduling of a divisible load application to: the heterogeneity of the workers, the unawareness of the total workload, both the unspecification of the execution parameters and their evolution over time. Let $\alpha_{\mathrm{w,i}}$ be the size of the chunk sent to a worker w for round i. Let τ and $\sigma_{\mathrm{w,i}}$ be respectively the wanted and the estimated time durations between the start of the sending of a chunk of size $\alpha_{\mathrm{w,i}}$ and the end of the reception of the corresponding result by the master. The basic idea of the AS4DR multi-round method is to adapt $\alpha_{\mathrm{w,i}}$ according to:

$$\alpha_{\mathrm{w,i}} := \alpha_{\mathrm{w,i-1}} \frac{\tau}{\sigma_{\mathrm{w,i-1}}} \qquad \text{for i} > 1. \tag{1}$$

A special feature of AS4DR is that it splits each chunk it has to deliver to a worker for a round into two subchunks that it delivers in a row to the worker. So, sending subchunks of arbitrarily chosen sizes $\dot{\alpha}_{\mathrm{w},1}$ and $\ddot{\alpha}_{\mathrm{w},1}$ to each worker w for the first round, the AS4DR scheduler then sends to worker w, for each round i, two subchunks $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ of respective sizes $\dot{\alpha}_{\mathrm{w,i}}$ and $\ddot{\alpha}_{\mathrm{w,i}}$, such that

$$\dot{\alpha}_{\mathrm{w,i}} + \ddot{\alpha}_{\mathrm{w,i}} = \alpha_{\mathrm{w,i}}. \tag{2}$$

Dividing the chunks in two parts allows the computation to overlap the communications between a worker and the master as can be seen in Figure 3. Let us suppose that the

ratio between $\dot{\alpha}_{\mathrm{w,i}}$ and $\alpha_{\mathrm{w,i}}$ is constant, and let us denote θ_{w} this ratio:

$$\theta_{\mathrm{w}} \equiv \frac{\dot{\alpha}_{\mathrm{w,i}}}{\alpha_{\mathrm{w,i}}}. \tag{3}$$

Figures 1 and 2 give the AS4DR scheduling algorithm for a M workers platform, and Figure 3 shows how computation overlaps communication with this method. As can be seen

- With CIP set τ and $(\alpha_{\mathrm{w},1}, \theta_{\mathrm{w},1})_{0 \leq \mathrm{w} \leq \mathrm{M}-1}$ ..... (Figure 6)
- Set $(\dot{\alpha}_{\mathrm{w},1}, \ddot{\alpha}_{\mathrm{w},1})_{0 \leq \mathrm{w} \leq \mathrm{M}-1}$ ..... (3), (2)
- After an appropriate delay, post $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ data to each worker

while (the last data item has not been acquired) **do**
- Get a $\dot{\mathrm{s}}$ result from some worker w
- Compute the size of the next $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ for worker w ..... (4), (1), (3), (2)
- Post $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ data to worker w
- Get previous $\ddot{\mathrm{s}}$ result from worker w

end while

Figure 1. AS4DR scheduling: master

while (the last subchunk has not been posted) **do**
- Get a subchunk data from master
- Process the subchunk data

if ($\dot{\mathrm{s}}$) **then**
- Post $\dot{\mathrm{s}}$ and previous $\ddot{\mathrm{s}}$ results to master

end if

end while

Figure 2. AS4DR scheduling: worker

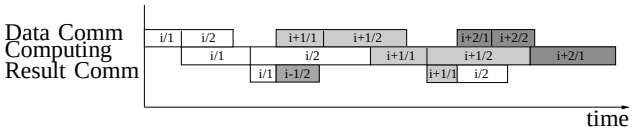

Figure 3. Overlapping between communication and computation

in Figure 3, worker round i for worker w is composed of three phases:

- transmission of the data from master to worker, lasting $\dot{\mathrm{D}}_{\mathrm{w,i}}$ and $\ddot{\mathrm{D}}_{\mathrm{w,i}}$ for subchunks $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ respectively,
- worker computation on the received data, lasting $\dot{\mathrm{C}}_{\mathrm{w,i}}$ and $\ddot{\mathrm{C}}_{\mathrm{w,i}}$ for subchunks $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ respectively,
- transmission of the computation result from worker to master, lasting $\dot{\mathrm{R}}_{\mathrm{w,i}}$ and $\ddot{\mathrm{R}}_{\mathrm{w,i}}$ for subchunks $\dot{\mathrm{s}}$ and $\ddot{\mathrm{s}}$ respectively.

It is worth noticing in Figure 3 that the result corresponding to the $\ddot{\mathrm{s}}$ subchunk of some round is returned to the master just after the result corresponding to the $\dot{\mathrm{s}}$ subchunk of the next round has itself been returned. As the scheduler does not make use of the return of $\ddot{\mathrm{s}}$ results in any way, AS4DR delays this return in order to make all actions have the same period; this helps contentions avoidance. Let us denote F_{w} the available computation speed (relative to the processing of one workload unit) of worker w. Likewise, $\mathrm{B}^{\mathrm{D}}_{\mathrm{w}}$ (resp. $\mathrm{B}^{\mathrm{R}}_{\mathrm{w}}$) is the available communication speed (relative to one workload unit) of the link from the master to worker w (resp. from worker w to the master). Finally $\mathrm{b}^{\mathrm{D}}_{\mathrm{w}}$, $\mathrm{b}^{\mathrm{R}}_{\mathrm{w}}$ and f_{w} are the respective latencies for a transfer of data from the master to worker w, for a transfer of result from worker w to the master and for a computation on worker w. We define

$$\sigma_{\mathrm{w,i}} \equiv \frac{\dot{\mathrm{C}}_{\mathrm{w,i}} - \mathrm{f}_{\mathrm{w}}}{\theta_{\mathrm{w}}} + 2\mathrm{f}_{\mathrm{w}}. \tag{4}$$

When the communications between master and workers are contention-free,

$$\lim_{\mathrm{i} \to +\infty} \sigma_{\mathrm{w,i}} = \tau.$$

B. Prevention of idleness

The AS4DR method could experience either of the workers idlenesses illustrated in figures 4: inter-round idleness and intra-round idleness. Let us respectively define:

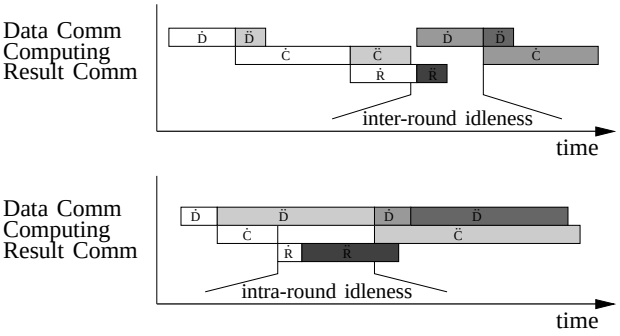

Figure 4. Example of inter-round and intra-round idlenesses

$$\theta^{\min}_{\mathrm{w,i+1}} \equiv \frac{\mathrm{D}_{\mathrm{w,i+1}} - \left(\mathrm{b}^{\mathrm{D}}_{\mathrm{w}} + \mathrm{f}_{\mathrm{w}}\right)}{\mathrm{D}_{\mathrm{w,i+1}} + \mathrm{C}_{\mathrm{w,i+1}} - 2\left(\mathrm{b}^{\mathrm{D}}_{\mathrm{w}} + \mathrm{f}_{\mathrm{w}}\right)}, \tag{5}$$

$$\theta^{\max}_{\mathrm{w,i+1}} \equiv \frac{\mathrm{C}_{\mathrm{w,i}} - \left(\mathrm{b}^{\mathrm{D}}_{\mathrm{w}} + \mathrm{f}_{\mathrm{w}} + \mathrm{b}^{\mathrm{R}}_{\mathrm{w}}\right)}{\mathrm{D}_{\mathrm{w,i+1}} + \mathrm{C}_{\mathrm{w,i}} + \mathrm{R}_{\mathrm{w,i}} - 2\left(\mathrm{b}^{\mathrm{D}}_{\mathrm{w}} + \mathrm{f}_{\mathrm{w}} + \mathrm{b}^{\mathrm{R}}_{\mathrm{w}}\right)} \tag{6}$$

Let us suppose that the communications between master and workers are contention-free. Then the AS4DR method prevents idleness, if and only if, for each worker w

$$\theta^{\max}_{\mathrm{w,i+1}} \geq \theta_{\mathrm{w}} \geq \theta^{\min}_{\mathrm{w,i+1}}, \ \forall \mathrm{i}. \tag{7}$$

C. Prevention of contentions

In order to take advantage of this result, we need to address the problem of contention avoidance. To make the instant each worker accesses to the master far enough from the instants the others access too, time delays d_{w} are

introduced before posting the very first subchunk to each worker w. Figure 5 illustrates the model of asymptotic τ-

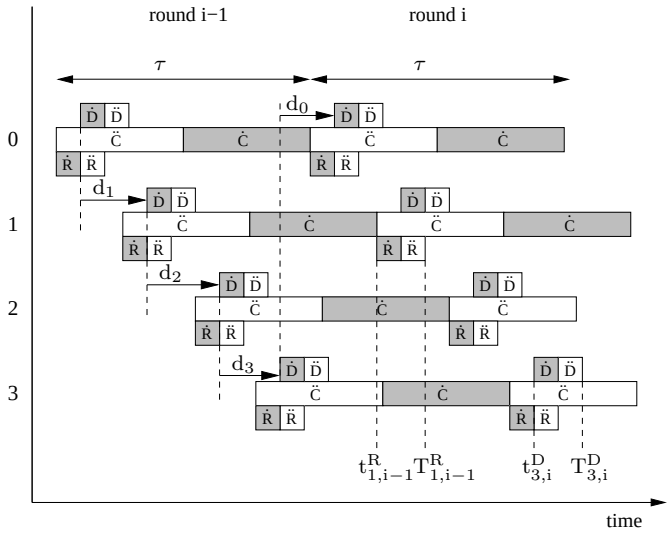

Figure 5. Contention-free asymptotic schedule

periodic (thus round-robin) scheduling we are looking for, in the case of a four workers platform. The delay $\mathrm{d_w}$, to be computed by the CIP algorithm is defined (modulo M) as follows:

$$\mathrm{d_w} \equiv (1+\lambda_\mathrm{w}) \max\left(\overline{\dot{\mathrm{D}}_{\mathrm{w-1}}} + \overline{\ddot{\mathrm{D}}_{\mathrm{w-1}}}, \overline{\dot{\mathrm{R}}_{\mathrm{w-1}}} + \overline{\ddot{\mathrm{R}}_\mathrm{w}}\right); \quad (8)$$

where: λ_w stands for a positive constant factor, $\overline{\mathrm{B_w^D}}$ (resp. $\overline{\mathrm{F_w}}$, $\overline{\mathrm{B_w^R}}$, $\overline{\mathrm{b_w^D}}$, $\overline{\mathrm{f_w}}$ and $\overline{\mathrm{b_w^R}}$) denotes an estimate of $\mathrm{B_w^D}$ (resp. $\mathrm{F_w}$, $\mathrm{B_w^R}$, $\mathrm{b_w^D}$, $\mathrm{f_w}$ and $\mathrm{b_w^R}$) and $\overline{\dot{\mathrm{D}}_\mathrm{w}}$ (resp. $\overline{\ddot{\mathrm{D}}_\mathrm{w}}$, $\overline{\dot{\mathrm{R}}_\mathrm{w}}$ and $\overline{\ddot{\mathrm{R}}_\mathrm{w}}$) is an estimate of $\dot{\mathrm{D}}_{\mathrm{w},1}$ (resp. $\ddot{\mathrm{D}}_{\mathrm{w},1}$, $\dot{\mathrm{R}}_{\mathrm{w},1}$ and $\ddot{\mathrm{R}}_{\mathrm{w},1}$).

$$\overline{\dot{\mathrm{D}}_\mathrm{w}} := \theta_\mathrm{w}\alpha_\mathrm{w}\frac{1}{\overline{\mathrm{B_w^D}}} + \overline{\mathrm{b_w^D}}, \quad \overline{\ddot{\mathrm{D}}_\mathrm{w}} := (1-\theta_\mathrm{w})\alpha_\mathrm{w}\frac{1}{\overline{\mathrm{B_w^D}}} + \overline{\mathrm{b_w^D}}, \quad (9)$$

$$\overline{\dot{\mathrm{R}}_\mathrm{w}} := \theta_\mathrm{w}\alpha_\mathrm{w}\frac{1}{\overline{\mathrm{B_w^R}}} + \overline{\mathrm{b_w^R}}, \quad \overline{\ddot{\mathrm{R}}_\mathrm{w}} := (1-\theta_\mathrm{w})\alpha_\mathrm{w}\frac{1}{\overline{\mathrm{B_w^R}}} + \overline{\mathrm{b_w^R}}. \quad (10)$$

$$\alpha_\mathrm{w} := \frac{\tau - 2\overline{\mathrm{f_w}}}{\frac{1}{\overline{\mathrm{F_w}}}} \quad (11)$$

$$\theta_\mathrm{w} := \phi_\mathrm{w}\theta_\mathrm{w}^{\max} + (1-\phi_\mathrm{w})\,\theta_\mathrm{w}^{\min}, \quad (12)$$

$$\theta_\mathrm{w}^{\min} := \frac{\frac{\alpha_\mathrm{w}}{\overline{\mathrm{B_w^D}}} + \overline{\mathrm{b_w^D}} - \overline{\mathrm{f_w}}}{\alpha_\mathrm{w}\left(\frac{1}{\overline{\mathrm{F_w}}} + \frac{1}{\overline{\mathrm{B_w^D}}}\right)}, \quad \theta_\mathrm{w}^{\max} := \frac{\frac{\alpha_\mathrm{w}}{\overline{\mathrm{F_w}}} + \overline{\mathrm{f_w}} - \overline{\mathrm{b_w^R}} - \overline{\mathrm{b_w^D}}}{\alpha_\mathrm{w}\left(\frac{1}{\overline{\mathrm{F_w}}} + \frac{1}{\overline{\mathrm{B_w^D}}} + \frac{1}{\overline{\mathrm{B_w^R}}}\right)}.$$

$$\phi_\mathrm{w} = 0.5. \quad (13)$$

So, before launching AS4DR, a preliminary step: CIP (for Contentions and Idleness Prevention), determines τ, $(\alpha_{\mathrm{w},1})_{0\le \mathrm{w} \le \mathrm{M}-1}$, $(\theta_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ and $(\mathrm{d_w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ as shown by Figure 6.

- Estimation of $\left(\mathrm{B_w^D},\ \mathrm{F_w},\ \mathrm{B_w^R},\ \mathrm{b_w^D},\ \mathrm{f_w},\ \mathrm{b_w^R}\right)_{0\le \mathrm{w} \le \mathrm{M}-1}$
- Set $(\phi_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (13)
- Set $(\lambda_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (15)
- $\tau := 2 \max\limits_{0\le \mathrm{s} \le \mathrm{M}-1} \overline{\mathrm{f_w}}$

repeat

- $\tau := \tau + \widehat{\tau}$
- Set $(\alpha_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (11)
- Set $(\theta_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (12)
- Set $\left(\overline{\dot{\mathrm{D}}_\mathrm{w}},\ \overline{\ddot{\mathrm{D}}_\mathrm{w}},\ \overline{\dot{\mathrm{R}}_\mathrm{w}},\ \overline{\ddot{\mathrm{R}}_\mathrm{w}}\right)_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (9),(10)
- Set $(\mathrm{d_w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ ..... (8)

until $\left(\tau \ge \sum_{\mathrm{w}=0}^{\mathrm{M}-1} \mathrm{d_w}\right)$

Figure 6. CIP algorithm

Besides, the time intervals $\mathrm{d_w}$ should allow all the workers to be served during the first round, i.e. within a τ period. Thus τ must verify:

$$\tau \ge \sum_{\mathrm{w}=0}^{\mathrm{M}-1} \mathrm{d_w}. \quad (14)$$

So, starting from an initial value of τ, the CIP algorithm enters an iterative process which increments τ with an arbitrarily fixed value $\widehat{\tau}$, then computes $(\alpha_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ and $(\mathrm{d_w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ successively and loops until (14) holds.

Once CIP is processed, proper AS4DR scheduling starts with a first round which sets the initial time-lags between successive round beginnings, according to the values $(\mathrm{d_w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ previously computed by CIP.

Taking into account the previous results about contentions and idleness avoidance, the CIP algorithm can avoid contentions and idleness during the AS4DR scheduling first round. For the next rounds, thanks to assignment (1), the AS4DR method helps maintain the duration of each round close to the reference value τ.

Let us define the value Λ_w for each worker w as

$$\Lambda_\mathrm{w} \equiv \frac{1}{\mathrm{M}}\frac{1}{\overline{\mathrm{F_w}}\max\left(\mathrm{K_w} + \frac{1}{\overline{\mathrm{B_w^D}}},\ \frac{1}{\overline{\mathrm{B_w^R}}}\right)} - 1;$$

$$\text{where } \mathrm{K_w} \equiv \frac{1}{\overline{\mathrm{B_w^R}}}\left(\phi_\mathrm{w}\frac{1}{1+\overline{\mathrm{F_w}}\left(\frac{1}{\overline{\mathrm{B_w^D}}}+\frac{1}{\overline{\mathrm{B_w^R}}}\right)} + (1-\phi_\mathrm{w})\frac{1}{1+\frac{\overline{\mathrm{B_w^D}}}{\overline{\mathrm{F_w}}}}\right).$$

Let us assume that $(\lambda_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ verifies:

$$0 \le \lambda_\mathrm{w} \le \min\left(\Lambda_\mathrm{w}, \Lambda_{\mathrm{w}-1}\right) \quad (\text{modulo M}), \quad (15)$$

then the CIP preliminary step provides a τ value and sets of values $(\theta_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$, $(\alpha_\mathrm{w})_{0\le \mathrm{w} \le \mathrm{M}-1}$ and $(\mathrm{d_w})_{0\le \mathrm{w} \le \mathrm{M}-1}$

that allow the AS4DR scheduling to start with neither contention nor idleness. Let us recall that the value of λ_{w} characterizes the importance of both the unspecification of the execution parameters and the variation of their magnitude over time, for any worker w. For a given set of execution parameters values, increasing the number M of workers will ultimately make the relation (15) false.

III. Related works

Details about related works can be found in [1] and the proof of the results presented in the previous section has been established in [2]. To the best of our knowledge, no scheduler aims at maximizing the efficiency of the use of the CPUs in a similar context.

IV. Experimental assessment of the adaptivity

In order to experimentally assess the ability of the AS4DR method to cope with inaccurately estimated execution parameters and with variation of their value over time, simulations have been conducted, with the SimGrid framework [3]. Let us consider a star-shaped platform (Figure 7 with M=1000) and 10 sets $(\mathrm{S_k})_{0\leq \mathrm{k} \leq 9}$ of values for the execution parameters of the worker nodes, given in Table I. Each of these sets is randomly allocated to 100 workers among the 1000 workers which constitute the platform. In order

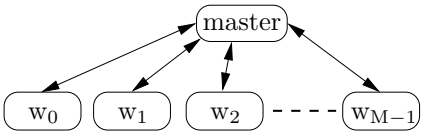

Figure 7. Star-shaped platform

	computation		communication master $\longrightarrow \mathrm{w_i}$		communication master $\longleftarrow \mathrm{w_i}$		number of workers
	speed	latency	speed	latency	speed	latency	
S_0	1.0e+4	1.0e-4	1.0e+8	1.0e-4	1.0e+8	1.0e-4	100
S_1	0.9e+4	1.5e-4	0.9e+8	1.0e-4	0.9e+8	1.0e-4	100
S_2	0.8e+4	1.4e-4	0.8e+8	1.0e-4	0.8e+8	1.0e-4	100
S_3	0.7e+4	1.3e-4	0.7e+8	1.0e-4	0.7e+8	1.0e-4	100
S_4	0.6e+4	1.2e-4	0.6e+8	1.0e-4	0.6e+8	1.0e-4	100
S_5	0.5e+4	0.8e-4	0.5e+8	1.0e-4	0.5e+8	1.0e-4	100
S_6	0.4e+4	0.9e-4	0.4e+8	1.0e-4	0.4e+8	1.0e-4	100
S_7	0.3e+4	0.8e-4	0.3e+8	1.0e-4	0.3e+8	1.0e-4	100
S_8	0.2e+4	0.4e-4	0.2e+8	1.0e-4	0.2e+8	1.0e-4	100
S_9	0.1e+4	0.5e-4	0.1e+8	1.0e-4	0.1e+8	1.0e-4	100

Table I
Reference values of the execution parameters

to assess the relevance of AS4DR adapting the workload (of each worker at each round), we compare this method to a scheduler called "Baseline". The Baseline method is identical to AS4DR except that, on the one hand, the wanted period τ is not computed but just set, and that, on the other hand, the workload is not adapted at each round; in other words, neither CIP algorithm nor assignment (1) are run. With τ and estimates of the execution parameters, it is possible, due to (11), to compute α_{w} an initial workload for each worker w. For our simulations, we set the estimate of each execution parameter for each worker as the time-average of the value of this execution parameter for this worker, during the simulation. In order to be able to compare the two methods efficiently, the value of τ for the Baseline scheduler is set to the one obtained by the CIP method when running AS4DR.

For the comparison of the simulation results, let us define $\mathrm{CPU_{eff}}$ the CPU-efficiency:

$$\mathrm{CPU_{eff}} \equiv 1 - \frac{\text{CPU idleness } + \text{ CPU latencies}}{\text{elapsed time}}.$$

When the execution parameters are exactly known and steady, both schedulers make the workers process data without idleness; except time spent in latencies. So, in the sequel, we will successively compare the schedulers when the parameters are either poorly estimated or time-varying.

A. Poor estimates in a steady context

In order to assess the effect of estimates' inaccuracy of the execution parameters on both schedulings, the effective workload initially allocated to each worker w is set by penalizing the value $(\alpha_{\mathrm{w}})_{\mathrm{ref}}$ computed using assignment (11) and the reference values contained in Table I:

$$\alpha_{\mathrm{w},1} := (1 \pm \iota_{\mathrm{k}})\,(\alpha_{\mathrm{w}})_{\mathrm{ref}}\,; \tag{16}$$

where ι_{k} is a strictly positive real number which values are given in Table II and where the operator $\pm$ means that the operation performed is randomly chosen between either addition or substraction. The values of ι_{k} characterize esti-

ι_0	ι_1	ι_2	ι_3	ι_4	ι_5
0.0	0.18	0.36	0.54	0.72	0.9

Table II
Estimates' inaccuracy parameter

mates' inaccuracy of the execution parameters; inaccuracy is minimum with ι_0, whereas it is maximum with ι_5. It is supposed to be the same for all the workers.

Figure 8 shows the measured CPU-efficiency of the whole platform for each scheduler, as a function of estimates' inaccuracy. For this simulation, which lasted 2000 seconds, the value of τ computed by CIP equals 3 seconds. When the execution parameters are exactly known (ι_0), both schedulers offered the same CPU-efficiency: 99.99%. The computation latencies are responsible for the gap with the theorical maximum CPU-efficiency: 100%. As expected, CPU-efficiency decreases for both methods when estimates' inaccuracy increases. This decrease is considerably faster for the Baseline scheduler than for AS4DR. For instance, when the inaccuracy is maximum (ι_5), the CPU-efficiency

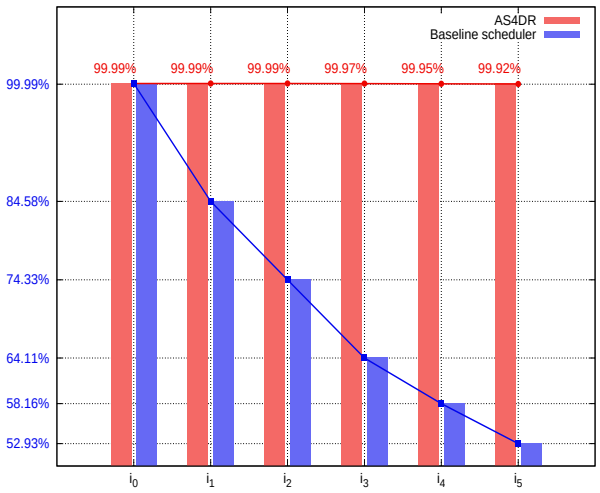

Figure 8. CPU-efficiency of the platform as a function of estimates' inaccuracy

with AS4DR is 2.13 times higher than with the Baseline scheduler.

B. Time-varying context

This subsection assesses to what extent one can put this ability to good use in adapting to the variations of the values of execution parameters over time, given that it has already been proved that the outbreak of these variations will not make the AS4DR scheduling unstable.

In this subsection, the reference values for the execution parameters, which are contained in Table I, are still randomly allocated to the workers at the initial instant. But, from this initial instant these values are likely to vary over time, according to the 10 profiles $(\mathrm{P_k})_{0\leq \mathrm{k}\leq 9}$ of variation shown by Figure 9. Whatever the profile $\mathrm{P_k}$ being allocated to a worker w and whatever an execution parameter, the higher value of $\mathrm{P_k}$ equals the reference value (in Table I) allocated to the worker w, for the execution parameter under consideration; the lower value of $\mathrm{P_k}$ is computed as a perturbation of the reference value:

$$\begin{aligned} \mathrm{B_w^D} &:= (1-\delta_\mathrm{k})\left(\mathrm{B_w^D}\right)_{\mathrm{ref}}, \mathrm{B_w^R} := (1-\delta_\mathrm{k})\left(\mathrm{B_w^R}\right)_{\mathrm{ref}}, \\ \mathrm{F_w} &:= (1-\delta_\mathrm{k})\left(\mathrm{F_w}\right)_{\mathrm{ref}}; \end{aligned}$$

where δ_k is a strictly positive number, the values of which are given in Table III. In this context, the coefficient δ_k characterizes the variations of the execution parameters and is called "dynamicity" in the sequel; with δ_4 the amplitude of the variation of the execution parameters is maximum, whereas it is minimum with δ_0. For each simulation the dynamicity is the same for all the workers. The same profile is used to simultaneously perturb the speeds of computation and communication. The profiles $(\mathrm{P_k})_{0\leq \mathrm{k}\leq 9}$ are randomly allocated to the 1000 workers as well. Figure 10 shows the

δ_0	δ_1	δ_2	δ_3	δ_4
0.0	0.2	0.4	0.6	0.8

Table III
DYNAMICITY PARAMETER

measured CPU-efficiency of the whole platform for each scheduler, as a function of dynamicity. For this simulation,

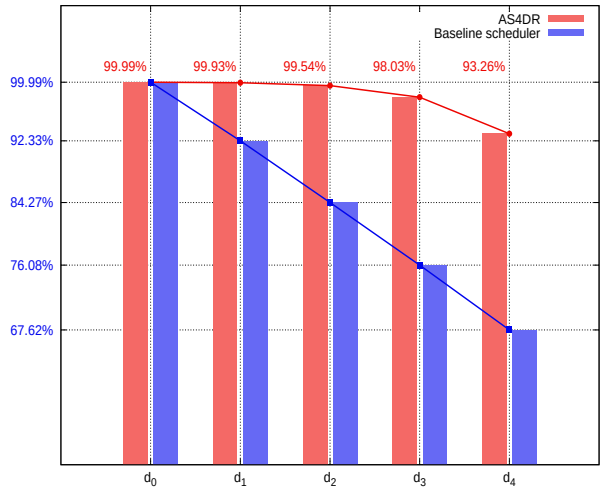

Figure 10. CPU-efficiency of the platform as a function of dynamicity

which lasted 2000 seconds, the value of τ computed by CIP equals 3 seconds. When the execution parameters are steady (δ_0), both schedulers offered the same CPU-efficiency: 99.99%. As expected, the decrease of the CPU-efficiency, when the dynamicity of the execution parameters increases, is significantly faster for the scheduler Baseline than for AS4DR. For instance, when the dynamicity equals δ_4, the CPU-efficiency with AS4DR is 1.38 times higher than with the Baseline scheduler.

V. CONCLUSION

The AS4DR method experimentally assessed in this paper succeeds in maximizing the CPU-efficiency when scheduling a divisible load of unknown total size on distributed resources with inaccurately specified or time-varying characteristics. Despite the fact that a bidirectionnal 1-port communication model is prone to contention, AS4DR can avoid the idleness of the CPU due to contentions, thanks both to the asymptotic periodicity it installs (for both data and results) and to its preliminary step CIP. AS4DR can only be efficient when the unspecification of the execution parameters and their variation over time are small enough; especially AS4DR cannot manage the complete stop of a worker. The pseudo-periodicity of the rounds limits the workload discrepancy when the last data is acquired by the master; when the cleanup phase begins. Thus AS4DR limits

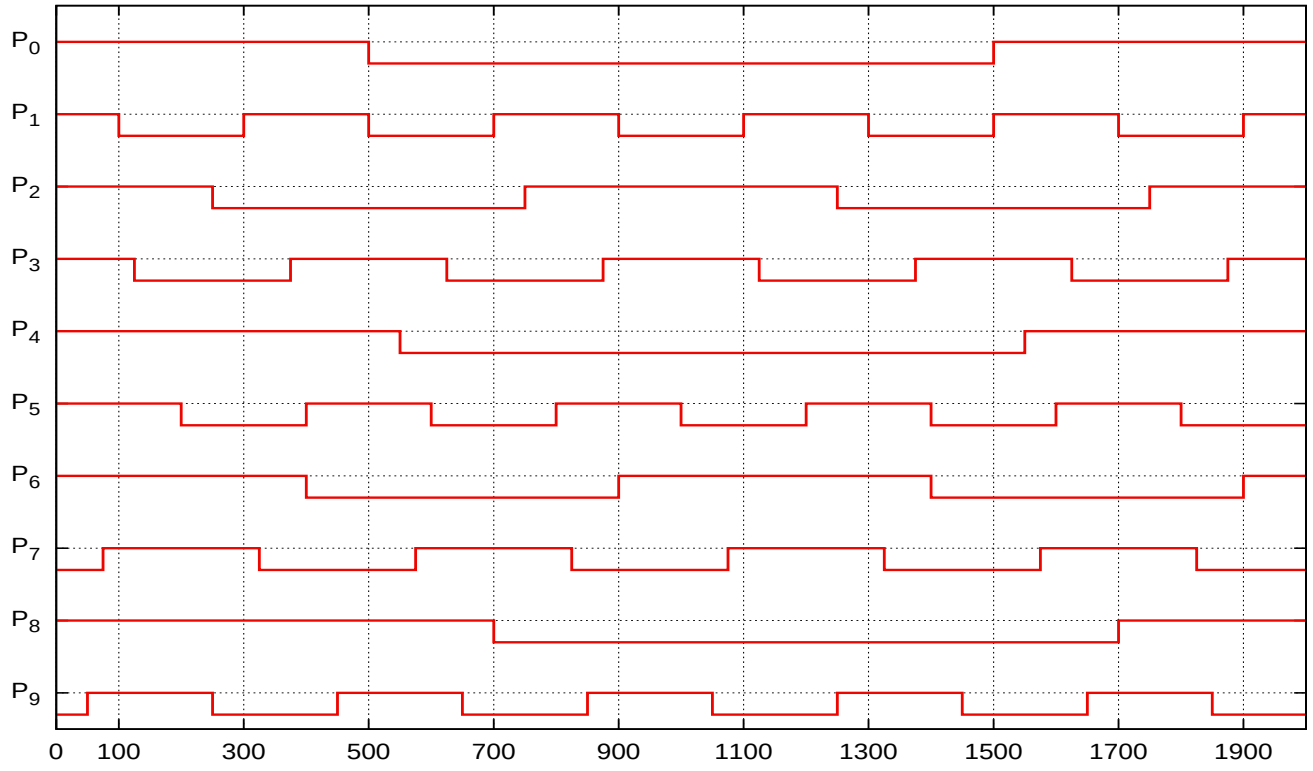

Figure 9. Perturbed execution parameter as a function of time (in seconds)

the cost of the communications for reallocating the extra load of the overloaded workers among the others ones; to make workers ending their processing simultaneously. The experimental results of this paper confirm that the adaptation to either poor estimates of the characteristics of the platform, or to their time-variation, are similar problems. Compared to using pure hardware performance figures, such as measured bandwidth or CPU frequency to adapt the workload at each round, AS4DR has the extra advantage of taking into account characterics of the software such as algorithmic complexity.

Up to now we have considered that the whole set of resources is used. Of course, for a given set of execution parameters values, using all the available resources will ultimately become impossible with an evermore increasing amount of resources. The CIP algorithm could help to select relevant subsets of resources. It is the aim of a future work.

References

[1] D.Millot and C.Parrot, "Scheduling on unspecified heterogeneous distributed resources," in *Proceeding of the 25th International Symposium on Parallel and Distributed Processing Workshops (IPDPSW'11)*, vol. 1, no. 1, IEEE Computing Society Press, May 2011, pp. 45–56.

[2] D. Millot and C. Parrot, "Fundamental results on the AS4DR scheduler," TELECOM sudParis, Évry(France), Tech. Rep. RR-11005-INF, December 2011.

[3] 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.