Some tests of adaptivity for the AS4DR scheduler

Daniel Millot, Christian Parrot

Some tests of adaptivity for the AS4DR scheduler

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

***Abstract*—This paper presents some tests of adaptivity for the AS4DR (Adaptive Scheduling for Distributed Resources) scheduler. The objective of AS4DR is to maximize the CPU use efficiency when executing divisible load applications on heterogeneous distributed memory platforms. Furthermore, this scheduler can operate when the total workload is unknown and when the execution parameters (available communication speed, available computing speed, etc.) are unspecified or may vary through time.**

The paper analyzes results obtained when simulating AS4DR scheduling on platforms characterized by such type of execution parameters.

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

I. INTRODUCTION

Let us consider the problem of maximizing the CPU utilization with useful work when scheduling a divisible load over a set of distributed resources, according to a master-worker model. The master receives a continuous input stream of data to be processed by the workers. 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. The considered execution platform is a 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. 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. As numerous schedulers do, AS4DR attempts to install periodic accesses to the master in order to ease the contention avoidance inherent to the master-workers model. Finally, we assume that master/workers have an unlimited buffering capability.

The rest of the paper is organized as follows. The next section deals with related works. Section III reminds the fundamental results on the AS4DR scheduler whereas section IV presents experimental results on a platform with 1000 workers; they assess the adaptivity of the scheduling obtained with this method. Finally, we conclude.

II. RELATED WORKS

A review of related works can be found in [1]. To the best of our knowledge, no scheduler aims at maximizing the efficiency of the use of the CPUs in a similar context.

III. THE AS4DR METHOD

The proof of the results presented in this section has been established in [2].

A. AS4DR method principle

Let's first recall briefly the context in which the AS4DR method has been proposed and studied. Its ultimate goal is to automatically adapt the scheduling of a divisible load application to the evolution of the execution parameters of the platform 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 periodicity and the estimated time duration 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 (1):

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

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 7)
- 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

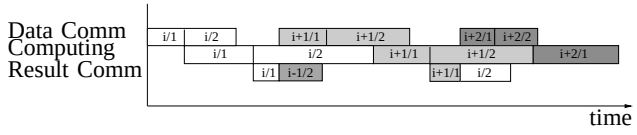

Figure 3. Overlapping between communication and computation

in Figure 3, 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.

As we suppose that the communication and computation costs are both roundwise affine in the size of the corresponding chunk, we have

$$\dot{\mathrm{D}}_{\mathrm{w,i}} = \theta_{\mathrm{w}} \frac{\alpha_{\mathrm{w,i}}}{\mathrm{B}^{\mathrm{D}}_{\mathrm{w}}} + \mathrm{b}^{\mathrm{D}}_{\mathrm{w}} \quad , \qquad \ddot{\mathrm{D}}_{\mathrm{w,i}} = (1-\theta_{\mathrm{w}}) \frac{\alpha_{\mathrm{w,i}}}{\mathrm{B}^{\mathrm{D}}_{\mathrm{w}}} + \mathrm{b}^{\mathrm{D}}_{\mathrm{w}},$$
$$\dot{\mathrm{C}}_{\mathrm{w,i}} = \theta_{\mathrm{w}} \frac{\alpha_{\mathrm{w,i}}}{\mathrm{F}_{\mathrm{w}}} + \mathrm{f}_{\mathrm{w}} \quad , \qquad \ddot{\mathrm{C}}_{\mathrm{w,i}} = (1-\theta_{\mathrm{w}}) \frac{\alpha_{\mathrm{w,i}}}{\mathrm{F}_{\mathrm{w}}} + \mathrm{f}_{\mathrm{w}},$$
$$\dot{\mathrm{R}}_{\mathrm{w,i}} = \theta_{\mathrm{w}} \frac{\alpha_{\mathrm{w,i}}}{\mathrm{B}^{\mathrm{R}}_{\mathrm{w}}} + \mathrm{b}^{\mathrm{R}}_{\mathrm{w}} \quad , \qquad \ddot{\mathrm{R}}_{\mathrm{w,i}} = (1-\theta_{\mathrm{w}}) \frac{\alpha_{\mathrm{w,i}}}{\mathrm{B}^{\mathrm{R}}_{\mathrm{w}}} + \mathrm{b}^{\mathrm{R}}_{\mathrm{w}}.$$

Where F_{w} is the available computation speed (relative to the processing of one workload unit) of worker w. It is worth noticing that this computation speed depends both on the platform and the application, by means of the clock frequency of worker w and the algorithmic complexity of the application respectively. 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

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

The value of $\sigma_{\mathrm{w,i}}$ is an extrapolation of the measured value of $\dot{\mathrm{C}}_{\mathrm{w,i}}$, in order to estimate the computation cost $\mathrm{C}_{\mathrm{w,i}}$ for the whole chunk. So,

$$\sigma_{\mathrm{w,i}} = \alpha_{\mathrm{w,i}} \frac{1}{\mathrm{F}_{\mathrm{w}}} + 2\mathrm{f}_{\mathrm{w}}. \tag{5}$$

The AS4DR method computes the next chunk size using equation (1). From (5) and (1), we get

$$\sigma_{\mathrm{w,i}} = 2\mathrm{f}_{\mathrm{w}} + \tau - \frac{2\mathrm{f}_{\mathrm{w}}\tau}{\sigma_{\mathrm{w,i-1}}}. \tag{6}$$

This sequence can be defined if and only if $\sigma_{\mathrm{w,i}}$ takes non zero values for all i. So, in the sequel, we will suppose that $\alpha_{\mathrm{w,i}}$ and f_{w} do not simultanously equal zero.

Let us suppose next that the communications between master and workers are contention-free. The sequence $(\sigma_{\mathrm{w,i}})_{\mathrm{i}}$, built according to the AS4DR method, linearly converges. More, the scheduling built according to the AS4DR method is asymptotically stable.

$$\text{if } \tau \geq 2\mathrm{f}_{\mathrm{w}}, \text{then} \qquad \lim_{\mathrm{i}\to+\infty} \sigma_{\mathrm{w,i}} = \tau,$$
$$\text{if } \tau < 2\mathrm{f}_{\mathrm{w}}, \text{then} \qquad \lim_{\mathrm{i}\to+\infty} \sigma_{\mathrm{w,i}} = 2\mathrm{f}_{\mathrm{w}}.$$

Practically, the duration τ, typically targeted for a round, is much larger than that of the computation latency $\mathrm{f_w}$. Therefore, only the very first case: $\tau \geq 2\mathrm{f_w}$, will be considered in the sequel. The dates when the master posts data, as well as those when it gets results, are asymptotically periodic, with the same period τ for all the workers.

B. Prevention of idleness and contentions

The AS4DR method could experience either of the workers idlenesses illustrated in Figures 4 and 5 : inter-round idleness and intra-round idleness. When the communications

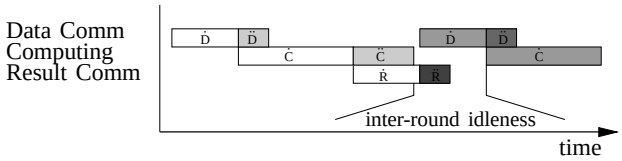

Figure 4. Example of inter-round idleness

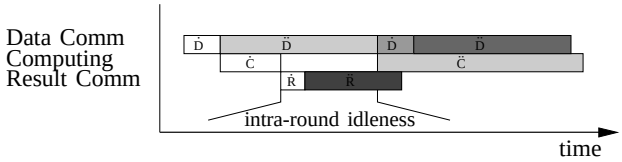

Figure 5. Example of intra-round idleness

between master and workers are contention-free, the value of the ratio θ_w can be set (e.g. according to (14)) such that the AS4DR method prevents idleness. So, in order to take advantage of both this result and the one of the previous section, the problem of contentions avoidance needs to be adressed.

To make the instant each worker accesses the master far enough from the instants the others access too, time delays $\mathrm{d_w}$ are introduced before posting the very first subchunk to each worker w. Introducing such laxity in the scheduling makes the AS4DR method more compliant with the errors on the estimate of the execution parameters. Figure 6 illustrates the model of asymptotic τ-periodic (thus round-robin) scheduling we are looking for, in the case of a four workers platform.

The AS4DR method prevents contentions, if and only if, $\forall$ i and for each worker w (modulo M),

$$\mathrm{d_w} \geq \max\Big(\dot{\mathrm{D}}_{\mathrm{w-1,i}} + \ddot{\mathrm{D}}_{\mathrm{w-1,i}}, \ddot{\mathrm{R}}_{\mathrm{w-1,i-1}} + \dot{\mathrm{R}}_{\mathrm{w,i}}\Big). \qquad (7)$$

The larger the $\mathrm{d_w}$ time intervals, the lower the risk of contentions in case of inaccurate execution parameters estimation, or in case of variation of these parameters over time. Before the launch of the AS4DR scheduler (see Figure 1) a step, called CIP (for Contentions and Idleness Prevention), determines the value for τ and the three sets of

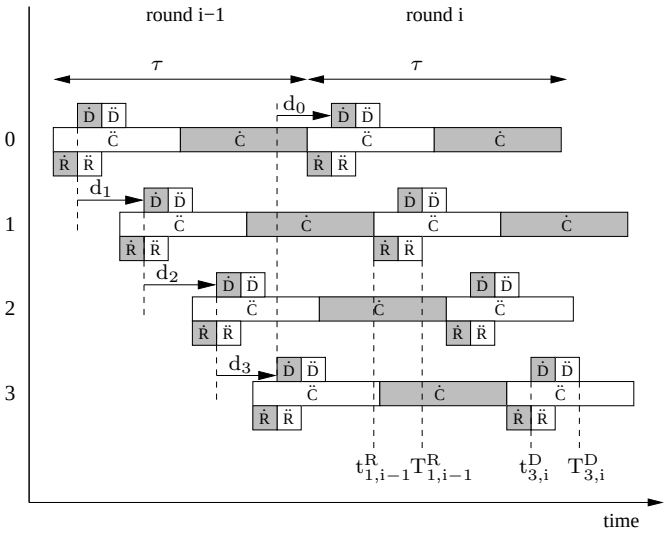

Figure 6. Contention-free asymptotic scheduling

values $(\alpha_{\mathrm{w},1})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$, $(\theta_\mathrm{w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ and $(\mathrm{d_w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$, as shown by Figure 7. Let us suppose that we have at our disposal an estimate of $\mathrm{B_w^D}$ (resp. $\mathrm{F_w}$, $\mathrm{B_w^R}$, $\mathrm{b_w^D}$, $\mathrm{f_w}$ and $\mathrm{b_w^R}$) which is denoted $\overline{\mathrm{B_w^D}}$ (resp. $\overline{\mathrm{F_w}}$, $\overline{\mathrm{B_w^R}}$, $\overline{\mathrm{b_w^D}}$, $\overline{\mathrm{f_w}}$ and $\overline{\mathrm{b_w^R}}$). The

- Estimation of $\left(\mathrm{B_w^D},\ \mathrm{F_w},\ \mathrm{B_w^R},\ \mathrm{b_w^D},\ \mathrm{f_w},\ \mathrm{b_w^R}\right)_{0\leq \mathrm{w}\leq \mathrm{M}-1}$
- Set $(\phi_\mathrm{w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (17)
- Set $(\lambda_\mathrm{w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (19)
- $\tau := 2 \max\limits_{0\leq \mathrm{s}\leq \mathrm{M}-1} \overline{\mathrm{f_w}}$

repeat

- $\tau := \tau + \widehat{\tau}$
- Set $(\alpha_\mathrm{w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (13)
- Set $(\theta_\mathrm{w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (14)
- Set $\left(\overline{\dot{\mathrm{D}}_\mathrm{w}},\ \overline{\ddot{\mathrm{D}}_\mathrm{w}},\ \overline{\dot{\mathrm{R}}_\mathrm{w}},\ \overline{\ddot{\mathrm{R}}_\mathrm{w}}\right)_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (9),(10),(11),(12)
- Set $(\mathrm{d_w})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ ..... (8)

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

Figure 7. CIP algorithm

delay $\mathrm{d_w}$, to be computed by the CIP algorithm is defined as follows:

$$\begin{aligned} \mathrm{d_w} \quad \equiv \quad & (1+\lambda_\mathrm{w}) \max\Big(\overline{\dot{\mathrm{D}}_{\mathrm{w-1}}} + \overline{\ddot{\mathrm{D}}_{\mathrm{w-1}}}, \\ & \overline{\ddot{\mathrm{R}}_{\mathrm{w-1}}} + \overline{\dot{\mathrm{R}}_\mathrm{w}}\Big) \ (\mathrm{modulo\ M}); \end{aligned} \qquad (8)$$

where λ_w is a positive constant factor and $\overline{\dot{\mathrm{D}}_\mathrm{w}}$ (resp. $\overline{\ddot{\mathrm{D}}_\mathrm{w}}$, $\overline{\dot{\mathrm{R}}_\mathrm{w}}$ and $\overline{\ddot{\mathrm{R}}_\mathrm{w}}$) are estimates of $\dot{\mathrm{D}}_{\mathrm{w},1}$ (resp. $\ddot{\mathrm{D}}_{\mathrm{w},1}$, $\dot{\mathrm{R}}_{\mathrm{w},1}$ and

$\ddot{\mathrm{R}}_{\mathrm{w},1}$) obtained thanks to

$$\overline{\dot{\mathrm{D}}_{\mathrm{w}}} := \theta_{\mathrm{w}}\alpha_{\mathrm{w}}\frac{1}{\overline{\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}}}} + \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{D}}}, \quad (9)$$

$$\overline{\ddot{\mathrm{D}}_{\mathrm{w}}} := (1-\theta_{\mathrm{w}})\alpha_{\mathrm{w}}\frac{1}{\overline{\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}}}} + \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{D}}}, \quad (10)$$

$$\overline{\dot{\mathrm{R}}_{\mathrm{w}}} := \theta_{\mathrm{w}}\alpha_{\mathrm{w}}\frac{1}{\overline{\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}}} + \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{R}}}, \quad (11)$$

$$\overline{\ddot{\mathrm{R}}_{\mathrm{w}}} := (1-\theta_{\mathrm{w}})\alpha_{\mathrm{w}}\frac{1}{\overline{\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}}} + \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{R}}}. \quad (12)$$

For relation (7) to hold, the greater the inaccuracy on the estimate of the execution parameters, the greater λ_{w} should be chosen.

The computation of $\overline{\dot{\mathrm{D}}_{\mathrm{w}}}$, $\overline{\ddot{\mathrm{D}}_{\mathrm{w}}}$, $\overline{\dot{\mathrm{R}}_{\mathrm{w}}}$ and $\overline{\ddot{\mathrm{R}}_{\mathrm{w}}}$ requires the values of α_{w} and θ_{w}:

$$\alpha_{\mathrm{w}} := \frac{\tau - 2\overline{\mathrm{f}_{\mathrm{w}}}}{\overline{\frac{1}{\mathrm{F}_{\mathrm{w}}}}} \quad (13)$$

$$\theta_{\mathrm{w}} := \phi_{\mathrm{w}}\theta_{\mathrm{w}}^{\max} + (1-\phi_{\mathrm{w}})\,\theta_{\mathrm{w}}^{\min}; \quad (14)$$

where $\theta_{\mathrm{w}}^{\min}$ and $\theta_{\mathrm{w}}^{\max}$ are respectively deduced from (15) and (16) as follows

$$\theta_{\mathrm{w}}^{\min} := \frac{\frac{\alpha_{\mathrm{w}}}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}} + \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{D}}} - \overline{\mathrm{f}_{\mathrm{w}}}}{\alpha_{\mathrm{w}}\left(\frac{1}{\mathrm{F}_{\mathrm{w}}} + \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}}\right)}, \quad (15)$$

$$\theta_{\mathrm{w}}^{\max} := \frac{\frac{\alpha_{\mathrm{w}}}{\mathrm{F}_{\mathrm{w}}} + \overline{\mathrm{f}_{\mathrm{w}}} - \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{R}}} - \overline{\mathrm{b}_{\mathrm{w}}^{\mathrm{D}}}}{\alpha_{\mathrm{w}}\left(\frac{1}{\mathrm{F}_{\mathrm{w}}} + \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}} + \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}\right)}. \quad (16)$$

When

$$\phi_{\mathrm{w}} = 0.5, \quad (17)$$

the risks of intra-round idleness and inter-round idleness are well balanced. But the value of θ_{w} could be obtained by averaging $\theta_{\mathrm{w}}^{\min}$ and $\theta_{\mathrm{w}}^{\max}$ with other weights than 0.5. Besides, the time intervals d_{w} should allow all the workers to be served during the first round, i.e. within a τ period. Thus τ must verify:

$$\tau \geq \sum_{\mathrm{w}=0}^{\mathrm{M}-1} \mathrm{d}_{\mathrm{w}}. \quad (18)$$

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 $(\alpha_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ and $(\mathrm{d}_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ successively, thanks to (13) respectively, and loops until (18) holds. Once CIP is processed, proper AS4DR scheduling starts with a first round which sets the initial time-lags between successive worker round beginnings, according to the values $(\mathrm{d}_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ 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

$$\Lambda_{\mathrm{w}} \equiv \frac{1}{\mathrm{M}}\frac{1}{\overline{\mathrm{F}_{\mathrm{w}}}\max\left(\mathrm{K}_{\mathrm{w}} + \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}},\ \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}\right)} - 1;$$

$$\text{where } \mathrm{K}_{\mathrm{w}} \equiv \frac{1}{\overline{\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}}}\left(\phi_{\mathrm{w}}\frac{1}{1+\overline{\mathrm{F}_{\mathrm{w}}}\left(\frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}} + \frac{1}{\mathrm{B}_{\mathrm{w}}^{\mathrm{R}}}\right)} + (1-\phi_{\mathrm{w}})\,\frac{1}{1+\frac{\overline{\mathrm{B}_{\mathrm{w}}^{\mathrm{D}}}}{\overline{\mathrm{F}_{\mathrm{w}}}}}\right).$$

Let us assume that for each worker w

$$0 \leq \lambda_{\mathrm{w}} \leq \min\left(\Lambda_{\mathrm{w}}, \Lambda_{\mathrm{w}-1}\right) \quad (\text{modulo M}). \quad (19)$$

The CIP preliminary step provides a τ value and sets of values $(\theta_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$, $(\alpha_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ and $(\mathrm{d}_{\mathrm{w}})_{0\leq \mathrm{w}\leq \mathrm{M}-1}$ that allow the AS4DR scheduling to start with neither contention nor idleness. Inequality (19) cannot hold when the right hand side member is negative. Roughly, this can take place when the number of workers M is too big compared to the computation over communication ratio.

IV. Tests of adaptivity for AS4DR

Simulations have been conducted in order to assess experimentally the AS4DR adaptivity. All the results presented in this section have been obtained with the SimGrid framework [3]. We consider a star-shaped platform (Figure 8 with M=1000) and 10 sets $(\mathrm{S}_{\mathrm{k}})_{0\leq \mathrm{k}\leq 9}$ of values for the execution parameters of the worker nodes, given in Table I. Each of these set is randomly allocated to 100 workers among the 1000 workers which constitute the platform.

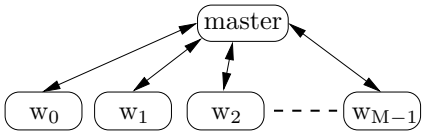

Figure 8. Star-shaped platform

In order 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

	computation		communication master $\longrightarrow$ w_i		communication master $\longleftarrow$ 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

is possible, due to (13), 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 $\mathrm{CPU}_{\mathrm{eff}}$ the CPU-efficiency:

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

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 the inaccuracy of the execution parameters estimates on both schedulings, the effective workload initially allocated to each worker w is set by penalizing the value $(\alpha_w)_{\mathrm{ref}}$ computed using assignment (13) and the reference values contained in Table I:

$$\alpha_{w,1} := (1 \pm \iota_k)\,(\alpha_w)_{\mathrm{ref}}\,; \tag{20}$$

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.

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
VALUES OF ι_k

The values of ι_k characterize the inaccuracy of the execution parameters estimates; 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 9 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 offer 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 with AS4DR is 2.13 times higher than with the Baseline scheduler.

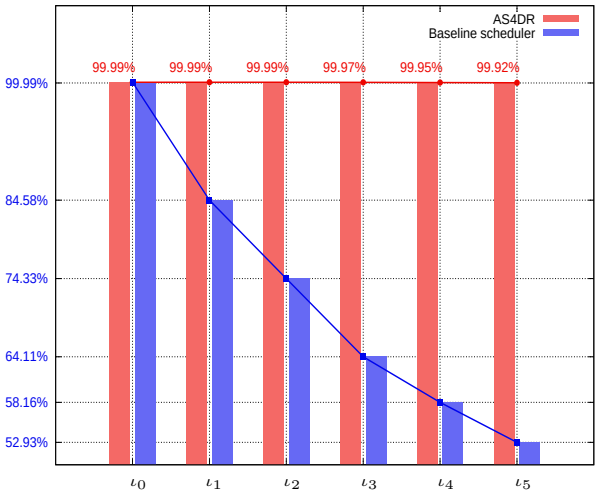

Figure 9. CPU-efficiency of the platform as a function of inaccuracy

Figures 10 and 11 show, for both schedulers, the variation (during the very first rounds) of the CPU-efficiency of a worker with maximum estimates' inaccuracy (ι_5). In abscissa are reported the dates of CPU-efficiency measurement; after the master received the result for the subchunk 1 of the first round and after it received the result for the subchunk 2 of the other rounds. Beyond 26 seconds, the CPU-efficiency of this worker becomes steady, for both schedulers. With AS4DR (Figure 11), the existence of an asymptotic limit to $\sigma_{w,i}$ explains the asymptotical character of the evolution of the CPU-efficiency. For both schedulers this asymptotic CPU-efficiency is numerically reached from the very first rounds. With AS4DR, the smallness of the computation latency of the worker under consideration: 10^{-4} seconds, in front of τ: 3 seconds, explains the magnitude of the speed of convergence; after just a few rounds, the CPU-efficiency with AS4DR for this worker is and stays 17.86 times higher than the one obtained with the Baseline

scheduler. On Figures 10 and 11 it can be noticed, for

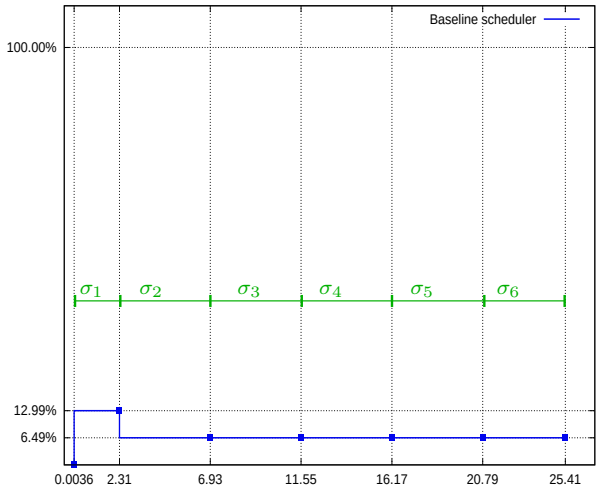

Figure 10. Baseline: CPU-efficiency of a worker as a function time (in seconds)

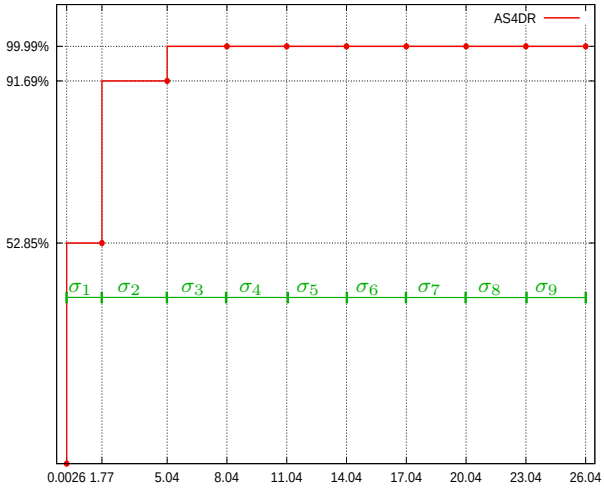

Figure 11. AS4DR: CPU-efficiency of a worker as a function time (in seconds)

the worker under consideration, that the dates of end of rounds (in abscissa) become asymptotically periodic. For instance, with AS4DR, the values $(\sigma_i)_{3\leq i\leq 9}$ are identical and equal τ (computed with CIP): 3, while the values $(\sigma_i)_{2\leq i\leq 6}$ are equal to 4.62, with the Baseline scheduler. The wanted asymptotical period τ: 3, is not reached for this worker with the Baseline scheduler.

In order to estimate the duration $(\sigma_{w,i})_{i\geq 1}$ of a round for all the workers of the platform, Figure 12 shows its average and its standard deviation over all the rounds and for all the workers, for different values of estimates' inaccuracy. For both methods, Figure 12 shows, on the one hand, that the

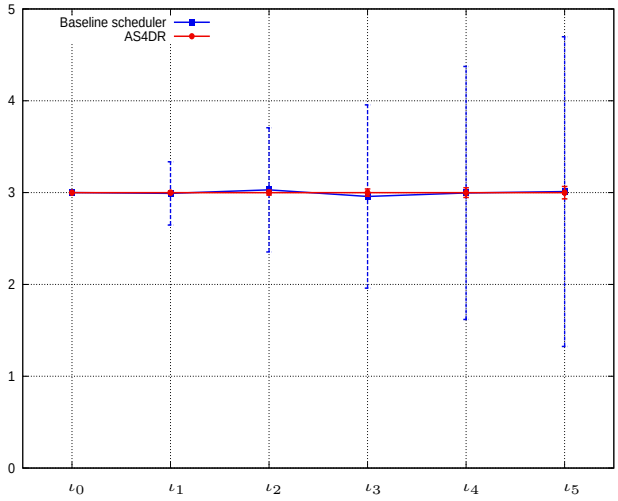

Figure 12. Average and standard deviation of $\boldsymbol{\sigma}$ as a function of inaccuracy

average of the values of $(\sigma_{w,i})_{i\geq 1}$ is close to the wanted value τ and, on the other hand, that the standard deviation of these values increases when estimates' inaccuracy of the execution parameters increases. It can be noticed that the standard deviation increases much faster with the Baseline scheduler, in relation to estimates' inaccuracy, than with AS4DR. For instance, the standard deviation with the Baseline scheduler is 24.98 times higher than with AS4DR for ι_5. The ratio: 17.86, observed between the CPU-efficiency of both schedulers for ι_5 on Figure 9, is a consequence of this disparity between the standard deviations of the values of $(\sigma_{w,i})_{i\geq 1}$. By adapting at each round the workload for each worker AS4DR reduces, on the one hand, the risk of contentions inherent to the 1-port communication model and, on the other hand, the risk (in Round-Robin mode) of waiting for the result of a previous worker.

B. Time-varying context

The previous subsection has shown the ability of the AS4DR method to adapt to a priori poor estimates of execution parameters in steady execution contexts. 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 $(P_k)_{0\leq k\leq 9}$ of variation shown by Figure 13. Whatever the profile P_k being allocated to a worker w and whatever an execution parameter, the higher value of 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:

$$\mathrm{B_w^D} := (1-\delta_\mathrm{k})\left(\mathrm{B_w^D}\right)_\mathrm{ref}, \mathrm{B_w^R} := (1-\delta_\mathrm{k})\left(\mathrm{B_w^R}\right)_\mathrm{ref}, \quad (21)$$

$$\mathrm{F_w} := (1-\delta_\mathrm{k})\left(\mathrm{F_w}\right)_\mathrm{ref}; \quad (22)$$

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.

δ_0	δ_1	δ_2	δ_3	δ_4
0.0	0.2	0.4	0.6	0.8

Table III
VALUES OF δ_k

Figure 14 shows the measured CPU-efficiency of the whole platform for each scheduler, as a function of the dynamicity. For this simulation, which lasted 2000 seconds, the value of

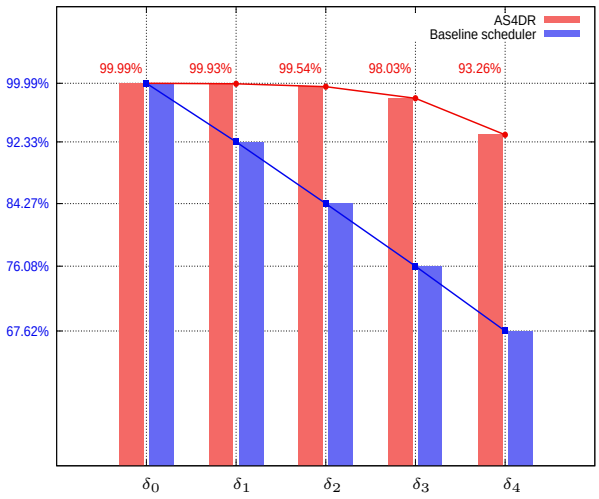

Figure 14. CPU-efficiency of the platform as a function of dynamicity

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 CPU-efficiency for both methods decreases when the dynamicity of the execution parameters increases. This decrease 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.

Figures 15 and 16 show, for both schedulers, the variation of the CPU-efficiency of a worker with profile $\mathrm{P_9}$ during the first reduction of the value of the execution parameters, when the dynamicity is maximum: δ_4. When time becomes equal

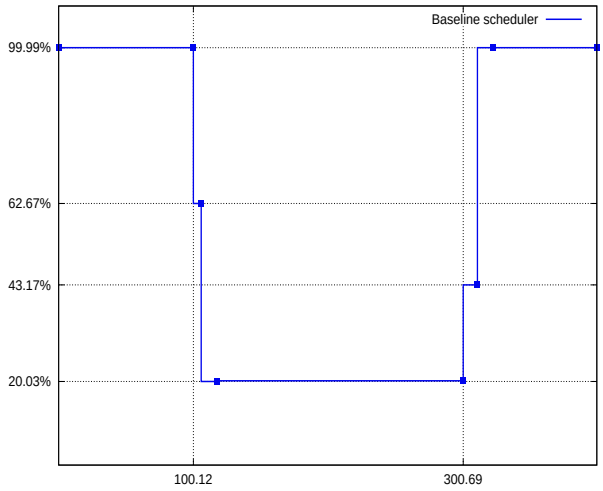

Figure 15. Baseline: CPU-efficiency of a worker as a function of time (in seconds)

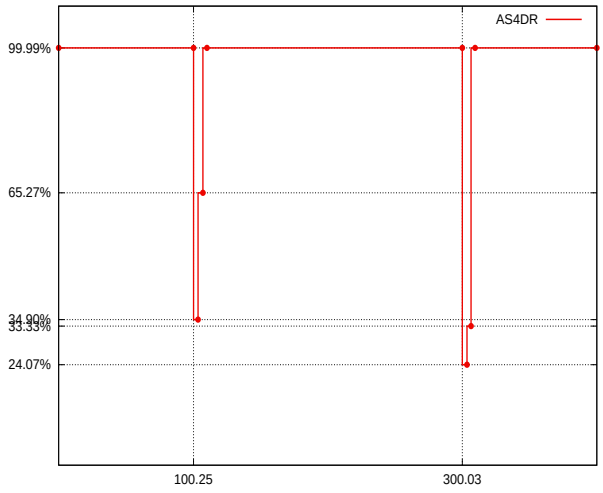

Figure 16. AS4DR: CPU-efficiency of a worker as a function of time (in seconds)

to 100 seconds each execution parameter, for the worker under consideration, is reduced by 80%. Then, when time becomes equal to 300 seconds each execution parameter recovers its reference value. With AS4DR (Figure 16), the smallness of the computation latency of the worker under consideration in front of τ, explains the quickness of the adaptation (3 rounds) to the change of context; on the falling edge as well as the rising one. In the absence of adaptation to the change of the characteristics of the platform, the CPU-

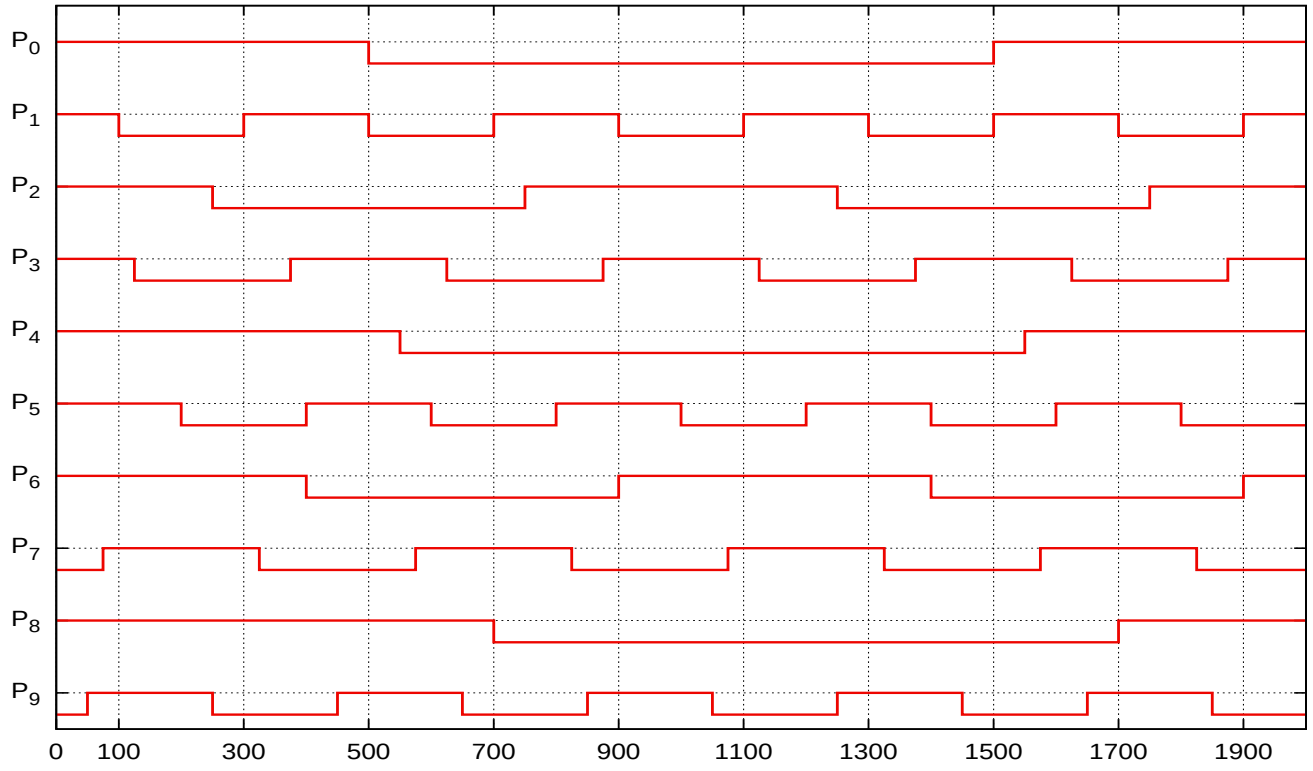

Figure 13. Perturbed execution parameter as a function of time (in seconds)

efficiency observed (Figure 15) with the Baseline scheduler is reduced during a long while (68 rounds).

Figure 17 shows the average and the standard deviation of $(\sigma_{w,i})_{i\geq 1}$ over all the rounds and for all the workers, for different values of the dynamicity. It can be noticed that, with the Baseline scheduler, the average of the values $(\sigma_{w,i})_{i\geq 1}$ is an increasing function of the dynamicity. The reason for this is that the perturbation of the context, due to (21)and (22), is a reduction of the reference value of the execution parameters, given in Table I; which systematically increases the duration of the rounds. On the contrary, with AS4DR, the average of the values $(\sigma_{w,i})_{i\geq 1}$ stays close to the wanted value τ. For both methods, Figure 17 shows that the standard deviation of these values is an increasing function of the dynamicity. Once again, the better CPU-efficiency observed with AS4DR in relation to the one observed with the Baseline scheduler, for instance for δ_4 on Figure 14, is a consequence of the disparity between the standard deviations of the values of $(\sigma_{w,i})_{i\geq 1}$, shown by Figure 17.

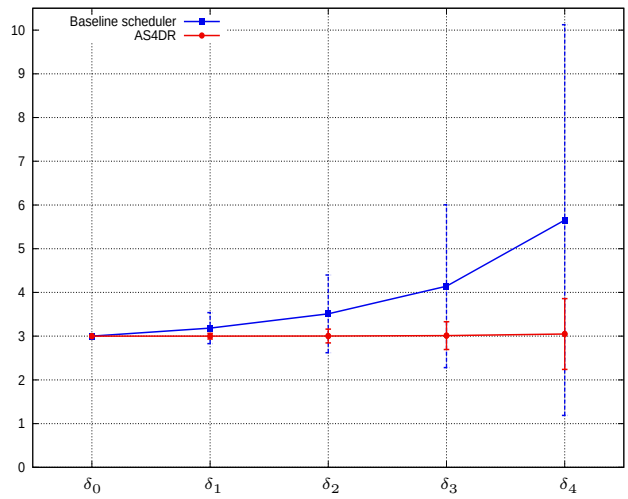

Figure 17. Average and standard deviation of $\boldsymbol{\sigma}$ as a function of dynamicity

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. The experimental

results presented in 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. The optimality of such a selection with respect to the global throughput will be 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.