Improvements and Study of the Accuracy of the Tasks Duration Predictor, New Heuristics

Yves Caniou, Emmanuel Jeannot

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Improvements and Study of the Accuracy of the Tasks Duration Predictor, New Heuristics

Yves Caniou — Emmanuel Jeannot

N° 5206

May 2004

ThŁme COM

ISSN 0249-6399 ISRN INRIA/RR--5206--FR+ENG

Improvements and Study of the Accuracy of the Tasks Duration Predictor, New Heuristics

Yves Caniou* , Emmanuel Jeannot

Thème COM — Systèmes communicants
Projets AlGorille

Rapport de recherche n° 5206 — May 2004 —52 pages

Abstract: The Historical Trace Manager is a task duration predictor module embedded in the agent of a Problem Solving Environment relying on the client-agent-server. The HTM is introduced in [CJ02a] and [CJ04]. In this paper, we explain some improvements built into the HTM and NetSolve, the Problem Soving Environment we use for our tests, in order to synchronize the HTM to the reality.

We also introduce two new scheduling heuritics relying on the HTM information: Advanced HMCT and Minimum Length.

We study the scheduling of several scenarios, including the simultaneous submissions of DAGS and independent tasks, on a real heterogeneous platform.

The excellent behavior of the HTM validates its estimations of the duration of each task concurrently running in the system. It can consequently predict the contention tasks may have on each other if scheduled and executed concurrently on the same computing resource.

Heuristics performances show the relevancy of the HTM information through the experiments: their ability of behaving with a constant quality between two executions of the same experiment as well as the quality of their respective scheduling choices to optimize several criteria at the same time. We also show that heuristics which rely on minimizing the contention give generally the best results regardless the criterion.

We finally compare the behavior of the heuristics previously tested in [CJ04] to the one observed here with more precise information on the global system state due to the synchronization mechanisms. Surprisingly, in the time-shared model, it does not necessarily improve the job repartition among the servers, performances can consequently decrease and the utilization of the fastest servers can become critical.

Key-words: time-shared resources, dynamic scheduling heuristics, historical trace manager, MCT, client-agent-server

* This work is partially supported by the Région Lorraine, the french ministry of research ACI GRID

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Amélioration et Étude de la précision du module de prédiction de la durée des tâches, Nouvelles heuristiques

Résumé : Nous revenons tout d'abord sur le gestionnaire de l'historique des tâches (HTM), présenté précédemment dans [CJ02a] et [CJ04], auquel des améliorations ont été apportées. Il est maintenant capable de se synchroniser avec ce qui est effectivement effectué sur chacun des serveurs.

Nous présentons aussi deux nouvelles heuristiques d'ordonnancement : Advanced HMCT et Minimum Length.

Nous étudions l'ordonnancement sur une même plate-forme hétérogène de plusieurs types de scénarios, comme la soumission concurrente d'applications sujettes à des contraintes de précédence concurrentes et de tâches indépendantes.

Nous montrons que la qualité des estimations du HTM permet de prendre correctement en compte le modèle temps partagé et de déduire le délai sur la terminaison des tâches occasionné par l'affectation et l'exécution concurrente de plusieurs tâches sur un même serveur de calcul.

Les observations sur les performances des différentes heuristiques montrent l'intérêt du HTM dans la prise de décision de l'ordonnanceur ainsi que dans la constance des résultats d'une exécution à l'autre d'une même expérience. De plus, nous observons que les heuristiques dont l'objectif prends en compte la minimisation de délais donnent de meilleures performances sur plusieurs critères comme le makespan ou le temps de réponse moyen par exemple.

Enfin, nous comparons les performances des heuristiques que nous avions précédemment testées dans [CJ04] à celles obtenues ici avec des informations plus précises sur l'état global du système. Nous expliquons que dans le cadre des exécutions temps-partagé, une meilleure précision n'apporte pas forcément une meilleure répartition des tâches et que l'utilisation des ressources les plus rapides peut devenir critique.

Mots-clés : ressources temps partagées, heuristiques dynamiques d'ordonnancement, gestionnaire d'historique des tâches, MCT, client-agent-serveur

1. Introduction

The historical Trace Manager is a task duration predictor embedded in the agent of a Problem Solving Environment which is built on the client-agent-server model. We know from the experiments undertaken in [CJ04] that it lets heuristics consider other criteria than the unique completion date of the last request entering the system. Indeed, it takes into account all the previous scheduling decisions done until the new submission to give according information to the scheduler.

Several heuristics have been designed in [CJ02a] in order to optimize at the same time the finishing date of the execution of the new request in the system as well as to give a better quality of service to still running tasks, for example the average response time. Indeed, running tasks can suffer from the delay the affectation of the new task can perform. Our heuristics rely on the information given by the HTM at each submission date. We have examined their performances on several criteria against Minimum Completion Time, a scheduling heuristic widely used in common Problem Sovling Environment like NetSolve or Ninf. The research rapport [CJ04] contains the experimentation procedural and the results obtained with real world experimentations.

Then, the main objective of the HTM is to let take into account other criteria in the scheduling decisions. If this does not imply 100% accurate estimations, a minimum of accuracy is naturally necessary to render the system state and *a priori* the most accurate the estimation is, the best the scheduling must be.

Thus, we have improved the HTM and the Environment. We explain in Section 2 how it is synchronizes to the reality. We present two new heuristics in Section 3. Experiments modalities are introduced in Section 4. We study the accuracy of the HTM in Section 5. We then compare the performances of each heuristics in Section 6 and, in Section 7, we compare their actual performances to the ones observed in [CJ04] when estimations were less precise. After exposing our future work in Section 8, we finally conclude in Section 9.

2. HTM Improvement: Synchronization to the Reality

The HTM, firsly developped for [CJ02b] and [CJ02a] for simulation studies and coded in NetSolve for real experimentations ([CJ04] and [CJ03]), has some new features to be synchronized to the real Problem Solving Environment state. We expect them to improve even further the accuracy that the HTM used to give.

The HTM is an environment simulator. Both the HTM and the scheduler, with which it cooperates, are modules embedded in the agent. It focus on simulating what is performed on each component of the system in order to be able to answer sharp questions to the scheduler: information are used by the heuristic which does not only rely on the completion date of the new task but also on the duration of each still running one.

It uses the time-shared model to simulate the environment: all networks links, which are consequently assumed in our work to be LAN, and servers can use their resource capacities equally between the works they have to perform. Hence, a server that computes n tasks at a given instant gives $1/n$ of CPU time to each of them. Further information can be obtained in previously mentionnned papers.

We have implemented some new features in the HTM, and have in consequence changed the code of NetSolve in order to build some mechanisms on which the HTM has to rely: when a client requests the agent for a job, it is given back a ranked list of possible servers like before but also a global ID for its request. This ID is transfered with the problem data to the server and when the task finishes, the ID is sent

back in a completion message to the agent. The agent uses it in two ways: it firstly testifies that the task has been effectively scheduled on the previously proposed server, and secondly this information is used to correct what has been simulated on the HTM and improve the quality of predictions (correct them in the former case). Indeed, all future finishing dates are re-computed and the heuristic uses those *a priori* more precise information for later scheduling.

The ID assignation and return mechanisms are also used in VisPerf [1], a soft utility to visualize what is performed on the environment controlled by NetSolve. This ID, in addition to certify that the client has contacted the server according to the scheduling decision, can also be used to correct the HTM data if this is not the case, e.g. if another server has executed the task. However, this particular case is not considered in this paper but we will study the response time (time to re-establish a good accuracy) and the performance of the HTM facing that kind of situation in future works.

3. New Heuristics

In addition to the three heuristics HMCT, MP and MSF that we have already presented and whose performances on real platforms are studied in [CJ04], we have designed and implemented two new ones: AHMCT and ML.

The mechanism is the same than for the previous ones: when a new request arrives, the HTM simulates the execution of the task on each server. Our heuristics use the HTM information, hence consider the perturbation that tasks induce on each other and compute the 'best' server given an objective which is explained in the corresponding subsection.

```
1 For each new task t
2     For each server j that can resolve the new submitted problem
3         Ask the HTM to compute M_j the makespan on server j, if t is executed on j
4     Map task t to server j_0 that minimizes M_{j_0}
5     Tell the HTM that task t is allocated to server j_0
```

Figure 1. *AHMCT algorithm*

3.1. Advanced Historical Minimum Completion Time: AHMCT

AHMCT is Minimum Completion Time (MCT) like it would have normally been in the time-shared model. Indeed, in the space-shared model, the completion date of the new request is also the makespan (maximum task completion date) on the server.

When a new task arrives, the HTM simulates the mapping of the task on each server. Therefore, the scheduler has an estimation of each still running task finishing date on each server before and after a possible execution of the task on the server.

AHMCT would normally computes each 'global' makespan (one per possible affectation) as the maximum between the makespan on the server where the task is simulated and the makespan on the other

[1] http://icl.cs.utk.edu/netsolvedev/applications/visperf.html

servers. Then it chooses the server leading to a minimum global makespan. But theses computations can be summarized by the $n-1$ comparisons, if n is the number of servers, between the makespan on each server after the simulation of the new request (see the algorithm in Fig. 1). Indeed, if we adopt the following notation: m'_k is the makespan on each server k before the simulation of the new task. We note $m_{i,j}$ the makespan on the server j after the simulation of the task on server i. In that case, we have for all $j \neq i, m_{i,j} = m'_j$ because the makespan is unchanged on the other servers.

With these notations, AHMCT chooses the server $s_0 = \min_{servers} \max_{serveri} m_{s,i}$.
Then $s_0 = \min_{servers} max(\max_{serveri \neq s} m_{s,i}, m_{s,s})$ e.g. $s_0 = \min_{servers} max(\max_{serveri \neq s} m'_i, m_{s,s})$. But $m_{i,i} > m_i$ then $s_0 = \min_{servers} m_{s,s}$.

The goal of AHMCT is the same as MCT's: it expects to minimize the makespan of the application by minimizing the makespan obtained after each new task.

The main drawback of this heuristic is that it tends to overload the fastest servers, which has two effects: unnecessarily delay running tasks completion date and servers may collapse, mainly due to a lack of memory or to the incapability to handle the throughput of requests.

3.2. Minimum Length: ML

ML was designed jointly to MSF: it aims to minimize the left quantity of time that all the concurrent running tasks require and that the server has to process. Therefore, after the new task is simulated on each server, it chooses the server that minimizes the sum of the left amount of time of all running tasks (see Fig. 2).

The main objective is to reduce the perturbation that tasks have on each other, while conducting a makespan policy: perturbations also induce the left amount of time (equal for it to the total amount of time) of the new task which is taken into account in the value to minimize.

1 **For each** new task t
2 **For each** server j that can resolve the new submitted problem
3 Ask the HTM to compute $P_j = \sum_{tasks\ i} T_i - a_t$
4 Map task t to server j_0 such as $P_{j_0} = \min_j P_j$
5 Tell the HTM that task t is allocated to server j_0

Figure 2. *ML algorithm*

4. Experiments Modalities

The HTM and all heuristics, on which information are available in [CJ02a], as well as improvements and new heuristics have been coded in NetSolve for the following test suite.

NetSolve [CD96] is a Problem Solving Environment (PSE) built on top of a GridRPC architecture instantiating the emerging standard promoted by the global grid forum (GGF)[2]. A GridRPC architecture is heterogeneous and composed of three parts: a set of clients, a set of servers and an agent (also called a

[2] http://www.ggf.org

Type	Machine	Processor	Speed	Memory	Swap	System
Server	spinnaker	xeon	2 GHz	1 Go	2 Go	Linux
	artimon	pentium IV	1.7 GHz	512 Mo	1024 Mo	Linux
	soyotte	sparc Ultra-1		64 Mo	188 Mo	SunOS
	fonck	sparc Ultra-1		64 Mo	188 Mo	SunOS
Agent	xrousse	pentium II bipro	400 MHz	512 Mo	512 Mo	Linux
Client	zanzibar	pentium III	550 MHz	256 Mo	500 Mo	Linux

Table 1. Resources composing the testing environment

server	task type 1	task type 2	task type 3
spinnaker	15	30	43
artimon	17	33.5	49.5
soyotte	128	256	382.5
fonck	127.5	254	380.9

Table 2. Duration of the tasks per type on the unloaded servers

registry). The agent has in charge to map a client request to a server.

All results that are presented in this paper are issued from experiments conducted on the same set of resources, whose characterics are given in Table 1. We consider the *heterogeneity coefficient of the platform* equal here to 8.9. It is the maximum on all servers and on all tasks of the division of the maximum computing time by the minimum computing time for the same task, e.g. $\max_{task_i} \frac{\max_{servers} d_{i,s}}{\min_{servers} d_{i,s}}$.

Experiments involve three types of tasks, whose resources requirements are given in Table 2. Tasks are computing intensive and do not need memory. Several submission scenarios have been examined. Scenarios (a) to (i) were performed before the new features exposed in Section 2, e.g the task ID assignment and the synchronization of the HTM to the reality. Heuristics performances for these scenarios as well as more information on the experimental procedures and practical details can be found in [CJ04]. Scenarios (a') to (i') are the same than the ones above, submitted to the improved HTM. A new scenario (j') is also considered to evaluate the heuristics facing a lower throughput of jobs and let do some remarks about the 'thoughput' and 'server load' notions. Table 3 gives a summary on the characteristics of each scenario, like the number of experiments undertaken.

Note that in order to validate each experiment in a scenario, several runs have been performed. We have already seen in [CJ04] that two runs could differ from each other as soon as a graph is involved: tasks durations differ and so the submissions date of daughter tasks. With the synchronization mecanisms that let register the real completion date of a task, submission dates differs as well as the recorded duration, then even for independent tasks submissions, two runs would likely differ from each other. We have consequently undertaken a number of runs of each experiment upon which average results should not change anymore.

5. HTM Accuracy: Results

To evaluate the HTM (Section 5), we have preferred to study its accuracy along all the experiments. Indeed, the submission of some tasks to validate its accuracy would only give an impression but not a real basement for its use: as we'll see, its utilization leads to consider the real capacity of the resources, thus because heuristics have different tendencies like to load the fastest servers first, the model can be treated roughly.

We present in this section the results on the accuracy of the HTM obtained from the experiments ordered by scenario. They are presented with graphs and tables: tables give the average percentage error made during the experiment and the standard deviation *per server*. The value depends greatly on the number of tasks that has been assigned to the server, and on the perturbations they had on each other. Thus, graphs present in the x axis the submission date of the task and two information in the y axis: the ratio of the

Scenario	Application(s)		Independent Tasks		Experiment	
	nbclients	width x depth	nbtasks	μ (sec)	nbseeds x nbrun	total nbtasks
(a) 500 independent tasks	-	-	500	20, 17 and 15	3 x 3 , 3 x 5	500
(b) 1D-mesh	10	1x50	-	-	4 x 6	500
(c) 1D-mesh	10	1x variable	-	-	4 x 6	500
(d) 1D-mesh + 250 independent tasks	5	1x50	250	20	4 x 6	500
(e) stencil (task 1)	1	10x50	-	-	1 x 6	500
(f) stencil (task 3)	1	10x50	-	-	1 x 6	500
(g) stencil + 174 independent tasks	1	10x25	174	28	2 x 3	424
(h) stencil + 86 independent tasks	1	10x25	86	40	4 x 6	336
(i) stencil + 86 independent tasks	1	5x25	86	25	4 x 6	211
Scenario	Application(s)		Independent Tasks		Experiment	
(a') 500 independent wastecpu tasks	-	-	500	20, 17 and 15	3 x 1	500
(d') 1D-mesh + 250 independent tasks	5	1x50	250	20	3 x 1	500
(e') stencil (task 1)	1	10x50	-	-	1 x 6	500
(h') stencil + 86 independent tasks	1	10x25	86	40	3 x 1	424
(i') stencil + 86 independent tasks	1	5x25	86	25	3 x 3	211
(j') 1D-mesh	2	1x50	253	25	3 x 3	353

Table 3. Scenarios, modalities and number of experiments

real flow divided by the flow estimated by the HTM and the real number of tasks that have impacted the considered task. If precised, the graph may be in log scale. A zoom of the ratio is generally provided: it gives the percentage of accuracy of the prediction on the flow for each task. Graphs are only given for the fastest servers spinnaker and artimon because regardless the heuristics, no task or a few are scheduled on the SUN servers: the accuracy is then nearly 100%. Except when mentionned otherwise and because the server where the same task has been mapped may change between two runs, graphs are the result of one representative run of one experiment of the scenario.

Two methods have been used to compute the different graphs, one for the graphs before the global ID assignment and one for after this improvement:

- Because no information were available before the implementation of the global ID assignment to recognize which task have finished, it was difficult to establish a connection between the real and the HTM walues. Therefore, values for real flow have been taken from the recorded entries on the client side. Then, the real flow is the flow observed from the client point of view. The HTM flow estimation is the one made at the submission of the task, because in that form the HTM considers himself to simulate the exact reality. Thus, the evaluation done at the submission of the task is never re-evaluated.
- Because of the global ID assignation, the connection between the HTM values and the real observed dates is possible. A parsing of the agent log file gives the real submission date and completion date of a task. Then, the HTM computes and records the duration of each task that it would have required on the same but unloaded server. Indeed, the completion of a previous submitted task is used to re-evaluate the estimation of the completion of each task on the server. The HTM flow estimation is taken from the last estimation before the completion of the task (at which moment, the estimation of the completion date is exactly the real one). With this method, only the agent log file is parsed.

As mentionned above, the ratio is the division of the real flow by the HTM estimation. In consequence, when inferior to 1, the estimation gives a longer flow than the real one. The immediate scheduling consequence is that the heuristic assumes that the server is more loaded than it really is and thus take scheduling decisions accordingly: it sub-utilizes the servers.

In the following subsections, we present when possible the accuracy observed for each scenarios with and without the HTM improvements ordered by heuristic. The goal is twofold: it firstly attempts to give the evolution of the accuracy with this new feature and the heuritic. Secondly, one can then compare the behavior of the same heuristic facing the same kind of experiment with *a priori* more accurate information on the system state and the consequences that it implies.

5.1. Scenario (a) and (a'): Submission of Independent Tasks

Values used in figures that are given in this subsection for Scenario (a), e.g independent tasks submitted to the HTM before the synchronization mechanisms, are the mean of a given number of runs of the same experiment (see Table 3). This is only possible because the scheduling decisions, if dynamic, are taken accordingly to the system state. But submissions are exactly the same between two runs of the same experiment (tasks are submitted at the same time) then chosen servers are the same from one run to another.

This is not the case anymore for scenario (a') submitted to the improved HTM. Real tasks completion time are now taken into account to compute information at a new request arrival and tasks have not exactly the same flow. Thus decisions are not identical between two runs and graphs result of one experiment.

We give in the graphs of Figures 3, 5 and 7 the accuracy observed for Scenario (a) at different rate scheduled by respectively HMCT, MP and MSF on spinnaker and artimon. When utilized, the accuracy on fonck and soyotte is more than 98% accurate (if any, there are only small perturbations involving no more than 2 ou 3 tasks on these servers). Graphs given in Figures 4, 6 and 8 are results of Scenario (a') scheduled by the same heuristics. For each heuristic, it is interesting to note the accuracy obtained in regard to the rate of the incoming jobs, as well as the number of tasks that have been scheduled to the servers.

5.1.1 HMCT

HMCT never uses the SUN servers fonck and soyotte before and after the HTM improvement, even at a rate equal to 15 second where it cannot handle the throughput of incoming jobs: at a given moment, the fastest servers refuse one or more job.

One can see from the graphs in Figure 3 that at rate $\mu = 20$ seconds, most of the task flows are sub-estimated: the HTM flow estimation is longer than the real one. The server is able to process a greater quantity of work than the one modelized by the time-shared model. Moreover, Table 4 shows that an error of 3.7% is done in average on spinnaker for Scenario (a).

The synchronization mechanisms induces a great improvement in the accuracy and consequently in the resource management: at $\mu = 17$ seconds, the HTM has a lower error regardless the server than before at $\mu = 20$ seconds. Moreover, the number of tasks scheduled on the fastest servers have increased and graphs in Figure 4, showing the accuracy of an experiment at rate $\mu = 17$ seconds, shows that more than 10 tasks have interfered with another one during its execution on the fastest servers. Although the great variation in the number of interfering tasks, the standard deviation is very small (see Table 5.

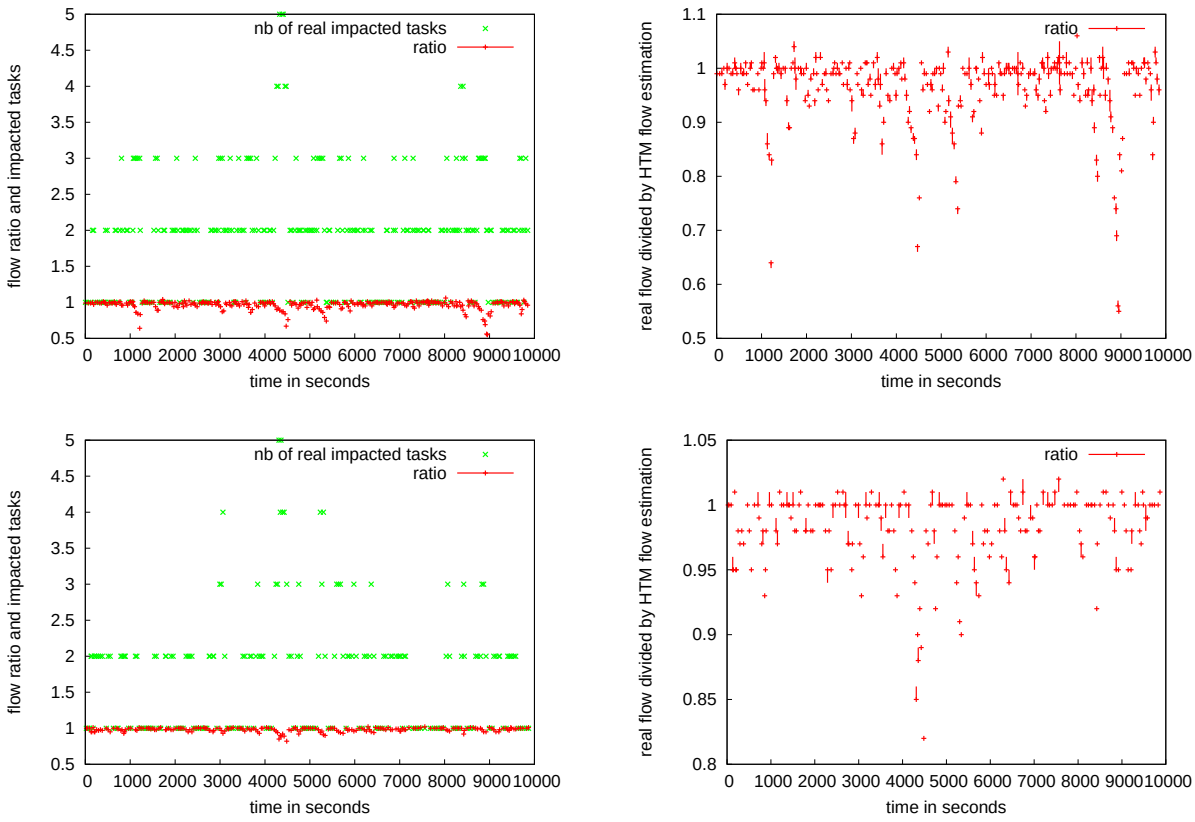

Figure 3. Accuracy of the HTM during scenario (a), $\mu = 20$ seconds, scheduled with HMCT, on spinnaker on the top and artimon on the bottom

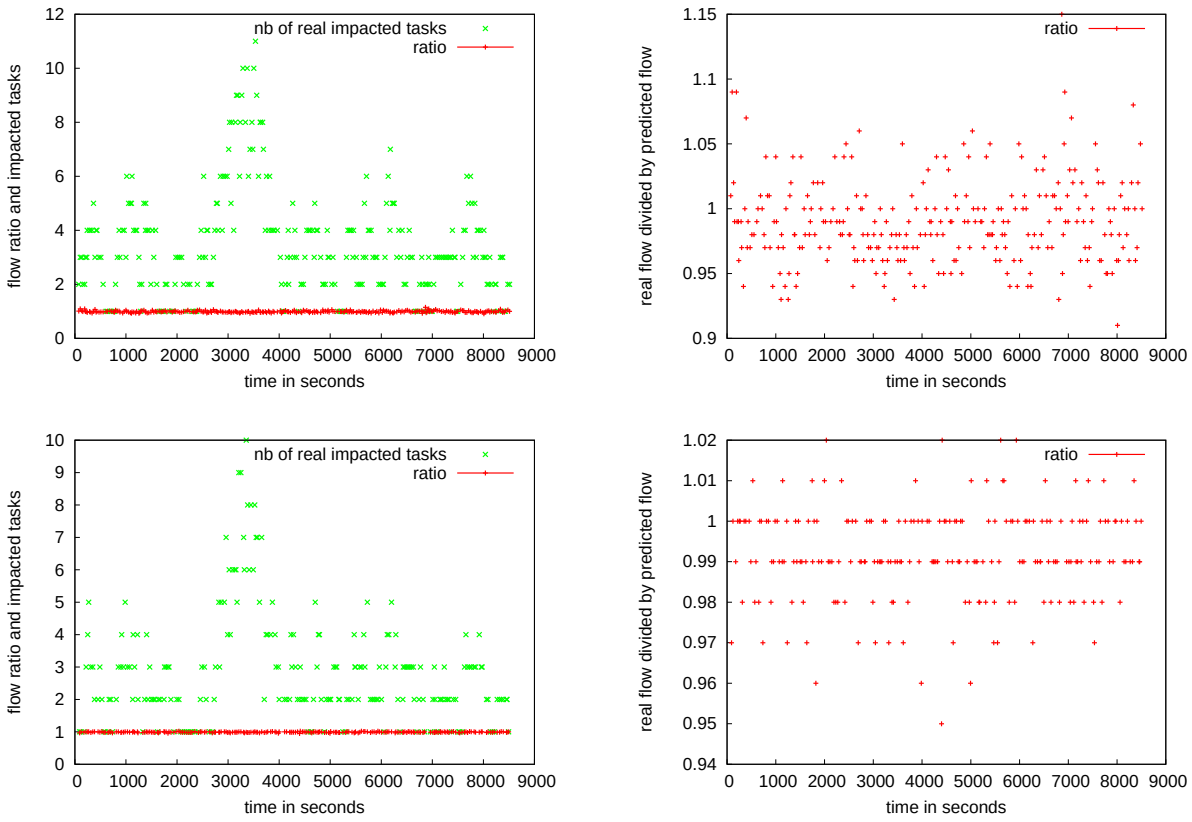

Figure 4. Accuracy of the HTM during scenario (a'), $\mu = 17$ seconds, scheduled with HMCT, on spinnaker on the top and artimon on the bottom

5.1.2 MP

We present in Figures 5 and 6 graphs of an experiment composed of independent tasks whose submission rate is $\mu = 17$ seconds. Again, we can establish the accuracy of the HTM and the consequent change in behavior of the heuristic: more tasks are assigned to the fastest servers as the average error is minimized. One can read in Tables 4 and 5 that for MP, and the consequent number of interfering tasks, the gain in precision is much more in the standard deviation than in the average error.

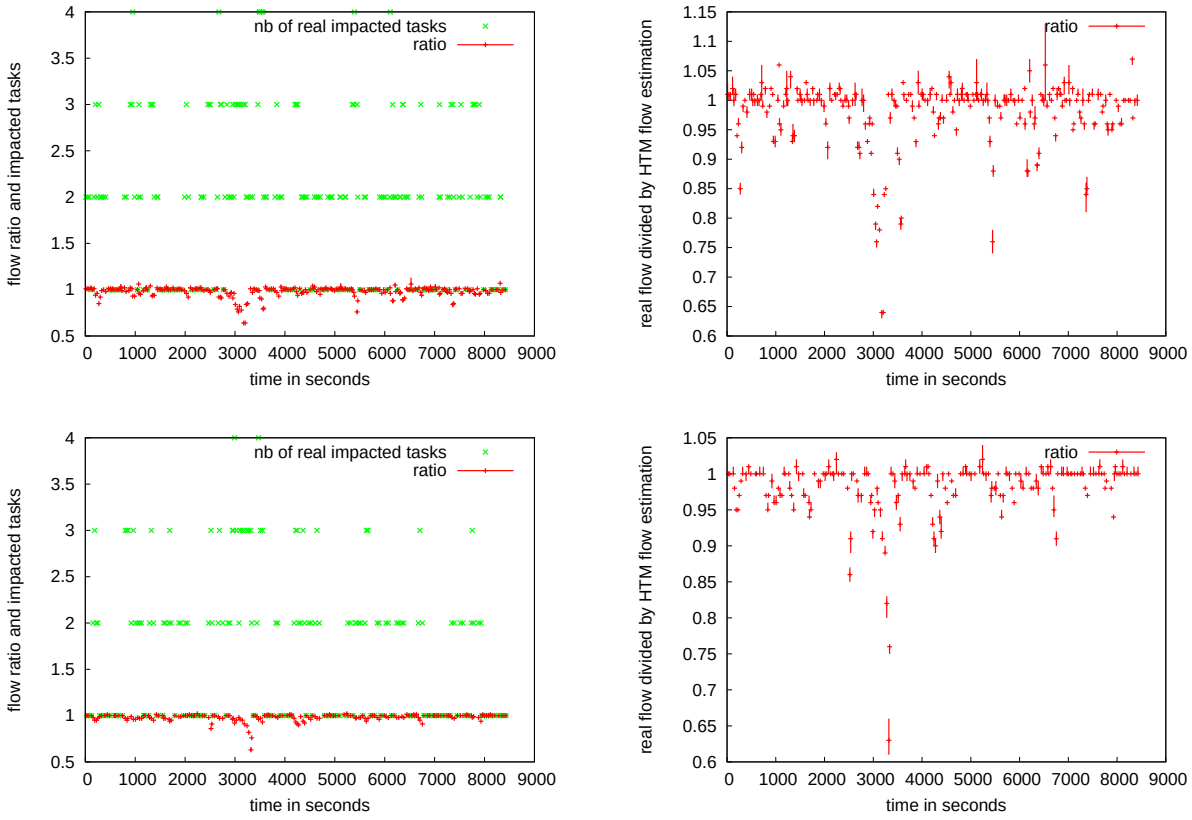

Figure 5. Accuracy of the HTM during scenario (a), $\mu = 17$ seconds, scheduled with MP, on spinnaker on the top and artimon on the bottom

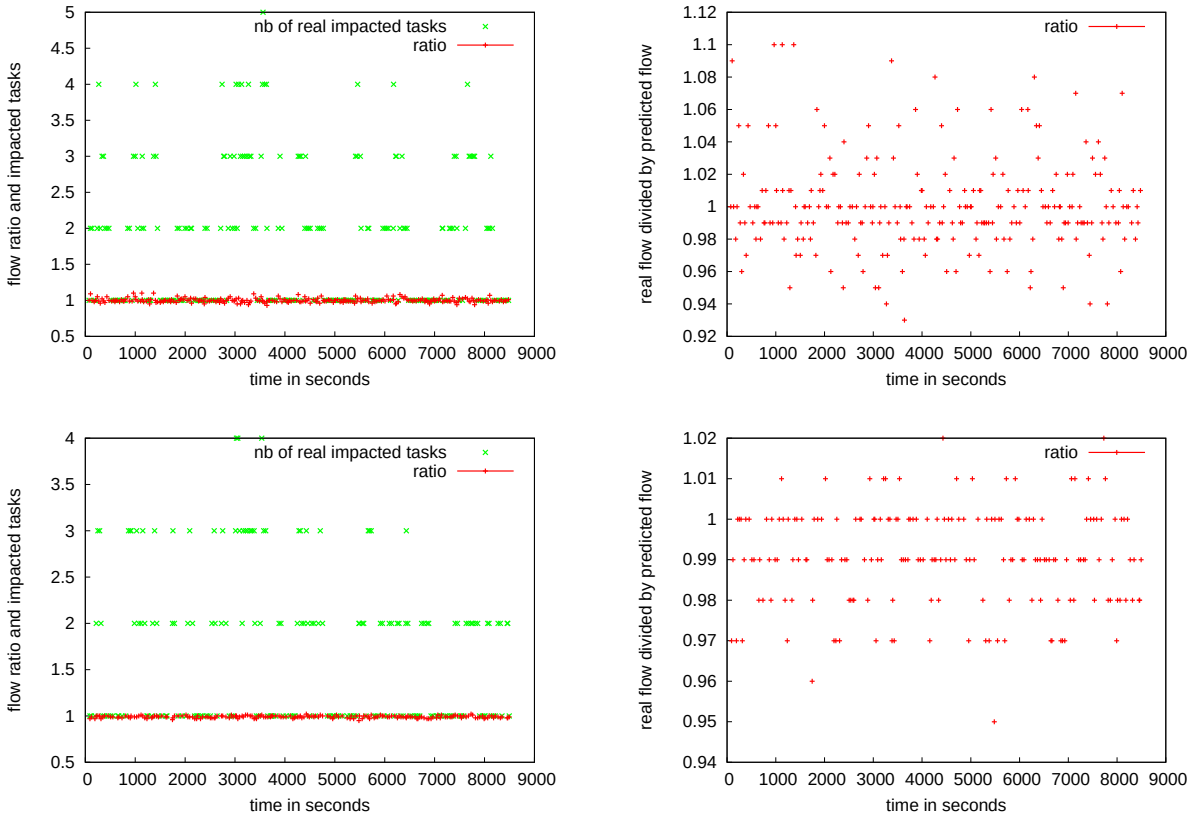

Figure 6. Accuracy of the HTM during scenario (a), $\mu = 17$ seconds, scheduled with MP, on spinnaker on the top and artimon on the bottom

5.1.3 MSF

Graphs in Figures 7 and 8 confirm all previous commentaries and their relevancy are more on the behavior of MSF: they express results of the same experiment, whose tasks are submitted at a rate equal to $\mu = 20$ seconds, submitted to a HTM that has not and has synchronization mechanisms. In the last scheme, the HTM has an average percentage error of 2.19% with a standard deviation of 2.10% on spinnaker, and 1.67% and 1.14% respectively on artimon.

Tables 4 and 5 show that MSF has at a rate $\mu = 15$ seconds information that have half the percentage of error that it had before at a rate $\mu = 17$ seconds !

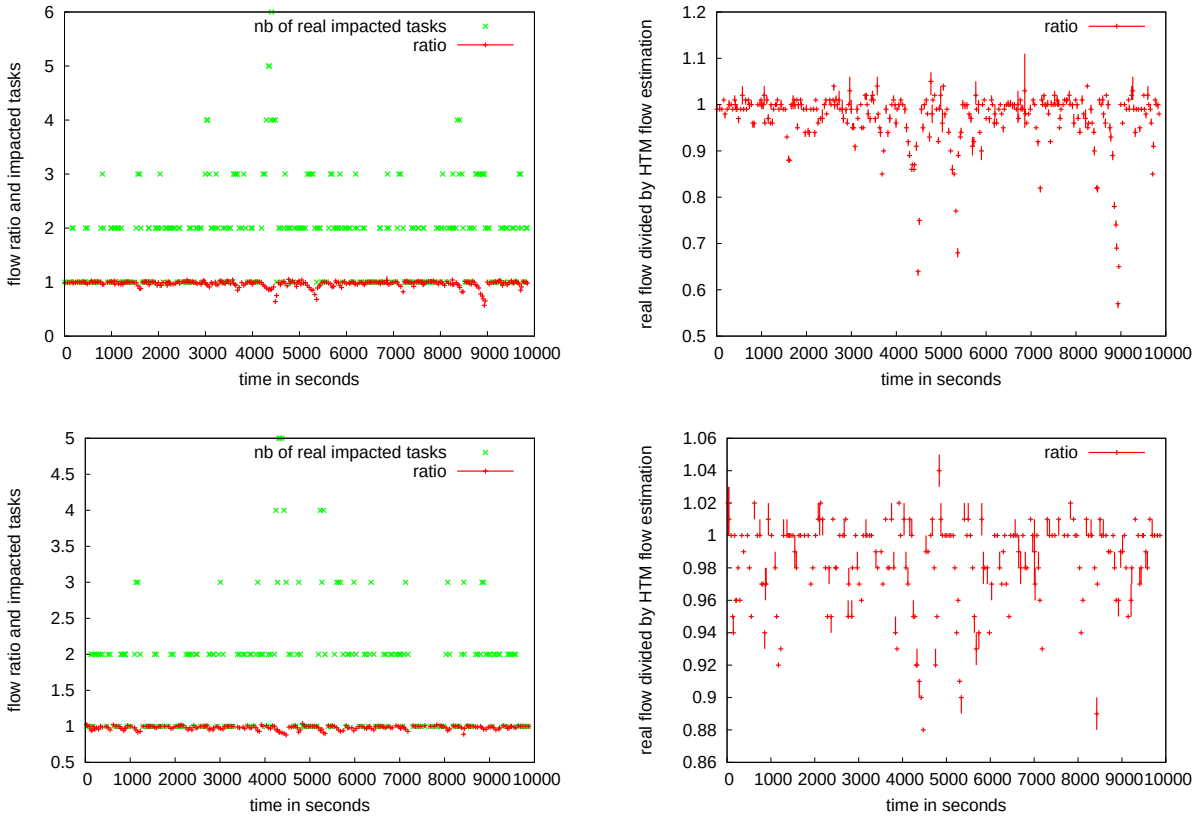

Figure 7. Accuracy of the HTM during scenario (a) scheduled with MSF, on spinnaker on the top and artimon on the bottom

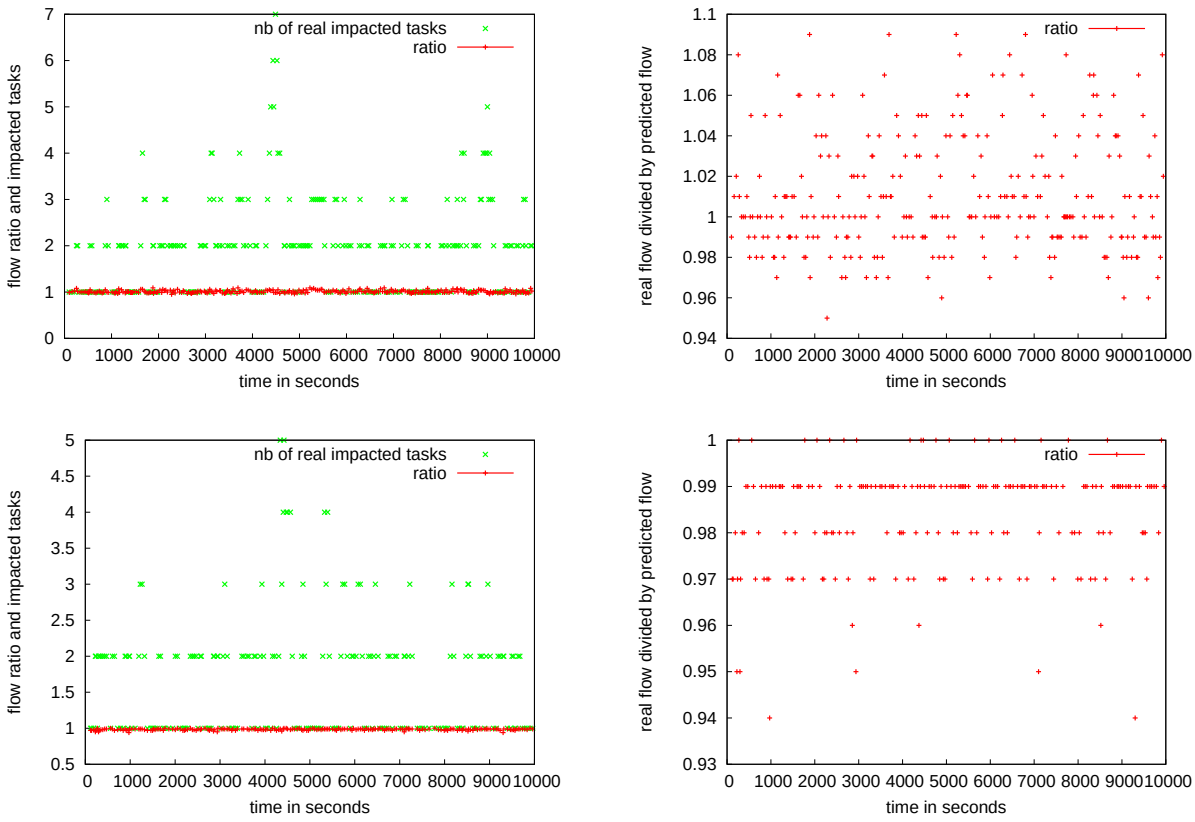

Figure 8. Accuracy of the HTM during scenario (a) scheduled with MSF, on spinnaker on the top and artimon on the bottom

5.1.4 Summary

As we could expect, the HTM provides now much better quality estimations than it used to be. The average percentage error is lower even though the number of tasks that interfer with one another is greater. This last issue also shows that the heuristics use now the resources at nearly their real capacity.

Indeed, one can see in Figure 9 that a high rate can decrease the HTM accuracy because of the too many interferences (in fact, one must also note that this experiment is one of the rare cases where HMCT has been able to handle the throughput of the jobs at that rate). This trend that we can also note in the next section will be explained later. One must note that the HTM always underestimates the ability of the server to execute the tasks. As it is plainly presented in Graphs 8 and 4 as well as in Table 5, this is not *a priori* the case anymore and the average error like the standard deviation is very small, even at a high rate.

Despite of a lesser precision, our heuristics gave really good results. It is well known that a better accuracy leads to better scheduling: we will see further if they achieve any gain in comparision to the results they used to perform.

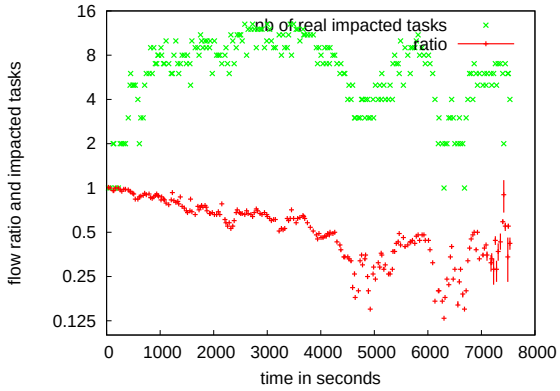

Figure 9. Scenario (a), $\mu = 15$ seconds, tasks scheduled by HMCT on artimon

	HMCT				MP				MSF			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	3.96 (5.40)	4.21 (6.57)	2.97 (3.91)	**3.71**	1.21 (1.84)	1.03 (1.32)	1.01 (1.59)	**1.08**	3.00 (3.82)	3.70 (6.05)	2.35 (2.72)	**3.02**
artimon	1.59 (2.10)	2.12 (2.76)	1.53 (2.35)	**1.75**	0.81 (1.37)	0.82 (1.29)	0.60 (0.94)	**0.74**	1.85 (2.26)	2.14 (2.43)	1.61 (2.15)	**1.87**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.25 (0.66)	0.14 (0.21)	0.15 (0.26)	**0.18**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.11 (0.16)	0.06 (0.09)	0.25 (0.14)	**0.14**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**

	HMCT					MP					MSF				
	$seed_1$	$seed_2$	$seed_3$	$seed_4$	**Avg**	$seed_1$	$seed_2$	$seed_3$	$seed_4$	**Avg**	$seed_1$	$seed_2$	$seed_3$	$seed_4$	**Avg**
spinnaker	15.45 (15.3)	19.90 (20.6)	9.03 (10.6)	11.22 (12.0)	**13.9**	3.40 (5.4)	3.43 (5.8)	2.31 (4.2)	3.18 (6.0)	**3.1**	14.30 (15.4)	15.86 (18.2)	7.95 (10.5)	8.52 (9.5)	**11.7**
artimon	9.49 (10.6)	17.64 (18.8)	6.57 (8.5)	5.74 (6.4)	**9.9**	2.19 (4.0)	2.49 (3.0)	1.76 (2.6)	7.83 (24.8)	**3.6**	9.34 (10.4)	11.92 (13.9)	7.02 (9.1)	6.82 (7.7)	**8.8**
soyotte	0.00 (0.0)	0.00 (0.0)	0.00 (0.00)	0.00 (0.00)	**0.0**	0.26 (0.2)	0.25 (0.4)	0.23 (0.3)	0.22 (0.2)	**0.2**	0.00 (0.0)	1.27 (0.0)	0.00 (0.0)	0.00 (0.0)	**0.3**
fonck	0.00 (0.0)	0.00 (0.0)	0.00 (0.0)	0.00 (0.0)	**0.0**	0.31 (0.3)	0.24 (0.3)	0.23 (0.3)	0.28 (0.3)	**0.3**	0.75 (0.9)	0.72 (0.8)	1.67 (0.0)	0.00 (0.0)	**0.8**

Table 4. Scenario (a), $\mu = 20$ and $\mu = 17$ seconds: percentage of error

	HMCT				MP				MSF			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	3.87 (3.58)	3.05 (3.05)	2.63 (2.82)	**3.18**	1.93 (2.92)	1.63 (1.89)	2.68 (2.63)	**2.08**	2.74 (2.81)	2.64 (3.73)	3.50 (3.60)	**2.96**
artimon	1.63 (1.02)	1.70 (1.16)	1.69 (1.06)	**1.67**	1.74 (2.15)	1.62 (1.07)	1.70 (1.08)	**1.69**	1.53 (1.12)	1.79 (1.09)	1.65 (1.00)	**1.66**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.17 (0.14)	0.20 (0.26)	0.00 (0.00)	**0.12**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.23 (0.25)	0.17 (0.18)	0.00 (0.00)	**0.13**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**

	AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	2.98 (3.04)	2.75 (2.97)	2.43 (2.32)	**2.72**	2.49 (2.46)	2.18 (2.20)	0.00 (0.00)	**1.56**
artimon	1.71 (1.16)	1.71 (1.04)	1.60 (0.98)	**1.67**	1.50 (0.99)	1.61 (0.91)	0.00 (0.00)	**1.04**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	1.16 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.39**

	HMCT				MP				MSF			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	2.75 (2.11)	3.09 (2.45)	2.66 (2.12)	**2.83**	2.04 (2.07)	2.09 (1.85)	2.08 (1.82)	**2.07**	2.81 (2.38)	3.06 (2.30)	2.69 (1.92)	**2.85**
artimon	1.01 (0.90)	1.45 (2.03)	1.31 (1.03)	**1.26**	1.13 (0.95)	1.11 (0.90)	1.25 (0.93)	**1.16**	2.26 (4.57)	1.14 (0.99)	1.23 (0.98)	**1.54**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.25 (0.61)	0.22 (0.37)	0.24 (0.45)	**0.24**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.18 (0.18)	0.15 (0.06)	0.24 (0.30)	**0.19**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**

	AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	2.82 (2.30)	2.92 (2.23)	2.56 (1.85)	**2.77**	2.67 (2.08)	2.87 (2.37)	2.43 (1.92)	**2.66**
artimon	1.36 (2.03)	1.43 (1.84)	1.30 (0.93)	**1.36**	1.15 (1.02)	1.09 (0.86)	1.23 (0.95)	**1.16**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.96 (nan)	0.98 (0.07)	0.96 (0.03)	**0.97**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.34 (0.43)	0.34 (0.41)	1.15 (1.70)	**0.61**

	MP		MSF		ML	
	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**
spinnaker	2.76 (2.18)	**2.76**	6.50 (6.02)	**6.50**	3.06 (2.01)	**3.06**
artimon	0.98 (0.80)	**0.98**	2.68 (4.20)	**2.68**	1.13 (1.04)	**1.13**
soyotte	0.34 (0.49)	**0.34**	0.36 (0.46)	**0.36**	0.42 (0.35)	**0.42**
fonck	0.39 (0.49)	**0.39**	0.63 (0.78)	**0.63**	0.36 (0.48)	**0.36**

Table 5. Scenario (a'), $\mu = 20$**,** $\mu = 17$ **and** $\mu = 15$ **seconds: percentage of error**

5.2. Scenarios Involving the Submission of DAGS

We have found that problems could arise if DAGS were involved in the submission: indeed, Figure 10 shows the results observed for an experiment consisting in the submission of a 1D-mesh application by six different cients in parallel with a log scale. This is a case where the agent faces a low/medium task rate but the nature itself of the application conducts to some errors.

We explain in Figure 11 why there exists some cases where the precision of the HTM can degrade with perturbations on the server. On can observe the submission of four tasks to a server. On the bottom where the real execution is depicted, we can see that there is no perturbation on the server because tasks begin to run after the completion of the previously assigned task. Nevertheless, on the top, we see that without synchronization of the HTM to the reality, the task 1 finishes at a date computed with the average duration of the task (obtained from benchmark). In consequence, the HTM notes a perturbation between task 1 and task 2, thus delays the completion time of both tasks. There are two consequences: the error observed on the ratio of the flow of both tasks is slightly worst. Of course, if this schema repeats itself, like for task 2 and task 3 and then task 3 and task 4, the error grows. The synchronization mechanisms are implanted to vouch for those cases to be correctly undertaken and consequently for the accuracy quality of the estimations to be constant over time and load.

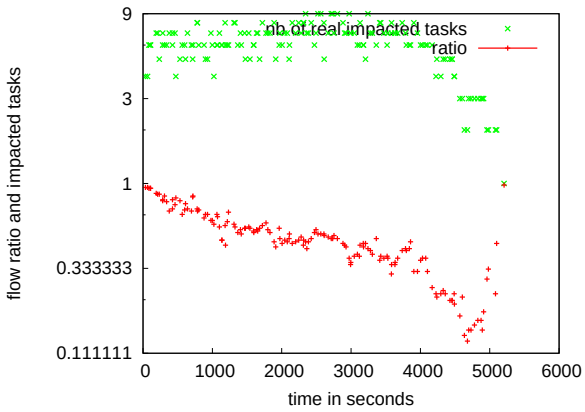

Figure 10. Ratio of the HTM estimated duration and the real post-mortem duration on a 6 1D-mesh applications submission scheduled with HMCT

We present in Figures 12, 13, 14, and 15 the accuracy of the HTM estimations during the execution of some experiments instantiating a scenario of Table 3, scheduled by different heuristics. The leftmost graphs are in a log 2 scale to better render the variation of the ratio. These let understand the variation of the accuracy of a prediction in fonction of the interfered tasks. Tables 6, 7, 8 and 9 give the corresponding average percentage of error done on each task flow estimation as well as the standard deviation, per heuristic and servers.

We can observe from Figure 12 that the error grows as the number of tasks that interfere with one another increases. Indeed, the zoom plot of both spinnaker and artimon presents a curve that has the inverse behavior than the number of impacted tasks: the servers behave better than the time-shared model tells. The errors grows to a maximum of 23% for a task that has suffered the interference of 18 tasks during its execution.

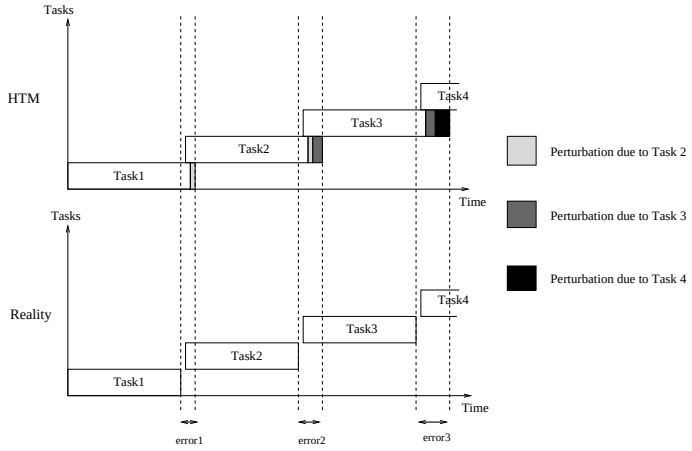

Figure 11. Difference between simulation and reality at high rate

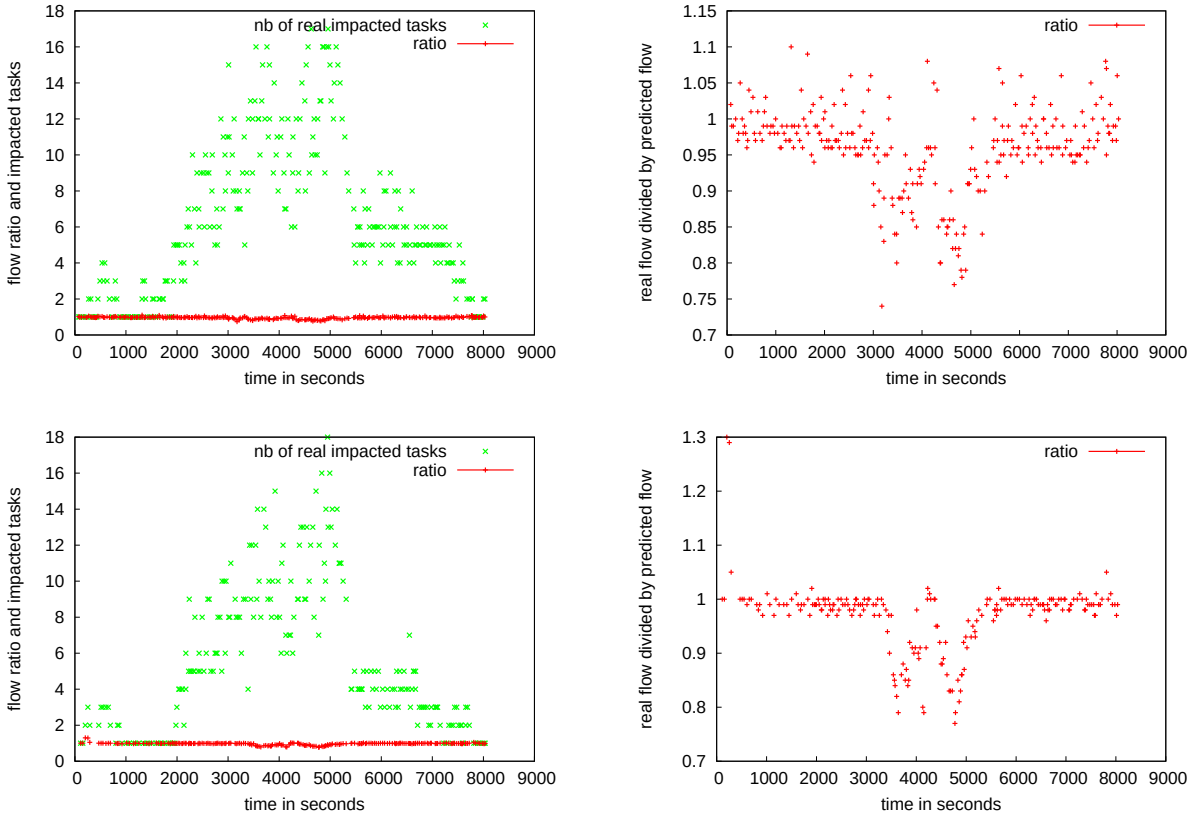

Figure 12. Accuracy of the HTM during scenario (d'), $\mu = 20$ seconds, scheduled with HMCT, on spinnaker on the top and artimon on the bottom

Although this too high number of tasks that must question the integrity of the heuristic, we can read in Table 6 that the HTM achieves a respectable 5.5% of error in average with a standard deviation equal to 5 (experiment generated with the first seed). Nonetheless, no experiments generated with the second and third seeds have correctly executed: a server, too much loaded, refused some job.

Figure 13 presents an experiment of Scenario (e'), e.g. the submission of a stencil application, scheduled by MSF. Firstly, we note that the plot of arrival dates is more discret than in any of the other experiment: tasks are submitted by groups. Secondly we note the number of tasks interfering with one another which is always very high on spinnaker and on artimon. Nonetheless, the accuracy of the HTM is great during all the experiment. In consequence, it predicts durations 96.6% accurate in average, with a standard deviation of 3.4 (see Table 7).

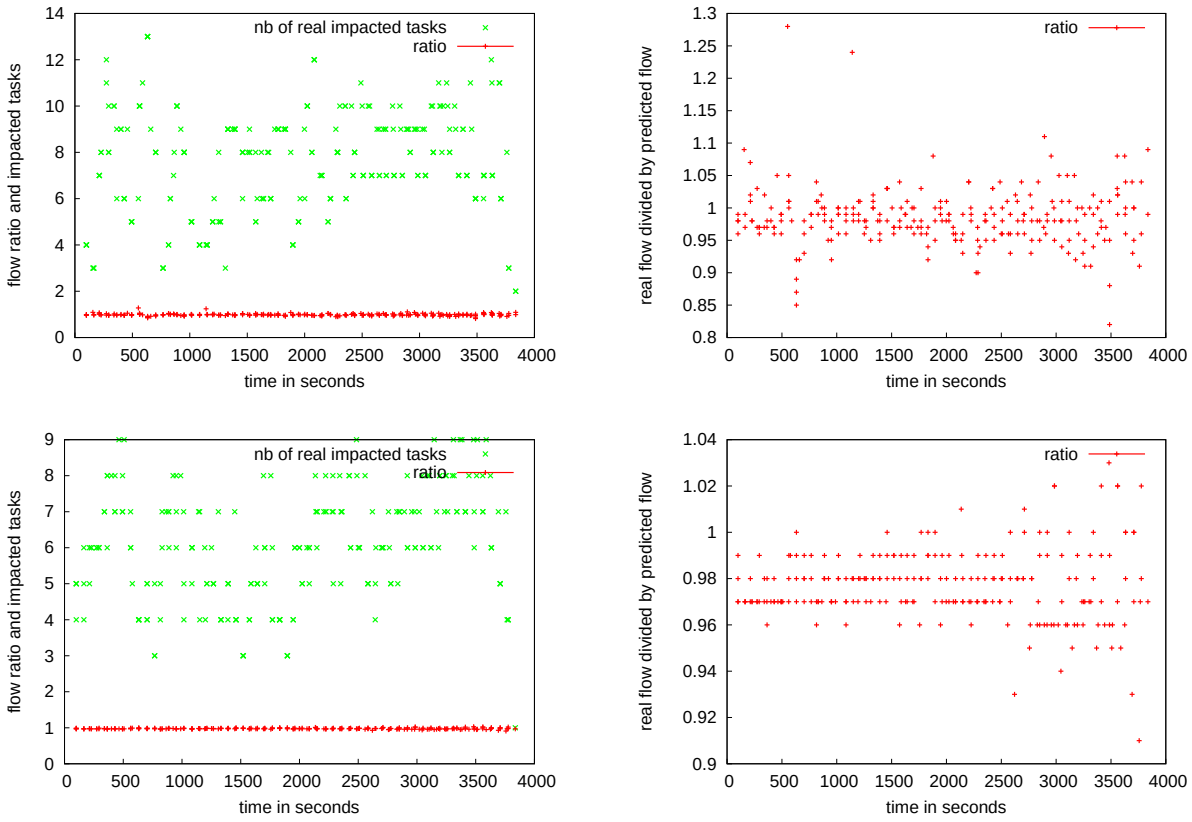

Figure 13. Accuracy of the HTM during scenario (e'), scheduled with MSF, on spinnaker on the top and artimon on the bottom

One must note that HMCT has not been able to schedule Scenario (h') generated with the first and the third seed. AHMCT didn't even achieve this performance: for both of them, spinnaker refused a request because it was overloaded. In consequence, AHMCT does not appear in Table 8.

Arrival dates are less discret than previoulsy, mainly due to the submission of independent tasks in parallel of the stencil application. One can see that even with the high number of interferences for each task, the HTM achieves a respectable accuracy on spinnaker, and a quasi-perfect one on artimon. The

average error is 4.6% with a standard deviation of 4.3 on spinnaker and only of 2% and 1.7 respectively on artimon. The agent has then a really good snapshot of the system when the scheduler computes its choice.

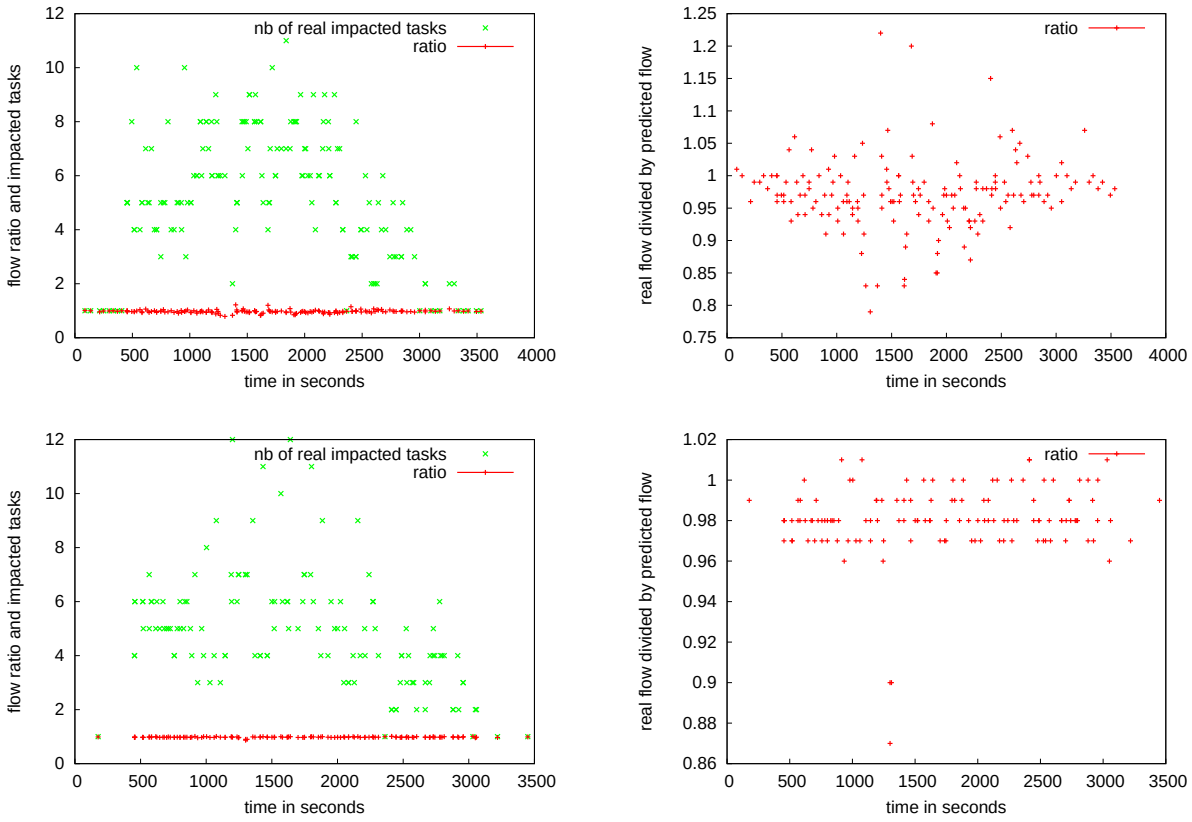

Figure 14. Accuracy of the HTM during scenario (h'), $\mu = 40$ seconds, scheduled with ML, on spinnaker on the top and artimon on the bottom

The same commentary can be done on Scenario (j') of which we give an experiment scheduled by MSF in Figure 15. The average error and the standard deviation are very small (under 2.5%).

This finishes to show that regardless the scenario, the HTM gives at any time accurate estimations of each task in the system, even when there is a high number of interferences between tasks and a high variation of instantaneous running tasks. We have shown that both events affect the accuracy. As they are heuristic dependent, the choice of the heuristic to implement in the agent must also rely on these information to be decided. Nevertheless, Tables 6, 7, 8 and 9 summarize the average error on the estimation of each task per heuristic and per server. If a mean of all the values would not have any sens, we can highlight that:
the error is the highest on spinnaker (there are more interferences due to a greater number of tasks mapped because of the heterogeneity of the platform and for performance reasons) ; this average error is usually under 3-5% which is a really good score. Moreover, if we had to order the heuristics on the basis of these tables, ML then MP then MSF would be the best ones. The next section will examine their respective performances.

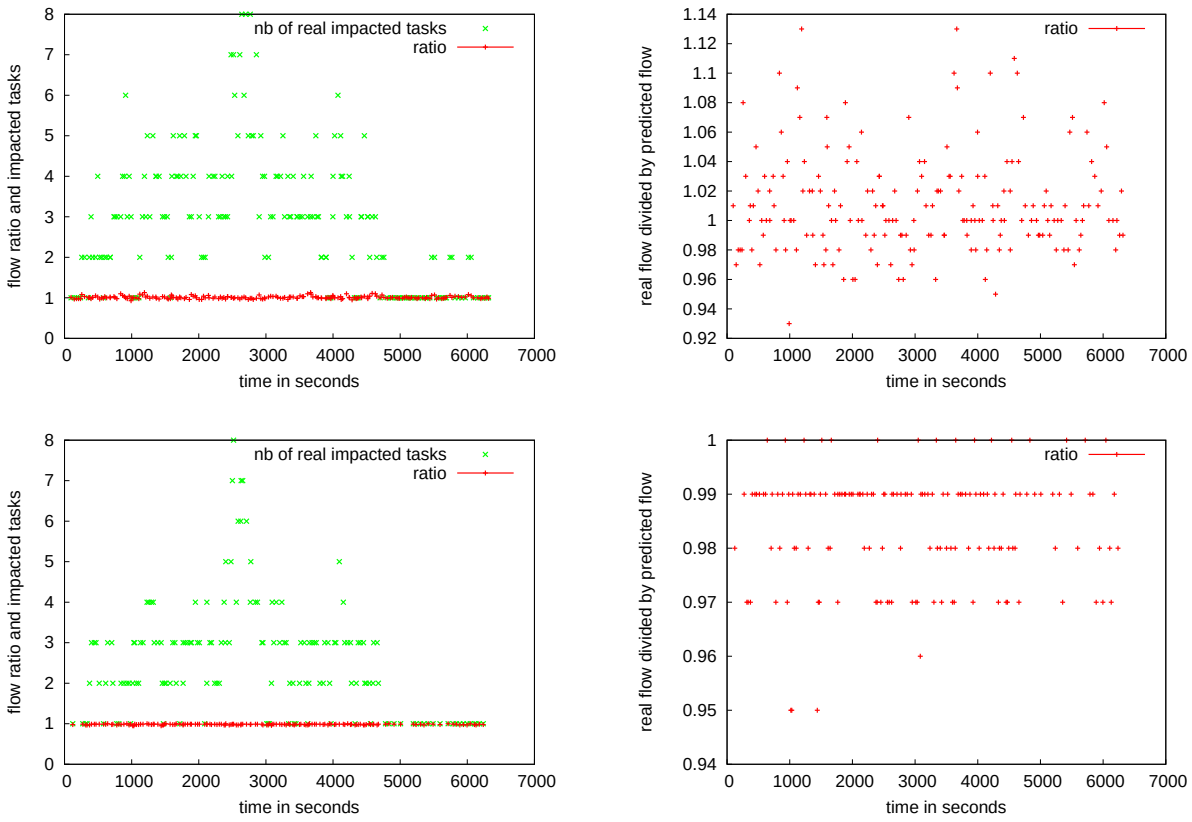

Figure 15. Accuracy of the HTM during scenario (j'), $\mu = 20$ seconds, scheduled with MSF, on spinnaker on the top and artimon on the bottom

	HMCT				MP				MSF			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	5.48 (5.06)	-	-	$<$ **5.48** $>$	3.52 (4.18)	(4.94)	3.51 (4.40)	**3.8**	5.74 (5.71)	7.47 (6.54)	7.24 (6.77)	**6.82**
artimon	3.95 (5.83)	-	-	$<$ **3.95** $>$	1.81 (3.89)	2.03 (3.14)	2.52 (3.89)	**2.12**	2.07 (3.25)	3.61 (5.37)	3.54 (4.94)	**3.07**
soyotte	0.00 (0.00)	-	-	$<$ **0** $>$	1.46 (1.66)	0.83 (0.97)	1.08 (0.98)	**1.12**	0.32 (0.42)	0.32 (0.43)	0.57 (0.61)	**0.40**
fonck	0.00 (0.00)	-	-	$<$ **0** $>$	0.21 (0.16)	0.28 (0.28)	2.12 (1.88)	**0.87**	1.02 (1.75)	0.18 (0.30)	0.53 (0.50)	**0.58**

	AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	4.77 (4.57)	7.88 (6.32)	-	$<$ **6.32** $>$	3.27 (2.57)	4.18 (4.20)	4.92 (5.21)	**4.12**
artimon	3.53 (5.11)	5.16 (6.27)	-	$<$ **4.34** $>$	1.07 (0.93)	2.11 (3.95)	2.06 (2.80)	**1.75**
soyotte	0.37 (0.5)	0.39 (0.45)	-	$<$ **0.38** $>$	0.19 (0.12)	0.54 (0.83)	0.78 (0.77)	**0.50**
fonck	0.55 (0.59)	0.66 (0.74)	-	$<$ **0.60** $>$	0.30 (0.37)	0.29 (0.44)	0.38 (0.28)	**0.32**

Table 6. Scenario (d'): percentage of error

	HMCT		MP		MSF		AHMCT		ML	
	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**
spinnaker	2.79 (3.15)	**2.79**	1.38 (1.11)	**1.38**	3.40 (3.37)	**3.40**	2.76 (4.12)	**2.76**	3.17 (3.05)	**3.17**
artimon	2.66 (2.52)	**2.66**	2.77 (0.84)	**2.77**	2.44 (1.27)	**2.44**	2.28 (2.16)	**2.28**	2.48 (1.02)	**2.48**
soyotte	0.00 (0.00)	**0.00**	0.35 (0.49)	**0.35**	0.15 (0.14)	**0.15**	0.00 (0.00)	**0.00**	0.23 (0.20)	**0.23**
fonck	0.00 (0.00)	**0.00**	0.22 (0.18)	**0.22**	0.35 (0.41)	**0.35**	0.00 (0.00)	**0.00**	0.26 (0.35)	**0.26**

Table 7. Scenario (e'): Percentage of Error

	HMCT				MP				MSF				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	-	9.32 (7.75)	-	**9.32**	4.80 (4.71)	5.57 (6.25)	6.47 (7.14)	**5.61**	10.81 (7.72)	10.70 (8.75)	11.85 (8.33)	**11.12**	4.60 (4.27)	3.86 (3.01)	4.24 (3.82)	**4.23**
artimon	-	8.19 (7.43)	-	**8.19**	2.50 (2.18)	3.06 (5.06)	3.96 (6.01)	**3.17**	7.36 (6.87)	9.49 (9.61)	7.51 (8.12)	**8.12**	2.05 (1.67)	1.85 (0.89)	2.52 (3.55)	**2.14**
soyotte	-	0.04 (0.0)	-	**0.04**	0.21 (0.31)	0.27 (0.31)	1.01 (1.06)	**0.50**	0.14 (0.07)	0.11 (0.06)	0.11 (0.07)	**0.12**	0.20 (0.15)	0.14 (0.08)	0.11 (0.06)	**0.15**
fonck	-	0.10 (0.09)	-	**0.10**	0.38 (0.39)	0.30 (0.17)	1.03 (2.12)	**0.57**	0.17 (0.15)	0.18 (0.18)	0.25 (0.16)	**0.20**	0.25 (0.31)	0.22 (0.14)	0.19 (0.13)	**0.22**

Table 8. Scenario (h'): percentage of error

	HMCT				MP				MSF			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	2.48 (3.35)	2.55 (2.83)	2.66 (2.68)	**2.56**	2.02 (2.18)	1.89 (2.27)	1.63 (1.69)	**1.85**	2.48 (2.55)	2.30 (2.31)	2.36 (2.27)	**2.38**
artimon	1.69 (1.61)	1.66 (0.98)	1.64 (1.05)	**1.66**	1.54 (1.06)	1.66 (1.06)	1.62 (1.04)	**1.61**	1.63 (1.06)	1.71 (1.25)	1.66 (1.01)	**1.67**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.25 (0.16)	0.35 (0.34)	0.32 (0.46)	**0.31**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.27 (0.49)	0.35 (0.35)	0.16 (0.11)	**0.26**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**

	AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
spinnaker	2.28 (2.06)	2.54 (2.79)	2.97 (3.22)	**2.60**	2.16 (1.90)	2.06 (2.32)	2.15 (2.17)	**2.12**
artimon	1.66 (1.02)	1.71 (1.04)	1.43 (0.98)	**1.60**	1.58 (1.04)	1.65 (1.00)	1.65 (1.05)	**1.63**
soyotte	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.58 (0.40)	0.00 (0.00)	0.00 (0.00)	**0.19**
fonck	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.00**	0.70 (0.68)	0.62 (0.68)	0.00 (0.00)	**0.44**

Table 9. Scenario (j): percentage of Error

6. Heuristics Performances: Results

We invite the reader to refer to the research report [CJ04] where the same kind of study has been presented. He may find more detailed information about how to read the results as well as the performances of HMCT, MP and MSF on scenarios given Table 3. There is nevertheless a difference here: graphs giving the sumflow can be given in a log scale in order to point clearly at the difference of performances between heuristics when differences are great: note that if so, it is mentionned in the section.

We compare next the performances of each heuristics relying on the HTM which has now some synchronization mechanisms to the instiation of MCT that can be found in NetSolve. They are examined in each subsection for each scenario.

6.1. Scenario (a')

HMCT and AHMCT have not been able to handle the experiment corresponding to the rate $\mu = 15$ seconds. At this rate, they aim to map too many tasks on spinnaker at a time which refuses the job. These two heuristics are ignored for this rate.

We present in Tables 10, 11 and 12 the utilization of the resources for each heuristic for the submission of independent tasks at rates $\mu = 15$, $\mu = 17$ and $\mu = 20$ seconds respectively. The percentage of task is given as well as the percentage corresponding to each type of task in parenthesis, and the sumflow, sum of the durations of all tasks, per server.

Tables 13, 14 and 15 gives the average gain that each task perceives when the agent uses a considered heuristic for $\mu = 15$, $\mu = 17$ and $\mu = 20$ seconds. Finally, a summary of all the results concerning independent tasks (that is here all the tasks) can be found in Tables 16, 17 and 18.

Results are the average done on respectively one, two and two runs per rate for each of the three experiments. In all, the results in this section required more than $12 * 6 = 72$ runs.

Our heuristics benefit from more precise information due to the use of the HTM (see Section 5). They take into consideration the perturbation that tasks have on each other. Thus, they clearly outperform MCT as it is shown in all the tables. This tendancy becomes more pronounced as the rate raises (e.g. μ decreases). Indeed, when $\mu = 15$ seconds, MP and ML achieve a 50% on the sumflow, on the average gain that each task can benefit from this scheduling, and 88% of the tasks finish sooner than if scheduled by MCT.

When $\mu = 17$ seconds, ML seems the best heuristics, providing average gains around 30% on the duration of each task, a lower sumflow, 72% of tasks finishing sooner than MCT, with 18% finishing at the same date (in a range of 2 seconds) (see Table 17).

Finally when $\mu = 20$ seconds, we can see that MSF and ML are very tight and give the best results regardless the metric. We must also note that MP performs very poorly at this rate mainly because it chooses unnecessarily a slower server than it could use.

experiment 1								
server	MCT		MP		MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	52.0 (16.4 14.8 20.8)	42312	48.4 (16.2 14.6 17.6)	11641	52.2 (15.8 16.6 19.8)	28257	49.6 (14.2 15.0 20.4)	17411
artimon	46.6 (15.6 14.6 16.4)	40228	39.6 (12.0 11.0 16.6)	11087	43.6 (12.6 13.0 18.0)	27033	43.4 (12.6 13.6 17.2)	16661
soyotte	0.6 (0.4 0.0 0.2)	642	6.0 (2.2 2.0 1.8)	7989	1.6 (1.6 0.0 0.0)	1023	3.0 (2.6 0.2 0.2)	2363
fonck	0.8 (0.2 0.2 0.4)	1149	6.0 (2.2 2.0 1.8)	7892	2.6 (2.6 0.0 0.0)	1661	4.0 (3.2 0.8 0.0)	3175
total sumflow		84331		38609		57974		39610

Table 10. Scenario (a'), $\mu = 15$ seconds: processors utilization

experiment 1

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.0 (19.0 17.0 20.0)	20527	56.0 (18.8 16.0 21.2)	14971	50.0 (17.4 14.2 18.4)	8689
artimon	44.0 (13.6 12.6 17.8)	16235	44.0 (13.8 13.6 16.6)	12076	38.2 (11.4 11.2 15.6)	7611
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.0 (2.2 2.0 1.8)	7533
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.8 (1.6 2.2 2.0)	7746
total sumflow		36762		27047		31579

experiment 2

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	54.8 (16.9 18.2 19.7)	20242	55.8 (16.0 21.4 18.4)	16730	50.4 (15.8 18.6 16.0)	8472
artimon	45.2 (13.7 17.0 14.5)	17863	44.2 (14.6 13.8 15.8)	13801	38.0 (11.4 12.4 14.2)	7833
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.0 (2.2 1.6 2.2)	7815
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.6 (1.2 2.6 1.8)	7589
total sumflow		38105		30531		31709

experiment 3

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	57.2 (20.9 19.8 16.5)	17342	55.4 (18.6 18.8 18.0)	12442	49.2 (15.8 17.8 15.6)	8104
artimon	42.8 (15.1 14.2 13.5)	13929	44.6 (17.4 15.2 12.0)	10113	39.2 (15.6 12.0 11.6)	6797
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.4 (2.0 1.8 1.6)	6680
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.2 (2.6 2.4 1.2)	7060
total sumflow		31271		22555		28641

MEAN

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.0 (18.9 18.3 18.8)	19370	55.7 (17.8 18.7 19.2)	14714	49.9 (16.3 16.9 16.7)	8422
artimon	44.0 (14.2 14.6 15.2)	16009	44.3 (15.3 14.2 14.8)	11997	38.5 (12.8 11.9 13.8)	7414
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.8 (2.1 1.8 1.9)	7343
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.9 (1.8 2.4 1.7)	7465
total sumflow		35379		26711		30643

experiment 1

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.4 (18.6 15.6 21.2)	14698	55.2 (17.2 16.8 21.2)	15111	54.8 (17.6 16.8 20.4)	12848
artimon	44.6 (14.0 14.0 16.6)	12613	44.8 (15.4 12.8 16.6)	12192	43.6 (13.4 12.8 17.4)	11293
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.2 (0.2 0.0 0.0)	129
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	1.4 (1.4 0.0 0.0)	893
total sumflow		27311		27303		25163

experiment 2

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.0 (17.2 20.4 18.4)	16331	55.2 (14.6 22.2 18.4)	16630	53.8 (15.0 19.8 19.0)	12878
artimon	44.0 (13.4 14.8 15.8)	14025	44.8 (16.0 13.0 15.8)	13887	44.2 (14.0 15.0 15.2)	11424
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.4 (0.4 0.0 0.0)	259
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	1.6 (1.2 0.4 0.0)	1274
total sumflow		30356		30517		25835

experiment 3

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.0 (19.0 18.8 17.2)	11975	55.4 (18.6 19.0 17.8)	12356	53.6 (17.6 19.4 16.6)	10896
artimon	45.0 (17.0 15.2 12.8)	10390	44.6 (17.4 15.0 12.2)	10302	44.8 (17.0 14.4 13.4)	9764
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.4 (0.4 0.0 0.0)	258
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	1.2 (1.0 0.2 0.0)	899
total sumflow		22365		22658		21817

MEAN

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.5 (18.3 18.3 18.9)	14335	55.3 (16.8 19.3 19.1)	14699	54.1 (16.7 18.7 18.7)	12207
artimon	44.5 (14.8 14.7 15.1)	12343	44.7 (16.3 13.6 14.9)	12127	44.2 (14.8 14.1 15.3)	10827
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.3 (0.3 0.0 0.0)	215
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	1.4 (1.2 0.2 0.0)	1022
total sumflow		26677		26826		24272

Table 11. Scenario (a'), $\mu = 17$ seconds: processors utilization

experiment 1

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.9 (18.7 16.5 21.7)	12463	59.5 (19.0 17.3 23.2)	10736	52.2 (16.8 15.5 19.9)	8138
artimon	43.1 (13.9 13.1 16.1)	9447	40.5 (13.6 12.3 14.6)	7254	35.4 (11.0 10.1 14.3)	6155
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.6 (2.0 1.8 1.8)	7018
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.8 (2.8 2.2 1.8)	8022
total sumflow		21910		17990		29333

experiment 2

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	57.3 (17.9 19.4 20.0)	13515	58.7 (17.6 21.1 20.0)	10703	52.4 (16.0 19.4 17.0)	8022
artimon	42.7 (12.7 15.8 14.2)	10662	41.3 (13.0 14.1 14.2)	7773	36.8 (11.4 12.4 13.0)	6152
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.0 (1.4 2.0 1.6)	6497
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.8 (1.8 1.4 2.6)	7887
total sumflow		24177		18476		28558

experiment 3

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.5 (20.0 19.4 17.1)	17586	55.2 (18.5 18.6 18.1)	12638	49.1 (16.0 17.3 15.8)	8141
artimon	43.5 (16.0 14.6 12.9)	13954	44.8 (17.5 15.4 11.9)	10154	39.2 (15.3 12.5 11.4)	6815
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.5 (2.1 1.8 1.6)	6776
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.2 (2.6 2.4 1.2)	7072
total sumflow		31540		22792		28804

MEAN

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.8 (18.6 18.5 18.6)	19868	55.6 (17.8 18.5 19.3)	15120	49.8 (16.4 16.6 16.8)	8483
artimon	44.2 (14.4 14.4 15.4)	16301	44.4 (15.3 14.4 14.7)	12172	38.5 (12.7 12.1 13.7)	7406
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.8 (2.2 1.9 1.8)	7327
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.9 (1.8 2.4 1.7)	7488
total sumflow		36169		27292		30704

experiment 1

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	57.9 (18.8 16.3 22.8)	10099	59.4 (18.8 17.2 23.4)	10709	57.5 (19.4 15.2 22.9)	9829
artimon	42.1 (13.8 13.3 15.0)	7669	40.6 (13.8 12.4 14.4)	7338	42.5 (13.2 14.4 14.9)	7891
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
total sumflow		17768		18047		17720

experiment 2

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	58.5 (18.5 21.2 18.8)	10336	58.2 (16.8 21.2 20.2)	10634	57.3 (17.8 20.1 19.4)	10076
artimon	41.5 (12.1 14.0 15.4)	8033	41.8 (13.8 14.0 14.0)	7925	42.5 (12.6 15.1 14.8)	8143
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.2 (0.2 0.0 0.0)	129
total sumflow		18369		18559		18348

experiment 3

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	54.5 (18.3 19.0 17.2)	12136	55.1 (18.3 18.9 17.9)	12605	53.6 (17.7 19.1 16.8)	11046
artimon	45.4 (17.6 15.0 12.8)	10406	44.9 (17.7 15.1 12.1)	10336	44.9 (17.0 14.7 13.2)	9824
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.4 (0.4 0.0 0.0)	258
fonck	0.1 (0.1 0.0 0.0)	64	0.0 (0.0 0.0 0.0)	0	1.1 (0.9 0.2 0.0)	833
total sumflow		22606		22941		21961

MEAN

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.2 (18.1 18.3 18.9)	14596	54.9 (16.7 18.9 19.4)	15089	53.7 (16.4 18.5 18.7)	12344
artimon	44.7 (14.9 14.7 15.1)	12489	45.1 (16.4 14.1 14.6)	12475	44.6 (15.1 14.2 15.3)	10965
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.4 (0.4 0.0 0.0)	258
fonck	0.1 (0.1 0.0 0.0)	43	0.0 (0.0 0.0 0.0)	0	1.3 (1.1 0.2 0.0)	979
total sumflow		27128		27565		24546

Table 12. Scenario (a'), $\mu = 20$ seconds: processors utilization

We note the increase in resource consumption with the rate for heuristics focusing on the fast servers. Interferences between tasks allocated on these servers grow thus the corresponding sumflow.

What seems remarquable is that AHMCT and HMCT give slighlty the same results, HMCT being quiet better. We expected the opposite in regard to their respective design. MP gives very good results when the throughput of jobs is high, but performs poorly when μ increases. We can conclude that among the three rates, ML gives in average the best results over all heuristics.

		mean flow MCT (sec)	gain in percentage MP	MSF	ML
	type 1	85.6	50.2	24.8	39.6
experiment 1	type 2	169.4	51.3	33.2	53.1
	type 3	239.7	57.1	32.1	57.1
MEAN	-	164.9	52.9	30.0	49.9

Table 13. Scenario (a'), $\mu = 15$ seconds: average percentage gain on each task given by type

		mean flow MCT (sec)	gain in percentage HMCT	MP	MSF	AHMCT	ML
	type 1	35.7	20.9	8.7	19.8	18.9	17.8
experiment 1	type 2	71.6	26.1	6.6	25.7	24.7	33.0
	type 3	107.6	28.2	19.6	27.4	28.2	34.8
	type 1	36.3	22.8	14.4	23.5	21.2	20.5
experiment 2	type 2	73.5	17.5	15.0	17.7	17.4	31.7
	type 3	114.7	20.6	18.7	21.2	21.2	35.9
	type 1	31.9	22.2	-1.4	23.5	21.6	19.2
experiment 3	type 2	59.6	27.7	-1.8	28.3	27.3	29.5
	type 3	102.6	30.1	18.8	30.5	29.9	34.8
MEAN	-	70.4	24.0	11.0	24.2	23.4	28.6

Table 14. Scenario (a'), $\mu = 17$ seconds: average percentage gain on each task given by type

		mean flow MCT (sec)	gain in percentage HMCT	MP	MSF	AHMCT	ML
	type 1	22.1	14.8	-48.6	16.3	14.1	16.6
experiment 1	type 2	42.6	19.6	-44.0	21.5	19.0	21.4
	type 3	63.5	17.9	-24.2	18.3	18.0	18.7
	type 1	23.2	21.7	-18.5	22.7	21.2	20.8
experiment 2	type 2	47.5	23.6	-11.3	24.1	23.4	25.1
	type 3	71.7	24.2	-22.7	24.4	23.7	24.4
	type 1	22.1	14.7	-48.5	16.3	14.1	16.5
experiment 3	type 2	42.7	19.5	-43.8	21.4	19.0	21.4
	type 3	63.7	17.9	-24.1	18.3	17.9	18.7
MEAN	-	44.4	19.3	-31.7	20.4	18.9	20.4

Table 15. Scenario (a'), $\mu = 20$ seconds: average percentage gain on each task given by type

	NetSolve's MCT		MP		MSF		ML	
	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**	$seed_1$	**Avg**
makespan	7713	**7713**	7922	**7922**	7648	**7648**	7634	**7634**
sumflow	84331	**84331**	38609	**38609**	57974	**57974**	39610	**39610**
maxflow	415.9	**415.9**	425.3	**425.3**	312.1	**312.1**	383.7	**383.7**
maxstretch	10.0	**10.0**	11.9	**11.9**	8.8	**8.8**	10.3	**10.3**
number of tasks that finish sooner than with NetSolve's MCT	-	–	86 (13)	**86(13)**	88 (10)	**88(10)**	88 (11)	**88(11)**

Table 16. Scenario (a'), $\mu = 15$ seconds: results in seconds

	NetSolve's MCT				HMCT				MP			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	8525	8513	8557	**8532**	8501	8504	8518	**8508**	8831	8821	8825	**8826**
sumflow	36762	38105	31272	**35380**	27047	30531	22555	**26711**	31579	31709	28641	**30643**
maxflow	257.6	256.0	271.3	**261.7**	192.1	206.3	200.6	**199.7**	408.3	412.8	393.6	**404.9**
maxstretch	6.6	7.0	7.3	**7.0**	5.0	5.4	5.3	**5.3**	10.0	10.1	9.4	**9.8**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	72 (21)	68 (23)	69 (19)	**70(21)**	75 (18)	75 (16)	69 (18)	**73(17)**

	MSF				AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	8501	8505	8518	**8508**	8501	8504	8517	**8507**	8501	8505	8517	**8507**
sumflow	27311	30356	22364	**26677**	27303	30517	22657	**26826**	25163	25835	21817	**24272**
maxflow	196.3	202.8	199.2	**199.4**	199.3	202.4	224.3	**208.7**	154.2	254.2	254.0	**220.8**
maxstretch	5.2	5.0	5.3	**5.2**	5.1	5.8	5.9	**5.6**	8.7	8.8	9.0	**8.9**
number of tasks that finish sooner than with NetSolve's MCT	71 (22)	68 (21)	69 (19)	**69(21)**	70 (22)	68 (23)	68 (20)	**69(22)**	73 (19)	74 (16)	69 (19)	**72(18)**

Table 17. Scenario (a'), $\mu = 17$ seconds: results in seconds

	NetSolve's MCT				HMCT				MP			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	10134	9934	10134	**10067**	10120	9916	10120	**10052**	10456	10233	10456	**10381**
sumflow	21910	24176	21910	**22665**	17990	18475	17990	**18152**	29334	28558	29334	**29075**
maxflow	136.4	163.5	136.4	**145.4**	78.6	127.8	78.6	**95.0**	387.9	383.3	387.9	**386.3**
maxstretch	3.6	4.2	3.6	**3.8**	2.5	3.2	2.5	**2.7**	9.1	9.0	9.1	**9.0**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	57 (25)	60 (21)	57 (25)	**58(24)**	57 (21)	63 (18)	57 (21)	**59(20)**

	MSF				AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	10119	9916	10119	**10052**	10120	9916	10120	**10052**	10119	9916	10119	**10052**
sumflow	17768	18369	17768	**17968**	18048	18559	18048	**18218**	17720	18348	17720	**17929**
maxflow	75.5	122.0	75.5	**91.0**	74.2	128.0	74.2	**92.1**	72.2	128.9	72.2	**91.1**
maxstretch	2.3	3.0	2.3	**2.5**	2.5	3.5	2.5	**2.8**	2.2	8.7	2.2	**4.4**
number of tasks that finish sooner than with NetSolve's MCT	59 (22)	62 (19)	59 (22)	**60(21)**	58 (25)	60 (22)	58 (25)	**59(24)**	58 (20)	61 (18)	58 (20)	**59(19)**

Table 18. Scenario (a'), $\mu = 20$ seconds: results in seconds

6.2. Scenario (d')

The exact composition of this scenario is given in Section 5.5 of the research report [CJ04]. Most of the presentation of the following results is identical.

There are several things to look at in this section: results for each of the 5 clients and of the metatask on metrics like the makespan, the sumflow, but we can also consider the quality of service given to each task of the metatask. Then, one can see the average results of the 4 subsets of experiment, on two runs each, resumed in Figures 16, 17, 18 and in Tables 19, 21, 20 and 22.

Note that the results for both HMCT and AHMCT do not appear in the graphs nor in the tables. During the numerous attempts, a task is refused at a given moment by the server that would have normally execute the task according to the scheduling decision. We do not deal with that kind of situation in this report like we have mentionned in Section 2 then the HTM simply aborts the experiment.

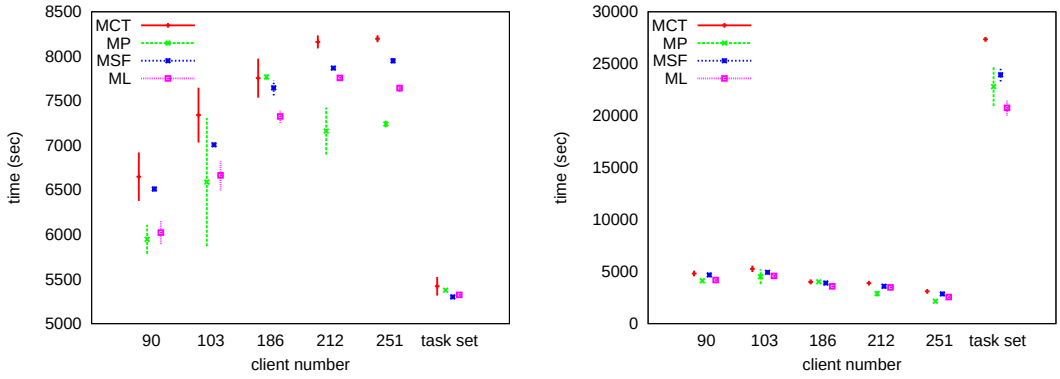

Figure 16. Scenario (d'): results on the makespan and on the sumflow for the first experiment

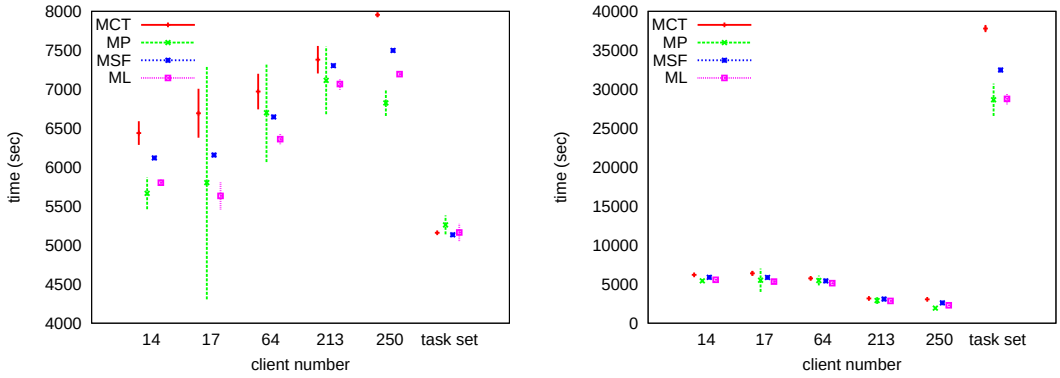

Figure 17. Scenario (d'): results on the makespan and on the sumflow for the second experiment

As already mentionned, when dealing with 1D-mesh applications, the makespan value is the sumflow value added to the application head arrival date. Then we observe the same variations on the results among

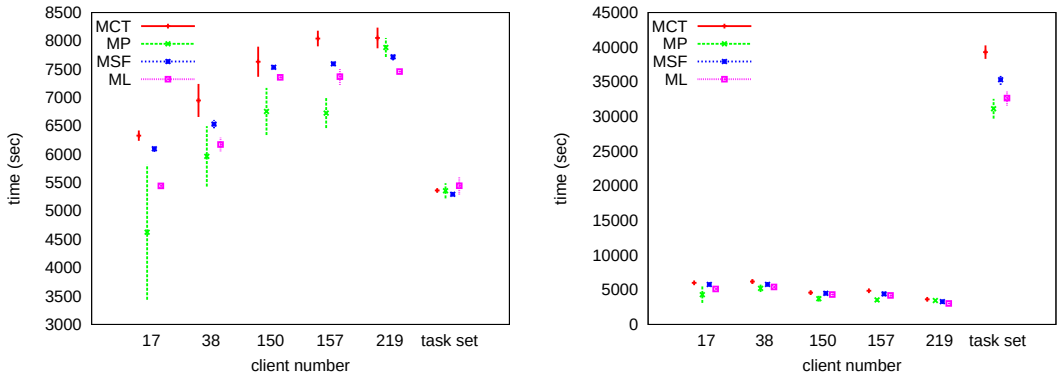

Figure 18. Scenario (d'): results on the makespan and on the sumflow for the third experiment

the heuristics between the makespan and the sumflow graph except of course for the set of independent tasks.

server	MCT		MP		MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
experiment 1								
spinnaker	55.6 (19.5 17.3 18.8)	26255	52.6 (20.3 15.5 16.8)	14640	54.0 (19.2 16.0 18.8)	22210	51.7 (17.5 15.2 19.0)	17049
artimon	44.2 (16.0 12.8 15.4)	21911	39.2 (13.2 12.3 13.7)	12870	43.7 (14.6 13.7 15.4)	19915	41.7 (13.0 13.6 15.1)	16194
soyotte	0.0 (0.0 0.0 0.0)	0	3.7 (0.8 0.9 2.0)	6269	0.9 (0.6 0.3 0.0)	784	3.0 (2.2 0.7 0.1)	2834
fonck	0.2 (0.1 0.1 0.0)	303	4.5 (1.3 1.5 1.7)	6741	1.4 (1.2 0.2 0.0)	1033	3.6 (2.9 0.7 0.0)	3114
total sumflow		48469		40520		43942		39191
experiment 2								
spinnaker	54.0 (17.7 18.7 17.6)	31283	50.6 (18.0 16.0 16.6)	18470	50.2 (13.7 18.3 18.2)	27117	47.1 (11.9 16.8 18.4)	20364
artimon	45.6 (15.4 13.9 16.3)	30505	40.4 (13.3 13.7 13.4)	16588	44.1 (14.4 13.9 15.8)	24236	42.5 (13.4 13.7 15.4)	19588
soyotte	0.2 (0.2 0.0 0.0)	227	4.4 (0.8 1.7 1.9)	7439	2.2 (1.9 0.3 0.0)	1595	4.7 (3.6 1.0 0.1)	4696
fonck	0.2 (0.1 0.0 0.1)	355	4.6 (1.3 1.2 2.1)	7410	3.5 (3.4 0.1 0.0)	2388	5.7 (4.5 1.1 0.1)	5317
total sumflow		62370		49907		55336		49965
experiment 3								
spinnaker	54.1 (15.1 20.6 18.4)	33084	54.0 (18.0 18.9 17.1)	18420	52.1 (14.6 19.0 18.5)	28876	49.7 (13.6 17.7 18.4)	22183
artimon	45.1 (13.9 14.8 16.4)	30053	37.8 (10.0 13.5 14.3)	18489	43.0 (11.2 15.1 16.7)	25795	41.6 (9.8 15.5 16.3)	21533
soyotte	0.4 (0.2 0.0 0.2)	513	4.0 (0.8 1.3 1.9)	6954	2.1 (1.2 0.9 0.0)	1988	4.2 (2.9 1.1 0.2)	5195
fonck	0.4 (0.0 0.2 0.2)	890	4.2 (0.4 1.9 1.9)	7463	2.8 (2.2 0.6 0.0)	2382	4.5 (2.9 1.3 0.3)	5800
total sumflow		64540		51326		59041		54711
MEAN								
spinnaker	54.6 (17.4 18.9 18.3)	30207	52.4 (18.8 16.8 16.8)	17177	52.1 (15.8 17.8 18.5)	26068	49.5 (14.3 16.6 18.6)	19865
artimon	45.0 (15.1 13.8 16.0)	27490	39.1 (12.2 13.2 13.8)	15982	43.6 (13.4 14.2 16.0)	23315	41.9 (12.1 14.3 15.6)	19105
soyotte	0.2 (0.1 0.0 0.1)	247	4.0 (0.8 1.3 1.9)	6887	1.7 (1.2 0.5 0.0)	1456	4.0 (2.9 0.9 0.1)	4242
fonck	0.3 (0.1 0.1 0.1)	516	4.4 (1.0 1.5 1.9)	7205	2.6 (2.3 0.3 0.0)	1934	4.6 (3.4 1.0 0.1)	4744
total sumflow		58460		47251		52773		47956

Table 19. Scenario (d'): processors utilization

We can see in Figures 16, 17, 18, the average makespan on the left hand and sumflow on the right hand of all applications including the set of independent tasks, computed on 2 runs. For this set, the makespan

		mean flow MCT (sec)	gain in percentage MP	MSF	ML
	type 1	54.8	19.5	5.9	-3.4
experiment 1	type 2	107.3	25.4	15.7	25.5
	type 3	169.6	11.0	13.0	32.9
	type 1	77.5	26.4	1.7	-10.1
experiment 2	type 2	160.6	26.5	17.8	28.5
	type 3	233.1	21.4	16.4	34.6
	type 1	76.1	31.9	2.0	-10.0
experiment 3	type 2	150.0	14.1	9.8	12.3
	type 3	239.5	21.6	12.7	27.6
MEAN	-	140.9	22.0	10.6	15.3

Table 20. Scenario (d'): average percentage gain for $\mu = 20$ sec on each task given by type

	1D-mesh clients makespan			1D-mesh clients sumflow			METATASK sumflow		
	MP	MSF	ML	MP	MSF	ML	MP	MSF	ML
experiment 1	8.9	2.9	7.2	17.0	5.5	12.9	16.6	12.5	24.1
experiment 2	9.4	4.9	9.7	15.3	7.3	14.4	22.2	13.6	24
experiment 3	14.2	4.1	8.9	19.0	6.2	12.7	21.3	11.6	20
MEAN	10.8	4.0	8.6	17.1	6.3	13.4	20	12.6	22.7

Table 21. Scenario (d'): average percentage gain against MCT on the makespan and the sumflow for each client

is nearly constant among the heuristics, and we have already shown this to be expected.

One can see from the graphs that our heuristics clearly show a significant improvement. It seems nonetheless hard to judge between MP and ML only on the average, but as one can easily see, MP has a much greater standard deviation. The use of tables gives the following information: MP and ML are tight on the resource consumptions (or management) (Table 19), from Tables 20 and 21 MP gives a better gain (22%) than ML (15%) on each independent task duration, on the makespan and the sumflow of each 1D-mesh client (11% for MP against 8.6% for ML) and both of them use around 22% less resources for executing the set of independent tasks. Nonetheless MP gives a slightly worst makespan on this set (see Table 22).

Despite the great deviation, sign that MP uses more the slowest servers than ML, MP seems to be the best heuristics facing that kind of scenario. Note that even with more runs, MP standard deviation would *a priori* not necessarily be highly reducde but maybe extend to each graph client. Finally, ML achieves very good and constant results.

	NetSolve's MCT				MP				MSF				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	5420	5160	5363	**5314**	5375	5261	5357	**5331**	5300	5134	5295	**5243**	5323	5164	5446	**5311**
sumflow	27350	37789	39284	**34808**	22800	28656	31118	**27525**	23943	32470	35311	**30575**	20761	28764	32658	**27394**
maxflow	375.9	399.9	515.3	**430.4**	484.7	509.0	568.6	**520.8**	289.3	278.2	325.2	**297.6**	283.6	416.2	700.6	**466.8**
maxstretch	12.8	10.5	13.5	**12.3**	14.1	15.5	15.4	**15.0**	9.3	10.6	12.1	**10.6**	14.4	16.3	20.6	**17.1**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	75 (16)	85 (15)	81 (16)	**80**(**16**)	62 (27)	69 (28)	61 (35)	**64**(**30**)	70 (19)	76 (22)	75 (23)	**74**(**21**)

Table 22. Scenario (d'): results in seconds for independent tasks

6.3. Scenario (e')

We propose to study the submission of a unique 10*50 stencil graph composed of tasks of type 1 (the shortest duration type) in this section. At most ten tasks are in the system at a given moment. We have already explained in the previous research report [CJ04] that we can not expect a good makespan with MP: the improvement in the HTM has no incidence in this because we have shown a very good accuracy with that kind of scenario.

We present in graphs of Figure 19 the makespan and the sumflow obtained on the experiment for each heuristic. Tables 23 shows the corresponding percentage of tasks that have been scheduled on each server as well as the sumflow. Finally, gains on the makespan and on the sumflow per heuristic is summarized in Table 24.

Surprisingly again, HMCT gives slightly better results than AHMCT. MSF achieves the best makespan, with a gain of 18.9% over MCT. ML has very tight performances because it achieves a 18.2% gain which is better than the 16.8% of HMCT. We note again that the heuristics specifically designed for the makespan, even relying on the HTM information, are not the best.

Considering the sumflow, MP gives the best results: mainly because most of tasks achives to run without or with less contention due to the use of the slowest servers. The drawback of this is naturally the makespan (27.5% worst than MCT). We must note the very good result of ML: it achieves to spare nearly half of the resources comparing to MCT ! It doubles the gain that can be done with MSF for nearly the same performance on the makespan.

ML is indubitably the best heuristic of this scenario.

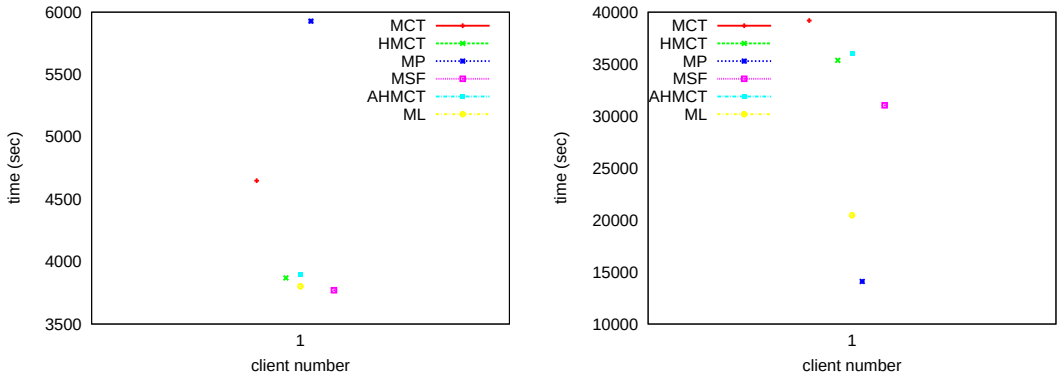

Figure 19. Scenario (e'): makespan and sumflow results

experiment 1						
server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	45.2 (45.2 0.0 0.0)	13804	53.0 (53.0 0.0 0.0)	18479	64.4 (64.4 0.0 0.0)	5065
artimon	54.8 (54.8 0.0 0.0)	25390	47.0 (47.0 0.0 0.0)	16883	25.0 (25.0 0.0 0.0)	2284
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.0 (4.0 0.0 0.0)	2549
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	6.6 (6.6 0.0 0.0)	4198
total sumflow		39194		35362		14096

experiment 1						
server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	52.0 (52.0 0.0 0.0)	15670	53.0 (53.0 0.0 0.0)	18911	49.8 (49.8 0.0 0.0)	8459
artimon	46.2 (46.2 0.0 0.0)	14215	47.0 (47.0 0.0 0.0)	17065	44.4 (44.4 0.0 0.0)	8303
soyotte	0.4 (0.4 0.0 0.0)	255	0.0 (0.0 0.0 0.0)	0	1.6 (1.6 0.0 0.0)	1022
fonck	1.4 (1.4 0.0 0.0)	904	0.0 (0.0 0.0 0.0)	0	4.2 (4.2 0.0 0.0)	2681
total sumflow		31044		35976		20465

Table 23. Scenario (e'): processors utilization

	makespan					sumflow				
	HMCT	MP	MSF	AHMCT	ML	HMCT	MP	MSF	AHMCT	ML
experiment 1	16.8	-27.5	18.9	16.1	18.2	9.8	64.0	20.8	8.2	47.8

Table 24. Scenario (e'): average percentage gain against MCT on the makespan and the sumflow for each client

6.4. Scenario (h')

Experiments consist in the submission of a 10*50 stencil graph, composed of tasks of type 1, in parallel of 86 independent tasks whose arrival dates are drawn from a Poisson distribution of parameter $\mu = 40$ seconds and that has a probability of one third to be of a given type (durations are given in Table 2).

Figures 20, 21 and 22 present per heuristic on the left side the makespan and on the right the sumflow of the applications submitted to the agent. The 1D-mesh application is launched as 9 independent tasks are already submitted, and possibly finished. In these figures, the independent tasks are assumed to be requested by one client.

Table 25 informs us about the management and the behavior of the heuristic: one can read the percentage of task (given by type in parenthesis) and the sumflow per server and per heuristic. However, one should be cautious because there is more tasks of type one due to the 1D-mesh graph. The gains that one can expect on the makespan and on the sumflow for both clients is given in Table, 27. Finally, Tables 26 and 28 give 'quality of service' information per heuristic assuming that the independent tasks are submitted by numerous clients: in that case, each task must obtain the best from the system.

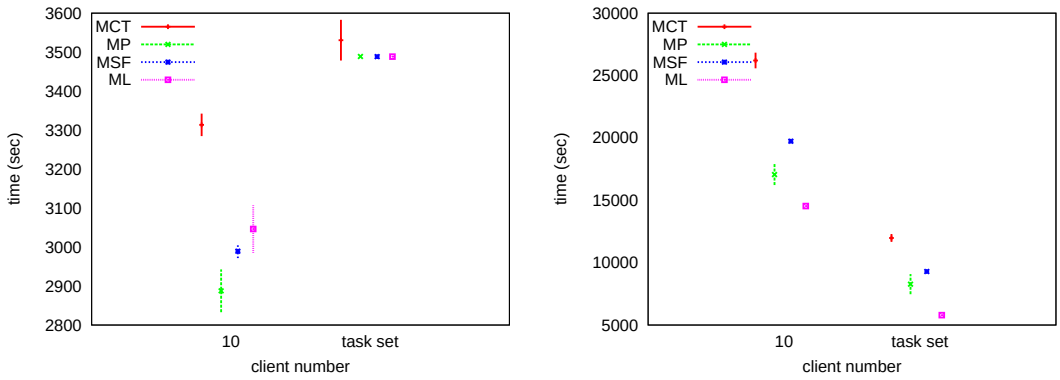

Figure 20. Scenario (h'): makespan and sumflow results for the first experiment

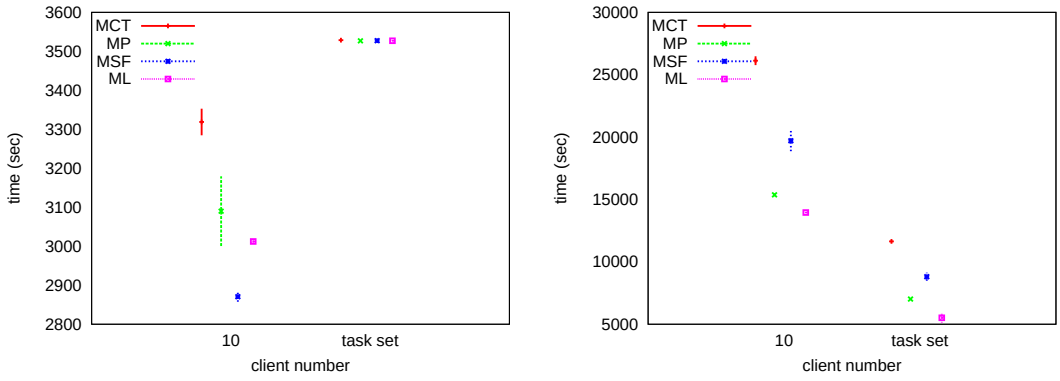

Figure 21. Scenario (h'): makespan and sumflow results for the second experiment

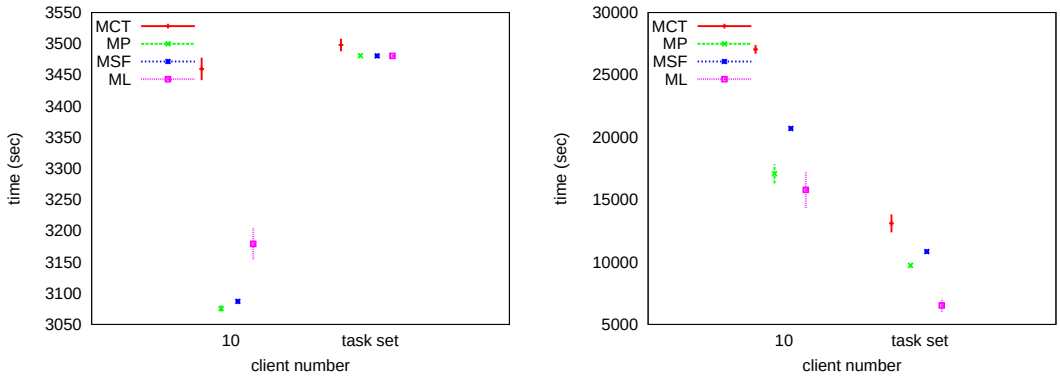

Figure 22. Scenario (h'): makespan and sumflow results for the third experiment

server	MCT		MP		MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
experiment 1								
spinnaker	54.8 (46.6 3.6 4.6)	19122	50.9 (42.6 3.7 4.6)	10104	51.2 (40.5 4.8 6.0)	13488	49.1 (38.8 4.8 5.5)	8035
artimon	45.1 (37.9 2.4 4.8)	18870	39.3 (33.6 1.8 3.9)	8799	40.3 (35.6 1.2 3.6)	11898	40.0 (34.8 1.2 4.0)	7369
soyotte	0.0 (0.0 0.0 0.0)	0	4.9 (4.2 0.3 0.4)	3093	3.7 (3.7 0.0 0.0)	1599	4.9 (4.9 0.0 0.0)	2216
fonck	0.1 (0.0 0.0 0.1)	191	4.9 (4.2 0.1 0.6)	3343	4.8 (4.8 0.0 0.0)	2044	6.0 (6.0 0.0 0.0)	2707
total sumflow		38183		25339		29029		20327
experiment 2								
spinnaker	53.4 (43.9 4.8 4.8)	19815	48.7 (39.9 3.4 5.4)	8553	49.4 (39.4 4.5 5.5)	13285	48.2 (37.9 5.1 5.2)	7685
artimon	46.4 (39.6 3.6 3.3)	17884	40.5 (34.4 3.9 2.2)	7533	41.7 (35.3 3.9 2.5)	11399	41.2 (35.1 3.3 2.8)	7181
soyotte	0.1 (0.1 0.0 0.0)	65	5.5 (4.6 0.9 0.0)	3159	4.2 (4.2 0.0 0.0)	1789	4.9 (4.9 0.0 0.0)	2112
fonck	0.0 (0.0 0.0 0.0)	0	5.4 (4.8 0.1 0.4)	3144	4.8 (4.8 0.0 0.0)	2042	5.7 (5.7 0.0 0.0)	2495
total sumflow		37764		22389		28515		19473
experiment 3								
spinnaker	54.0 (44.8 4.2 5.1)	22067	50.9 (42.4 4.0 4.5)	10851	49.7 (39.7 4.3 5.7)	13999	47.8 (36.9 4.2 6.7)	8674
artimon	45.8 (38.1 3.0 4.8)	18016	39.3 (32.4 2.5 4.3)	9257	40.2 (33.2 2.8 4.2)	13169	40.6 (34.5 3.0 3.1)	8162
soyotte	0.0 (0.0 0.0 0.0)	0	5.5 (5.1 0.1 0.3)	3349	4.6 (4.6 0.0 0.0)	2025	5.4 (5.4 0.0 0.0)	2485
fonck	0.1 (0.1 0.0 0.0)	65	4.3 (3.1 0.4 0.7)	3358	5.5 (5.5 0.0 0.0)	2355	6.2 (6.2 0.0 0.0)	2991
total sumflow		40148		26815		31548		22312
MEAN								
spinnaker	54.1 (45.1 4.2 4.8)	20335	50.1 (41.6 3.7 4.8)	9836	50.1 (39.9 4.5 5.7)	13591	48.4 (37.9 4.7 5.8)	8131
artimon	45.8 (38.5 3.0 4.3)	18257	39.7 (33.5 2.7 3.5)	8530	40.7 (34.7 2.6 3.4)	12155	40.6 (34.8 2.5 3.3)	7571
soyotte	0.0 (0.0 0.0 0.0)	22	5.3 (4.6 0.4 0.2)	3200	4.2 (4.2 0.0 0.0)	1804	5.1 (5.1 0.0 0.0)	2271
fonck	0.1 (0.0 0.0 0.0)	85	4.9 (4.0 0.2 0.6)	3282	5.0 (5.0 0.0 0.0)	2147	6.0 (6.0 0.0 0.0)	2731
total sumflow		38698		24848		29697		20704

Table 25. Scenario (h'): processors utilization

We must firstly note that HMCT and AHMCT did not handle the throughput of the scenario regardless the number of experiment we have tried. Thus, no real information can be processed and they do not ap-

		mean flow MCT (sec)	gain in percentage MP	MSF	ML
	type 1	72.4	32.1	13.9	44.1
experiment 1	type 2	113.1	22.1	21.1	49.3
	type 3	226.8	33.3	25.8	55.0
	type 1	83.7	56.3	28.4	54.2
experiment 2	type 2	150.1	30.9	23.2	50.6
	type 3	179.2	38.5	23.0	53.5
	type 1	75.5	27.0	21.7	52.5
experiment 3	type 2	153.2	30.3	22.9	49.1
	type 3	219.2	22.9	13.0	50.0
MEAN	-	141.5	32.6	21.4	50.9

Table 26. Scenario (h'): average percentage gain for $\mu = 40$ sec on each task given by type

	1D-mesh clients makespan			1D-mesh clients sumflow			METATASK sumflow		
	MP	MSF	ML	MP	MSF	ML	MP	MSF	ML
experiment 1	12.8	9.8	8.1	34.9	24.7	44.5	30.9	22.1	51.7
experiment 2	6.9	13.5	9.2	41.2	24.6	46.6	36.7	23.7	52.3
experiment 3	11.1	10.8	8.1	36.9	23.4	41.6	30.1	19.8	51
MEAN	10.3	11.4	8.5	37.6	24.2	44.3	32.6	21.9	51.7

Table 27. Scenario (h'): average percentage gain against MCT on the makespan and the sumflow for each client

pear in the results.

As far as the makespan is concerned, our heuristics achieves around 10% of gain against MCT. MP and MSF are tight, and for the first time better than ML (Table 27). But as soon as we are interested in quality of service or in resource management, ML gives the best results: it achieves to spare 8% on the sumflow for the stencil against MP, e.g. 45% better than MCT ! With 53%, ML is the best on the sumflow of the set of independent tasks, in front of MP which is 36% better than MCT. In addition, ML even uses each server less than any other: if resources had a cost proportional to their power, using ML would represent a large benefit (Table 25).

ML beats all heuristics on the quality of service as one can see in Tables 28 and 26. For example, an independent task has a 81% probability in average to finish sooner with ML than with MCT (against 13% to finish later) and 52% shorter.

MP gives very good results too, but is a large step behind ML performances, except on the makespan of the stencil. In conclusion, ML is the best heuristic facing this scenario, with excellent performances against MCT and a step higher than MP and MSF.

	NetSolve's MCT				MP				MSF				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	3531	3529	3498	**3519**	3489	3527	3481	**3499**	3488	3527	3481	**3499**	3488	3527	3481	**3499**
sumflow	11980	11638	13100	**12239**	8277	7017	9736	**8343**	9292	8811	10842	**9648**	5792	5516	6527	**5945**
maxflow	401.2	373.6	387.1	**387.3**	450.2	412.8	489.1	**450.7**	250.7	261.2	279.2	**263.7**	198.9	167.3	198.1	**188.1**
maxstretch	9.7	10.0	10.0	**9.9**	12.4	11.7	13.2	**12.4**	6.7	8.7	8.6	**8.0**	6.6	7.2	4.8	**6.2**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	64 (28)	78 (18)	68 (25)	**70**(**24**)	60 (32)	67 (26)	57 (36)	**61**(**31**)	72 (20)	82 (14)	80 (13)	**78**(**16**)

Table 28. Scenario (h'): results in seconds on independent tasks

6.5. Secnario (i')

This section presents the results that we have obtained submitting a 5*25 stencil in parallel of 86 independent tasks to the improved HTM. The details of the whole experimentation, the composition of the independent tasks set is given in [CJ04].

There are three instantiations of this scenario, each of which has been executed three times. Figures 23, 24 and 25 and Tables 29, 30, 31 and 32 show the average results on these three runs as well as the average on the scenario per heuristic.

We consider only two clients in the graphs: one who submits the stencil and the other responsible of the set of tasks thus considered here as one unique application. Furthermore, the mean makespan and sumflow is given but one should note the standard deviation computed on the three runs for each heuristic.

Table 29 contains the average resource consumption: the average percentage of task (also given par type in parenthesis) and the sumflow (sum of all the flow of the tasks that have been mapped to the server) per server. Table 30 summarizes the information contained in the figures (except the deviation) and gives the gain one can expect against MCT if the scheduler is the corresponding heuristics. For example, using MSF let *the graph* finish 6.1% sooner than MCT (0.9% for ML) and achieve a 13.5% gain on the sumflow (34.5% for ML). The gain on the makespan for the tasks set is not given as already explained.

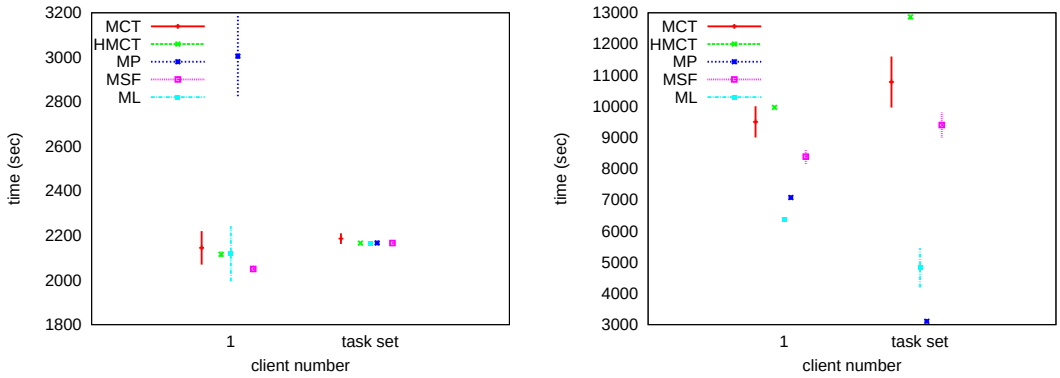

Figure 23. Scenario (i'): makespan and sumflow results for the first experiment

One can see that MCT, MP and ML have a greater standard deviation on the makespan of the graph than the other. This is due to the use of the slow servers for our heuristics (see Table 29) and certainly due to the censor report (and so to the quality of the information on the system state) for MCT. MSF and HMCT on the opposite have small or no variation. Table 30 show that the gain on the makespan is low (a maximum of 6% for MSF) and that the gain on the sumflow depends greatly on the heuristics used, for the graph and for the tasks set. We have explained in [CJ04] the behavior of MP which has a bad pert on the graph makespan and consequently a good benefit on the sumflow of the graph and of the set. But we can note that HMCT does not give any improvement against MCT and that MSF outperform MCT. The good surprise is that ML gives slightly the same makespan than MCT and that it outperforms from far MSF with results even better than MP !

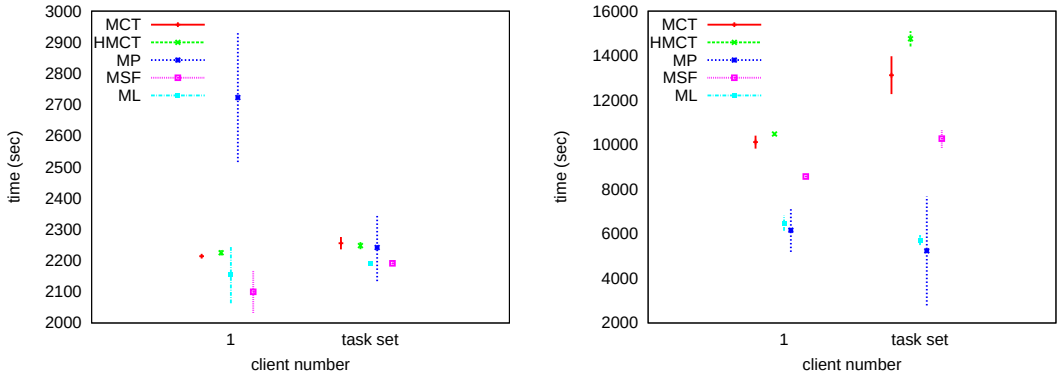

Figure 24. Scenario (i'): makespan and sumflow results for the second experiment

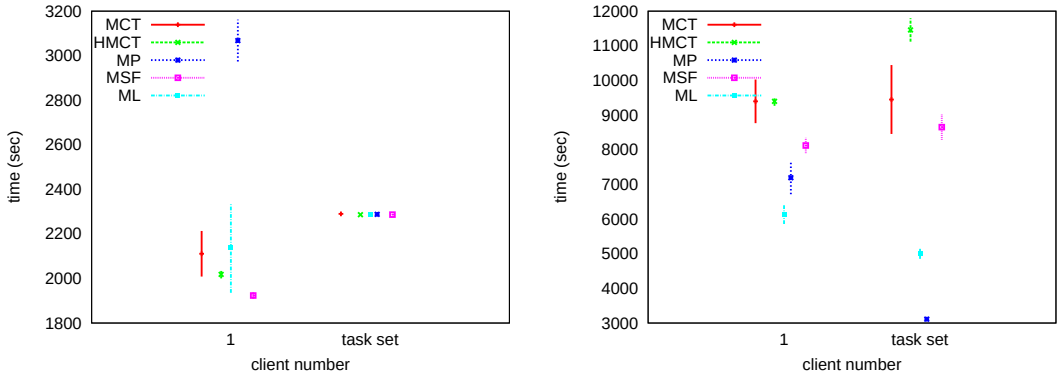

Figure 25. Scenario (i'): makespan and sumflow results for the third experiment

experiment 1						
server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	53.7 (40.6 7.1 6.0)	10549	52.4 (39.3 6.5 6.6)	12281	48.5 (34.8 6.8 7.0)	2950
artimon	46.3 (35.7 4.7 5.8)	9734	47.6 (37.0 5.4 5.2)	10553	32.2 (22.3 5.1 4.9)	2057
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	8.2 (8.2 0.0 0.0)	2208
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	11.1 (11.1 0.0 0.0)	2968
total sumflow		20283		22834		10183
experiment 2						
server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.3 (42.0 6.0 7.3)	12698	53.4 (38.9 7.3 7.3)	13635	47.1 (32.5 7.3 7.3)	3452
artimon	44.7 (31.4 6.3 7.0)	10546	46.6 (34.6 5.1 7.0)	11604	37.1 (26.9 4.4 5.8)	2738
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	7.0 (6.0 0.2 0.8)	2426
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	8.8 (8.1 0.5 0.3)	2787
total sumflow		23244		25239		11403
experiment 3						
server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	53.9 (38.7 9.6 5.5)	9797	52.4 (37.0 9.3 6.2)	10905	44.5 (28.6 9.5 6.5)	2593
artimon	46.1 (34.8 6.5 4.9)	9051	47.6 (36.5 6.8 4.3)	9949	34.8 (24.2 6.6 3.9)	2154
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	9.5 (9.5 0.0 0.0)	2548
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	11.2 (11.2 0.0 0.0)	3010
total sumflow		18848		20854		10305
MEAN						
server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	54.3 (40.4 7.6 6.3)	11015	52.8 (38.4 7.7 6.7)	12274	46.7 (32.0 7.8 6.9)	2998
artimon	45.7 (34.0 5.8 5.9)	9777	47.2 (36.0 5.7 5.5)	10702	34.7 (24.4 5.4 4.9)	2316
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	8.2 (7.9 0.1 0.3)	2394
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	10.4 (10.1 0.2 0.1)	2922
total sumflow		20792		22976		10630

experiment 1				
server	MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow
spinnaker	52.0 (39.5 6.3 6.2)	8765	46.6 (32.7 6.5 7.4)	4486
artimon	43.9 (32.7 5.5 5.7)	7918	43.8 (34.0 5.4 4.4)	4128
soyotte	1.1 (1.1 0.0 0.0)	299	3.6 (3.6 0.0 0.0)	978
fonck	3.0 (3.0 0.0 0.0)	807	6.0 (6.0 0.0 0.0)	1614
total sumflow		17789		11206
experiment 2				
server	MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow
spinnaker	51.2 (36.7 7.7 6.8)	9170	47.4 (32.5 6.6 8.2)	5023
artimon	43.4 (31.4 4.6 7.4)	8235	42.7 (31.0 5.7 6.0)	4508
soyotte	1.6 (1.6 0.0 0.0)	426	3.8 (3.8 0.0 0.0)	1020
fonck	3.8 (3.8 0.0 0.0)	1021	6.2 (6.2 0.0 0.0)	1656
total sumflow		18852		12207
experiment 3				
server	MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow
spinnaker	51.0 (36.5 8.2 6.3)	8322	49.3 (34.3 10.1 4.9)	4569
artimon	45.0 (33.0 7.9 4.1)	7390	41.7 (30.2 6.0 5.5)	4126
soyotte	1.6 (1.6 0.0 0.0)	426	3.3 (3.3 0.0 0.0)	918
fonck	2.4 (2.4 0.0 0.0)	638	5.7 (5.7 0.0 0.0)	1527
total sumflow		16776		11140
MEAN				
server	MSF		ML	
	% of tasks	sumflow	% of tasks	sumflow
spinnaker	51.4 (37.5 7.4 6.4)	8752	47.8 (33.2 7.7 6.8)	4693
artimon	44.1 (32.4 6.0 5.7)	7848	42.7 (31.7 5.7 5.3)	4254
soyotte	1.4 (1.4 0.0 0.0)	384	3.6 (3.6 0.0 0.0)	972
fonck	3.1 (3.1 0.0 0.0)	822	6.0 (6.0 0.0 0.0)	1599
total sumflow		17806		11518

Table 29. Scenario (i'): processors utilization

	STENCIL								TASKS SET			
	makespan				sumflow				sumflow			
	HMCT	MP	MSF	ML	HMCT	MP	MSF	ML	HMCT	MP	MSF	ML
experiment 1	1.3	-40.1	4.4	1.2	-4.9	25.5	11.7	32.8	-19.4	71.2	12.8	55.3
experiment 2	-0.5	-23.0	5.2	2.6	-3.6	39.1	15.2	35.9	-13.9	62.5	19.8	56.2
experiment 3	4.4	-45.3	8.9	-1.2	0.0	23.5	13.6	34.6	-18.5	65.4	12.7	50.5
MEAN	1.7	-36.1	6.1	0.9	-2.8	29.4	13.5	34.5	-17.3	66.4	15.1	54

Table 30. Scenario (i'): average percentage gain against MCT on the makespan and the sumflow for each client

		mean flow	gain in percentage			
		MCT (sec)	HMCT	MP	MSF	ML
	type 1	66.5	-19.1	69.1	9.4	48.8
experiment 1	type 2	135.7	-19.4	71.7	12.1	54.6
	type 3	199.8	-19.4	71.8	15.0	58.8
	type 1	79.3	-14.7	60.5	21.0	50.0
experiment 2	type 2	148.4	-16.0	61.6	18.9	56.5
	type 3	229.5	-9.7	59.1	23.5	58.5
	type 1	53.3	-39.8	55.6	-11.1	29.9
experiment 3	type 2	113.7	-24.1	67.3	7.6	46.3
	type 3	181.1	-11.0	71.4	17.1	54.8
MEAN	-	134.2	-19.2	65.3	12.6	50.9

Table 31. Scenario (i'): average percentage gain on each task given by type

	NetSolve's MCT				HMCT				MP			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	2186	2256	2290	**2244**	2166	2248	2286	**2233**	2167	2242	2288	**2232**
sumflow	10781	13123	9450	**11118**	12867	14757	11458	**13027**	3107	5236	3112	**3818**
maxflow	311.8	330.2	329.3	**323.8**	288.0	324.1	310.6	**307.6**	90.6	316.1	99.1	**168.6**
maxstretch	8.3	8.3	9.1	**8.5**	6.9	8.0	7.6	**7.5**	2.9	11.0	4.7	**6.2**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	30 (67)	31 (67)	32 (66)	**31**(**67**)	95 (3)	92 (8)	91 (7)	**93**(**6**)

	MSF				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	2167	2191	2287	**2215**	2166	2191	2288	**2215**
sumflow	9401	10275	8653	**9443**	4824	5723	4997	**5181**
maxflow	223.1	239.6	219.8	**227.5**	131.7	142.5	147.2	**140.5**
maxstretch	6.5	7.1	6.7	**6.8**	6.9	6.9	9.5	**7.7**
number of tasks that finish sooner than with NetSolve's MCT	61 (37)	78 (21)	51 (46)	**63**(**35**)	87 (11)	92 (7)	79 (18)	**86**(**12**)

Table 32. Scenario (i'): results in seconds on independent tasks

6.6. Senario (j')

This scenario consists in two 1D-mesh submissions in parallel of a set of independent tasks whose arrive at date with an inter-arrival drawn from a Poisson distribution of mean $\mu = 25$ seconds. The two 1D-mesh client submits their first task around respectively 300 and 2000 seconds.

Makespan and Sumflow graphs are given in Figures 26, 27 and 28 on respectively the left and right hand. Resource management is summarized in Table 33, where one can read the percentage of tasks, also given by type in parenthesis, affected per server as well as the sumflow. The gain over MCT per heuristic for each client (we assume in this table that there is two clients, one per 1D-mesh graph, and a third that submits the independent tasks) on both the makespan and the sumflow is presented in Table 35. Finally, Tables 34 and 36 give information about the quality of service of the heuristics: assuming that the independent tasks are submitted by several clients, each of them wants *a priori* to see their task(s) finish the soonest. One can read here the probability for its task to finish sooner as well as the expected gain on the duration.

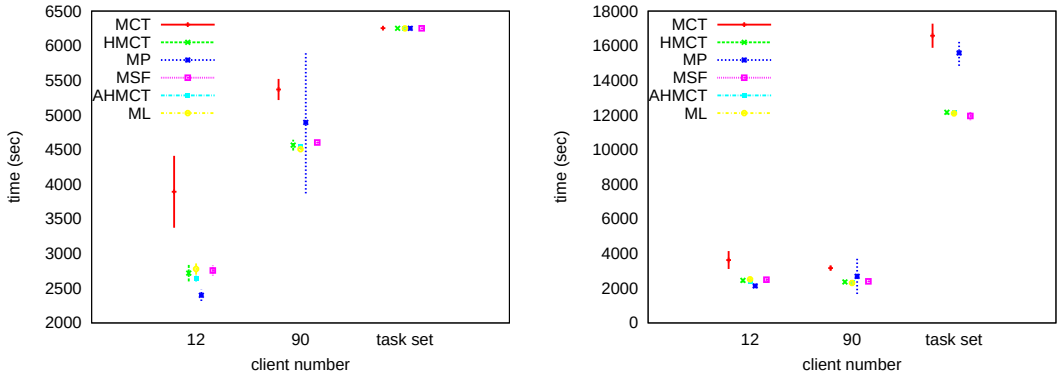

Figure 26. Scenario (j'): makespan and sumflow results for the first experiment

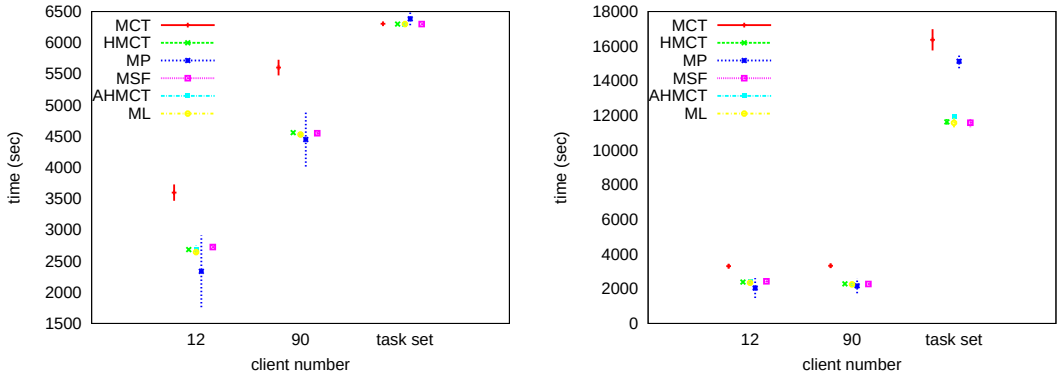

Figure 27. Scenario (j'): makespan and sumflow results for the second experiment

Except for MP, which has low performances on some criteria (like on the expected gain on duration and the sumflow of the set of independent tasks), all of our heuristics give very tight performances here, then no

experiment 1

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.5 (19.5 18.5 18.5)	13630	56.4 (18.8 18.5 19.1)	9577	52.5 (18.6 16.6 17.3)	6280
artimon	43.5 (13.7 15.2 14.6)	9747	43.6 (14.4 15.2 14.1)	7405	37.9 (11.0 14.7 12.1)	5223
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.3 (1.5 0.9 1.9)	4127
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.3 (2.0 1.4 1.9)	4782
total sumflow		23377		16982		20412

experiment 2

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	54.6 (16.1 20.6 17.8)	13103	56.2 (16.7 21.1 18.4)	9228	51.9 (15.7 19.5 16.7)	6386
artimon	45.4 (15.6 16.8 13.0)	9897	43.8 (15.0 16.3 12.5)	7078	38.5 (12.3 14.3 12.0)	4973
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.6 (2.3 1.3 1.0)	3690
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.9 (1.5 2.3 1.1)	4296
total sumflow		23000		16306		19345

experiment 3

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.1 (16.9 18.7 19.5)	12152	56.7 (18.5 17.1 21.1)	9020	54.0 (18.8 17.6 17.7)	6468
artimon	44.9 (15.7 14.7 14.4)	8711	43.3 (14.1 16.3 12.9)	6753	37.3 (12.1 12.2 13.0)	4985
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.0 (0.9 1.3 1.7)	3989
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.7 (0.8 2.4 1.6)	4673
total sumflow		20863		15773		20115

MEAN

server	MCT		HMCT		MP	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	55.4 (17.5 19.3 18.6)	12962	56.4 (18.0 18.9 19.5)	9275	52.8 (17.7 17.9 17.2)	6378
artimon	44.6 (15.0 15.6 14.0)	9452	43.6 (14.5 16.0 13.2)	7079	37.9 (11.8 13.7 12.4)	5060
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	4.3 (1.6 1.2 1.5)	3935
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	5.0 (1.4 2.0 1.5)	4584
total sumflow		22413		16354		19957

experiment 1

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.0 (18.5 19.4 18.1)	9317	56.2 (18.0 19.6 18.5)	9541	56.2 (19.1 19.5 17.6)	9013
artimon	44.0 (14.6 14.4 15.0)	7527	43.8 (15.1 14.1 14.6)	7303	42.7 (12.9 14.2 15.6)	7393
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.4 (0.4 0.0 0.0)	171
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.8 (0.8 0.0 0.0)	341
total sumflow		16844		16844		16918

experiment 2

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.2 (17.7 21.4 17.1)	9006	56.7 (17.4 21.4 17.8)	9175	54.7 (16.1 20.8 17.8)	8553
artimon	43.8 (14.1 16.0 13.8)	7283	43.3 (14.4 16.0 13.0)	7393	45.0 (15.3 16.6 13.1)	7499
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.3 (0.3 0.0 0.0)	128
total sumflow		16289		16568		16180

experiment 3

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.0 (17.7 18.7 19.6)	8299	56.3 (17.4 18.2 20.7)	8827	54.9 (16.1 19.1 19.6)	8132
artimon	44.0 (14.9 14.7 14.4)	6891	43.7 (15.2 15.2 13.3)	6769	45.1 (16.4 14.4 14.4)	7081
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0
total sumflow		15190		15596		15213

MEAN

server	MSF		AHMCT		ML	
	% of tasks	sumflow	% of tasks	sumflow	% of tasks	sumflow
spinnaker	56.1 (17.9 19.8 18.3)	8874	56.4 (17.6 19.8 19.0)	9181	55.2 (17.1 19.8 18.3)	8566
artimon	43.9 (14.5 15.0 14.4)	7234	43.6 (14.9 15.1 13.7)	7155	44.3 (14.9 15.0 14.4)	7324
soyotte	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.1 (0.1 0.0 0.0)	57
fonck	0.0 (0.0 0.0 0.0)	0	0.0 (0.0 0.0 0.0)	0	0.3 (0.3 0.0 0.0)	156
total sumflow		16108		16336		16104

Table 33. Scenario (j'): processors utilization

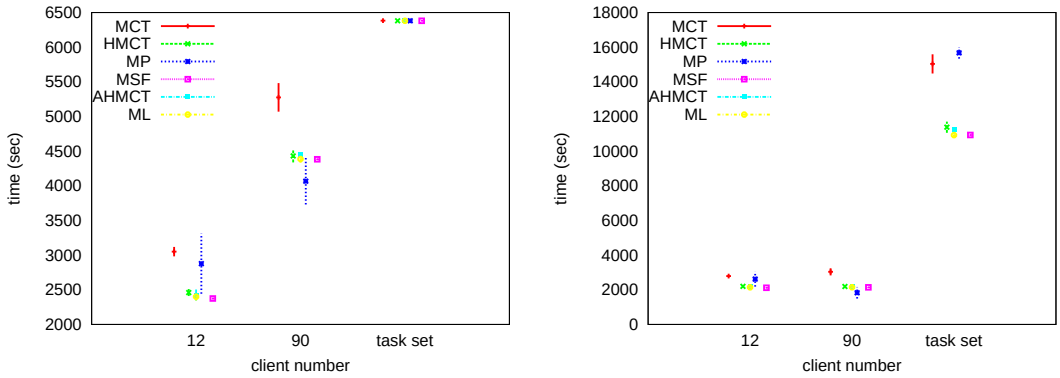

Figure 28. Scenario (j'): makespan and sumflow results for the third experiment

		mean flow	gain in percentage				
		MCT (sec)	HMCT	MP	MSF	AHMCT	ML
	type 1	30.6	18.5	-1.0	21.4	19.0	11.4
experiment 1	type 2	63.8	25.1	21.3	27.2	24.4	27.2
	type 3	104.8	30.3	-2.8	30.5	31.1	31.8
	type 1	29.4	21.0	-13.6	20.3	16.6	18.7
experiment 2	type 2	67.6	27.9	7.6	29.0	27.0	29.1
	type 3	99.8	32.3	14.3	32.3	30.6	32.8
	type 1	30.3	25.9	17.5	27.7	24.9	28.3
experiment 3	type 2	58.6	21.6	-11.3	25.4	22.8	24.1
	type 3	88.1	25.6	-6.7	28.4	27.2	29.1
MEAN	-	63.6	25.4	2.8	26.9	24.8	25.8

Table 34. Scenario (j'): average percentage gain on each task given by type

real discussion can be provided for this scenario. The low throughput has two main incidences that explains the previous remarks: sometimes, MP chooses a slow server and this results in the standard deviation that we can observe on the graphs and to its performances ; the other heuristics behave much alike because there are no real perturbation: in fact, the few perturbations, and the corresponding scheduling decisions, do not produce a sufficient impact to be observable on the heuristic performances.

Remark We clearly see here that the HTM let take much better scheduling decisions. Moreover, the heuristics behaves much alike when the throughput of the request is low and previous scenarios are necessary to compare them. The server load has not a really correlation with the throughput in the sens that they are always the most sollicited: a nuance must be done between the throughput of submitted jobs seen by the agent and the rate of incoming tasks seen by servers, which is heuristic dependent.

	STENCIL										TASKS SET				
	makespan					sumflow					sumflow				
	HMCT	MP	MSF	AHMCT	ML	HMCT	MP	MSF	AHMCT	ML	HMCT	MP	MSF	AHMCT	ML
experiment 1	22.5	23.6	21.7	23.8	22.3	28.8	28.1	27.7	30.3	28.9	26.7	6.0	27.9	26.9	27.0
experiment 2	22.0	27.8	21.5	22.3	22.9	29.5	36.4	29.0	29.9	30.6	28.4	7.2	28.9	27.1	28.7
experiment 3	17.7	14.3	19.5	18.1	19.1	24.5	23.0	26.8	24.9	26.3	25.6	-0.5	27.8	25.9	27.8
MEAN	20.7	21.9	20.9	21.4	21.4	27.6	29.2	27.9	28.4	28.6	26.9	4.2	28.2	26.6	28.8

Table 35. Scenario (j'): average percentage gain against MCT on the makespan and the sumflow for each client

	NetSolve's MCT				HMCT				MP			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
makespan	6253	6304	6383	**6313**	6251	6300	6381	**6311**	6251	6383	6381	**6339**
sumflow	16577	16369	15036	**15994**	12159	11632	11382	**11724**	15581	15130	15671	**15461**
maxflow	202.0	200.3	177.4	**193.2**	147.4	128.3	112.9	**129.5**	398.7	401.7	399.2	**399.9**
maxstretch	5.5	5.3	4.6	**5.1**	4.0	3.3	3.1	**3.5**	9.6	9.6	9.6	**9.6**
number of tasks that finish sooner than with NetSolve's MCT	-	-	-	–	69 (22)	72 (20)	69 (22)	**70**(**21**)	72 (19)	74 (19)	73 (19)	**73**(**19**)

	MSF				AHMCT				ML			
	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**	$seed_1$	$seed_2$	$seed_3$	**Avg**
	6251	6299	6381	**6310**	6251	6299	6381	**6310**	6251	6299	6382	**6310**
sumflow	11946	11583	10933	**11487**	12123	11922	11225	**11757**	12094	11578	10927	**11533**
maxflow	150.3	130.2	115.2	**131.9**	139.7	125.0	116.0	**126.9**	132.9	128.1	123.0	**128.0**
maxstretch	4.1	3.3	2.8	**3.4**	3.9	3.5	2.9	**3.4**	8.7	6.9	3.0	**6.2**
number of tasks that finish sooner than with NetSolve's MCT	71 (19)	72 (20)	71 (20)	**71**(**20**)	69 (21)	71 (22)	68 (23)	**69**(**22**)	73 (18)	72 (20)	71 (20)	**72**(**19**)

Table 36. Scenario (j'): results in seconds

7. Heuristics Performances and Behavior, HTM Improvement: Correlation ?

It is common knowledge that a better accuracy leads to better scheduling decisions thus implying better performances in the space-shared model. The mechanisms that we have implemented in NetSolve and in our HTM leads to synchronize the HTM to the reality: indeed, the HTM is aware of what is being conducted on the environment. In consequence, the accuracy is improved as we have shown in Section 5 and the heuristic uses the resources nearly to their real capacity.

The immediate question is: do heuristics behave now like they used to before the synchronization mecanisms implementation ? If not, how do their performances evolute, and do they significantly ? We will attempt to answer these questions in this section. We invite the reader to refer to the research rapport [CJ04] to have the exact and entire study of the scenarios and thus the results against which we will compare the new ones.

7.1 Metatask

Concerning the submission of a set of independent tasks, there is not a big difference between heuristics behaviour if the rate is low. For example when $\mu = 20$ seconds, the average response time (and the other quality of service metrics in general) is slightly the same, there is nevertheless a difference in the resource management only on the third experiment.

When $\mu = 17$ seconds, HMCT and MSF increase the number of tasks mapped to the fastest servers spinnaker and artimon. Indeed, MSF does not use fonck and soyotte anymore. This leads for both of them to a slightly pert in performance on QOS criteria because the scheduling decisions involve a little more perturbations.
When the rate is high ($\mu = 15$ seconds), consequences are more obvious: HMCT cannot handle the job throughput anymore because of a too high load on spinnaker who refuses some tasks.

On the opposite, MP improves even slightly its performances with the increase of the rate: the policy of minimizing the perturbation seems to pay here.

The reason of the heuristics behaviour is the following: we have shown good accuracy for a low and medium rate. Then, HMCT, MP and MSF behaves the same for $\mu = 20, 17$. But when $\mu < 17$ seconds, the HTM sub-estimated the fastest servers computing capacity as soon as 5 or 6 tasks were interfering with another one. With the increase in accuracy, the heuristics have a precise evaluation of the situation on the fast servers: depending on the design of the heuristic, more tasks will be consequently mapped on the fastest servers. This change does not occur with MP which aims to minimize the perturbation: on the opposite, a better accuracy leads necessarily to less perturbation and then to a better quality of service.

Scenario (d) vs (d')
Firstly, we note that the HMCT, which has more precise information on the system state, does not appear in the results: scheduling tasks on the fastest servers make them refuse some jobs. Moreover, we have seen through all the experiments that this policy does not give good results on quality of service for independent tasks and, even on the makespan, results can be much improved using another heuristic (ML for example).

When MSF is the heuristic embedded in the agent, spinnaker and artimon are more loaded than before: it was predictable. Indeed, as information on the whole environment are more accurate, fast servers are less sub-utilized like they used to be before. In consequence, we see more perturbations and performances suffer it like the respective gains show: before, gains against MCT were on average $(7.8\%; 11.4\%)$ on the

makespan and sumflow of the five 1D-mesh clients and 20.6% on the sumflow of the task set ; they are now $(4\%; 6.3\%)$ and 12.6%. Even the average mean flow gains decrease from 15.7% to 10.6%. Nonetheless, we also observe that the maxtretch and maxflow decrease. The HTM information leads also to less standard deviation in the heuristic behavior on the makespan and sumflow during numerous runs.

MP seems to improve its performances (makespan, mean flow, max-stretch) or be constant (sumflow, percentage of tasks finished sooner than MCT).

Scenario (e) vs (e')
HMCT does not use fonck or soyotte anymore, thus increasing the sumflow on spinnaker and artimon due to more perturbations. For HMCT and MSF, the same behavior is observed: performances seem to decrease (but are 3% significant ?). MP seems marked even deeper by its drawback fully explained in [CJ04], and in consequence the makespan is higher and the sumflow lower.

Scenario (h) vs (h')
Like for the previous scenarios, HMCT make servers to refuse some jobs. The quality of the information can be considered here as a drawback for this heuristic. MP and MSF increase their consumption in fast resources and MSF has a negligeable standard deviation on its behavior between two runs.

We can do the same commentaries than above: the new features implemented in NetSolve and in the HTM to increase the accuracy of its prediction has nearly no influence on MP performance (mainly because the accuracy was fairly good) but MSF performances decrease due to more perturbations on the fastest servers. Indeed, MSF average mean flow (or response time) decreases from 38.4% to 21.4% for example.

Scenario (i) vs (i')
We observe again that the heuritics map more tasks on the fastest servers (see tables concerning the processors utilization). Nonetheless, the difference resides in that some heuristics already used to load the fastest servers, thus an increase has more obvious consequences on some criteria like we will see below.

If a low difference between results of Scenario (i) and (i') may not be in direct relation with a behavior evolution, one must note that for example HMCT 5.6% gain on the sumflow of the application is now a 3% pert. MSF performance against MCT decreases from 27.5% to 13.5%. Performance evolutions for the tasks set sumflow and on the average duration of independent tasks are even worst. Only MP achieves to maintain the same level.

Most of these behaviors are direct implications of more perturbations on the fastest servers. Tasks are delayed and so is the makespan. The average duration is longer. HMCT and MSF suffer the most as MP is nearly untouched mainly beacause the HTM predictions were already accurate: MP delays the completion of the graph by submitting some critical tasks on slow servers. It thus faces a lower job throughput than the others in this scenario.

Assessment It is moderate: HMCT cannot face the throughput that it could before thus minimizing its interest. MSF real performances with highly accurate information is less good than before on the quality of service metrics but acts with a higher predictable behavior (we observe a lower standard deviation between two runs). It deals with the real capacity of computing resources and then maps more tasks to the fastest servers then producing higher contention. Nonetheless, it still stay a good heuristics which clearly outperforms MCT (and the two derivatives HMCT and AHMCT).

MP behaves slightly better because of its design to minimize the perturbation, then a higher quality of service can be observed. It is also easily explained as the accuracy was already good.

Moreover, the most accuracte the predictions are, the better the model has to be because fast servers are even more sollicited. Consequently, they may refuse to execute a request and the scheduling decision is not put into effect. The other consequence is that simulation experiments may not be relevant if several parameters are not taken into account like: a given determinist task have not exactly the same duration from one execution to another, and this can affect the modelization of the system ; if the scheduling decisions are taken accordingly to the load given by some sensors, and that tasks are executed in a time-share system, the environement is really hard to modelize to reflect the reality ; results of the modelization are generally lower than they would be in the reality (systems behaves slightly better than the model seems to show) and they depends on the perturbation created on each server at any time.

We have observed an execution of a run where someone logged on the server (spinnaker). The HTM estimations were consequently lower than the real durations because of the perturbations that existed in the real world and that the HTM was not aware of. Nonetheless, it regained a good accuracy with time as shown in Figure 29. Although we are not able to determine what job has been performed (no information were given in the log files), nor its characteristics like its duration, this result is encouraging.

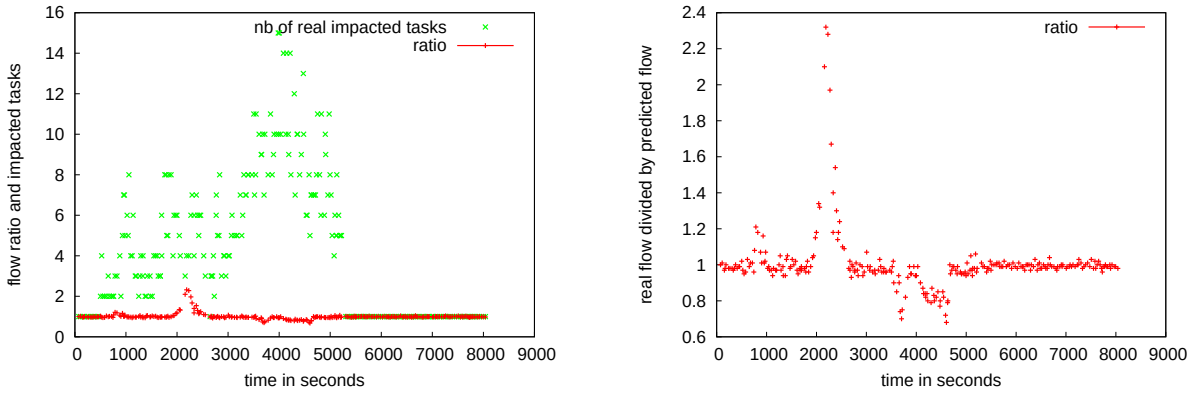

Figure 29. The HTM is able to regain a good accuracy on a non-dedicated server

Finally, the difference of behavior and the performances tha we have observed on HMCT, MP and mostly on MSF highlight the excellent performances of ML which achives to take the advantages of both MP and MSF without their respective drawbacks.

8. Improvements and Future Work

We will consider the memory management in further work. It is implemented in the actual HTM by a simple filter on the servers which do not dispose of sufficient memory to handle the job. It assumes a constant memory load during the execution of the task, equal of the memory peak requirement. Of course, heuristics implementing a trade-off between the memory requirement and other criteria can be developped to advantageously replace the filter.

The accuracy at our disposal with the HTM information invite to go further and propose a system that do not only rely on the duration of a task on idle servers anymore (obtained by benchmark), but on the

use of the complexity polynomial and possibly on further acknowledgements on the task (like in [Sch98] where Schopf has at her disposal the number of operations or the results of the profiling of the application). Indeed, when a task is submitted, the HTM can easily compute an estimation of the duration of the new task by extrapolation and uses its completion date to evaluate an estimation of one more coefficient of the polynomial. This implies a better accuracy in the next estimation of the duration of the task on the idle server etc. In that case, the system would achieve an increase in task duration prediction with time.

NetSolve in the 2.0 version is able to end a task if the client asks so to the agent. Shortly, the HTM will take into account this kind of situation. Information on the system state will thus be up to date at any moment.

Another possible extend of the functionnalities of the HTM is a scheduling by slices:
Firstly, the SCALAPACK functions perfectly know the graphs of the underlying computing applications. When receiving that kind of request, the agent can give the graph to the scheduler with can in turn simulates and prepare some basis for further scheduling like not map critical tasks on slow servers. This idea can naturally also be used for a client who has the knowledge of his application: at each job submission, the agent answer the scheduler choice, but has knowledge of further submission and take decisions accordingly.
Secondly, if the agent faces a high throughput, requests can be delayed. Requests are recorded as a set of tasks which has to be scheduled when the thoughput lowers. In that case, the agent acts also as a regulator for the system.
Finally, relying on the same idea, the HTM can simulate the new request for different arrival dates (naturally chosen as the date where a task finish and leaves the system). The scheduler can then take its decision about where to map the task and at which time the task must enter the environment.

We also want to deal with non-dedicated servers in future work. Hence, we need to exploit the information of sensors that are already executing on servers (NetSolve's for the moment). The HTM can compare the workload that the agent receives and the one that would normally be: if higher in a given range, then it can conclude that some perturbations must be included in the computation of the Gantt chart, basis of the information given to the scheduler. In that case, other estimations, obtained for example using probabilistic behavior, can also be used and used in addition within the HTM.

9. Conclusions

We have introduced two new heuristics AHMCT and ML in this paper, which rely on the Historical Trace Manager, a module embedded in the agent whose role is to predict the duration that any task requires on the system. The HTM and the other heuristics HMCT, MP and MSF have already been presented in [CJ02a] and [CJ04].

We have explained some new mechanisms: a global ID assignation for each new request in the system and a completion message sent to the agent when a task finishes. Both mechanisms have been implemented in NetSolve which required to change or improve all parts (client, agent and server). The HTM is now able to take each task completion date into account to compute higher accurate information on the system state.

We have presented several scenarios that were scheduled and executed in real world experiments. Three main studies have been conducted: the validation of the HTM estimations before and after the new features ; the comparison between all heuristics performances on several criteria including the makespan, the sumflow, the mean flow and probability of an independent task to finish sooner than if scheduled with MCT.

The HTM achives an excellent estimation of the duration of any task in the system at any moment. Thus, heuristics can take according scheduling decisions with the real computing capacity of the resources. Some previous bad behaviors that were encountered and explained have disappeared with the synchronization mechanisms. It generally leads for the heuristics to higher load the fastest resources. Then the most accurate is the module, the most accurate is has to be because the time-share model is treated roughly during some experiments.

AHMCT performances confirm that trying to optimize the makespan by minimizing the makespan at each request may not an efficient policy in an heterogeneous time-shared environment (it behaves better than MCT because of better information on the system state but is much outperformed by the heuristics that are designed to minimize the perturbations tasks have on each other).

Our heuristics are a real improvement in taking good scheduling decisions. The synchronization mechanisms have changed some performances (HMCT and MSF) that appear to behave better if sub-utilizing the resources. ML appears to mix the advantages of MP and MSF (quality of service criteria and makespan) and may be used regardless the application type of the submissions.

References

[CD96] H. Casanova and J. Dongarra. Netsolve : A network server for solving computational science problems. In *Proceedings of Super-Computing -Pittsburg*, 1996.

[CJ02a] Y. Caniou and E. Jeannot. Dynamic mapping of a metatask on the grid: Historical trace, minimum perturbation and minimum length heuristics. Technical Report 4620, LORIA, Nancy, oct 2002.

[CJ02b] Y. Caniou and E. Jeannot. Ordonnancement pour la grille : une extension de mct. In *Proceedings of RenPar 2002*, pages 58–65, Hammamet, Tunisia, April 2002.

[CJ03] Y. Caniou and E. Jeannot. New dynamic heuristics in the client-agent-server model. In *Proceedings of the 13th Heteregeneous Computing Workshop (HCW03)*, april 2003.

[CJ04] Y. Caniou and E. Jeannot. Study of the behaviour of heuristics relying on the historical trace manager in a (multi)client-agent-server system. Technical Report 5168, LORIA, Nancy, 2004.

[Sch98] J. Schopf. *Performance Prediction and Scheduling for Parallel Applications on Multi-Users Clusters*. PhD thesis, University of California, San-Diego, 1998.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399