Dynamic Scheduling in Heterogeneous Multiprocessor Architectures. Efficiency Analysis.

Laura C. De Giusti[1], Marcelo Naiouf[1], Franco Chichizola[1], Emilio Luque[2], Armando E. De Giusti[1]

[1]Instituto de Investigación en Informática LIDI (III-LIDI) – School of Computer Sciences – Universidad Nacional de La Plata. Argentina
[2]Universidad Autónoma de Barcelona (UAB) - Computer Architecture and Operating System Department (CAOS) Spain
{ldegiusti, mnaiouf, francoch}@lidi.info.unlp.edu.ar, emilio.luque@uab.es, degiusti@lidi.info.unlp.edu.ar

Abstract. A MPAHA (Model for Parallel Algorithms on Heterogeneous Architectures) model that allows predicting parallel application performance running over heterogeneous architectures is presented. MPAHA considers the heterogeneity of processors and communications.
From the results obtained with the MPAHA model, the AMTHA (Automatic Mapping Task on Heterogeneous Architectures) algorithm for task-to-processors assignment is presented and its implementation is analyzed.
DCS_AMTHA, a dynamic scheduling strategy for multiple applications on heterogeneous multiprocessor architectures, is defined and experimental results focusing on global efficiency are presented.
Finally, current lines of research related with model extensions for clusters of multicores are mentioned.

Keywords: Dynamic Scheduling – Parallel Algorithm Model - Distributed Architectures – Heterogeneity

1 Introduction

The problem of *automatic task-to-processor mapping* in heterogeneous architectures is highly complex [1]. This complexity can be briefly represented considering the two main elements relating the parallel application to the supporting architecture: each node processing capacity and the cost of inter-processor communications in time [2].

In Computer Science, models are used to describe real entities such as processing architectures and to obtain an "abstract" or a simplified version of the physical machine, capturing crucial characteristics and disregarding minor details of the implementation [3][4]. In the case of parallel systems, the most currently used architectures – due to their cost/performance relation - are heterogeneous clusters and multiclusters; for this reason, it is really important to develop a model that fits the characteristics of these platforms. An essential element to be considered is the

potential heterogeneity of processors and communications among them, which adds complexity to the modeling [5][6].

At present, there are different graph-based models to characterize the behavior of parallel applications in distributed architectures [7]. Among them, we can mention TIG (Task Interaction Graph), TPG (Task Precedence Graph), TTIG (Task Temporal Interaction Graph) [8], TTIGHA (Task Temporal Interaction Graph on Heterogeneous Architectures) [9] and MPAHA (Model on Parallel Algorithms on Heterogeneous Architectures) [10].

Once the graph modeling the application has been defined, the *scheduling* problem [11] is solved by an algorithm that establishes an automatic mechanism to carry out the task-to-processor assignment, searching for the optimization of some execution parameter (usually, time) [12]. Among the known mapping/scheduling algorithms, we consider AMTHA (Automatic Mapping Task on Heterogeneous Architectures), a mapping algorithm, to carry out the assignment of tasks of the application to the processors of the architecture [10]. This algorithm considers the heterogeneous characteristics of the architecture taken into account in the MPAHA (Model on Parallel Algorithms on Heterogeneous Architectures) model [10].

Usually, scheduling/mapping algorithms are used to obtain the best assignment of the processes that make up an application to the processors of the architecture in which it will be run. In this paper, the DCS_AMTHA (Dynamic Concurrent Scheduling) algorithm to carry out the scheduling of multiple parallel applications on heterogeneously distributed architectures (cluster) is defined. This algorithm is based on AMTHA and the goal is to optimize the efficiency achieved by the whole system.

In Section 2, the MPAHA model is briefly described. In Section 3, the AMTHA scheduling algorithm is dealt with. In Section 4, DCS_AMTHA scheduling algorithm is detailed. Section 5 describes the experimental work, and Section 6 presents results of speedup and efficiency of DCS_AMTHA for different processing architectures and working loads. Finally, Section 7 presents the conclusions and the future lines of work.

2 MPAHA Model

The MPAHA model is based on the construction of a directed graph *G (V, E)*, where:

- *V* is the set of nodes representing each of the tasks T_i of the parallel program.
- *E* is the set of edges representing each of the communications between the nodes of the graph.

2.1 Model parameters

Nowadays, most applications are formed by a set of tasks that perform different functions and that communicate among themselves to exchange information at any point. Each of these tasks can in turn be split in various blocks (subtasks) which do not communicate with other blocks from other tasks.

In the first parameter of the graph (*V*), each node represents a task T_i of the parallel program, including its subtasks (St_j) and the order in which they should be executed to perform the task. If there is a heterogeneous architecture available, the computation times for each of the composing processors should be taken into account. That is, node *i* ($V_i \in V$) stores the computation time at each of the different types of processor for each subtask of task T_i. Therefore, V_i *(s,p) = execution time of subtask s in processor type p.*

In the second parameter of the graph (*E*), the edges represent the communications exchanged between each pair of tasks. In this set, an edge *A* between two tasks T_i and T_j contains the communication volume (in bytes), the source subtask ($\in T_i$), and a target subtask ($\in T_j$). That is, $E_{i,j}$ *(o,d) = communication volume between the source subtask* ($o \in T_i$) *and the target subtask* ($d \in T_i$).

It should be noted that, given the heterogeneity of the interconnecting network, instead of representing the time required for the communication, the corresponding communication volume between two subtasks is represented. Fig. 1 shows an example of a graph generated with this model.

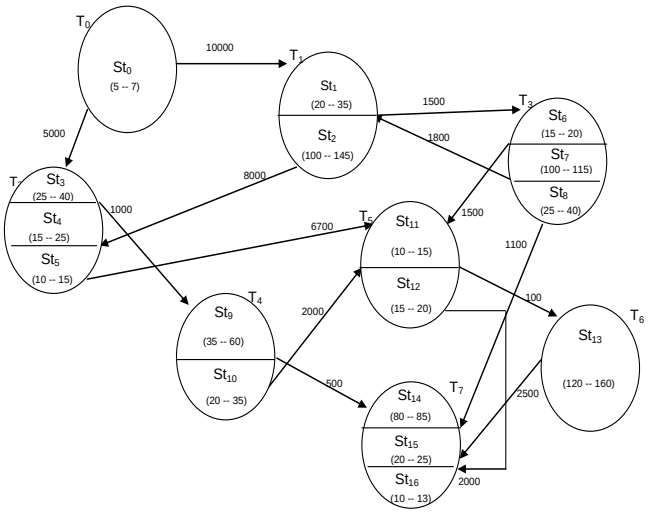

Fig. 1. Example of a graph generated by the model.

3 AMTHA mapping algorithm

AMTHA is a static mapping algorithm that is applied to the graph generated by the MPAHA model. It allows determining the assignment of tasks to the processors of the architecture to be used to minimize the execution times of the application in that architecture. This algorithm must also provide the order in which subtasks (forming the task) assigned to each processor should be executed (*task scheduling*). AMTHA considers an architecture with a limited number of heterogeneous processors. As

regards the interconnecting network, the algorithm also considers that bandwidth and transmission speed can be heterogeneous.

The AMTHA algorithm uses the values of graph G generated by the MPAHA model. These values are: the time required to compute a subtask in each type of processor, the communication volume with adjacent processors, and the task each subtask belongs to.

The AMTHA algorithm assigns one task at a time until all tasks have been assigned. Figure 2 shows the pseudo-code with the main steps of the algorithm:
When the execution of the algorithm ends, all the tasks have been assigned to one of the processors and the order in which the subtasks forming the tasks assigned to these processors will be executed has also been determined.

Calculate ***rank*** for each task.
Whereas (not all tasks have been assigned)
1. Select the next task ***t*** to assign.
2. Chose the processor ***p*** to which task ***t*** should be assigned.
3. Assign task ***t*** (selected in step 1) to processor ***p*** (selected in step 2).
4. Update the ***rank*** of the tasks involved in step 3.

Fig. 2. Pseudo-code with the basic steps of the AMTHA algorithm.

The following paragraphs describe each of the three steps followed during the execution of the AMTHA algorithm.

3.1 Calculating the rank of a task

Given a graph *G*, the rank of a task *Rk(T)* is defined as the sum of the average times of the subtasks forming it and that are ready for execution (all predecessors have already been assigned to a processor and are already there). Equation 1 expresses this definition:

$$Rk(T) = \sum_{i \in L(T)} W_{avg}(St_i) \quad (1)$$

where:

L(T) is the set of subtasks that are ready for task *T*.

$W_{avg}(St)$ is the average time of subtask *St*. The average time is calculated as shown in Equation 2.

$$W_{avg}(St_i) = \frac{\sum_{p \in P} V_{St_i}(type\ of\ \ processor\ \ p)}{\#P} \quad (2)$$

where *P* is the set of processors present in the architecture and *#P* is the number of processors forming this set.

3.2 Selecting the task to execute

After obtaining the *rank* of each application task, the task that maximizes it is selected. If there are two or more tasks that have the same maximum value, the

algorithm breaks this tie by selecting the one that minimizes the total execution time average for the task. Equation 3 shows this calculation:

$$Tavg\ (T) = \sum_{i \in T}\ W_{avg}(St_i) \tag{3}$$

3.3 Selecting the processor

Selecting the processor involves choosing the computer within the architecture that minimizes the execution time when the selected task is assigned to that processor.

In order to understand how the time corresponding to processor *p* is calculated, the fact that each processor keeps a list of subtasks LU_p that were already assigned to it and that can be executed (all its predecessors are already placed), and another list that contains those subtasks that were assigned to *p* but whose execution is still pending LNU_p (some of their predecessors have not been placed yet) must be taken into account.

Therefore, to calculate which processor *p* will be selected, two possible situations are considered:

1. All subtasks of task *t* can be placed in *p* (that is, all its predecessors have been placed).
2. Some of the subtasks of *t* cannot be placed in *p* (this happens when some predecessor of this subtask has not been placed).

In the first case, the time *Tp* corresponding to processor *p* is given by the moment in which *p* finishes the execution of the last subtask of *t*. However, in the second case, the time *Tp* corresponding to processor *p* is given by the time when the last subtask of LU_p will finish plus the addition of all execution times in *p* for each of the subtasks on LNU_p.

3.4 Assigning the task to the selected processor

When assigning a task *T* to a processor *p*, there is an attempt to place each subtask St_k belonging to *T* to the processor, at a moment when all the adjacent subtasks have already finished (including the predecessor subtask within *T*, if any). This can be a free interval between two subtasks that have already been placed in *p*, or an interval after them. If subtask St_k cannot be placed, it is added to the LNU_p list. Each time a subtask St_k is added to the *LU* list of one of the processors, an attempt is made to place all the predecessors belonging to the already assigned tasks.

3.5 Updating the rank value of pending tasks

The first action within this step consists in assigning *-1* as rank value to the task *t* that has been assigned to processor *p*. The reason for this is to prevent task *t* from being re-selected for assignment.

Also, the following situation is considered in this step: for each subtask St_k placed in step 3.4, the need to update the rank of the tasks successor subtasks St_{succ} of St_k

belong to is analyzed; that is, if all predecessors of St_{succ} are already placed, then the rank of the task St_{succ} belongs to is updated by increasing it by $W_{avg}(St_{succ})$.

4 Dynamic Concurrent Scheduling (DCS_AMTHA) Algorithm

Given a parallel application, the AMTHA algorithm generates an assignment of its tasks so as to achieve an efficient use of the heterogeneous architecture in which it will be executed [10]. As the multiprocessor architectures increase the number of processors, a way to obtain high efficiency is to increase the volume of parallel work, simultaneously dealing with multiple applications. In this work, DCS_AMTHA algorithm is developed in order to carry out the scheduling of multiple applications on a heterogeneously distributed architecture.

This algorithm allows overlapping two or more applications when using the architecture. The application A_i scheduling is carried out using the AMTHA algorithm; the tasks that make up A_i can be assigned to empty gaps generated by the assignment of applications prior to A_i. In this case, A_i time zero ($\mathbf{t_0}$=time on which execution starts) is the time on which A_i reaches the system (S_i).

The *DCS_AMTHA* algorithm allows the reassignment of already assigned tasks to a processor that are not being run yet.

Application A_i is assigned through the AMTHA algorithm, considering $\mathbf{t_0}$ as the moment of reaching S_i. Those tasks belonging to previously assigned applications ($A_0..A_{i-1}$) whose starting time in the scheduling occurs after de-assigning S_i, and together with the A_i tasks, are part of the new scheduling process.

Among the most outstanding features of this algorithm, the following are to be pointed out:

- At some moment, it requires process migration or reconfiguration of location of each application task. The generated communication overhead depends on the physical distribution of processors.
- It prioritizes to obtain *minimum system end time*, not regarding the order in which applications reach the system.

5 Experimental Work

In previous works, *DCS_AMTHA* was compared to two alternative algorithms for scheduling multiple applications (*SCS_AMTHA* and *SS_AMTHA*). As a result, it was concluded that the *DCS_AMTHA* algorithm yields best response times for the whole system, that is, the architecture is used in a more efficient way [13].

Based on these results, the behavior of the algorithm after scaling the architecture and/or the number of applications in the system is analyzed.

A set of applications was selected, in which each of them varies in terms of the number of application tasks, task size, number of subtasks making up a task, and communication volume among subtasks. In all the applications, the total computing time exceeded that of communications (coarse grained application). In this work, 150

synthetic applications were generated and placed on a queue *Q*. Each application has the following characteristics:

- The number of tasks in the application is between 25 and 50.
- The number of subtasks in each application task is between 5 and 10.
- The computation time for each subtask is between 100 and 650 seconds.
- The degree of interaction between the subtasks of different tasks is high.

To run the applications, a heterogeneous architecture formed by two clusters interconnected by a switch is used. The first (*cluster 1*) is composed by 25 processors (Pentium IV 2.4Ghz, 1Gb RAM), and the second (*cluster 2*) by 25 processors (Celeron, 2 GHz, 128Mb RAM). The connection is made through a 100-Mbit Ethernet network. This architecture was chosen so that the clusters forming it are of different characteristics in terms of the processors' computing power.

Various tests were carried out, identified by two parameters: the computing power of the subset of processors used (*WPT*), and the number of applications used (*AN*). That is, the test *WPT* - *AN* uses the first *AN* applications of *Q* and runs them over an architecture whose total computing power is *WPT*. The applications of the experiment reach it at different times (between 100 and 600 seconds difference between two successive applications), so that the following is observed: $S_i < S_{i+1} \quad \forall i \in [0..148]$

6 Results

To assess the behavior of the *dynamic concurrent scheduling* algorithm (*DCS_AMTHA*) when the architecture and/or the number of applications in the system are scaling, the *speedup and efficiency* parameters are analyzed (in this case, over heterogeneous architectures).

The speedup metrics is used to analyze the algorithm performance in the parallel architecture as indicated in Equation (4).

$$Speedup = \frac{SequentialTime}{ParallelTime}. \tag{4}$$

In heterogeneous architectures, the "*Sequential Time*" is given by the time of the best sequential algorithm executed in the machine with the greatest calculation power. In this multiple application case, the time each application requires is added. "*Parallel Time*" is the whole system end time.

To evaluate how good the speedup obtained is, efficiency is calculated. To this aim, the speedup obtained is compared with the total computing power (*WPT*) of the architecture upon which work is being carried out (which determines the theoretical speedup), as indicated in Equation (5).

$$Efficiency = \frac{Speedup}{WPT}. \tag{5}$$

The total computing power considers the relative calculation power of each machine with respect to the power of the most powerful machine. The *WPT* is calculated with Equation (6), where *#P* is the number of machines of the architecture

used, and WP_i is the relative calculation power of machine i regarding the best machine power.

$$WPT = \sum_{i=1}^{\#P} WP_i . \quad (6)$$

Figure 3 shows, for the most representative tests, the speedup achieved by the whole system when scheduling was done with the DCS_AMTHA algorithm. As it can be observed in the graph, speedup increases as the number of applications *NA* increases, with a tendency towards stabilization from *NA*=120.

On the other hand, it can also be seen that speedup increases with computing power *WPT* up to a point where increasing this power does not yield a significant improvement (*WPT* ≈ 30).

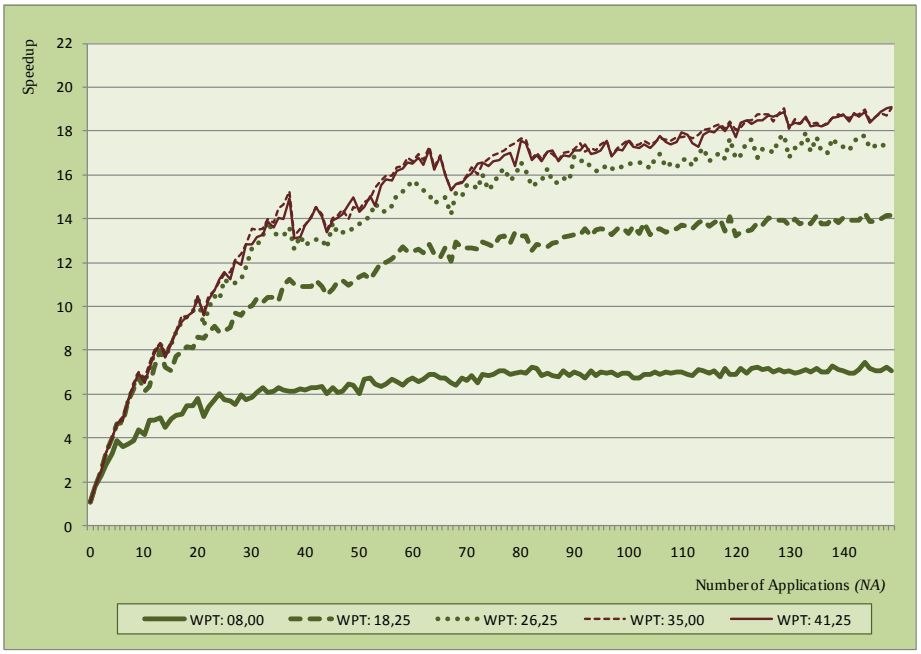

Fig. 3. Speedup obtained from each test.

Figure 4 shows the efficiency achieved for the same tests represented in Fig. 3. The same as with speedup, efficiency increases with *NA*, and it stabilizes after *NA*=120. However, unlike speedup, efficiency decreases as the architecture grows (*WPT*).

By combining both results, the limit up to which *WPT* can be increased to achieve relevant improvements in response times can be inferred. For these particular tests, this limit is given by an architecture with *WPT* = 30.

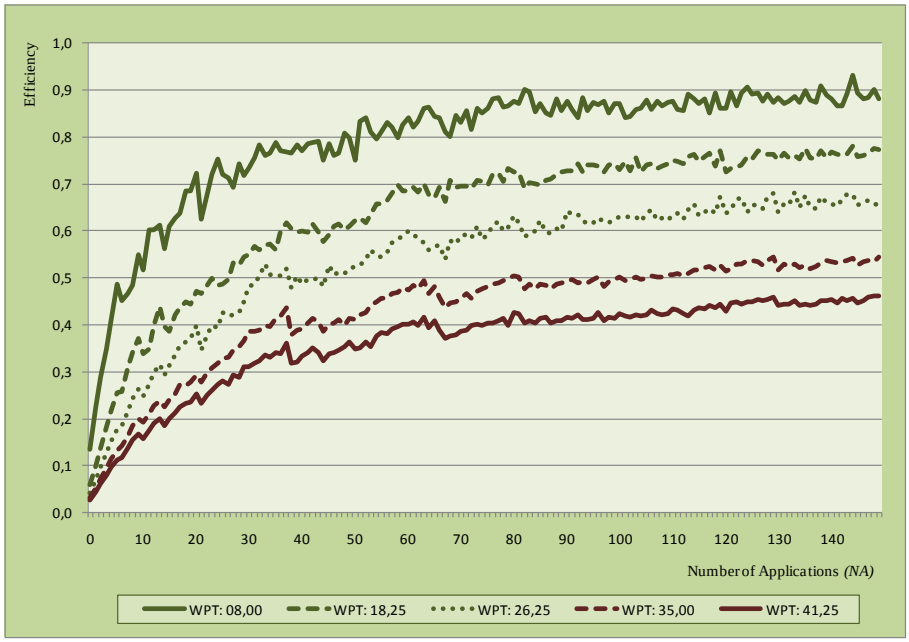

Fig. 4. Efficiency obtained from each test.

7 Conclusions and Work Guidelines

DCS_AMTHA, a dynamic scheduling strategy for multiple applications on heterogeneous multiprocessor architectures, is defined. It is based on AMTHA task-to-processor assignment on heterogeneous architectures.

Experimental results for different processing configurations and different working loads are analyzed, focusing on the global efficiency obtained.

Within the future work lines, research will be carried out on the evaluation of the scheduling algorithm presented (*DCS_AMTHA*) on clusters of multicore processors, taking into account the particular features of such architectures.

References

1. Grama A., Gupta A., Karypis G., Kumar V., "An Introduction to Parallel Computing. Design and Analysis of Algorithms. 2nd Edition". Pearson Addison Wesley (2003).
2. Kalinov A., Klimov S., "Optimal Mapping of a Parallel Application Processes onto Heterogeneous Platform". Proc. of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), pp 123. IEEE CS Press (2005).

3. Leopold C., “Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches”. Wiley, New York (2001).
4. Attiya H., Welch J., “Distributed Computing: Fundamentals, Simulations, and Advanced Topics. 2nd Edition”. Wiley-IEEE, New Jersey (2004).
5. Topcuoglu H., Hariri S., Wu M., “Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing”. IEEE Transactions on Parallel and Distributed Systems, vol. 13, pp. 260-274 (2002)
6. Goldman, “Scalable Algorithms for Complete Exchange on Multi-Cluster Networks”. CCGRID'02, pp. 286 – 287. IEEE/ACM, Berlin (2002).
7. Roig C., Ripoll A., Senar M.A., Guirado F., Luque E., “Modelling Message-Passing Programas for Static Mapping”. Euromicro Workshop on Parallel and Distributed Processing (PDP’00), pp. 229--236, IEEE CS Press, USA (1999).
8. Roig C., Ripoll A., Senar M., Guirado F., Luque E., “Exploiting knowledge of temporal behavior in parallel programs for improving distributed mapping”. EuroPar 2000. LNCS, vol. 1900, pp. 262--71. Springer, Heidelberg (2000).
9. De Giusti L., Chichizola F., Naiouf M., De Giusti A. “Mapping Tasks to Processors in Heterogeneous Multiprocessor Architectures: The MATEHa Algorithm”. International Conference of the Chilean Computer Science Society, pp. 85-91. IEEE CS Press (2008).
10. De Giusti L. “Mapping sobre Arquitecturas Heterogéneas”. PhD Dissertation, Universidad Nacional de La Plata (2008).
11. Cuenca J., Gimenez D., Martinez J., “Heuristics for Work Distribution of a Homogeneous Parallel Dynamic Programming Scheme on Heterogeneous Systems”. Proceeding of the 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar’04), pp 354-361. IEEE CS Press (2004).
12. Cunha J.C., Kacsuk P., Winter S., “Parallel Program development for cluster computing: methodology, tools and integrated environments”. Nova Science Pub., New York (2001).
13. De Giusti L., Chichizola F., Naiouf M., De Giusti A., “Scheduling Strategies in Heterogeneous Multiprocessor Architectures using AMTHA Algorithm (DCS_AMTHA, SCS_AMTHA, SS_AMTHA)”. Technical Report, March 2009.