A GENERIC SMA FOR MULTI CRITERIA OPTIMIZED ALLOCATION OF APPLICATIONS ON HETEROGENEOUS ARCHITECTURES

Saouli Rachida , Akil Mohamed** ,and Henni Abderrazak ****
** Computer science department, BP 145, Mohamed Khider University, Algeria.*
Email: saoulir@esiee.fr
** ESIEE, BP 99 2 BD Blaise Pascal, F 93162 Noisy le Grand cedex, France
akilm@esiee.fr
*** INI, Oued Smar, Alger 16000, Alegria
henni@ini.dz

Abstract

We are interested in this work, by the optimized static allocation multi criteria in a real time distributed system when the tasks are subject to precedence's constraints. Within this framework it is necessary to suppose before execution, that all the possible scenarios of executions satisfy the temporal constraints while minimizing the cost and the size of material architecture, as well as the use at best of its resources. In this problem of resource allocation (placement and scheduling), which is NP-Complete, the satisfaction of several criteria can be contradictory. For the resolution of this problem, we propose in this work, a generic multi agent system, which is a dynamic component, coupled with a static strategy of scheduling in order to particularly integrating the criterion of load balancing. Thus, the need for a dynamic model appeared to us with the consideration of the heuristic based on list scheduling [1], [12]. An experimental Analysis was realized under programming parallel environment PVM (Parallel Virtual Machine)}, and shows the interest of our method. This for any heuristics using the dates of execution of the tasks (operations) in particular for method AAA (Algorithm, Architecture, Adequacy) developed with the INRIA and which was the subject of several extensions.

Keywords: static placement/scheduling, load balancing, system distributed real time, multi agent systems

1. Introduction

The real time systems are founding in fields such as aeronautics, control of industrial processes or telemedicine also embarked systems are in systems of brakes control and engines. These critical systems, whose majority consists of treatments, must imperatively respect all their temporal constraints. A way to satisfy these real time constraints lies in the use of parallel machines multiprocessors, which are possibly heterogeneous. In the study of the real time distributed systems the problems of placement and scheduling are simultaneously encountered and if the model of the graph's algorithm is guided by precedence, it is the total execution time of the system which is considered not only or without [12] load balancing. Thus within this framework we consider the list scheduling algorithms where, in each stage, a choice is made for the most advantageous placement (faster and satisfying a criterion) and for a given operation [9]. This adapts perfectly to the systems subjected to temporal constraints. They make it possible to select one processor, for an operation given by primarily using two functions. They correspond to the start and the end dates of the operation on the operator, and load balancing is performing progressively. Then the algorithm maintains up to date two lists: one containing the ready tasks and the other unoccupied processors. It chooses a new task to assign with the free processor (minimal utilization ratio) for which imbalance after placement is minimal. The choice of the task to be scheduled is more difficult. It defines a priority between the tasks available to a given moment. This priority is a function of the temporal characteristics of the tasks or/and structural of the tasks graph and so that the expiries are respected. The selection criterion of a task, takes into account in this case the urgency of the task. In [13], the attribution of a priority is according to the task's duration time and the maximum of its successors duration time, but this method gives satisfactory results when the communications are null. Extensions to this work [14] duplicate tasks, what increase the processors use ratio. Other work based on a dating of the events; the rule giving the priority can take into account the duration of the task [10] or its date of activation at the latest, without considering the constraints neither of precedence nor over resources. In [9] according to the date of reception of the messages, the earliest ready task on a given processor, is selected what reduces the choice of the processors and makes difficult the load balancing. In addition, the selection criteria of a task can been restricted by constraints of placement, which limits the selection of a task. These constraints can be related to the heterogeneous operators [3] intended execute different operations (tasks) who's urgency to be scheduled, follows a function cost called pressure of scheduling. This function corresponds to a difference between the values of penalty and flexibility but without to consider the balancing of load. In [6] the constraints of placement relate to the mechanisms of faults tolerance. Two criteria are taking into account, and are varying dynamically with an order of evaluation, at each request for reconfiguration. Moreover, the heuristics based on these criteria gives a low having imbalance only, for configurations of the too strong constraints, related to the bonds available enter the nodes.

Then we consider that in the problem of resources allocation, the application of several criteria was not solve in the literature by traditional linear methods. To solve this problem we propose in this work a new methodology, based agent whose model is present in *section 2*. *In section 3,* we consider our model coupled particularly with heuristic developed at INRIA [12], [3], in order to integrate initially the load-balancing criterion not considered yet. In the section *4,* we present the environment of simulation under PVM as well as the results obtained.

2. Multi agent system multi criteria (SMA-MC)

The methodology used, is basing on the concept of agent. Among the definitions for the concept of agent, we use the one that defines the agent as an autonomous entity, real or abstracted. It is able to act on itself and its environment. In a multi agent universe, it can communicate with other agents. The behavior is the consequence of its observations, its knowledge and interactions with other agents [4].

2.1. Conceptual structures of agents

An agent is located in an environment. To model the structure of the agent, it is necessary to have a model of this environment. The latter can be in a state among a whole of states. It can change its state either in a spontaneous way or like result of the actions of the agent. The evolution of the environment is modeling differently, according to its characteristics, which one takes into account, and simplifications that one is authorized [5].
The characteristics of the environment influence the way in which one designs an agent because it is necessary to take account the environment's evolution, and the capacity of the agent to seize this evolution.

2.2 The general model of the SMA

The system proposed is based on:

- *A Meta agent*: from one placement/scheduling heuristic Hi, characterized by use of specifications models of tasks algorithm to allocate, and architecture target of processors, the meta agent create (instancing) agents of system. It constitutes *accountancies* of each agent (sure knowledge that an agent has on the other agents of its environment). It establishes the model (*ENV)* of its environment, which corresponds to allocation model. It defines the whole of objectives to reach, as well as the set of the strategies using communication actions. We define the knowledge of the Meta agent as being the whole of the Meta rules, which make it possible to adapt heuristics of placement/scheduling to its SMA.

- *A generated SMA:* where each agent corresponds to a processor composing the model of distributed architecture used in the considered heuristics. This fact by using the base of knowledge BC, the Meta agent generates the system of cognitive agents [8], which act in the same definite environment. Each agent will perceive the environment like dynamics and nondeterministic. The environment state can change after actions of other agents or of environment, and the same action in a certain state will have different results according to the actions from the other agents.

Detailed description of the agent

To determine the agent's actions of this system, it is necessary to define them like their interactions [2], with environment and agents. The definition of an agent bases: on the one hand of a universal definition of cognitive agents, and on the other hand of a detailed definition corresponding to the logical model. This latter, bases on architecture BDI (Believe, Desire, and Intention) [8], and presented in the Figure1.

Figure 1

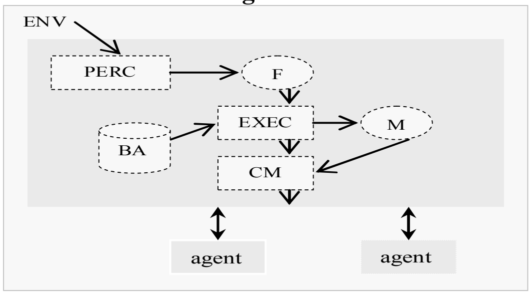

Figure1. The logical agent's model

In Figure1, we represent the environment ENV by a plan surrounding the agents that are located there. Each function of the agent is representing by a rectangle surrounding its name. The set of knowledge of the agent (strategies: knowledge and objectives that are Desires and Intentions) corresponds to knowledge base *BA.* The result of functions modeling an agent is representing by an oval. In addition, double direction arrow represents interaction enters agents and the environment.

The functions of the agent's model locat in its environment are defining below:

- *PERC: ENV → F* is the function, which perceives the model of the environment *ENV,* and establishes the facts F, which are the *beliefs* of the agent.
- *EXEC: S x F x B → M* is the kernel function, which allows the achievement of the objectives *B* and established the change(s) M on the environment. This is according to strategies *S* available to the agent, to established facts *F.*
- *CM: M x C → ENV* is the function, which makes it possible the communication of the change (actions) on the environment according to the established *accointances.*

S***trategies*** of each agent are the set of rules, which, allow to make a change on the environment, and to perform interactions (*SI)* with the agents of its *accountancies.* An interaction allows generating a new model of the state of the environment (*S1)*, sending tasks since and from other agents (*S2)*, sending messages of information or a new decision (*S3),* and modification on the model of environment (*S4).*

Interaction between the agents of this model is realizing according to their *accountancies*. The agents concerned with this interaction are those, which satisfy the same constraints on the placement/scheduling, or bound by a neigh board's relation. This interaction is performing with the use of a master agent, which is supervising execution of the slave agents' functions. The master maintains, for a certain time the objectives until their achievements (limited obligation) by slave agents of the system. The result of the perception of environment implies the possibility that the selected objective can be still realizable, or possibly changes by another according to the result of actions performed on the environment. In this case, the takeover by a master agent implies master slave relations with the agents of the system, by guiding their actions and their wait, as well as the resolution of possible conflict generated by the mechanisms of decisions of each agent slave.

3. The criterion of load balancing and the methodology AAA

Within the framework of real time distributed applications, the methodology AAA (Algorithm, Architecture, Adequacy) developed with the INRIA *heuristic* (H_A) [3] takes into account all the steps of the development of an application, of its high level specification until the execution of the code in the components. An optimized allocation is obtaining by transformation of graphs. It corresponds to one static placement/scheduling of elements of the algorithm on target distributed and heterogeneous architecture. The latter is modeled by a directed graph constituting a network of automates. The set of the nodes of a graph, defining in this case a processor, is of four types: operator, transfer, memories and bus. The required solution bases on *the latency,* which is in direct relationship to minimization of the communication's cost and corresponds in this case to the length of the critical path of the allocation graph. This methodology bases on a preliminary characterization of the elements of the graph of algorithm and architecture (maximum execution times of the operations on each operator able to execute it …). It makes it possible to predict the behavior of the application and to build its executive distributed and optimized. Nevertheless, the predictions off lines, in particular the duration time of the operations on the operators of architecture, can be deviated of the real values at the execution time. Indeed these durations, which are at worst case, do not adapt to the data (changes) of execution what can land the system in degradation: an important imbalance in the use of the resources, a not respected latency implying a static adaptation and new execution of the allocation.

The definition of the objective (criterion) of load balancing is relating to the improvement of the establishment in heuristics H_A considered. The latter bases on a cost function, which evaluates the urgency to schedule an operation so that the higher it is for an operator, more the value of corresponding flexibility is small and more the critical path is lengthened (penalty). It is thus a question of scheduling each operation candidate on each operator able to execute it, and retain the operator minimizing the function cost used. Our system makes it possible to integrate the objective defined according to Figure2.

Figure 2

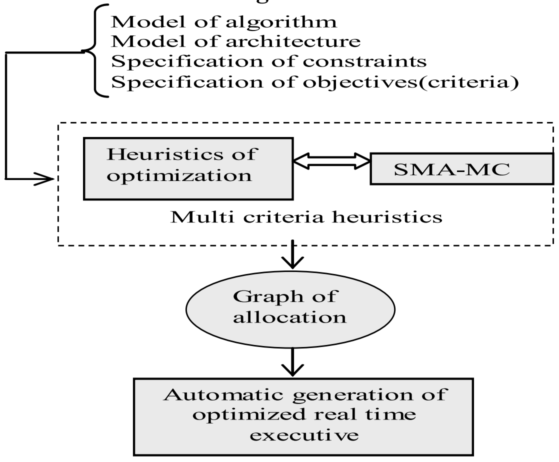
Model of algorithm
Model of architecture
Specification of constraints
Specification of objectives(criteria)
Heuristics of optimization
SMA-MC
Multi criteria heuristics
Graph of allocation
Automatic generation of optimized real time executive

Figure2. Mixed system of optimized allocation

The SMA-MC generates the corresponding SMA and the Meta *Agent* establishes:

- the slave agents *sl* and the main agent,
- the *accountancies* of each agent while basing on the constraints of placements,
- the model of the environment state *ENV that* corresponds to the diagram of the allocation sequence,
- the objectives considered (load balancing), and the strategies (sets of rules or methods) for there realization.

Duration times used: In our method, the strategies of the agent base on a principal characteristic, which relates to the rule of use exact durations of operations on the operators, and not an approximation with the average duration as it is the case in heuristics H_A considered. Indeed let us recall that in the static allocation, if the exact durations can be possibly deviated of the real values at the execution time, the approximate values will only increase these deviations. These execution times correspond to an approximate time calculated for each *Oi* operation, by an average compared to the number n of *Opj* operators able to execute it according to the expression:

$\Delta_{app}(Oi) = 1/n \sum \Delta(Oi , Opj)$ such as

$\Delta(Oi,Opj)$ is the exact duration characterized in H_A.

In addition, the earliest end date E(Oi) of an operation Oi corresponds:

$E(Oi) = S(Oi) + \Delta_{app}(Oi)$ such as S(Oi) is the earliest start date which is the greatest date E of its predecessors.

From these dates, the *critical path R* of the graph of algorithm is the duration of the longest path:

R=Max [E(Oi)].

Consequence of the use of exact durations in our method, is that we characterize each operation Oi by a start date S (Oi,Opj) on each operator Opj.

Principle: the approach proposed aims to balance the loads of the operators to each stage of the heuristics of scheduling. For this, we based on the principle of the dynamic method of balancing *SID (Sender Initiative diffusion*). However, we have modifying the quantity (=1) of operations to be negotiated for a new scheduling. We have regarding the field of scheduling as being constraints of placement. Finally, we have adding the execution kernel *EXEC,* to take into account the precedence of the operations. It is thus a question of negotiating the migration of only one operation among those allocated with an agent overloaded according to the following mechanism:

- *Perceive scheduling information.*
- *Each agent estimates its local load: an overloaded agent disseminates information of balancing to the agents of its. Accepted operation generates a minimal scheduling path, which does not lengthen the critical path with this stage.*
- *Resolves conflict and decision of the scheduling of the selected operations.*

The result of the perception *PERC (ENV)* of the environment makes it possible to establish facts for each agent slave: h*is local load*, *the length of local scheduling*, as well as the *average load* in a number of allocated operations defined by:

$\sum$*load (sl) / n* where *n* is the number of agents slaves *sl* pertaining to the *accountancies.*

The set of the rules of agents' strategies is applying by kernel function *EXEC* defined below by the functions *EXEC1* and *EXEC2* of each agent and makes it possible to evaluate its load *LU* compared to a value (threshold) preset by using the average of the loads:

Any overloaded agent communicates to its group, the preceding operation maximum earliest end date Emax. This date corresponds to each operation allocated to him (according EXEC1).

Any discharged agent tests the operation, which does not lengthen critical path Rn with this stage, based in the exact duration time and the local scheduling path Rlocal (according EXEC2).

Using information received from each agent slave, the master agent applies the actions of change *SI* of environment. Therefore, for any operation concerned with the action of (*new*) allocation, a slave will be select after resolution of conflict. The decision of the master is according to whether this operation generates a new minimal local way *Rnouv* on a slave.

The resolution of conflict relates to the case where a several slave agents select the same operation and in this case, the chosen operation is that which checks:

Min (Rn - Rnouv) such as *Rn* is the path obtained from stage n of the heuristics.

Function 1: ***EXEC1 (F, B, S)***

```
REQUIRE: LPRED {lists of operations scheduled with its preceding ones}
REQUIRE: Q {list of the operations scheduled on the agent overloaded}
REQUIRE: LCRIT {list of operation in critical path}
ENSURE : Emax
ENSURE: M {actions of communication}
If (LU –Moy_load) > threshold then
  For all O such that operator (O) = sl do
     Emax = maximum (LPRED)
     {The maximum earliest end date
     of the precedents operations}
  End For
M = (S3, Emax (Q)) {return the list of Emax of each operation locally scheduled}
End If
If Oi in LCRIT then
Emax = maximum (LPRED (Oi))
{Operation Oi scheduled with this stage}
M = (S3, Emax (Oi))
{Return Emax of the precedent of Oi}
End If
```

Function 2 EXEC2 (F,B,S)

```
REQUIRE: Q {list of the operations scheduled on the agent overloaded}
REQUIRE: Emax
REQUIRE: Δ (O, sl) {exact duration time of each operation O scheduled on agent sl}
ENSURE: M {actions of communications}
If (LU –Moy_load) < 0 then
  For all O in Q do
   If (Emax (O) < Rlocal)
   and (Rlocal + Δ (O, sl)) < Rn then
   Rnouv = Rlocal + Δ (O, sl)
   min (O) = MINIMUM (Rnouv) {retain
   the operation which does not lengthen the
   critical path}
   End If
   If (Emax (O) > Rlocal) and (Emax (O) +
   Δ (O, sl) < Rn) then
   Rnouv = Emax (O) + Δ (O, sl)
   min (O) = MINIMUM (Rnouv)
   End if
   End for
   M = (S3, min (O)) {communications'
   Actions of the local path min generated}
End if
```

4. Experimental Analysis under PVM

PVM (Parallel Virtual Machine) is a communication system, made up of a library and a demon. It makes the communications independent of the operating system, where applications use whole machines, potentially heterogeneous and inter-connected by a communication network, like only one computer. Execution applications on *pvm* distribute tasks on one or more computers constituting the parallel machine. Each task of the virtual machine can emit an unspecified number of messages toward any other task of the machine. Thus, *pvm* library consists of two principal parts. The first one, which is independent of the operating system, gathers all the functions of management of the tasks *pvm* and of the groups as well as the high-level communication functions. The second, moreover low level, includes the transfer's functions of the packages on the network, recognizes the system of the host machine, and adapts to his architecture. *XPVM* is the X-Windows version of *pvm*; it is a representation of machine PVM with the name(s) and type(s) of computer(s) composing it. This menu of window make it possible to control the tasks (*menu Tasks*), to control the machines (*menu Hosts*), to stop pvm (*menu Reset, Quit, Halt*) like having of the assistance (*menu Help*).

We have realized our simulation by using *xpvm* under a host *Linux*: the principal task (*level n0*) is the *task xpvm,* which enabled us to activate the execution of the system's agents. In level n1*,* a group of tasks, main agent (T_M) and k agents slaves (T_sl), are creating and synchronizing with mechanism of *barrier*. This later makes it possible to block all the agents on this level, until the specified number of agents in simulation was been activated, while arriving at the same point of equivalence. In addition, it allows a dynamically manage of addition other members (agents) to the group. The spawn associates in this level an identifier (TID) with each task created (T_M, T_SL) and which will make it possible pvm to recognize it in the virtual machine. In addition, we used an index *me* of membership of the group to identify each task agent member, value 0 allows to initially launching the execution of the main agent (T_M) of our system. The execution scheme (*level n2)* realized in our method is presented in Figure3.

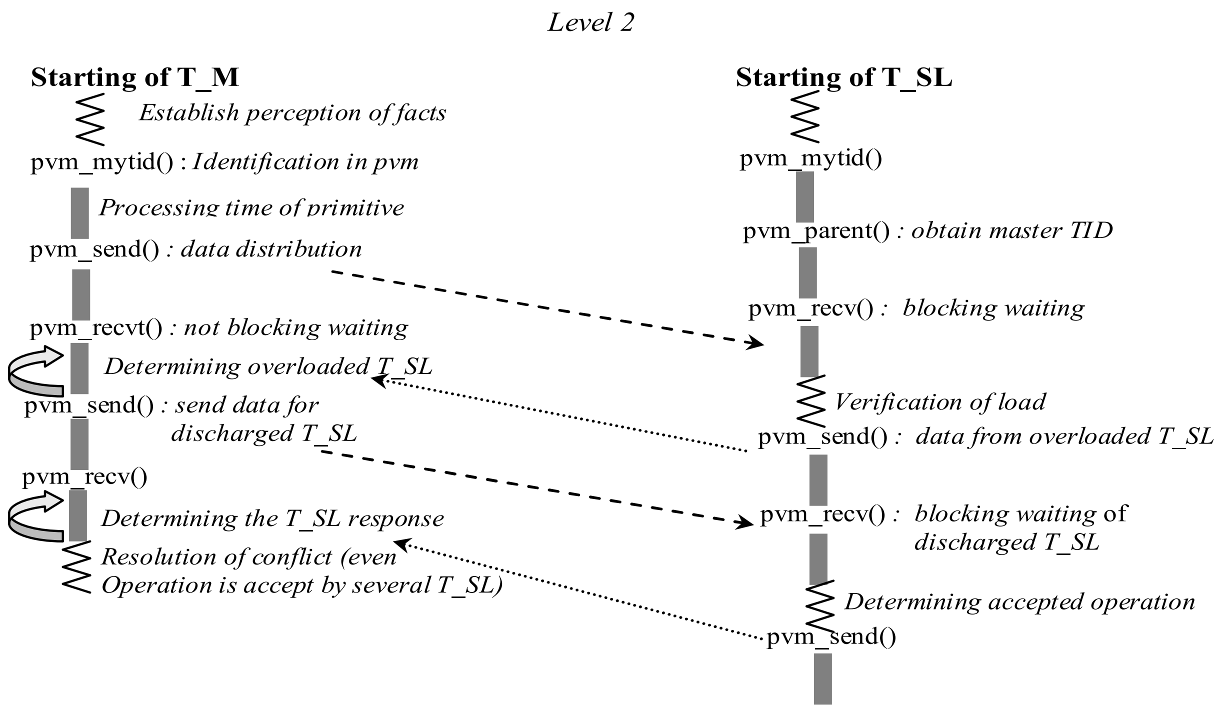
Level 2
Starting of T_M
Establish perception of facts
pvm_mytid() : Identification in pvm
Processing time of primitive
pvm_send() : data distribution
pvm_recvt() : not blocking waiting
Determining overloaded T_SL
pvm_send() : send data for discharged T_SL
pvm_recv()
Determining the T_SL response
Resolution of conflict (even Operation is accept by several T_SL)
Starting of T_SL
pvm_mytid()
pvm_parent() : obtain master TID
pvm_recv() : blocking waiting
Verification of load
pvm_send() : data from overloaded T_SL
pvm_recv() : blocking waiting of discharged T_SL
Determining accepted operation
pvm_send()

Figure3. Model of the agents' execution

Restrictions were been made because of indeterminism in the creation of the tasks pvm. Then we have limited our tests to a scenario of extreme imbalance where all operations are been scheduled on only one slave agent. This is why a reduced number (2) of slave agents, was considered. Simulation uses a file test of entries using allocations graphs examples, generated by the heuristics considered. This determines the perception function, realized in this diagram by the main agent. The files tests are been structured in tables, which characterize for each operation, the date's values below:

- *Earliest start date S of scheduled operation on allocated agent*
- *Precedents operations of scheduled operations,*
- *Exact duration of operations on agents*
- *Average duration time in the group*

The result of simulation relates to the determination of the operation and the new operator (agent slave) of scheduling in a stage of the heuristic. Indeed, in our case, we use the exact duration over the slave and not the average duration as in heuristic H_A, to test the acceptance of an operation by, a slave. Let us recall that they are here operators modeled by agents. An agent is overload according to following rules:

- *R1: If (local load - average load in the group)> threshold (=1)* then *process* corresponding to the imbalance generated by H_A in the allocation.
- *R2: If (an operation belonging to the critical path was scheduled with this stage)}* then *process corresponding to temporal* imbalance because of use of average duration in H_A.

Process is the consequence of application of the rules R1 or R2, which allows the agent discharged to determine the operation of which it can be its new agent (operator) of scheduling.

Application of rule R2: That is to say, the sequence of scheduling Figure 4(a) generated by heuristics H_A such as operations Oi: (A, B, C, D) scheduled on two operators Op1 and Op2 and characterized by:

- Diagram of precedence: PRED(B)=PRED(C)=A and PRED(D)=C,B
- Average durations times:
  $\Delta_{app}(A) = \Delta_{app}(C)= \Delta_{app}(D)= 1$
  $\Delta_{app}(B)= 4$

Evaluation of balancing by our system: We have established the values of duration time below, to obtain the same duration time's average Δ_{app}:

	Δ_A	Δ_B	Δ_C	Δ_D	S_A	S_B	S_C	S_D
sl1	1	5	1	1	0	1	1	6
sl2	1	3	1	1	0	1	1	4

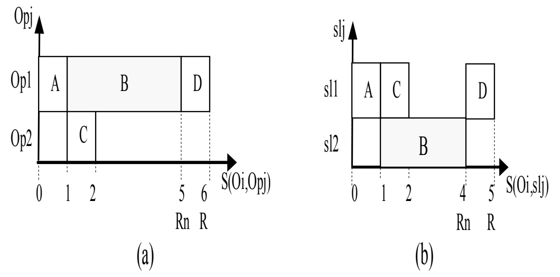

Figure4. Diagram temporal before (a) and after load balancing (b)

Our method relates to the stage of scheduling operation B that belongs to the critical path and its flexibility is null. Thus, what determined its urgency to be scheduled by H_A. After it's scheduling and with this stage, in our method it is the slave agent sl2, which will be concerned and applied the rule R2 by using the exact durations such as:

Min (Δ(B, sl1), Δ(B, sl2))=3 and Emaxpred > Rlocal_{sl2}= (1 > 0)

What implies Rn - (Emaxpred + 3) > 0: the critical path Rn corresponding to this stage, does not increase. This agent (sl2) will accept operation B. Here Emaxpred is the end date of the operation A which is the preceding operation of B. Recall that these values (Emax) are calculated and transmitted in our method, by the agent overloads and used by the slave discharges to test if the operation can be scheduled in it.

Figure 4(b) presents the result of the primitive *pvm_send ()* generated by the agent sl2. It shows the consequence of this result on the following stages of scheduling of operations C and D.

Application of the rule R1: We illustrate in this example, the application of the rule R1 by our method. It is about the stage after scheduling of three no critical operations Oi:(A,B,C) characterized by:

- Diagram of precedence given by:
  PRED (B) = A, PRED (C) =B
- Values of duration times Δ and Starting date S:

	Δ_A	Δ_B	Δ_C	S_A	S_B	S_C
sl1	1	4	1	0	1	5
sl2	1	4	1	0	1	5

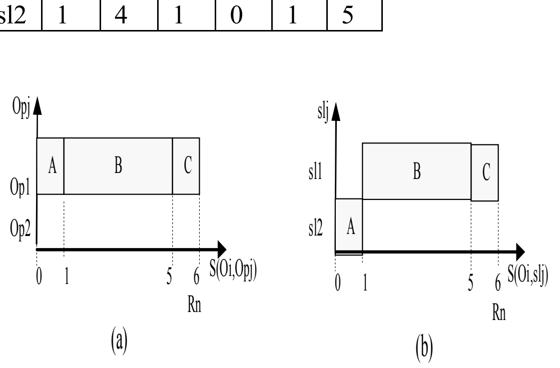

Figure5. Diagram temporal before (a) and after load balancing (b)

Figure5 (a) presents the temporal diagram of the sequence of scheduling obtained by heuristics H. Figure 5(b) presents the result of our method after the application of the rule R1 for this stage. It is thus the agent sl1, which is overload such as:
(LU_{sl1} – average load$_{)}$ = 3 – (3/2) > 1. Operation A is that accepted by the agent sl2, for it generates a local path minimal (Rlocal) on this agent.

Thus let us note that the improvement of the heuristics, by the integration of the criterion of load balancing allows a better use of operators and/or a profit in the length of scheduling sequence, if the operation concerned with balancing is characterized by a smaller exact duration. Possibly if the rules R1 and R2 are not applied, a decision can related to the reduction of the size of architecture according to operators' which are allocated with no operation.

5. Conclusion

We considered in this work that the problem of the integration of several criteria, to each stage of the placement/scheduling heuristics, is presented in the form of an open system (parallel, asynchronous, indeterminism). We propose in this work a mixed system. The latter 'cohabits' a dynamic model based cooperative agent with a static model of allocation. Initially, the criterion of load balancing was introducing with taking into account, precedence over operations. These latter are connecting to real time distributed systems that base on the assumption of synchronism extremely related to the logical succession of events and decisions. The system that we propose is generic which, through the Meta agent will try *to learn* in the past and to found, which and if a criterion is to performing at each stage of the heuristic. These remain then to solve the problem of *the adaptation* of the multi agent system for heuristics $H_{A,\ or}$ others in order to introduce well other criteria not taken into account such as the degree of freedom [7], the fault-tolerance.

REFERENCES

[1] Bertrand Braschi. Principle of list scheduling algorithms with tasks assignment priorities. Doctorate Thesis. University of Grenoble. Nov 90.

[2] Chaïbdraa B. Interaction between agents in routines, familiar and unfamiliar situations. International Journal of cooperative Information systems (5): 1-25. 1996.

[3] Thierry Grand Pierre. Modeling of heterogeneous parallel architectures for automatic generating an optimized real time distributed executives. PHD Thesis. Paris XI Orsay University. Nov 2000.

[4] Jacques Ferber. Multi agent systems towards a collective intelligence. Inter Editions. 1995.

[5] Florea Adina, Daniel Kayser and Stefan Pentiuc. Agents Intelligents. Web course. Polytechnic University of Bucharest. 2002.

[6] Lanet.J. Tasks allocation in a distributed system. RTS 95, pp 222-231, Paris, Jan 95.

[7] LU and Carey. Load balanced allocation in locally distributed computer systems. Computer Science Technical Report 633. University of Wisconsin. Feb 1986.

[8] Jean Pierre Müller. Organizational Modeling multi agent systems. S.E of ARCo. jui2000.

[9] Shepard Gagné. A Pre Run Time Scheduling algorithm for hard real time systems. IEEE transactions on software. Vol 17. n7, pp 669-677. Jui 1991.

[10] Shirazi, Wang. Analysis and evaluation of heuristic method for static task sheduling. Journal of Parallel and Distributed Computing. N.10. pp 222-232. 1990.

[11] Munteau Traian, Talbi El Ghazali. Method of static processes placement on parallel architectures. TSI V.10. n5, 1991.

[12] Vicard.A. Formalization and optimization of distributed real time embarked systems. PHD Thesis. University of Paris-XIII, Galileo institute, JUI.1999.

[13] Wu and Sweeting. Heuristic Algorithms for task Assignement and Scheduling in Processor Network. Parallel Computing, n20, pp1-14. 1993.

[14] Yang and Gerasoilis. List Scheduling With and Without Communication Delays. Parallel Computing. n19, pp 1321-1344. 1993.