Journal of Theoretical and Applied Information Technology
15th November 2011. Vol. 33 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

A TASKS ALLOCATION ALGORITHM FOR DISTRIBUTED SYSTEMS

[1]MOSTAPH ZBAKH, [2]MOHAMED DAFIR EL KETTANI

Information Security Research Team Laboratory

Université Mohamed V - Souissi

Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes-ENSIAS,

Avenue Mohammed Ben Abdallah Regragui, Madinat Al Irfane, BP 713, Agdal

Rabat, Marroco

E-mail : [1]zbakh@ensias.ma, [2]dafir@ensias.ma

ABSTRACT

In this paper, we complete and implement the general problem of tasks allocation. The theoretical part of this work is developed in our previous work [14] and here we focus our research on the practical aspect.
The problem is modelled as a non-cooperative game between several players. For this game, we adopt the Nash equilibrium structure and on the basis of this structure, we draw a distributed tasks allocation algorithm that can find this equilibrium. We implement this algorithm by using the MPI environment and a system of 10 computer sources that generate the tasks and 10 processing computers. The theoretical and numerical results show that the tasks allocation strategy obtained leads to a good load balancing of computers.

Keywords: *Tasks Allocation, Game Theory, Non-Cooperative Game, Nash Equilibrium, Distributed Algorithm, MPI Environment.*

1. INTRODUCTION AND RELATED WORKS

In recent years, heterogeneous systems have become a key platform for the execution of heterogeneous applications. The major problem encountered when programming such a system is the problem of tasks allocation. A good allocation of tasks leads to a good load balancing of the system. Several articles deal with the problem of load balancing and routing taking into account the characteristics of communication links between machines. For example, in [3], the authors address the problem of load balancing on linear platforms and in [4, 7, 8] the authors address the problem of routing in a network of several parallel links with an origin and a destination machine. In [5, 12], the authors seek a routing strategy that allows the balancing of a heterogeneous system.

The general formulation of such a problem is as follows. We assume that we have a set of any m machines and n tasks (selfish) of sizes T_1, T_2... T_n. We suppose that the jobs are divisible and each one can be processed by all the machines M_i (i = 1... m). The load L_i of a machine i is defined as the sum of execution times of tasks which it treats and the cost of a task as the sum of loads of the machines that treats it. The general problem of tasks allocation is to find an allocation that minimizes the costs of tasks

To clarify the idea of allocation of tasks on homogeneous machines, consider the following simple example where the jobs are not divisible and each one is processed only by one machine. We consider two identical machines (M_1 and M_2) and five tasks with execution times 1U, 2U, 3U, 4U, and 1U (U = unit of time).

We consider the allocation shown in Figure 1 and in Table 1 we present the cost of each task obtained by

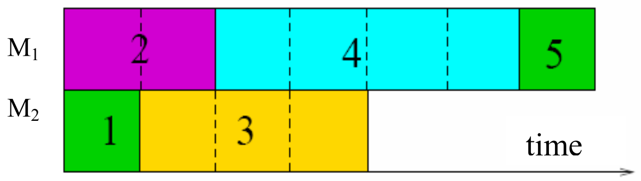


Figure1: First allocation

Task	Task1	Task2	Task3	Task4	Task5
Cost	4	7	4	7	7

Table 1: Cost of tasks achieved by the first allocation

this assignment. It is clear that a task can improve its cost by choosing the following assignment:

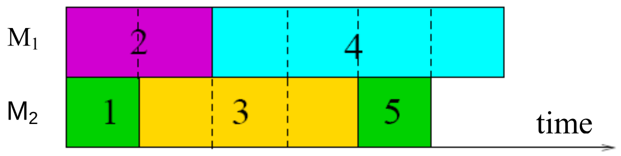


Figure2: Second allocation

Task	Task1	Task2	Task3	Task4	Task5
Cost	5	6	5	6	5

Table2: Cost of tasks achieved by the 2nd assignment

Several studies in the literature show the existence of such allocations on homogeneous machines (identical) without specifying the nature of this balance [4, 13]. Our goal here is to generalize this problem to any machines (homogeneous and heterogeneous) on the one hand, and find a structure of such an assignment on the other hand. To do this, we formalize this problem as a non-cooperative tasks allocation game.

This article is structured as follows. In section 2, we formalize this problem as a non-cooperative game and we derive a distributed algorithm for our tasks allocation in section 3. In section 4, we give an implementation for this algorithm by using MPI environment and in section 5 we draw a conclusion and perspectives for this work.

2. PROBLEM OF TASKS ALLOCATION AS A NON–COOPERATIVE GAME

Given n tasks of sizes T_1, T_2... T_n and m machines of speed V_1, V_2... V_m; each task should be handled by at least one of m machines. The load of a machine is defined as the sum of the execution times of these tasks and the cost of a task as the sum of the loads of the machines that handle it. Our goal is to find an allocation of these tasks that minimizes the cost of all tasks.

Let S_{ji} be a real between 0 and 1 which represents the portion of the job j processed by machine i. We call the vector $s_j = (s_{j1}, S_{j2}\ldots S_{jm})$ the allocation strategy of the task j (j = 1... n) and the vector $s=(s_1, s_2... s_n)$ the strategy profile for this tasks allocation game.

In order to modelize the response time of each machine, we assume that a scheduling exists and modelize each machine as a M/M/1 queuing system.

We also assume that tasks are distributed with a rate µ.

The response time of machine i is given as $t_i(s)=\dfrac{1}{v_i-\sum_{j=1}^{n} s_{ji}T_j\mu}$. The cost of a job j is therefore given as : $c_j(s)=\sum_{i=1}^{m} s_{ji}t_i(s)=\sum_{i=1}^{m}\dfrac{s_{ji}}{v_i-\sum_{j=1}^{n} s_{ji}T_j\mu}$.

Our goal is to find a feasible tasks allocation strategy (s_1, s_2… s_j ... s_n) which minimizes all $c_j(s)$. The decision of each job j depends on the decisions of other tasks since c_j is a function of s. Therefore, this strategy will lead to a good load balancing of machines.

Definition 1: A feasible strategy profile of tasks allocation is a strategy profile that verifies the following conditions:

1) Positivity: $s_{ji} \geq 0$, i =1,…,m; j =1,…,n
2) Conservation: $\sum_{i=1}^{m} s_{ji} = 1,\ j = 1\ldots n$
3) Stability : $\sum_{j=1}^{n} s_{ji}T_j\mu \prec v_i$, i = 1…m

Definition 2: The non-cooperative game of tasks allocation is a set of players, a set of strategies and preferences between the profiles of strategies. The players are the n tasks. Each task T_j has its set of feasible strategies for the allocation of tasks s_j, and the task j prefers the profile of strategies s than the profile s' if and only if $c_j(s) < c_j(s')$.

The solution to this problem is to find the Nash equilibrium [1, 2] for this allocation game.

Definition 3: The Nash equilibrium for this tasks allocation game [1,2,5] is a profile of strategies s such that for each task j (j=1 ... n):

S_j is such that

$$c_j(s_1, s_2, \dots, s_j, \dots, s_n) = \min_{\hat{s}_j} c_j(s_1, s_2, \dots, \hat{s}_j, \dots s_n)$$

.

In other words, the Nash equilibrium is a profile of strategies such that no player can improve its cost by choosing another allocation strategy.

For this game of tasks allocation there is a unique Nash equilibrium because the response time functions of the machines are continuous, convex and increasing [6].

To determine a solution to our game of tasks allocation, we consider an alternative definition of the Nash equilibrium: "Nash equilibrium can be defined as the profile of strategies for which the allocation strategy of each task is a best response to strategies of other tasks [5]". The best response of a task provides a minimum response time, assuming that the strategies of the other tasks are kept fixed. This definition gives us a method for determining the structure of the Nash equilibrium.

First, we determine the strategies of the best responses s_j for each task j, and then we find a profile of strategies $s=(s_1,s_2 \dots s_n)$ where sj is the best response of the task j, for j = 1, 2...n.
We begin by determining the best response of the task j, for j = 1, 2 ... n, assuming that the strategies of other users are always kept fixed.

Let $v_i^j = v_i - \sum_{k=1, k\neq j}^{m} s_{ki} T_k \mu$ be the available processing rate of the processor i as seen by the task j. The problem of calculating the best response strategy of the task j (j = 1 ...n) is reduced to the problem of allocating a single job on m machines having v_i^j as processing rates, that is to say, calculating the optimal allocation strategy for this task. This can be translated into the following optimization problem (Best_Response$_j$):

$$\begin{cases} \min_{s_j} c_j(s) \\ \text{under constraints:} \\ \quad s_{ji} \in [0, 1],\ i = 1, \dots, m, \\ \quad \sum_{i=1}^{m} s_{ji} = 1 \\ \quad \sum_{k=1}^{n} s_{ki} T_k \mu \prec v_i^j,\ i = 1, \dots, m \end{cases}$$

There are several algorithms for solving similar optimization problems as in our case which are based on Lagrange parameters. In [5.11], the authors have addressed the problem of optimization with the same objective function but with different constraints. We draw on this work to solve our optimization problem.

Theorem [14]: Assuming that the machines are ranked in decreasing order of their available processing rates $\left(v_1^j \geq v_2^j \geq \dots \geq v_n^j\right)$, the solution s_j of the optimization problem , Best Response, is given by:

$$s_{ji} = \begin{cases} \dfrac{1}{T_j}\left(v_i^j - \sqrt{v_i^j}\,\dfrac{\sum_{i=1}^{c_j} v_i^j - T_j \mu}{\sum_{i=1}^{c_j} \sqrt{v_i^j}} \right) & \text{if } 1 \leq i < c_j \\ 0 & \text{if } c_j \leq i \leq m \end{cases}$$

where c_j is the minimum index that verifies the inequality: $\sqrt{v_{c_j}^j} \leq \dfrac{\sum_{k=1}^{c_j} v_k^j - T_j \mu}{\sum_{k=1}^{c_j} \sqrt{v_k^j}}$.

Example:

The following table shows the values of S_{1i} (i=1…4) if we apply this algorithm to allocate one task of size 4U on four computers in the following two cases:

v_1^1	v_2^1	v_3^1	v_4^1	s_{11}	s_{12}	s_{13}	s_{14}
10.0	8.0	6.0	4.0	0.627	0. 324	0.049	0
10.0	12.0	6.0	2.0	0.352	0.648	0	0

Table 3: The values of S_{1i} (i=1…4) to allocate one task of size 4U on 4 computers.

3. A DISTRIBUTED ALGORITHM FOR TASKS ALLOCATION

Based on the work presented in articles [5,11], we describe a distributed algorithm to compute the Nash equilibrium. For this and to characterize this equilibrium, we proceed with a generalization of this problem in the following way. Instead of considering a task j, there will be a generation source of tasks j. The source j will produce the same tasks with the same size T_j.

The idea of the algorithm is as follows. The sources generate tasks in parallel for several iterations. In each iteration, we measure the standard L_1 norm as $\sum_{j=1}^{m}\left|c_j^{(l-1)}-c_j^{l}\right|$, which is the sum of differences between the costs of source j in iteration l and iteration l-1. We stop when we obtain a difference less than a predefined error threshold.

The computation of the Nash equilibrium may require some coordination between sources (sources must coordinate among themselves to obtain information on the load of each machine). We use the following notations in addition to those of the previous section:

$j \leftarrow$ the number of the source j;
$l \leftarrow$ the iteration number;
$s_j^{(l)} \leftarrow$ the strategy of the source j computed in iteration l;
$c_j^{(l)} \leftarrow$ execution time of the source j at iteration l;
$\varepsilon \leftarrow$ the threshold error;
$norm \leftarrow$ the norm L_1 at iteration l defined as $\sum_{j=1}^{n}\left|c_j^{(l-1)}-c_j^{l}\right|$;
$send(j,msg) \leftarrow$ sends the message msg to source j;
$receive(j,msg) \leftarrow$ receives the message msg from the source j;

Each source j executes the following algorithm:

```
1- Initialisation :
   $s_j^{(0)} \leftarrow 0;$
   $c_j^{(0)} \leftarrow 0;$
   $l \leftarrow 0;$
   $norm \leftarrow 1;$
   $sum \leftarrow 0;$
   $state \leftarrow CONTINUE;$
   $left \leftarrow [(j-2) \bmod n]+1;$
   $right = [j \bmod n]+1;$

2- While (1) do

     if (j=1) {source 1}
      if ($l \neq 0$)
        receive(left,(norm,l,state)) ;
        if ($norm \prec \varepsilon$)
          send(right,(norm,l,STOP)) ;
          exit;
          $sum \leftarrow 0;$
          $l \leftarrow l+1;$
     else {others sources}
          receive(left,(sum,l,state)) ;
          if (state=STOP)
            $\left(v_i^j \leftarrow v_i - \sum_{k=1,k\neq j}^{m} s_{ki}T_k\mu\right)$

     if $(j \neq n)$ receive(right,(sum,l,STOP)) ;
          exit ;

     For $i := 1,\ldots,m$ do
       Obtain $v_i^j$ by examining the queue
             of each machine
       $s_j^{(l)} \leftarrow \mathrm{Best\_Response}_j\left(v_1^j,\ldots,v_m^j,T_j\right)$ ;
       Compute of $c_j^{(l)}$ ;
       $sum \leftarrow sum+\left|c_j^{(l-1)}-c_j^{(l)}\right|$ ;
       send(right,(sum,l,CONTINUE)) ;
     endwhile
```

4. EXPERIMENTS RESULTS

a- Simulation Environment

The simulations are done using the programming environment with message passing MPI (Message Passing Interface) [15]. This environment has a set of communications functions that connect different nodes in the system.
The simulation model consists of a collection of computers interconnected through a communications network and the tasks are distributed according to the load balancing scheme established.

The system is simulated as follows:

- 10 computers that are the sources generating tasks so that the source j (computer j) generates tasks Tj with the same size.
- 10 processing computers for receiving the portion of the tasks according to the allocation scheme.

b- Simulation Results

The algorithm runs several steps before convergence to the desired allocation strategy as follows.
In each iteration, the source computers simultaneously send tasks to the processing computers and calculate their cost times cj (s) (j = 1 ... m) and the difference
$\left|c_j^{(l-1)} - c_j^{l}\right|$, knowing that $c_j^{(l)}$ is the cost time of source j in the Step l.

So that we can compute the norm $\sum_{j=1}^{n}\left|c_j^{(l-1)} - c_j^{l}\right|$
a standard coordination between different sources is required as follows: each source j>1 sends the difference to its source at right.

The latter adds the value received from its source at left with its own value and sends the sum to its source at right. Source 1 is designated to calculate the final sum. If this sum exceeds the error threshold, we move to the next iteration.

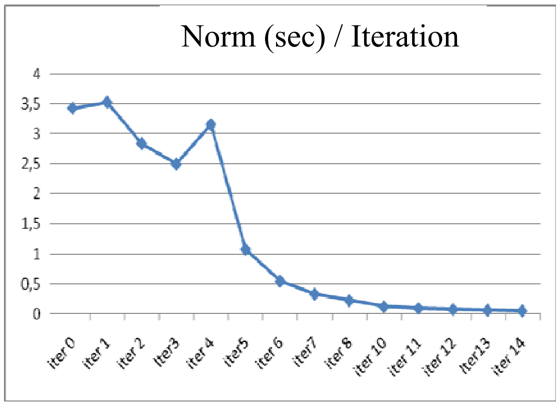
With a threshold of 5% error, the algorithm converges in step 16 with the following strategy of allocation.

Source j	Sj0	Sj1	Sj2	Sj3	Sj4	Sj5	Sj6	Sj7	Sj8	Sj9	Somme
Source 0	0,7	0,3	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	**1,08**
Source 1	0,45	0,3	0,17	0,1	0,01	0,01	0,01	0,01	0,01	0,01	**1,08**
Source 2	0,33	0,26	0,2	0,13	0,07	0,05	0,01	0,01	0,01	0,01	**1,08**
Source 3	0,26	0,22	0,18	0,14	0,1	0,07	0,03	0,03	0,01	0,01	**1,05**
Source 4	0,22	0,19	0,16	0,14	0,11	0,09	0,06	0,04	0,02	0,01	**1,04**
Source 5	0,19	0,17	0,15	0,13	0,11	0,1	0,08	0,06	0,04	0,01	**1,04**
Source 6	0,17	0,15	0,14	0,12	0,11	0,1	0,08	0,07	0,05	0,01	**1**
Source 7	0,16	0,15	0,13	0,12	0,11	0,1	0,09	0,08	0,06	0,01	**1,01**
Source 8	0,15	0,14	0,13	0,12	0,11	0,1	0,09	0,08	0,07	0,01	**1**
Source 9	0,15	0,14	0,13	0,12	0,11	0,1	0,1	0,09	0,08	0,01	**1,03**

Table4 : The allocation strategy found Sji (j=0…9 ; i=0…9) with 10 generating source computers (Source) and 10 processing computers.

The element of index ji (j = 1 ... m, i = 1 ... n) in the table represents the portion of the task Tj sent to the computer i. So each task is divided into n portions and each portion is processed by a computer. The last column of the table represents the sum of portions for each task that is approximately equal to 1.

In the following graph, we show the behavior of the norm $\sum_{j=1}^{n} \left| c_j^{(l-1)} - c_j^{l} \right|$ according to the iterations of the algorithm.



Graph1 : Representation of Norms according to Iterations

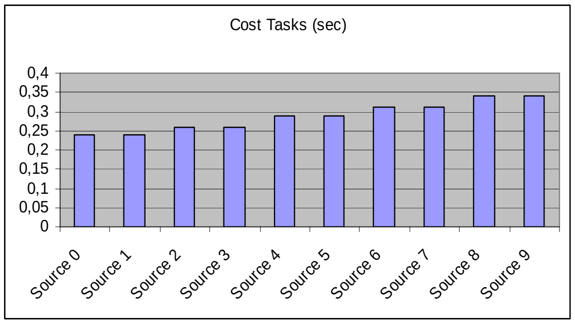
The Norm converges in a decreasing way to the error threshold (5%) according to iterations and the convergence is clearly seen from iteration 8.

The Cost Times are represented in the following table and their histogram representation is drawn in the following graphic.

SJ	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
CJ	0,24	0,24	0,26	0,26	0,29	0,29	0,31	0,31	0,34	0,34

Table 5 : Cost tasks in seconds for each source Sj (j=1…9)

We note that this allocation strategy leads to a good load balancing of processing computers.



Graph 2: Costs tasks in seconds

5. CONCLUSION AND PERSPICTIVES

In this paper, we have completed and implemented the general problem of allocating tasks. The theoretical part of this work is developed in [14] and here we focus our research on practical aspects.
We have modelized this problem as non-cooperative game between several players and we find that the Nash equilibrium for this game provides a good allocation of tasks for our system.
We have proposed the structure of the Nash equilibrium and on the basis of this structure, we have described a distributed algorithm to discover it.
We have implemented this algorithm by using the MPI environment and a system of 10 computer sources that generate the tasks and 10 processing computers. The numerical results show that the tasks allocation obtained leads to a good load balancing of processing computers.
Several adjustments and extensions are possible for this work on the Internet, parallel and distributed systems also in computing grids.
Communication between tasks is overlooked in this work; our next step will take into account this constraint.

REFERENCES

[1] M. Osborne, “An Introduction to Game Theory”, Oxford University Press, New York, 2004
[2] J. Nash, “Non-cooperative games”, Ann. Math. 54 (2) 286-295, 1951
[3] A. Czumaj, B.Vöcking, ”Tight Bounds for Worst-Case Equilibrium”, ACM Transactions on Algorithms, Vol. 3, N. 1, Article 4, 2007
[4] A. Legrand, H. Renard, Y. Robert et F. Vivien "Mapping and load-balancing iterative computations on heterogenous clusters with shared links", IEEE Trans. Parallel and Distributed Systems, Vol. 15, N 6, 546-558, 2004
[5] D. Grosu, A.T. Chronopoulos, “Noncooperative load balancing in distributed systems”, J. of Parallel Distrib. Comput. 65, 1022-1034, 2005
[6] A. Orda, R. Rom, N. Shimkin, "Competitive routing in multiuser communication networks", IEEE/ACM Trans. Networking 1 (5), 510-521, 1993

[7] E. Altman, T. Bassar, T. Jimenez, N Shimkin, "Routing in two parallel links: game-theoric distributed algorithms", J. Parallel Distributed Comput. 61 (9), 1367-1381,

[8] T. Boulonge, E. Altman, O. Pourtallier, "On the convergence to Nash equilibrium in problems of distributed computing", Ann. Oper. Res. 109 (1), 279-291, 2002

[9] D. G. Luenberger, "Linear and Nonlinear Programming, Addison-Wesly", Reading, MA, 1984

[10] T. Basar, G.L. Olsder, "Dynamic noncooperative game Theory", SIAM, Philadelphia, PA, 1998

[11] X. Tang, S. T. Shanson, "Optimizing Static job scheduling in a network of heterogeneous computers", in Proceeding of the International Conference on Parallel Processing, 373-382, 2000

[12] M. Zbakh, "Equilibrage de Nash dans le problème d'allocation des tâches", in Proceeding of RenPar'2009, Toulouse, France, 2009

[13] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, Y. Yang, "Scheduling divisible loads on star and tree networks :results and open problems", in IEEE Trans. Parallel and Distributed System, 16(3): 207-218, 2005

[14] M. Zbakh, S. El Hajji, "Tasks Allocation Problem as a non – cooperative game", in Journal of Theoretical and Applied Information Technology, Vol. 16, N°.2, June 2010, pp 110-115

[15] W. Gropp, E. Lusk, D. Ashton, D. Buntinas, R. Butler, A. Chan, R. Ross, R. Thakur, B. Toonen, "MPICH2 User's Guide, V1.0.3", November 2005