Journal of Theoretical and Applied Information Technology

www.jatit.org

TASKS ALLOCATION PROBLEM AS A NON - COOPERATIVE GAME

[1]MOSTAPHA ZBAKH, [2]SAID EL HAJJI

[1]Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes-ENSIAS, Avenue Mohammed Ben Abdallah Regragui, Madinat Al Irfane, BP 713, Agdal Rabat, Maroc

[2]Faculté des Sciences de Rabat, Avenue Ibn Batouta, Rabat, Maroc

ABSTRACT

We present the problem of tasks allocation in its general form as a non-cooperative game between players. For this game, we give the Nash equilibrium structure and on the basis of this structure, we draw a distributed tasks allocation algorithm that can find this equilibrium.

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

I. INTRODUCTION AND RELATED WORKS

In recent years, heterogeneous systems have become the 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 of the machines. For example, in [3], the authors address the problem of load balancing on the 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 the heterogeneous system.

The general formulation of such a problem is as follow. 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 minimize the costs of tasks

To clarify the idea of allocation of tasks on homogeneous machines, consider the simple following 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 presented the cost of each task obtained by

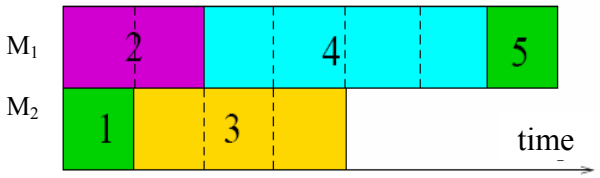

Figure1: First allocation

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

Table 1: Cost of tasks achieved by the first allocation

Vol. 16 No ne, 2010 pp [110 – 115]

ISSN: 1817-3195 / **E-ISSN**: 1992-8615

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

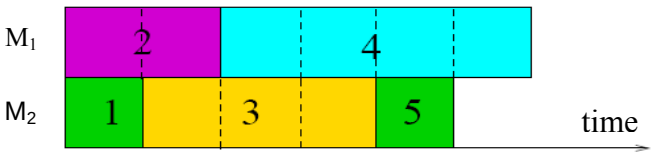

Figure2: Second allocation

Several studies exist in the literature that 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 draw a conclusion and perspectives for this work.

II. 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

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

Table2: Cost of tasks achieved by the 2nd assignment

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

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: $\sum_{i=1}^{m} s_{ji}=1, j=1...n$
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 prefer 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):

ISSN: 1817-3195 / E-ISSN: 1992-8615

S_j is such that
$c_j(s_1, s_2, ..., s_j, ..., s_n) = \min_{s_j^{\wedge}} c_j(s_1, s_2, ..., s_j^{\wedge}, ...s_n)$

.

In other words, 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 times 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 $s=(s_1,s_2...s_n)$ 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$_j$):

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

There are several algorithms for solving similar optimization problems of our case which are based on the 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: Assuming that the machines are ranked in decreasing order of their available processing rates $\left(v_1^j \geq v_2^j \geq \ldots \geq v_n^j\right)$, the solution s_j of the optimization problem Best_Response$_j$ is given by:

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

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

Proof:

From the formula $C_j(s)$, we find that
$\frac{\partial c_j(s)}{\partial s_{ji}} \geq 0$ and $\frac{\partial^2 c_j(s)}{\partial (s_{ji})^2} \geq 0$ for i=1 …n.
This shows that the function $c_j(s)$ is convex in s_j. This optimization problem should minimize the convex function and the Kuhn-Tucker conditions of first order are necessary and sufficient for optimization [9].

Let $\alpha \geq 0$ and $\eta_i \geq 0$, i=1,…,m, be the Lagrange multipliers [9]. The Lagrangian is:
$L(s_{j1}, ..., s_{jm}, \alpha, \eta_1, ..., \eta_m) =$
$\sum_{i=1}^{m} \frac{s_{ji}}{v_i^j - s_{ji} T_j \mu} - \alpha\left(\sum_{i=1}^{m} s_{ji} - 1\right) - \sum_{i=1}^{m} \eta_i s_{ji}$

ISSN: 1817-3195 / **E-ISSN**: 1992-8615

The Kuhn-Tucker conditions imply that s_{ji} , i=1…, m, is the optimal solution if and only if there exist $\alpha \geq 0$ and $\eta_i \geq 0$, i=1,…,m such that :

$$\frac{\partial L}{\partial s_{ji}} = 0 \text{ and } \frac{\partial L}{\partial \alpha} = 0 \text{ and}$$

$$\eta_i s_{ji} = 0, \eta_i \geq 0, s_{ji} \geq 0, i = 1...m$$

These three conditions become:

$$\frac{v_i^j}{(v_i^j - s_{ji} T_j \mu)^2} - \alpha - \eta_i = 0, i = 1...m \text{ and}$$

$$\sum_{i=1}^{m} s_{ji} = 0 \text{ and}$$

$$\eta_i s_{ji} = 0, \eta_i \geq 0, s_{ji} \geq 0, i = 1...m$$

ie :

$$\alpha = \frac{v_i^j}{(v_i^j - s_{ji} T_j \mu)^2}, \quad \text{if}$$

$$s_{ji} \succ 0, 1 \leq i \leq m \ (*)$$

$$\alpha \leq \frac{v_i^j}{(v_i^j - s_{ji} T_j \mu)^2}, \text{ if } s_{ji} = 0, 1 \leq i \leq m$$

(**)

$$\sum_{i=1}^{m} s_{ji} = 0, s_{ji} \geq 0, i = 1...m$$

Since the machines are ordered according their v_i^j , ($v_1^j \geq v_2^j \geq ... \geq v_m^j$), we have the same thing for s_{ji} ($s_{j1} \geq s_{ji} \geq ... \geq s_{jm}$). This implies that there exist some situations where low power machines have no tasks to treat. In other word, there exist an index $c_j, (1 \leq c_j \prec m)$ such that $s_{ji} = 0$, for $i = c_j ... m$.

According to the formula (*), we obtain the equality:

$$\sum_{i=1}^{c_j - 1} \sqrt{v_i^j} = \sqrt{\alpha} \left(\sum_{i=1}^{c_j - 1} v_i^j - \sum_{i=1}^{c_j - 1} s_{ji} T_j \mu \right)$$

and according to (**):

$$\sqrt{\alpha} = \frac{\sum_{i=1}^{c_j - 1} \sqrt{v_i^j}}{\sum_{i=1}^{c_j - 1} v_i^j - \sum_{i=1}^{c_j - 1} s_{ji} T_j \mu} \leq \frac{1}{\sqrt{v_i^{c_j}}}$$

i.e:

$$\sqrt{v_i^{c_j}} \sum_{i=1}^{c_j} \sqrt{v_i^j} \leq \sum_{i=1}^{c_j} v_i^j - T_j \mu$$

The parameter c_j is the minimum index that satisfies the above equation, and

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

III. A DISTRIBUTED ALGORITHM FOR TASKS ALLOCATION

Based on the work presented in the 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 the 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 $\sum_{j=1}^{n} \left| c_j^{(l-1)} - c_j^l \right|$;

ISSN: 1817-3195 / E-ISSN: 1992-8615

$send(j, msg) \leftarrow$ 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)* of each machine

$\left(v_i^j \leftarrow v_i - \sum_{k=1, k \neq j}^{m} s_{ki} T_k \mu \right)$

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

exit ;

For $i := 1, ..., m$ ***do***

Obtain v_i^j by examining the queue

$s_j^{(l)} \leftarrow \text{Best\_Response}_j\left(v_1^j, ..., v_m^j, T_j\right);$

Compute of $c_j^{(l)}$;

$sum \leftarrow sum + \left| c_j^{(l-1)} - c_j^{(l)} \right|$;

send*(right,(sum,l,CONTINUE)) ;*

endwhile

IV.CONCLUSION AND PERSPICTIVES

We have formulated the general problem of allocating tasks as non-cooperative game between several players. For this game, the Nash equilibrium provides a good allocation of tasks for our system. We propose the structure of the Nash equilibrium and on the basis of this structure; we have described a distributed algorithm to discover it. Several adjustments and extensions are possible for this work on the Internet, distributed systems and computing grids.

In this work, we have neglected the communication between tasks; our next step will take into account this constraint on the one hand and implement the above algorithm on the other hand.

V. 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,

ISSN: 1817-3195 / **E-ISSN**: 1992-8615

[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

Vol. 16 No.2 June, 2010 pp [110 – 115]

ISSN: 1817-3195 / **E-ISSN**: 1992-8615