Asian Journal of Information Technology 5(8): 818-822, 2006

Development of a Load Balancing Algorithm in an Environment of Grid Computing

[1]Abdallah Boukerram, [2]Samira Ait Kaci Azzou and [2]Mabrouk Nekkache
[1]Institut Supérieur d'Informatique et de Modélisation Appliquées,
Université B. Pascal Clermont II, 63 000 Cedex, France
[2]Département Informatique, Université F. Abbas, 19000 Sétif, Algérie

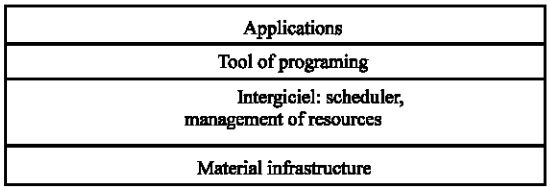
Abstract: The distribution of the load of calculation, takes into account the problems of the minimization of the time of answer, the maximization of the throughput of the system, the minimization of the time of inoccupation of the nodes of treatment and makes sure that the workloads are balanced enough on all the node of the grid. This study describes an algorithm of distribution, for load balancing in an environment of grid computing. This algorithm is inspired by a certain number of classic algorithms of load balancing. The conception of the algorithm is conceived for a grid whereas the implementation is established on an organized cluster in tabular form bi-dimensional of processors 4-related. The matrix of distribution and the number of iterations proving the convergences of the algorithm established in a experimental way, are presented well and widely commented.

Key words: Grid computing, load balancing, cluster, node, scheduler

INTRODUCTION

The distributed data processing wants, an inescapable alternative for the big projects of research or development today, such as the e-Busines, the e-Government, the e-Health, the high-throughput computing or all which collaborative calculation. The grid are defined as a material and software infrastructure supplying a reliable, coherent access, at high rate of penetration cheap and especially at capacity of stocking and very high treatment going even beyond capacities offered to there by super calculator[1]. The grid pulls their power of calculation of the coordination of several hundreds of thousand diverse functional units. The exploitation of the sub-used resources, the access to certain special resources (dedicated processors, electron microscope, ...etc), the parallel processing are the major trump cards of the grid. To make profitable such systems, it is necessary to manage to make a fair organization of spots on all the resources. The algorithms of load distribution are expensive: The more the algorithms are complex, the more the costs are raised and better are the performances because the programs take care of all the complexity and the heterogeneity of systems in the grid[2].

General architecture of a grid computing: Architecture at four levels, is inspired by the benchmark model GLOBUS, supplying all the basic services for the construction and the management of grid computing[3,4].



- Location and allowance of resources
- Communication
- Information about the resources (typology, state and availability).
- Access to the data and the mechanism of security
- Creation and launch of jobs

We note all the same a set of weaknesses of the grid computing. These weaknesses are bound to the cohabitation of several incompatible standards, because leaning on operational systems and different protocols[2,3]:

- Weakness at the level of the security
- Slowness of the access times connected to a single point of access determined by the UI (User Interface).
- Tolerance in the faults
- Weakness of the tools of redistribution of the power and the load of computing.

It is in this last point, namely the load balancing that we were interested in this study to bring a solution.

Corresponding Author: Abdallah Boukerram, Institut Supérieur d'Informatique et de Modélisation Appliquées, Université B. Pascal Clermont II, 63 000 Cedex, France

ALGORITHMS OF LOAD BALANCING

We distinguish a wide range of algorithms of load balancing. In the literature, a distinction is established between the determinist and stochastic methods in the iterative load balancing[5,6]. The iterative algorithms in which we are interested, lean on the Eq. (a), the mathematical developments are established in[7,8]:

$$W_i^{(t+1)} = W_i^{(t)} + \sum_{i,j=1..n} \alpha_{ij}(W_j^{(t)} - W_i^{(t)}) + \mu_i^{(t+1)} - K \quad (a)$$

Where:

$W_i^{(t)}$ load with the node i at the time t.
α_{ij}: parameter of exchange between knots i and j.
$\mu_i^{(t)}$: load supported by the node i at the moment t.
K: represent the load realized by a node at the end of iteration.

The methods applying such models for their implementation use the structures of following data: modelling of the network in the form of graph of type G (X.E) where:

- X: represent nodes of the graph (grid)
- E: all the bows of the network
- |X| = n, is the number of nodes of the grid
- (i,j): physical connection, connecting the node i with the node j
- d(i) is the degree of the node i
- d(G): degree of the graph.

Among these works, we can quote those of Boilat[8].

From the Eq. (a), simplifying hypotheses, bring this Eq. under vectoriel shape:

$$W_i^{(t+1)} = MW_i^{(t)}$$

W: Vector of load of all nodes
M: Matrix of distribution, its dimension is (nxn) with:

The Cybenko[9],principle leaves that all the nodes of a network are identical and have the same degree. Thesimplifying assumptions bring back the Eq. (a) in form:

$$m_{ij} = 1/\max(d(i),d(j)) + 1) \quad \text{for } i \neq j$$

$$m_{ij} = 1 - \sum_{i \neq j} m_{ij} \quad \text{else}$$

For i,j = 1..n

ADOPTED STRATEGY

The distribution of the load is static. She is made after the system made the collection of the information of loads on all the nodes of the grid, to redistribute then the load. The centralization of the collection of the piece of information is justified by a certain number of advantages namely:

- It allows to avoid the problem of distribution all to all, what thus reduces considerably the traffic in the grid.
- The time of the collection of the piece of information is reduced, because the wait of the answer is dependent only on a node at the same moment.
- Any new node integrating the grid is easily considered in this strategy.

Developed method: Contrary to the algorithms seen above the proposed algorithm part of the principle:

- we do not know beforehand the number of nodes of the grid
- we do not thus arrange a matrix of adjacency: on the other hand in every node corresponds a table of routing.

That is to say the Eq. noted (a):

$$W_i^{(t+1)} = W_i^{(t)} + \sum_{i,j=1..n} \alpha_{ij}(W_j^{(t)} - W_i^{(t)}) + \mu_i^{(t+1)} - K$$

We consider that the load is static it is to say constant in every iteration of balancing of load let be:

$$W_i^{(t+1)} = W_i^{(t)}$$

furthermore $\mu_i^t = 0$ one create by no means residual load.

This what reduces the Eq. (a) as:

$$W_i^{(t+1)} = (1 - \sum \alpha_{ij})W_i^{(t)} + \alpha_{ij}W_j^{(t)}$$ what is the shape: $W_i^{(t+1)} = MW_i^{(t)}$

$W_i^{(t)}$ is the vector of dimension n containing the load of all the node of calculation at the moment t. M is the matrix of distribution is calculated, while taking as a starting point the genetic algorithms: a node takes a half and diffuses the other on the whole of its neighbours.

$m_{ij} = \alpha_{ij}$ where $\alpha_{ij} = V/2$ if $i \neq j$,

V = number of neighbours of i

$m_{ij} = 1/2$ if $i = j$

$m_{ij} = 0$ if i is not connected to j

Algorithm:

- Access to a central node (scheduler) where from is launched the load balancing algorithm initially.
- Calculation of the matrix of diffusion, the vector of load for each node and the total of number of iterations.
- Stop of the process of balancing as soon as there is equality of load enters all the nodes more or less five percent.
- Preservation of two values (balise min=minimal load, balise max=maximum load). These two values represent, thresholds of release of the algorithm of load balancing.

If w(t) > balise max or w(t) < balise min then activation of the load balancing algorithm. A supplementary algorithm comes to add: algorithm of marking to cross all the nodes of the grid.

Course of the nodes of the grid: Entry: node (scheduler)

- Read table of routing associated to mark every node accessible not marked
- Mark the accessible nodes not already
- Take each marked node and remake the stages of reading and marking;
- Stop, when there is no more not marked node.

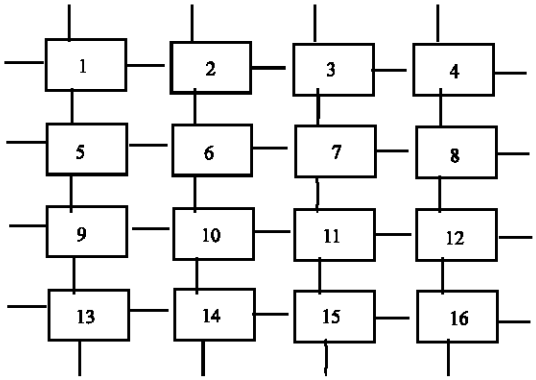


Fig. 1: Cluster of 16 processors

The nodes of the grid all were traversed.

Environment of the development: The implementation is made on a cluster of 16 processors of Pentium type III put rhythm by clocks of 2Ghz, under Java[10]. This cluster can be seen as a node of a grid computing or symbolizing he, even a grid[11,12]. The network of interconnection is a matrix type: any node of the cluster is directly connected with these four neighbours. The nodes of the first column see their fourth physical connection connected with those of the last column and those of the first line to those of the last one as indicated on the Fig. 1.

The processor 1, plays the role of the scheduler, it is him who launches the program of initial load balancing. More all the jobs called to be executed on the cluster pass by scheduler: It is the only access point to the network. The global state of load or the collection of the load of the various nodes of the system is also made by this one.

noeuds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0.50	0.25	0	0.16	0.06	0	0	0	0	0	0	0	0.03	0	0	0
2	0.25	0.50	0.16	0	0	0.06	0	0	0	0	0	0	0	0.03	0	0
3	0	0.16	0.50	0.25	0	0	0.06	0	0	0	0	0	0	0	0.03	0
4	0.16	0	0.25	0.50	0	0	0	0.06	0	0	0	0	0	0	0	0.03
5	0.06	0	0	0	0.50	0.25	0	0.16	0.03	0	0	0	0	0	0	0
6	0	0.06	0	0	0.25	0.50	0.16	0	0	0.03	0	0	0	0	0	0
7	0	0	0.06	0	0	0.16	0.50	0.25	0	0	0.03	0	0	0	0	0
8	0	0	0	0.16	0.16	0	0.25	0.50	0	0	0	0.03	0	0	0	0
9	0	0	0	0	0.03	0	0	0	0.50	0.25	0	0.16	0.06	0	0	0
10	0	0	0	0	0	0.03	0	0	0.25	0.50	0.16	0	0	0.06	0	0
11	0	0	0	0	0	0	0.03	0	0	0.16	0.50	0.25	0	0	0.06	0
12	0	0	0	0	0	0	0	0.03	0.16		0.25	0.50	0.25	0	0	0.06
13	0.03	0	0	0	0	0	0	0	0.06	0	0	0	0.50	0.25	0	0;16
14	0	0.03	0	0	0	0	0	0	0	0.06	0	0	0	0.50	0.16	0
15	0	0	0.03	0	0	0	0	0	0	0	0.06	0	0	0.16	0.500	0.25
16	0	0	0	0.03	0	0	0	0	0	0	0	0.06	0.16	0	0.25	0.500

Vector of load: Initialization: 10 000 units for node 1 all the others are 0

Noeuds 1-16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Itération1	5000	1280	0	1240	1240	0	0	0	0	0	0	0	1240	0	0	0
itération2	2900	1010	725	1200	1250	1580	0	1030	695	0	0	0	1050	790	0	0
itération3	1320	700	500	675	800	910	740	610	5000	480	500	860	600	500	500	505
.....																
Itération 17	625	630	750	631	630	750	452	747	753	456	900	444	629	750	462	438
.....																
itération32	626	600	606	603	608	592	661	586	607	707	753	593	608	592	728	568
itération33	620	623	649	626	631	625	565	619	640	620	685	632	631	661	608	601

RESULTS

Matrix of distribution:

COMMENT

- The algorithm ends in a number limited by iterations, thus convergent. 33 iterations, necessary for the convergence are executed on a station of work put rhythm by a clock of 2Ghz, within 200 μs.

- The obtained results are similar to those of Boilat with a network of seven nodes, organized in star, while our turn on a network of 16 nodes.
- We notice the importance of the network of interconnection, in these strategies of load balancing of: more the network of interconnection is more tightened the convergence is accelerated.
- It is noticed that the nodes closest to to the node of scheduler are the first to reach balanced load.

This algorithm would win more in reliability, by taking care of the weights of communication, so that knots served lastly can benefit from a lowering in charge of work.

SECURITY AND FAULT TOLERANCE

If ever a breakdown arises on a node, other one than scheduler, it will be without consequence on the algorithm of load balancing. At the risk of not losing the job which was attributed to him, this problem can be settled by guarding a copy on the father's node[13].

The security is a crucial point, in the world of grid[13]. Every user on the grid is subject to an authentification further to a certificate delivered by appropriate authorities, followed by an authorization suiting to the virtual organization to which it is up and defining the resources in which he has access. The control is made in the entry of the user interface.

The environment of distribution of the load is global, what allows us to have an effective and synchronous control. It remains while to choose the algorithm of distribution. Our choice is motivated, by all the aforementioned criteria.

CONCLUSION

The load balancing algorithm developed converges in an acceptable number of iterations. The conception and the implementation of this algorithm are established in a perfect logic of a grid computing, although the tests are made on a cluster. The network of interconnection of the grid and the busy band, are the essential variables to be considered in the research works to come for a total equity of load between the different processors of a grid. We shall note the portability of this specific algorithm of load balancing on any of grid computing in a centralized system.

REFERENCES

1. Csajkowski, K. and I. Foster, 2002. A ressource management architecture of metacomputing system. Lecture notes in Computer Science.
2. Badidi, E., 2000. Architecture and service for load balancing on the distributed system. Thesis of Phd in Computer Science, Montréal
3. The globus alliance is developing fundamental technologies needed to build computationals grids.Http://www.globus.org/
4. Foster, I. and C. Kesselman. Globus: A metacomputing Infrastructure Toolkit http://www.globus.org/
5. F5 Networks, 2000. Local Hygh-availability, intelligent load balancing. http://www.f5.com/bigip/index.html
6. Jiming, L., J. Xialing and W. Yuanshi, 2005. Agent based load balancing on homogenious mini-grids. IEEE trans. On parallel and distributed system, pp: 16.
7. Vernier, F., 2004. Algorithmique itérative pour l'équilibrage de charge dans les réseaux dynamiques. Thesis of Phd in Computer Science, F.Comté
8. Boilat, J.E., 1990. Load balancing and poisson equation in a graph. Pratice & Experience, 2: 289-313.
9. Cybenko, 1989. Dynamic load balancing for distributed memory multiprocesseurs. J. Parallel and Distributed Computing, pp: 279-301.

10. Nemeth, Z. and V. Sunderam, 2003. Characterezinggrids: Attributions, définitions and formalisms. J. Grid Comput., 1: 9-23.
11. Kielman, T., P. Hatcher, L. Bougé and H. Bal, 2003. Enabling java for hygh performance computing: Exploiting distributed shared memory and remote.
12. Aumage, A. and G. Mercier, 2003. A cluster of clusters enbled mpi implementation. In 3rd IEEE/ACM international Symposium on cluster Computing and the Grid. ACM, pp: 110-117.
13. Humphrey, M., M.R. Thompson and K.R. Jackson, 2005. Security for grids Proceeding of the IEEE, 93: 644-652.