American Journal of Applied Sciences 3 (4): 1810-1813, 2006
ISSN 1546-9239

Implementation of Load Balancing Algorithm in a Grid Computing

[1]Abdallah Boukerram and [2]Samira Ait Kaci Azzou
[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 emergence of the grid computing constituted by several resources stood out as platform of development for the industrial applications treating big quantities of data. In these environments of high- throughput computing, numerous researches were dedicated these last five years to the load balancing to make profitable the power of calculations in the grid. This paper describes the complete implementation of an algorithm of load balancing in an environment of grid computing. The implementation of the algorithm is realized on a cluster of processors in a logic of portability on grids. The number of iterations proving the convergence of the algorithm, vectors of the loads of calculations and the matrix of distribution are well clarified.

Key words: Grid computing, load balancing, cluster, node, interconnexion network

INTRODUCTION

The grid computing federating a big number of computer resources stand out as solutions of the big projects of industrial researches and development so as projects: datagrid[1], e-etoile[2], globus[3], teragrid[4].

The grid computing offer a material and software infrastructure supply a reliable access with capacity of very high stocking and treatment exceeding the performances of the great computers[5].

This power of calculation and stocking of data is conjugated to a report performance / cost very advantageous.

To make profitable in best such systems, it is necessary to obtain an organization of fair loads on all the nodes of the grid.

The algorithms of the load balancing are expensive at time CPU, so that they can take care of all the complexity and the heterogeneity of grid[6].

We note 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[6, 7]:

* Weakness at the level of the security
* Slowness of the access times connected to a single point of access determined by the User Interface.
* Tolerance in the faults
* Security in the grid
* 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 paper to bring a solution.

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[7,8].

The different levels are as follow :

Level 1: Material infrastructure

Level2: Intergiciel: scheduler, management of resources

Level 3: Tool of programming

Level 4: Applications

The Intergiciel offer a set of services such as:

* Location and allowance of resources
* Communication between processors
* Information about the resources
* Access to the data and the mechanism of security
* Creation and launch of jobs

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[9,10].

The iterative algorithms in which we are interested, lean on the equation (a), the mathematical developments are established in[11,12]:

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

Where:

$W_i^{(t)}$ load with the node i at the time t.

α_{ij} : parameter of exchange between the node 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:

Corresponding Author: Abdallah Boukerram, Institut Supérieur d'Informatique, et de Modélisation Appliquées, Université B. Pascal Clermont II, 63 000 Cedex France, Tel: (33) 06 82 02 74 16

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[12].

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

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

W: Vector of load of all nodes

M: Matrix of distribution, its dimension is (nxn) with:

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

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

for i, j = 1..n

The Cybenko[13], principle leaves that all the nodes of a network are identical and have the same degree. The simplifying assumptions bring back the equation (a) in form: $W_i^{(t+1)} = M W_i^{(t)}$

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

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

for i, j = 1..n

MATERIALS AND METHODS

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.

a. Developed method: We adopted the basic algorithm of load balancing to the grid computing:

* Integration of table of routing in the structures of data[14].
* Introduction of simplifying hypotheses relieving the algorithm.

Contrary to the classic algorithm the proposed algorithm part of the principle where we do not know beforehand the number of nodes of the grid.

Hypotheses: That is to say the equation 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 :

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

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

This what reduces the equation (a) as:

$$W_i^{(t+1)} = (1 - \sum \alpha_{ij}) W_i^{(t)} + \alpha_{ij} W_j^{(t)}$$

for i, j = 1., n

what is the shape: $W_i^{(t+1)} = M W_i^{(t)}$

$W_i^{(t)}$ 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 ≠ j,

V = number of neighbours of i

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

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

b. Developed algorithm

Initialization

* Matrix of distribution: node 1=1 all the others are 0.
* Vector of load 10 000 units for node 1 all the others are 0.
* Access to a central node where from is launched the load balancing algorithm initially.

Calculation of loads: Calculation of the matrix of diffusion, the vector of load for each node and the total of number of iterations.

Criterion of stop: Stop of the process of balancing as soon as there is equality of load enters all the nodes more or less five percent.

Introduction of optimal thresholds: 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. The nodes of the grid all were traversed.

c. Environment of the development: The implementation is made on a cluster of 12 processors of Pentium type IV put rhythm by clocks of 2 Ghz, under Java[15]. This topology (Fig.1) in star is identical to those used in the local networks. She has the advantage of:

* Mapping of quite the applications client / server.
* Offer an easy parallelism.
* Simplicity of realization with network equipments (hub, concentrator or multiplexer).

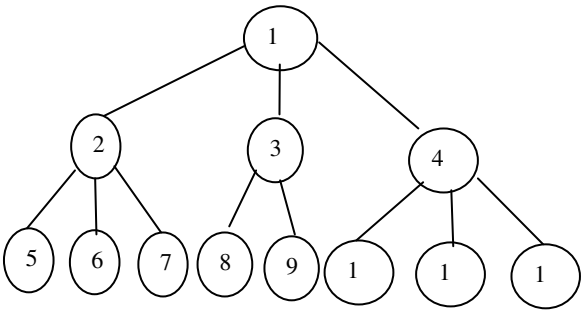

Fig. 1: Grid of 12 nodes

This cluster with this topology can be see as a node of a grid computing or symbolizing he, even a grid[16,17].

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.

d. 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[18].

The security is a crucial point, in the world of grid[18]. Every user on the grid is subject to an authentication 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.

RESULTS

Matrix of distribution:

nodes	1	2	3	4	5	6	7	8	9	10	11	12
1	0.50	0.25	0.15	0.10	0	0	0	0	0	0	0	0
2	0.25	0.50	0	0	0.13	0.08	0.04	0	0	0	0	0
3	0.25	0	0.50	0	0	0	0	0.15	0.10	0	0	0
4	0.25	0	0	0.50	0	0	0	0	0	0.15	0.10	0.04
5	0	0.50	0	0	0.50	0	0	0	0	0	0	0
6	0	0.50	0	0	0	0.50	0	0	0	0	0	0
7	0	0.50	0	0	0	0	0.50	0	0	0	0	0
8	0	0	0.50	0	0	0	0	0.50	0	0	0	0
9	0	0	0.50	0	0	0	0	0	0.50	0	0	0
10	0	0	0	0.50	0	0	0	0	0	0.50	0	0
11	0	0	0	0.50	0	0	0	0	0	0	0.50	0
12	0	0	0	0.50	0	0	0	0	0	0	0	0.50

Vector of load: Initialization:

Noeuds 1-16	1	2	3	4	5	6	7	8	9	10	11	12
Iteration 1	5000	2500	1500	1000	0	0	0	0	0	0	0	0
Iteration 2	3000	1500	800	1000	1200	1250	1250	0	0	0	0	0
Iteration 3	3000	1500	400	1000	1200	1250	1250	200	200	0	0	0
Iteration 4	3000	1500	400	1000	1200	1250	1250	200	200	200	200	100
……												
Iteration 11	625	630	750	631	630	750	452	747	753	456	900	444
.....												
Iteration 22	840	830	831	835	840	831	850	814	840	923	1042.	363
Iteration 23	841	829	832	834	836	827	843	821	830	833	842	833
Stop												

Interpretation of the results

* We notice that the algorithm converges in a number limited by iterations, that is 23 iterations
* The convergence is more accelerated in the first 11 iterations than in the last ones.
* It is noticed that the nodes closest to to the node of scheduler are the first to reach balanced load.
* The ten percent residue added to the criterion of stop indeed assures the convergence of the algorithm for 90 % returns.

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.

CONCLUSION

The load balancing algorithm developed leans on a structure of data of network type WAN, what guarantees its portability on any grid computing. The distribution of loads indeed assures the convergence of the algorithm in acceptable time. For an optimal equity of loads, we have to integrate into the works future, both variables which are the networks of interconnection and the bandwidth.

REFERENCES

1. Projet datagrid. http://www.datagrid-international.com
2. Projet e-toile. URL : http://www.urec.cnrs.fr/etoile/.
3. Projet globus. http://www.globus.org.
4. Projet teragrid. http://www.teragrid.org/.
5. Csajkowski, K. and I. Foster, 2002. A ressource management architecture of metacomputing system. Lecture notes in Computer Science.
6. Badidi, E., 2000: Architecture and service for load balancing on the distributed system. Ph.D. Thesis of Computer Science, Montréal.
7. The globus alliance is developing fundamental technologies needed to build computationals grids.Http://www.globus.org/
8. Foster, I. and C. Kesselman. Globus: A metacomputing Infrastructure Toolkit. http://www.globus.org/
9. F5 Networks 2000: local Hygh-availability, intelligent load balancing,. http://www.f5.com/bigip/index.html
10. Jiming, L., L. Xialing and W. Yuanshi, 2005: Agent based load balancing on homogenious mini-grids. IEEE Trans. Parallel and Distributed System, 16: 6.
11. Vernier, F., 2004: Algorithmique itérative pour l'équilibrage de charge dans les réseaux dynamiques. Ph. D. Thesis of Computer Science, F.Comté.
12. Boilat, J.E., 1990. Load balancing and poisson equation in a graph. Pratice & Experience, 2: 289-313.
13. Cybenko, 1989. Dynamic load balancing for distributed memory multiprocessors. J. Parallel and Distributed Comput., pp: 279-301.
14. Pujolle, G., 2000 Les Réseaux Edition Eyrolles
15. Kielman, T., P. Hatcher, L. Bougé and H. Bal, 2003: Enabling java for high performance computing: Exploiting distributed shared memory and remote.
16. Nemeth, Z. and V. Sunderam, 2003: Characterizing grids: attributions, definitions and formalisms. J. Grid Comput., 1: 9-23.
17. Aumage A. and G. Mercier, 2003. A cluster of clusters enabled MPI implementation. In 3rd IEEE/ACM international Symposium on cluster Computing and the Grid. ACM, pp: 110-117.
18. Humphrey, M., M.R. Thompson and K.R. Jackson, 2005. Security for grids. Proc. IEEE, 93: 644-652.