Dynamic Load Balancing on Non-homogeneous Clusters

Marcelo R. Naiouf[1], Laura C. De Giusti[2], Franco Chichizola[3],
and Armando E. De Giusti[4]

Instituto de Investigación en Informática LIDI (III-LIDI)[5]
Facultad de Informática – Universidad Nacional de La Plata
La Plata - Buenos Aires - Argentina
{mnaiouf, ldgiusti, francoch, degiusti}@lidi.info.unlp.edu.ar

Abstract. This paper discusses the dynamic and static balancing of non-homogenous cluster architectures, simultaneously analyzing the theoretical parallel speedup as well as the speedup experimentally obtained.

A classical application (Parallel N-Queens) with a parallel solution algorithm, where processing predominates upon communication, has been chosen so as to go deep in the load balancing aspects (dynamic or static) without distortion of results caused by communication overhead.

Four interconnected clusters have been used in which the machines within each cluster have homogeneous processors although different among clusters. Thus, the set can be seen as a N-processor heterogeneous cluster or as a multi-cluster scheme with 4 subsets of homogeneous processors.

At the same time, three forms of load distribution in the processors (Direct Static, Predictive Static and Dynamic by Demand) have been studied, analyzing in each case parallel speedup and load unbalancing regarding problem size and the processors used.

Keywords: Parallel Processing, Load Distribution, Static and Dynamic Load Balancing.

1 Introduction

1.1 Cluster and Multi-cluster Architectures

A cluster is a type of parallel/distributed processing architecture consisting of a set of interconnected computers that can work as a single machine. The machines that make up a cluster can be homogeneous or heterogeneous, this being an important factor for the analysis of performance that can be obtained from a cluster as a parallel machine [1][2][3].

[1] Full-time Professor, School of Computer Sciences. UNLP.

[2] PhD student. UNLP Scholarship. Assistant Profesor, School of Computer Sciences. UNLP.

[3] PhD student. CONICET. Assistant Profesor, School of Computer Sciences. UNLP.

[4] CONICET Main Researcher. Full-time Professor, School of Computer Sciences. UNLP.

[5] This project is financially supported by the CIC and the YPF Foundation.

G. Min et al. (Eds.): ISPA 2006 Ws, LNCS 4331, pp. 65 – 73, 2006.

A multi-cluster architecture consists in interconnecting two or more clusters to configure a new parallel machine. The characterization of global performance parameters of a multi-cluster is complex owing to the number of intervening clusters, the degree of heterogeneity of processors and the inter-cluster communication system. On occasions, a combination of interconnected homogeneous clusters, configuring a heterogeneous multi-cluster is used.

1.2 Load Balancing in Heterogeneous Architectures

For the type of known work problems (e.g. matrix multiplication) a "predictive" static load balancing considering the calculation power of the multi-cluster processors can be obtained; however, many real problems have a variable or dynamic workload depending on the data [4][5][6][7]. In these cases, it is necessary to adjust data or processes allocation dynamically while the application is being executed.

Besides, in a multi-cluster scheme in which applications are resolved with the Master-Slave paradigm, any dynamic balancing solution used, implies a communication overhead that will be affected by the complexity of the communication scheme among the nodes of the different clusters.

1.3 Types of Problems with Variable Workload

There are certain types of data parallelism problems for which it is possible to perform a static balancing allocation of the total workload. In these cases, provided there is a heterogeneous architecture, it will be possible to define a predictive $F(P_i,Wt)$ function where P_i is the calculation power of processor i and Wt the total work This function allows to distribute data "a priori" among processors [8].

If there is a variable workload due to the data particular characteristics (e.g. data arrangement, identification of image patterns), it is not possible to have a predictive function that assures load balancing among processors. Thus, it will be necessary to have a dynamic allocation policy that can be combined with a predictive initial distribution of a percentage of the total data [5][9].

Any dynamic allocation policy used implies some overhead degree of communication, which will be more complex to model and predict in a heterogeneous multi-cluster architecture.

2 Characterization of Type of Application of Interest

As analyzed in the introduction, there are different research axes on dynamic load balancing problems in multi-cluster architectures.

An architecture model in which heterogeneity appears only in machines with different clusters and can be compared to a calculation power function of the machines of each cluster has been determined.

Finally, the focus of this experimental work has been put on one type of the problems in which communication time among *Tc* processes is not significant, considering *Tp* (*Tp* >> *Tc*) local processing time.

This restriction allows to identify the differences among the static and dynamic load balancing schemes more clearly without overlapping an important communication overhead not relate to the distribution.

3 Load Distribution Models to be Studied and Theoretical Speedup to Be Achieved

Three ways of data parallelism implementation will be used:

- *Direct Static Distribution (DSD)* where the total workload *Wt* will be allocated to the architecture *B* processor in a homogeneous manner, so that each processor will have *Wt/B*, regardless the $F(P_i, Wt)$ function. This distribution is used as a lower bound reference.
- *Predictive Static Distribution (PSD)* where the total workload *Wt* will be allocated to the architecture *B* processor at the moment of starting the application, according to the prediction $F(P_i, Wt)$ function.
- *Dynamic Distribution upon Demand (DDD)* where a *Li* percentage of the total *Wt* workload will be allocated to the architecture *B* processor at the moment of starting the application, according to the prediction $F(P_i, W_i)$ function and then, each processor will demand more work on the part of the Master, as its task is being completed.

The *Li* value and the amount of additional work to be allocated to each processor on demand are experimental research parameters that depend on the application and the relation between *Tp* and *Tc*.

The theoretical speedup to be achieved by multi-cluster architecture will be a $G(P_i)$ function. The experimental measuring of the real speedup should directly correlate with the degree of balancing achieved with the total *Wt* work allocation during the execution of the application.

4 Contribution of This Work

- An expression for heterogeneous cluster calculation power is presented, considering individual processor power and heterogeneity. Also theoretical analysis of unbalance and maximum speedup attainable is presented.
- A Master-Slave model with 4 heterogeneous clusters among them operating as a (*B*=42) multi-cluster with an additional processor as Master has been studied, checking the theoretical analysis on processors heterogeneity and maximum speedup attainable.
- One problem case was studied, which responded to the hypothesis *Tp* >> *Tc*, with the three load distributions proposed (DSD, PSD, DDD) to carry out the data parallelism, specially comparing with the theoretical parallel speedup. This speedup was achieved in view of the calculation power of the processors, and the load unbalancing taking into account the parameters *B*, *Wt*, P_i y *Li* mentioned before.

5 Application to Parallel Solution on a Heterogeneous Multi-cluster of the N-Queens Problem

The N-queens problem consists in placing *N* queens on an *NxN* board in such a way that they do not attacks one another [10][11][12]. A queen attacks another one if they are in the same diagonal, row or column .

5.1 Sequential Solution

An initial solution to the N-queens problem, using an sequential algorithm, consists in trying all possible location combinations of the queens on the board, keeping those that are valid and disrupting the search whenever this is not achieved. Considering that a valid combination can generate up to 8 different solutions, which are rotations of the same combination, the number of distributions to be evaluated can be reduced. The best sequential algorithm found for this problem is based on this fact [13][14][15].

5.2 Parallel Solution Proposed Based on the Function of the Load Distribution Models

For the parallel solution of this problem, the queen is placed on one or more rows, and all the solutions for that initial arrangement are obtained. Each processor is in charge of solving the problem for a subset of said solutions, in this way, the whole system works with all the possible combinations of those rows.

When working with a heterogeneous architecture, the amount of work (combinations) that each processor must solve vary according to the existing relation regarding calculation power. To be able to distribute the work in a balanced way, it is convenient to use "fine grain", that is, many combinations of little work each, so as to level up the work done by each machine, and resolve several of them. To this aim, the first four rows are used to form each of the combinations to resolve [16].

In this way, different N^4 combinations are obtained to be distributed among all the heterogeneous processors, *N* being the board size. This distribution is carried out by using those motherhoods mentioned in III.

6 Experimental Results Obtained

In this section, the tests carried out are presented together with the results obtained, regarding the speedup metrics and the unbalancing described below.

6.1 Metrics Used

To measure the load unbalancing among the processors that intervene in a parallel application, the relative work difference obtained is calculated with formula (1), where $Work_i = machine\ \ time_i$ [2].

$$Unbalance = \frac{\max_{i=1..B}(Work_i) - \min_{i=1..B}(Work_i)}{average_{i=1..B}(Work_i)} . \qquad (1)$$

The speedup metrics is used to analyze the algorithm performance in the parallel architecture as indicated by formula (2).

$$Speedup \;=\; \frac{SequentialTime}{ParallelTime} . \qquad (2)$$

In the case of a heterogeneous architecture, the "Sequential Time" is given by the time of the best sequential algorithm executed in the machine with the greatest calculation power [1][17][18].

To evaluate how good the speedup obtained is, it is compared with the theoretical speedup of the architecture upon which work is being carried out. The speedup considers the relative calculation power of each machine with respect to the power of the most powerful machine [19]. The theoretical speedup is calculated with formula (3), where *B* is the number of machines of the architecture used, y P_i is the relative calculation power of the *machine i* regarding the best machine power. This relation is expressed in the formula (4).

$$TheoreticalSpeedup = \sum_{i=1}^{B} P_i . \qquad (3)$$

$$P_i = \frac{sequentialTime(powerfulMachine)}{sequentialTime(m_i)} . \qquad (4)$$

6.2 Experiments

The experiments were done on a multi-cluster architecture consisting of four clusters: an 16 Pentium IV 2.4 Ghz homogeneous cluster of 1 Gb memory.

- an 10 Celeron 2 Ghz homogeneous cluster of 128 Mb memory.
- an 8 Duron 800Mhz homogeneous cluster of 256 Mb memory.
- an 8 Pentium III 700 Mhz cluster homogeneous cluster of 256 Mb memory.

Communication within each cluster is done via an Ethernet web, using a switch for communication among clusters.

The language used for the implementations is C together with the MPI library to handle communications among processors.[20]

Tests were carried out using 42 machines, adding one for the dynamic distribution, acting as master, and with different board sizes. (*N* = 17, 18, 19, 20, 21).

In the case of dynamic distribution, it was experimented with different percentages of initial distribution. (*Li* = 0, 5, 10, 15, 20, 25, 50).

6.3 Results

The data of Table 1 shows the percentage of load unbalancing produced by the algorithm for the Direct Static, Predictive Static and Dynamic upon Demand distributions with different *Li* values. Some of these results can be seen in figure 1.

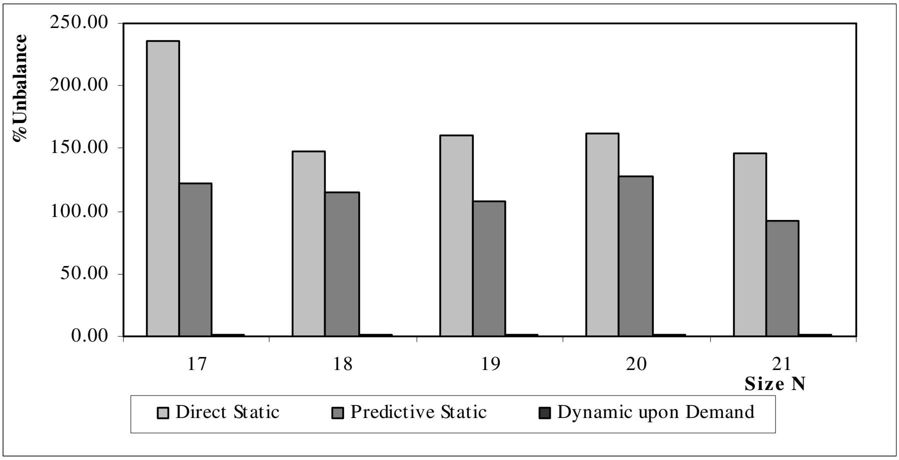

Fig. 1. Graph of Percentage of Load Unbalancing of Direct Static, Predictive Static and Dynamic upon Demand Distributions (*Li*=15). *N*=17,18,19,20,21

Table 1. Percentage of Unbalancing for each test

Size	Direct Static	Predictive Static	Dynamic upon Demand						
			0%	5%	10%	15%	20%	25%	50%
17	236.09	122.37	2.81	1.65	0.13	0.14	0.12	0.12	106.09
18	148.30	115.20	20.93	12.10	0.04	0.04	1.63	27.29	150.29
19	160.66	107.62	152.91	92.72	0.04	0.05	3.91	9.92	131.61
20	162.02	128.21	0.03	0.03	0.03	0.03	0.03	24.09	131.19
21	145.60	92.91	0.03	0.03	0.03	0.03	0.03	16.48	117.34

Table 2 presents the speedup obtained for each test mentioned before together with the optimal speedup (or theoretical) calculated for this machine combination. Table 3 shows the total time for each test.

Table 2. Speedup

Size	Optimum	Direct Static	Predictive Static	Dynamic upon Demand						
				0%	5%	10%	15%	20%	25%	50%
17	31	10.62	17.75	24.13	24.72	24.96	25.14	25.38	23.30	13.49
18	31	12.99	17.31	30.07	30.15	30.20	30.25	29.78	24.00	12.00
19	31	12.21	18.22	30.79	30.59	30.76	30.90	30.12	28.63	13.28
20	31	12.46	15.99	30.85	30.89	30.91	30.95	30.99	24.90	13.21
21	31	13.52	19.76	30.98	30.98	30.99	31.00	30.99	27.10	14.30

Figure 2 shows the speedup obtained with each of the distribution algorithms for some of the tests in Table 2, together with the optimal speedup of this architecture.

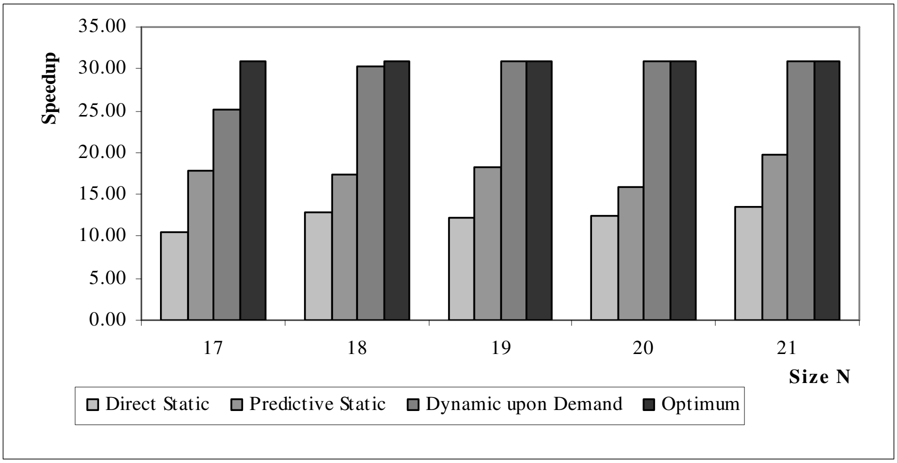

Fig. 2. Speedup of the Direct Static, Predictive Static and Dynamic upon Demand Distributions (*Li*=15) and Optimum. *N*=17, 18, 19, 20, 21.

Table 3. Algorithm Total Time

Size	Direct Static	Predictive Static	Dynamic upon Demand						
			0%	5%	10%	15%	20%	25%	50%
17	4.70	2.81	2.07	2.02	2.00	1.99	1.97	2.14	3.70
18	27.91	20.94	12.06	12.02	12.00	11.98	12.17	15.10	30.19
19	228.16	152.95	90.50	91.09	90.57	90.18	92.51	97.33	209.76
20	1795.20	1399.46	725.15	724.19	723.68	722.92	721.90	898.45	1693.51
21	13957.76	9554.65	6094.24	6094.46	6090.92	6090.52	6090.63	6966.64	13205.15

7 Conclusions and Work Guidelines

Analyzing the results obtained from the experimental work, we can come to the following conclusions:

- Experimental results are coherent with theoretical analysis of unbalance and maximum speedup attainable.
- For the type of problems where Tp>>Tc (as the N-Queens problem that requires a minimal communication among machines), if the work is data-dependent it's essential the choice of data distribution among clusters, to achieve an almost optimal speedup.
- Naturally, algorithms that take into account the calculation power of each machine for work distribution have a better behavior than Direct Static distribution. This improvement is clearly expressed in the load balancing and the speedup.
- Among the algorithm that take into account the calculation power, it can be seen that the algorithms that distribute dynamically can assign work in a more balancing way among the machines (as seen in Graph 1), without much affecting the final time of execution (as shown by the speedup en Graph 2 and the data of Table 3).

- In dynamic distribution, the speedup obtained is quite close to the optimum according to the parallel architecture used in this case, all of which becomes more evident as N increases.

At present, tests are being done with clusters outside the UNLP, particularly at the UNSur (Bahía Blanca), UNComahue (Neuquen), UA Barcelona(Spain) and the Universidad Católica del Salvador (Brasil), through a WAN network. This requires a previous evaluation of the communication costs, for considering them in the computation power model.

References

1. Al-Jaroodi J, Mohamed N, Jiang H, Swanson D. “Modeling parallel applications performance on heterogeneous system”. IEEE Computer Society, 2003.
2. Bohn C, Lamont G. “Load balancing for heterogeneous clusters of PCs”. Future Generation Computer Systems, Elsevier Science B.V., Vol 18, 2002, pp 389-400.
3. Leopold C. "Parallel and distributed computing. A survey of models, paradigms, and approaches". Wiley Series on Parallel and Distributed Computing. Albert Zomaya Series Editor, 2001.
4. Baiardi F, Chiti S, Mori P, Ricci L. “Integrating load balancing and locality in the parallelization of irregular problems”. Future Generation Computer Systems, Elsevier Science B.V., Vol 17, 2001, pp 969-975.
5. Naiouf M. “Procesamiento paralelo. Balance dinámico de carga en algoritmos de sorting”. Tesis doctoral. Universidad Nacional de La Plata, 2004.
6. Watts J, Taylor S. “A practical approach to dynamic load balancing”. IEEE Transactions on Parallel and Distributed Systems, 9(3), March 1998, pp. 235-248.
7. Dongarra J, Foster I, Fox G, Gropp W, Kennedy K, Torczon L, White A. “The Sourcebook of Parallel Computing”. Morgan Kauffman Publishers. Elsevier Science, 2003.
8. Ross K, Yao D. “Optimal load balancing and scheduling in a distributed computer system”. Journal of Association for Computing Machinery, 38 (3): 676-690.1991.
9. Hui C, Chanson S. “Improve strategies for dynamic load balancing”. IEEE Concurrency, pages 58-67. 1999.
10. Dongarra J, Foster I, Fox G, Gropp W, Kennedy K, Torczon L, White A. “The Sourcebook of Parallel Computing”. Morgan Kauffman Publishers. Elsevier Science, 2003.
11. Bruen A, Dixon R. “Then n-queens problem. Discrete mathematics”. 12:393-395, 1997.
12. De Giusti L, Novarini P, Naiouf M, De Giusti A. “Parallelization of the N-queens problem. Load unbalance analysis”. Workshop de Procesamiento Paralelo y Distribuido (WPPD), Congreso Argentino de Ciencias de la Computación (CACIC’03), 2003.
13. Hedetniemi S, Hedetniemi T, Reynolds R. “Combinatorial problems on chessboards: II”. Chapter 6 in domination in graphs: advanced topic, pag 133-162, 1998.
14. Bernhardsson B. ”Explicit solution to the n-queens problems for all n”. ACM SIGART Bulletin,2:7,1991.
15. Somers J. “The N-queens problem a study in optimization”. www.jsomers.com/ nqueen_demo /nqueens.html.
16. Takaken, “N-queens problem (number of solutions)”. http://www.ic-net.or.jp/home/ takaken/e/queen/.
17. De Giusti L., Chichizola F. “Optimización de N-queens Paralelo”. Technical report III-LIDI. 2006.

18. Grama A, Gupta A, Karypis G, Kumar V. "Introduction to parallel computing". Second Edition. Pearson Addison Wesley, 2003.
19. Jordan H, Alaghband G. "Fundamentals of parallel computing". Prentice Hall, 2002.
20. Tinetti F. "Cómputo paralelo en redes locales de computadoras". Tesis Doctoral. Universidad Autónoma de Barcelona, 2004.
21. Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J., "MPI: The Complete Reference", The MIT Press, Cambridge, Massachusetts ,1996.