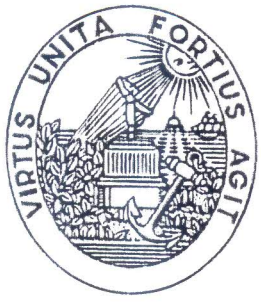
4th international meeting

VECPAR

vector and parallel processing

Proceedings

Part II (June 22)



Faculdade de Engenharia
da Universidade do Porto

2000 June, 21 22 23

VECPAR'2000

4rd International Meeting on Vector and Parallel Processing

2000, June 21-23

Conference Proceedings
Part II
(Thursday, June 22)

FEUP
Faculdade de Engenharia
da Universidade do Porto

Performance Analysis and Modeling of Regular Applications on Heterogeneous Workstation Networks

Andrea Clematis[1] and Angelo Corana[2]

[1] Istituto Matematica Applicata - Consiglio Nazionale Ricerche,
Via De Marini 6, 16149 Genova, Italy
E-mail: clematis@ima.ge.cnr.it
[2] Istituto Circuiti Elettronici - Consiglio Nazionale Ricerche,
Via De Marini 6, 16149 Genova, Italy
E-mail: corana@ice.ge.cnr.it

Abstract. Heterogeneous networks of workstations and/or personal computers (NOW) are increasingly used as a powerful platform for the execution of parallel applications.
Sometimes applications are developed having in mind this type of heterogeneous environment, but in most cases applications already developed for traditional parallel machines (homogeneous and dedicated) are ported to NOWs, resulting in performance degradation due in part to less efficient communications but more often to unbalancing.
In this work we propose a simple model able to analyze and predict performance on heterogeneous NOWs of regular data-parallel applications originally developed for ring or 2-D mesh topologies. To improve performance, the computation time on the various nodes must be as balanced as possible. This can be obtained in two ways: by heterogeneous data partitioning or by assigning to each node a number of processes proportionally to its relative power.
A test case based on matrix multiplication is analyzed and the results predicted by the model are compared with the ones collected experimentally.
Our analysis shows that an efficient porting of homogeneous data-parallel applications on heterogeneous NOWs is possible and can be achieved in most cases in a quite straightforward and effective way.

1 Introduction

In recent years networks of workstations and/or personal computers are increasingly used for the execution of parallel applications [7, 11]. Indeed technological advances make available nodes with high computing power and interconnecting networks with sufficiently high communication speed.

These systems constitute a viable alternative to classical parallel machines (which are homogeneous and dedicated) and have the advantages of a wide availability and a good price/performance ratio.

Main features of NOWs are: heterogeneity, since in most cases the various nodes are different, making a good balancing among nodes a critical aspect;

communication latency that is normally higher that the one in the 'true' parallel machines, imposing limits on fine grain computation.

A simple and effective way to achieve good efficiency on such platforms is the use of the master-worker programming model with the pool of tasks paradigm, which is self-balancing [10]. However, this approach is only feasible if tasks are independent. Moreover, it cannot be adopted if we are interested in the efficient and straightforward porting on NOWs of parallel applications which have been developed with different programming models for homogeneous and dedicated parallel systems.

Particularly, a number of data-parallel applications have been implemented on homogeneous systems with regular topologies such as ring and mesh using the SPMD model, obtaining loosely synchronous applications, well balanced and therefore providing a good efficiency.

If we execute applications belonging to this class on heterogeneous NOWs, the various nodes have in general different speeds, thus the fastest ones exibit a high idle time, resulting in a overall performance degradation. In order to minimize idle time, the computational work in each node must be as close as possible proportional to the computing power of the node.

Similar problems have been recently addressed by other authors. In [1] the problem arising with the use of grid algorithms on heterogeneous workstation networks is addressed, and solution based on sophisticated data allocation methods are proposed.

In this work we consider two possible strategies to obtain a good load balancing: a single process per node with heterogeneous data partitioning; homogeneous data partitioning assigning a different number of processes to each node, according to its computing power.

We propose a simple model able to evaluate performance in the various cases, taking into account the involved parameters at the application level (e.g. computational work and communication amount), at the architectural level (e.g. interconnection network speed) and at both levels (e.g. relative speed of nodes).

A test case based on matrix multiplication is analyzed and the results obtained with the model are compared with the ones collected experimentally.

2 Regular SPMD applications

Many applications are suitable for the parallelization on regular topologies (e.g ring or 2-D mesh) with a even distribution of data among processors.

The code in each node consists normally of an initialization phase, a loop and a termination phase (Fig. 1). In each loop iteration there are a computation phase and a communication phase with neighbouring nodes, i.e. nodes connected by direct links on the considered topology.

For the generic l-th loop iteration ($l = 1, \ldots, L$), the elapsed time T_i on the i-th node can be expressed as

$$T_i = T_i^{comp} + T_i^{comm} + T_i^{idle} \quad (1)$$

5 Simulation and experimental results

We set up a simple model able to simulate the execution of regular applications on NOWs, with the three different approaches outlined in the previous section. The model uses some parameters at the hardware level (i.e. the number of processors p and the network speed, expressed by α and β), and some parameters which also depend on the selected application (i.e. the atomic time τ on the reference node and the relative node speeds s_i). The third approach also requires the total number of processes q. From such low level parameters, the model computes for the given application the computation, communication and idle times at the loop iteration level for each processor. In this way the model yields the figures of speed-up and efficiency of the whole application.

The model is tested using the matrix multiplication algorithm that computes $C = A \times B$, with A, B and C $n \times n$ matrices, on a logical ring of processes, as described in [13].

In the original SPMD implementation with homogeneous data partitioning each processor i stores a slice of matrix A and a slice of matrix B, each comprising rows from $(i-1)n/p$ to $i \cdot n/p$. Slices of A remain local to the various processors, whereas slices of B circulate along the ring. The whole computation requires p loop iterations and at the end processor i has computed n/p rows of C, from row $(i-1)n/p$ to row $i \cdot n/p$.

So, the computation time of node i during the l-th iteration is

$$T_i^{comp} = 2\frac{n^3}{p^2}\frac{\tau}{s_i}, \quad i = 1, \ldots, p \tag{23}$$

and in each iteration n^2/p elements of B are moved between neighbouring nodes.

Using the heterogeneous data partitioning approach means in this case to assign slices of matrix A to each node with a number of rows proportional to its relative speed, whereas matrix B is still evenly partitioned among nodes.

The third approach is exactly the same as the first, with the exception that q processes (with $q > p$) are generated and the optimal q_i are given by eq. (19).

The various versions of this test program are implemented using C language and PVM v. 3.4 and executed on a variable number of nodes belonging to a NOW of six workstations connected by a switched Ethernet. Table 2 shows the characteristics of the various nodes and the total power of the different configurations.

The trials are executed on dedicated nodes and with a low traffic on the network. The measured value of the time per element on the reference node is $\tau = 0.56\ \mu sec$. We measure on the network the values $\alpha = 1\ msec$ and $\beta \simeq 1\ \mu sec$.

Experimental data has been collected using 1000×1000 floating point matrices. Table 3 reports the measured and simulated speed-up for the three different approaches. As expected, the speed-up of the straightforward homogeneous partitioning is well below the ideal one, while the two proposed strategies to reduce unbalancing yield considerably better speed-up figures.

Table 2. The first column identifies the configuration, that includes nodes up to the current row; for each configuration the type and the relative speed of the nodes, the total computing power and the degree of heterogeneity h are reported

Config. Id.	Workstation	Relative speeds	Available computing power	h
-	Sparc-20	1.00	1.00	-
C1	SGI-O2	1.87	2.87	0.31
C2	SGI-O2	1.90	4.77	0.37
C3	Sparc-Ultra 5	1.87	6.64	0.40
C4	Sparc-Ultra 5	1.85	8.49	0.41
C5	Indigo 2	5.87	14.36	0.58

Table 3. The first column gives the configuration identifier; the SPMD columns provide speed-up for homogeneous SPMD application measured (M) and simulated(S); HD columns summarize speed-up for heterogeneous data partitioning; the VP colums provide speed-up for homogeneous data partitioning but with a number of processes on each node proportional to its relative speed (the total number of processes q is reported in the last column)

Config. Id.	SPMD-M	SPMD-S	HD-M	HD-S	VP-M	VP-S	q
C1	2.06	1.99	2.89	2.86	2.71	2.80	3
C2	3.09	3.00	4.69	4.76	4.68	4.66	5
C3	4.62	4.00	6.62	6.62	6.14	6.51	7
C4	5.80	5.00	8.36	8.48	8.18	8.35	9
C5	6.54	5.95	13.23	14.24	12.67	13.6	15

Measured and experimental data are in most cases in good agreement, thus confirming that the proposed model is quite reliable. The maximum errors occur in the case of SPMD homogeneous implementation, and it is due to an underestimation of the relative speed of the slowest nodes. In fact we assume that the relative speed of each node does not vary with the data size handled by the node. Indeed, we can sometimes observe a gain in processor speed when the amount of local data decreases, for example due to better use of the hierarchy of memories. This is more relevant in the homogeneous data partitioning case where the relative weight of the slowest nodes is greater.

6 Conclusions

We analized the problem of porting data-parallel applications originally developed for homogeneous parallel systems with regular topologies (e.g. ring or mesh) to network of workstations and/or personal computers.

For this kind of computing resources, maintaining the even partitioning of data among processors yields poor performance, since efficiency is limited by unbalancing, that increases with the degree of heterogeneity of the network.

Two strategies are considered to overcome this problem: heterogeneous data partitioning or allocation to each node of a number of processes proportionally to its relative power.

A simple model is proposed to analyze and predict performance of the considered class of applications using the various approaches.

The model is tested using a matrix multiplication algorithm with processes arranged in a ring topology. A good agreement is obtained between simulated and experimental figures of performance both for the naive unbalanced implementation and for the two improved implementations.

References

1. Beaumont, O., Boudet, V., Rastello, F., Robert, Y.: Data Allocation Strategies for Dense Linear Algebra Kernels on Heterogeneous Two-dimensional Grids. Int. Parallel and Distributed Processing Symposium (IPDPS'2000). Cancun (Mexico), 1-5 May 2000
2. Cermele, M., Colajanni, M., Necci, G.: Dynamic Load Balancing of Distributed SPMD Computations with Explicit Message Passing Systems. Proc. 6-th Heterogeneous Computing Workshop. IEEE CS Press (1997) 2-16
3. Clematis, A., Corana, A.: Modeling Performance of Heterogeneous Parallel Computers. Parallel Computing **25** (1999) 1131-1145
4. Clematis, A., Dodero, G., Gianuzzi, V.: A Resource Management Tool for Heterogeneous Networks. Proc. 7-th Euromicro Workshop on Parallel and Distributed Processing. IEEE CS Press (1999) 367-373
5. Corana, A.: Parallel Computation of the Correlation Dimension from a Time Series. Parallel Computing **25** (1999) 639-666
6. Fox, G.C., Johnson, M.A., Lyzenga, G.A., Otto, S.W., Salmon, J.K., Walker, D.W.: Solving Problems on Concurrent Processors. Vol. 1. Prentice-Hall, Englewood Cliffs, NJ (1988)
7. Khokhar, A.A., Prasanna, V.K., Shaaban, M.E., Wang, C.: Heterogeneous Computing: Challenges and Opportunities. Computer **26** (1993) 18-27
8. Mazzeo, A., Mazzocca, N., Villano, U.: Efficiency Measurements in Heterogeneous Distributed Computing Systems: from Theory to Practice. Concurrency: Practice and Experience **10** (1998) 285-313
9. Ranka, S., Sahni, S.: Hypercube Algorithms with Applications to Image Processing and Pattern Recognition. Springer-Verlag, Berlin Heidelberg New York (1990)
10. Schmidt, B.K., Sunderam, V.S.: Empirical Analysis of Overheads in Cluster Environments. Concurrency, Practice and Experience **6** (1994) 1-32

11. Sunderam, V.S.: Methodologies and Systems for Heterogeneous Concurrent Computing. In: Joubert, G.R., Trystram, D., Peters, F.J., Evans, D.J. (eds.): Parallel Computing: Trends and Applications. Elsevier, Amsterdam (1994) 29-45
12. Sunderam, V.S., Geist, G.A., Dongarra, J., Manchek, R.: The PVM Concurrent Computing System: Evolution, Experiences, and Trends. Parallel Computing **20** (1994) 531-545
13. Yan, Y., Zhang, X., Song, Y.: An Effective and Practical Performance Prediction Model for Parallel Computing on Nondedicated Heterogeneous NOW. J. Parallel and Distributed Computing **38** (1996) 63-80
14. Zaki, M.J., Li, W., Cierniak, M.: Performance Impact of Processor and Memory Heterogeneity in a Network of Machines. Proc. Fourth Heterogeneous Computing Workshop. Santa Barbara, California (1995)