Installation routines for linear algebra libraries on LANs *

Domingo Giménez[1] and Ginés Carrillo[2]

[1] Departamento de Informática: Lenguajes y Sistemas Informáticos.
Univ de Murcia. Aptdo 4021. 30001 Murcia, Spain.
domingo@dif.um.es

[2] Departament de Tecnología
Universitat Pompeu Fabra.
La Rambla, 30-32. 08002 Barcelona, Spain.
gines.carrillo@tecn.upf.es

Abstract. In this paper we study the design of installation routines for linear algebra routines on networks of processors. The main idea is to develop those installation routines in such a way that they allow inexpert users to execute the parallel linear algebra routines with an optimum number of processors and distribution of data. The designing methodology of these routines has been analyzed for homogeneous and heterogeneous networks, and the experimental results obtained with a gaussian elimination routine are shown.

1 Introduction

In the last years parallel/distributed computing has become widely popular due in part to the possibility of using some processors connected by a communication network as a parallel system. In that way the use of parallel systems has become cheaper and easier, and new users are begining to use parallel programming. In particular, users with great computational necessities (scientists and engineers) now have the possibility of solving their problems using the machines they have access to, connecting them by means of some communication network (ethernet, fastethernet, myrinet, ...) and using message-passing parallelism, without high additional cost. But the design of message-passing parallel programs is not an easy task, specially for inexpert users. The problems these users need to solve are in many cases linear algebra problems: solution of linear systems, or eigen-problems. This is why we are working on the design of linear algebra routines especially for LANs (Local Area Networks) [1].

There is other research in which the design of linear algebra routines for heterogeneous networks of processors is analysed [2–4]. In some cases the distribution of data in the system is obtained dynamically [2], and statically in others [3]. Our goal is to statically obtain data distributions close to the optimum.

* Partially supported by Comisión Interministerial de Ciencia y Tecnología, project TIC96-1062-C03-02.

Since the users we are thinking of are not experts in parallel programming, one possibility is to develop installation routines for the linear algebra routines, in such a way that the user can execute a linear algebra routine in a processor. This routine consults information generated by the installation routine and decides the number of processors to use and the best data distribution to obtain the lowest execution time.

Each linear algebra routine has an associated installation routine which obtains approximate optimum values of the number of processors and the block size in which the matrices are divided. The installation routines are executed during the installation of the linear algebra library, but they can be re-executed each time the conditions of the system change, i. e. more memory in some machines, modifications in the management of the file system or addition or elimination of a processor.

As the system can be formed by processors with different capacities, it may be preferable to develop linear algebra routines and the corresponding installation routines for heterogeneous systems.

In this paper we analyse the methodology for the design of these installation routines for homogeneous and heterogeneous LANs. Experimental results obtained with a gaussian elimination routine and with variations in the heterogeneity of the network are shown.

2 Installation routines

A gaussian elimination routine has been used to study the methodology of the design of installation routines and the behaviour of these routines in systems with different characteristics.

The matrix is considered as divided in blocks of adjacent rows, and the blocks are assigned to the processors using a rowwise block-cyclic-striped mapping [5].

If the system is homogeneous the blocks can be all the same size (b). When the block size increases the imbalance increases, but the number of communications decreases. Thus, for a given matrix size (n), the installation routine must obtain the number of processors (p) and the block size, with which the lowest execution time is obtained.

In the case of a heterogeneous system, the block size is not the same for the different processors, and the blocks assigned to processors with higher computational capacity would be larger. The installation routine also obtains the optimum number of processors and block size, but in the linear algebra routine the size of the blocks in each processor is obtained by the formula:

$$b_i = \frac{v_i}{\sum_{i=1}^{p} v_i} pb \tag{1}$$

where v_i is the speed of processor i (i. e. in Mflops) and b_i the size of blocks assigned to processor i. This is the way in which other software for heterogeneous computing works [2] when dynamically deciding the data distribution in the system. In our case the assignment is not done dynamically since to dynamically obtain the optimum number of processors and block size would be too costly. What could be done in our case is to dynamically obtain the values of v_i by executing in each processor some basic matrix operation (i. e. a matrix multiplication), but we have preferred to leave this work to the installation routine because we consider LAN systems are normally well controlled. In any case, the modification to obtain the block sizes b_i dinamically is only a small one, and it could be done in the linear algebra routine.

The installation routine could work by obtaining the values of p and b from a formula of the theoretical execution time (depending on the linear algebra routine) and the values of the cost of an arithmetic operation, of the start-up time and word-sending time obtained experimentally for the system. Some problems remain: to obtain the theoretical execution time we normally make some assumptions which are valid when the matrix size increases, but not with small matrices, and it is more difficult to predict the experimental results in LANs than in multicomputers, due to the characteristics of the communication network.

Another possibility is to obtain the optimum number of processors and block size by performing a number of executions. In that case the system manager decides the minimum and maximum matrix size and the increment of the matrix size. Experiments are performed for these matrix sizes, but to perform the experiments for all the possible numbers of processors and block sizes could be too expensive, and what the routine does is to obtain the number of processors and block size for the smallest matrix and it uses the values obtained for a matrix size as initial values for the next matrix size. Since the optimum values of p and b vary in a continuous manner with the matrix size, the cost of the installation routine thus becomes acceptable.

In the homogeneous case, a file with the matrix sizes and the associated number of processors and block size is generated, and the linear algebra routine consults this file for the entry with the matrix size nearest to the actual matrix size in order to decide the number of processors and the distribution of the matrix in the execution.

In the heterogeneous case, that file is also generated along with an additional file with the proportional speeds of the processors, classified from fastest to slowest. The linear algebra routine takes the number of processors from the first file, the processors to use in the execution from the second file, and the block sizes are obtained using formula 1 with the value b obtained from the first file and the values v_i from the second file.

3 Experimental results

In this section the experimental results obtained using different installation routines for a gaussian elimination routine are shown. The entries of the matrices have been randomly generated, and a network of SUN Ultra workstations connected by ethernet has been used. Three networks are considered:

- A network of five SUN Ultra 1 with the same computational capacity, but with one of them managing the file system, and consequently this machine works more slowly. This is the more homogeneous system of the three, and it will be called HOM.
- A network obtained by adding a SUN Ultra 5 to HOM, which is quicker than the other processors. This network will be called HIB.
- A network with three processors: the SUN Ultra 1 which manages the file system, another SUN Ultra 1, and the SUN Ultra 5. Since this network can be considered the more heterogeneous, it will be called HET.

In figures 1, 2 and 3 the results obtained with the different installation routines are compared with those obtained experimentally. The figures show the quotients of the execution time obtained with the optimum number of processors and block size given by the different installation routines and the optimum execution time obtained experimentally. Figure 1 shows the quotient for HOM, figure 2 for HIB, and figure 3 for HET. The installation routines used are: HOMEXP, the homogeneous-experimental routine; HETEXP, the heterogeneous-experimental; and THEOR, the theoretical routine. THEOR is used in HOM to obtain the theoretical optimum number of processors and block size to use in each processor and in HIB and HET the same number of processors is used, but the block sizes are different in the different processors.

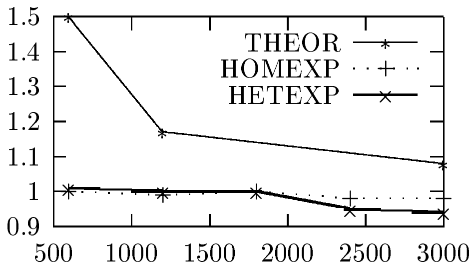

Fig. 1. Comparison between the different installation routines in HOM. Quotient of the execution time obtained with the values of p and b provided by the installation routines and the best execution time.

Some considerations can be made:

- The theoretical routine does not predict the number of processors and matrix distribution well for small matrices but when the matrix size increases the prediction is better. In some cases (HET) the theoretical prediction is as good as the experimental prediction. Theoretical prediction could be performed for big matrix sizes, since in that case the experimental routines are too expensive.

– The homogeneous-experimental routine works well when the network is close to a homogeneous network, but the heterogeneous-experimental routine is preferable when the heterogeneity increases.
– The heterogeneus-experimental routine works well in all the systems, so this type of routine is the best as installation routine.

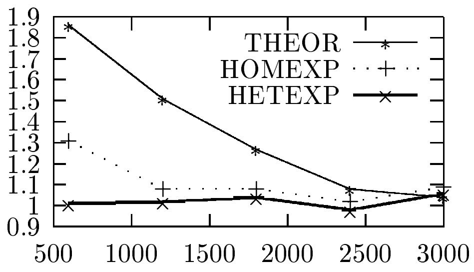

Fig. 2. Comparison between the different installation routines in HIB. Quotient of the execution time obtained with the values of p and b provided by the installation routines and the best execution time.

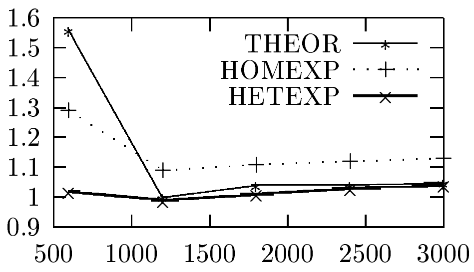

Fig. 3. Comparison between the different installation routines in HET. Quotient of the execution time obtained with the values of p and b provided by the installation routines and the best execution time.

4 Conclusions

We have shown a methodology to design installation routines for linear algebra libraries for non parallel programmers in LANs. These routines can be used to provide the users with the number of processors and the block sizes for solving the problems in a time close to the optimum time.

The experiments performed show satisfactory outcomes.

Our idea is to design a linear algebra library using installation routines of the type analysed in this paper.

References

1. D. Giménez, C. Jiménez, M. J. Majado, N. Marín and A. Martín. Solving Eigenvalue Problems on Networks of Processors. In José M. L. M. Palma, Jack Dongarra and Vicente Hernández, editor, *Vector and Parallel Processing-VECPAR-98*, number 1573 in Lecture Notes in Computer Science, pages 85–99. Springer, 1999.
2. A. Kalinov and A. Lastovetsky. Heterogeneous Distribution of Computations While Solving Linear Algebra Problems on Networks of Heterogeneous Computers. In P. Sloot, M. Bubak, A. Hoekstra and B. Hertzberger, editor, *High Performance Computing and Networking*, number 1593 in Lecture Notes in Computer Science, pages 191–200. Springer, 1999.
3. Pierre Boulet, Jack Dongarra, Fabrice Rastello, Yves Robert and Frédéric Vivien. Algorithmic issues on heterogeneous computing platforms. *Parallel Processing Letters*, 9(2):197–213, 1999.
4. Vincent Boudet, Antoine Petitet, Fabrice Rastello and Yves Robert. Data Allocation Strategies for Dense Linear Algebra Kernels on Heterogeneous Two-dimensional Grids. In *IASTED Parallel and Distributed Computing and Systems*, 1999.
5. Vipin Kumar, Ananth Grama, Anshul Gupta and George Karypis. *Introduction to Parallel Computing. Design and Analysis of Algorithms.* The Benjamin Cummings Publishing Company, 1994.