Contemporary Mathematics
Volume 157, 1994

Implementation of Domain Decomposition Techniques on nCUBE2 Parallel Machine

A. BRAMBILLA, C. CARLENZOLI, G. GAZZANIGA,
P. GERVASIO, G. SACCHI

ABSTRACT. In this paper some aspects related to the parallel implementation of the Dirichlet-Neumann procedure for solving bidimensional elliptic problems with domain decomposition techniques are described. In order to asses the efficiency of the implementation on a parallel distributed memory computer, elapsed times and speed-up factors are considered. Their dependence upon the number of processors involved, the degree of the polynomials and the kind of linear system solver used in the spectral approximation of the problem is also discussed.

1. Multidomain formulation of the mathematical problem

We are interested in solving the following elliptic boundary value problem

$$(1) \qquad \begin{cases} Lu = -\Delta u + \alpha u = f & \text{in } \Omega \\ u = g & \text{on } \Gamma^D \\ \dfrac{\partial u}{\partial n} = \phi & \text{on } \Gamma^{Ne} \end{cases}$$

where $\Omega \subset \mathbb{R}^2$ is an open bounded domain with boundary $\partial\Omega$ such that:

$$\partial\Omega = \bar{\Gamma}^D \cup \bar{\Gamma}^{Ne}, \ \Gamma^D \cap \Gamma^{Ne} = \emptyset; \ \alpha(x,y) \geq 0 \ \forall \ (x,y) \in \Omega, \ \alpha \in C^0(\Omega),$$

f, g, ϕ are given suitable functions on Ω, Γ^D and Γ^{Ne} respectively; n is the outward unit normal vector to $\partial\Omega$.

1991 Mathematics Subject Classification. Primary 65Y05, 65N55.
Partially supported by:
"Progetto Finalizzato: Sistemi Informatici e Calcolo Parallelo" of C.N.R.
This paper is in final form and no version of it will be submitted for publication elsewhere

0271-4132/94 $1.00 + $.25 per page

The variational formulation of problem (1) and related results for the existence and uniqueness of the solution can be found for instance in [2].

In order to obtain a multidomain formulation of problem (1), the computational domain Ω is partitioned as in Fig. 1 into M nonoverlapping subdomains Ω_i,
$i = 1, .., M$, such that $\Omega = \cup_{i=1}^{M} \Omega_i$, $\Gamma_i = \partial\Omega_i \cap \partial\Omega_{i+1}$, $i = 1, .., M-1$ and $\Gamma = \cup_{i=1}^{M-1} \Gamma_i$.

Let us denote by $u_i = u|_{\Omega_i}$ $i = 1, .., M$, then problem (1) becomes

$$(2) \qquad \begin{cases} Lu_i = -\Delta u_i + \alpha u_i = f & \text{in } \Omega_i \quad i = 1, .., M \\ u_i = g & \text{on } \Gamma^D \cap \partial\Omega_i \\ \dfrac{\partial u_i}{\partial n} = \phi & \text{on } \Gamma^{Ne} \cap \partial\Omega_i \end{cases}$$

with the following regularity conditions of the solution on the interface:

$$(3) \qquad u_i = u_{i+1} \text{ on } \Gamma_i, \ i = 1, \ldots, M-1 \quad \text{(continuity of the solution } u\text{)}$$

$$(4) \qquad \frac{\partial u_i}{\partial \nu} = \frac{\partial u_{i+1}}{\partial \nu} \text{ on } \Gamma_i, \ i = 1, \ldots, M-1 \quad \text{(continuity of the flux)}$$

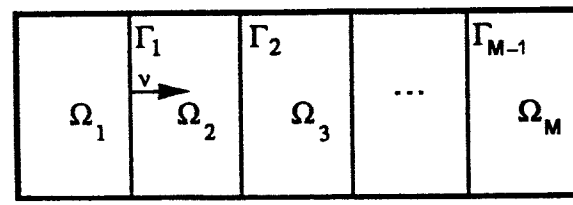

FIG. 1: STRIPWISE SUBDIVISION OF Ω.

Similar approaches to the partitioning of domains have been dealt with by, for instance, Max Dryja and Olof B. Widlund. A complete description of domain decomposition methods, we refer to in this paper, can be found in [4].

The solution of the multidomain formulation (2)-(4) can be carried out decoupling the problem on Ω_i, with i odd, and on Ω_i, with i even, and assigning transmission conditions as follows:

- Dirichlet condition (3) to the problems on Ω_i with i odd
- Neumann condition (4) to the problems on Ω_i with i even.

In this way an iterative scheme, known as Dirichlet/Neumann procedure (D/N), is produced.

On each subregion Ω_i, $i = 1, \ldots, M$ the numerical approximation of problem (2) is carried out by using spectral collocation methods (see [3], cap. 2).

2. The implementation of Dirichlet/Neumann scheme

Referring to a partition of Ω as in Fig. 1, the solution of problem (2)-(4) by means of the D/N procedure can be implemented, step by step, as follows:

Step 1 Construction of matrices on each Ω_i, $i = 1, \ldots, M$

Step 2 Solution of Dirichlet problems on Ω_i, i odd, and evaluation of the interface conditions

Step 3 Solution of Neumann problems on Ω_i, i even

Step 4 Relaxation on Γ of the solutions computed in Step 2 and Step 3

Step 5 Redo from Step 2 until convergence.

The computations involved in the previous scheme can be carried out following the natural large grain parallelism suggested by the scheme itself. More precisely in Step 1 M different processor elements can simultaneously compute the matrices associated to M different subdomains with degree of parallelism equal to M.

In Step 2 and in Step 3 $M/2$ processor elements can solve independently all the Dirichlet problems as well as the Neumann problems. We remark that Step 3 must follow Step 2, so the degree of parallelism of these steps is $M/2$. Step 4 and Step 5 are devoted to the evaluation of the new Dirichlet conditions and of convergence condition respectively. So they can be carried out by the processor elements devoted to the solution of Dirichlet problems.

This scheme was implemented on a parallel distributed memory machine using macrotasking techniques following the natural mathematical parallelism of the D/N procedure. In this way each subproblem (on Ω_i) is allocated to a different processor: for example processors with odd identification number are devoted to Dirichlet problems, while processors with even identification number are devoted to Neumann problems. Each processor P_i must construct the matrix related to the subproblem on Ω_i and solve the linear system arising from spectral approximation with a direct (LU factorization) or iterative (Bi-CGStab) method. Obviously the execution of all Dirichlet (Neumann) problems can be carried out concurrently.

We point out that in our approach the parallelism is achieved at the mathematical level with regard to the D/N algorithm, whereas the linear system

solvers (both direct and iterative) inside each subdomain are implemented in a sequential mode.

Data structure, too, was splitted on the local memory of the different processors, so the execution of the algorithm requires information exchange among the processors. Precisely, odd processors must send the right-hand side of the Neumann equation, computed in Step 2, to even processors. On the contrary even processors must send the odd processors the solution on interface computed in Step 3 and their contribution to the evaluation of the relaxation parameter (Step 4). The exchange of information requires processors synchronization and it doesn't allow the parallel execution of Steps 2-3 and Steps 3-4.

3. Numerical results

The parallel machine used for the numerical experiments was a nCUBE2 hypercube computer, hosted by a SUN 4/330 Sparcstation. The cube consists of 16 processor elements with 4 Mbytes of local memory each. This kind of machine is equipped with a high speed (2.22 Mbytes/sec of rate) hardware message routing which allows direct pass through of messages.

In order to evaluate the performances of the D/N algorithm described in the previous section we introduce in addition to the usual speed-up factor $S_p = \frac{T_1}{T_p}$ (T_1 and T_p = elapsed time to run the algorithm on one and p processors respectively), the so called *maximum expected speed-up* (see [1])

$$S_{max} = \frac{M}{p_1(CP) + 2 \star p_2(CP)}$$

where $p_1(CP)$ is the relative cost (in percent of the total CPU-time) of the part of the program with degree of parallelism equal to the number of subdomains (i.e. processor elements) M and p_2 (CP) is the relative cost of the part with degree of parallelism equal to $M/2$.

The numerical experiments differ for the number M of the subdomains (and processors) involved and for the degree N of the polynomial used in the spectral approximation of the elliptic boundary value problems on each Ω_i.

The linear systems $A_i u_i = f_i$, $i = 1, ..., M$ arising from spectral approximation are solved on each Ω_i using both direct and iterative methods.

The use of direct methods allows the construction and factorization of the matrices A_i to be carried out once at the beginning of the procedure (Step 1). The direct solver implemented was Doolittle LU factorization. In this case, if

$(N+1)^2$ is the number of d.o.f. in each Ω_i, the computational cost of the factorization is $O(\frac{N^6}{3})$ floating point operations for each matrix A_i. The factorization of the matrices can be carried out, as already said, concurrently on M different processors and with parallelism equal to M. At each iteration of the D/N algorithm, the triangular linear systems on each subdomain are solved by means of backward and forward substitution. The computational cost of this step is $O(N^4)$ floating point operations per subdomain and the degree of parallelism is $M/2$. Then we observe that the most expensive part of the computation is carried out with degree of parallelism equal to M.

The iterative method we used to solve the M linear systems was the preconditioned Bi-CGStab (see [5]). As preconditioning matrices we used the five diagonal finite difference matrices approximating the same elliptic subproblems. In this case, in Step 1 one can only perform the construction and factorization of the preconditioners with a computational cost equal to $O(2N^3)$ floating point operations. Then at each iteration of the D/N algorithm some Bi-CGStab iterations are needed on each subdomain. One Bi-CGStab iteration costs $O(8N^3)$ floating point operations and so the most relevant part of the computation is carried out with a degree of parallelism equal to $M/2$.

In Figg. 2 and 3 the speed-up factors obtained using both the direct and the iterative solvers are shown. Bars stand for real speed-up S_p and lines stand for maximum expected speed-up S_{max}. According to previous remarks, both S_p and S_{max} obtained using the direct solver are higher than those obtained using the iterative solver.

The different values of the speed-up factors obtained can also be explained by the need of information exchange due to the distributed memory architecture. Although the amount of data to be exchanged is the same both in direct and iterative solver, the iterative method might require different numbers of iterations depending on the boundary conditions used on each domain. In this case the parallel execution needs longer synchronization times to be spent waiting data from domains requiring more iterations. On the contrary the direct method allows a better synchronization among the processors, since the forward and backward resolution is independent of the domain.

The numerical experiments point out that, as predicted by the theory [4], the number of iteration NIT of the D/N procedure is independent of both the polynomial degree N and the solver used on each subdomain. Moreover NIT is increasing as a function of the number M of subdomains.

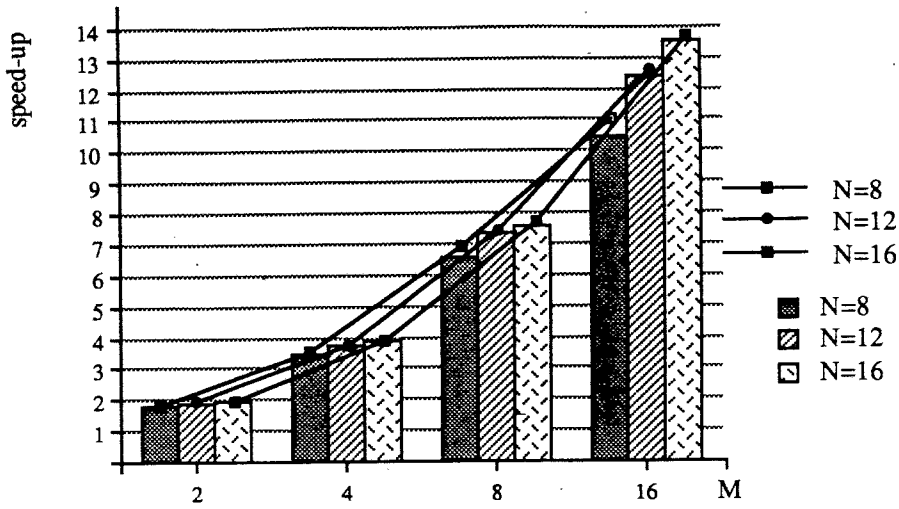

FIG. 2: SPEED-UP FACTOR OBTAINED USING A DIRECT SOLVER.

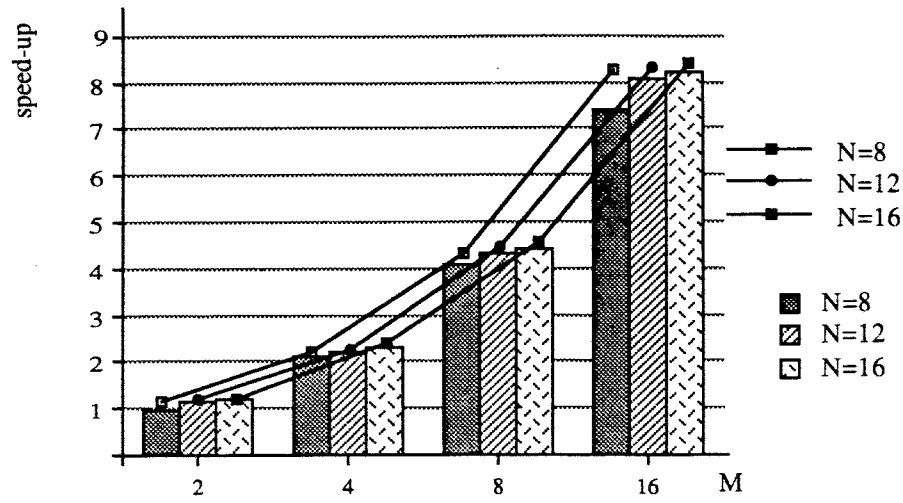

FIG. 3: SPEED-UP FACTOR OBTAINED USING AN ITERATIVE SOLVER.

As far as the elapsed time is concerned, it can be pointed out that, following the structure of the implemented algorithm, the direct solver is more convenient than the iterative one for small values of N ($N \leq 8$) as well as for large values of NIT.

More details about the implementation, the numerical results and the performances of the procedure can be found in [1].

4. Conclusions

The main interest in this work was to asses the efficiency of parallel distributed memory architectures with regard to the implementation of the wellknown D/N algorithm. This kind of implementation seems to be very effective in the solution of differential problems. So the treatment of more complex geometries (such as subdivisions with internal cross-points) could be interesting in order to extend the procedure to more general problems.

A second improvement could be the implementation of the D/N procedure with a finer grain parallelism. It requires the use of parallel solvers for the linear system on each subdomain. This could increase the amount of the information to be exchanged between the processors, reducing the global performance of the algorithm. So the choice and the implementation of the inner solver become a crucial point for an effective use of this method.

REFERENCES

1. Brambilla A., Carlenzoli C., Gazzaniga G., Gervasio P., Sacchi G., Solving elliptic boundary value problems on a distributed memory parallel computer, in preparation.
2. Brezis H., *Analyse Functionelle*, Paris, 1983.
3. Canuto C., Hussaini M.Y., Quarteroni A., Zang T.A., *Spectral methods in fluid dynamics*, Springer Verlag, New York, 1988.
4. Quarteroni A., Domain Decomposition and Parallel Processing for the Numerical Solution of Partial Differential Equations, Surv. Math. Industry, **1** (1991), 75-118.
5. Van der Vorst H.A., A fast and smoothly converging variant of Bi-CG for the solution of non symmetric linear systems, SIAM J. Sci. Statist. Comput., **13**, 2 (1992), 631-644.

ISTITUTO DI ANALISI NUMERICA DEL C.N.R., C.SO C. ALBERTO 5, 27100 PAVIA, ITALY
E-mail address: gia07@ipvian.ian.pv.cnr.it

DIPARTIMENTO DI MATEMATICA, UNIVERSITA' CATTOLICA, VIA TRIESTE 17, 25121 BRESCIA, ITALY
E-mail address: ucxaan@imicilea.cilea.it