Parallel computing for electromagnetic field computation

Christian Vollaire, Laurent Nicolas, Alain Nicolas

HAL Id: hal-00141576
https://hal.science/hal-00141576

Submitted on 20 Apr 2007

Parallel Computing for Electromagnetic Field Computation

C. Vollaire, L. Nicolas, A. Nicolas

CEGELY - UPRESA CNRS 5005 - Ecole Centrale de Lyon

BP163 - 69131 Ecully cedex - France

Abstract–This paper deals with parallel computation in electrical engineering. Shared memory and distributed memory architectures are presented, with their implication in the development of parallel numerical algorithms. The necessity of *optimizing* the parallel performances is highlighted. Both Cray C98 and Cray T3E are finally compared.

Index terms–Finite element methods, parallel algorithms, shared memory systems, distributed memory systems, electromagnetic scattering.

I. INTRODUCTION

Numerical computation is more and more used in engineering sciences to develop new devices or to optimize operating apparatus. Many of the numerical modeling packages make use of the Finite Element (F.E.) method. Unfortunately, this method requires the 3D mesh of the whole studied domain. This is especially expensive for open boundary problems. Furthermore, the size of the 3D mesh makes the modeling of large, coupled or complex problems difficult or even impossible to perform.

Only parallel computers provide the increase in computing performance required to solve such types of problems today. Two reasons may be highlighted: when large memory is needed because of a large amount of data, or when speed is needed to obtain the solution [1]. In recent years, several papers in parallel computational electromagnetics were published [2-6], especially since new distributed memory architectures appeared. Several parallelizations of existing codes were also reported [7]. However, electrical engineering seems to be behind other domains in parallel computation. For example, 3D optimization using genetic algorithms for Navier-Stokes computations is today currently performed in fluid dynamics [8].

The cost related to the parallel computation is an important point to emphasize. This cost has to be estimated in terms of computational developments -how much have numerical algorithms to be modified in order to take advantage of parallel computation?- and in terms of use of parallel computers. A good match between the algorithms and the architecture of the computer has to be found to obtain the optimum performances. This is especially true for new developed codes. For existing industrial codes, the crucial

question is: how easily can they be ported onto different parallel architectures?

The object of this paper is to show the interest of the parallel computing for electrical engineering. Shared and distributed memory parallel computers are first presented. The implications of the parallel architecture on the algorithms is then highlighted. A comparison between the Cray C98 and the Cray T3E is finally presented as example.

II. SHARED AND DISTRIBUTED MEMORY PARALLEL COMPUTERS

We are concerned only by Multiple Instruction Multiple Data (MIMD) parallel computers: they are multi-purpose and the most adapted to the type of problems that we want to solve. These computers may be classified under two categories, depending on the type of access to the memory they provide: shared memory and distributed memory.

A. Shared Memory Parallel Architectures

They are composed of multiple processors, the memory is shared by all the processors, and the communication is performed via a high speed interconnection network. These computers may also have vector capabilities, such as the Cray YMP or the Cray C98.

The exchange of data is performed by accessing the same memory address. Semaphores are used to prevent the problem of simultaneous access to the same data by various processors. The programming environment is generally well developed on such computers. Automatic vectorization and parallelization tools are available. Many profiling tools also give information about the performances of the codes (parallelism ratio, parallel/sequential portion of the code, average vector length, ...).

It seems however that this type of technology has reached a ceiling: it becomes difficult to increase the number of interconnected processors and the size of the memory. Obviously, computers like Cray C94/C98 will be replaced before the end of the century. Furthermore, this type of computer is expensive, requiring small companies to share their costs and thus their use.

B. Distributed Memory Parallel Architectures

They are composed of independent subsystems. Each processor has its own memory, and the communication is achieved using message passing. An additional cost due to

the communications is then unavoidable and the interconnection network is crucial for the parallel performances. Furthermore, the question of synchronization between the processors arises. At present, no tool seems really efficient to automatically parallelize codes on such architectures.

Cray T3E or Intel Paragon are examples of distributed memory parallel computers. Access to such a computer may be cheaper than the access to a shared memory one. For example, it is possible to obtain good computation performances by using a cluster of workstations or a cluster of personal computers.

C. Present Trend

Distributed memory parallel computers then become unavoidable. Presently, they are the only known way allowing the efficient use of a large number of processors together with a large physical memory. New hybrid parallel architectures are arising: they combine shared and distributed memory and they are equipped with powerful vector processors, such as the FUJITSU VPP300. This allows a better quality price-performance ratio and the operating costs are reduced.

III. PARALLEL ALGORITHMS

Some important points have to be taken into account. First, software tools provided with the computer, for the analysis of the parallel performance or for the automatic parallelization/vectorization, have a great impact on the way that parallelism is realized. Secondly, numerical algorithms have to be lightly or deeply modified in order to take advantage of parallel processing. Thirdly, the presence or the lack of portability is a choice to make: it seems clear that, the more a code is optimized for a given computer, the more it will be non-portable on other machines.

Parallel algorithms have to be adapted to suit the architecture of the computer in order to obtain the best parallel performance. Whatever the type of architecture, a good load balancing and a minimization of the overhead introduced by the parallelism are required.

A. Shared Memory Parallel Computers

On shared memory computers, parallelism is introduced by splitting the loops. The main work to parallelize the code concerns the study of the scope of the variables and the tiling [9]. This can be performed automatically or manually.

As example, automatic vectorization and parallelization tools are available on the Cray C98. This is called autotasking. But it is necessary to add compiler directives manually and to find the scope of each variable to increase parallel performance to an acceptable level (Fig. 1). This technique is called microtasking. Vector performances have
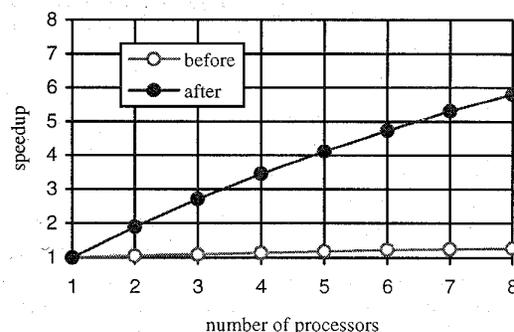

Fig. 1. Cray C98-speedups before and after manual optimization.

also to be taken into account. The main problem on such architectures consists in finding a good compromise between vectorization and parallelization.

B. Distributed Memory Parallel Computer

On distributed memory computers, several programming models are possible (SPMD, MPMD, MS), depending on the nature of the application [10]. The main points to respect are partitioning the data and the computations, and minimizing the communications. The domain decomposition may be a natural way to achieve parallelism [2, 7, 11]. But it requires an additional expensive preprocessing step to slice the mesh. Moreover it is not adapted to massively parallel computers, because the number of partitions possibly generated is often limited.

The main optimization on such architectures consists in minimizing the overhead due to the communications. An example is in the use of the message passing library. By switching from PVM to SHMEM[1], speedups have been deeply increased on a Cray T3E (Fig. 2). On the other hand, this optimization has been made to the detriment of
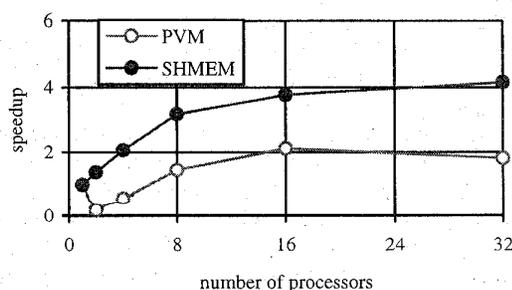

Fig. 2. Cray T3E-speedups with PVM or with SHMEM for message passing.

---

[1] SHMEM is a library developed by Cray. The memory space is physically distributed but virtually shared. The date have then to be allocated at the same address in the memory on every processor.

portability. Note that the PVM performances seem to drop as the number of processors increases from 16 to 32. This is due to the overhead introduced by the parallelism: the number of degrees of freedom of the test problem used for these computations is not large enough to obtain good performances on a high number of processors.

IV. EXAMPLE: COMPARISON BETWEEN THE C98 AND THE CRAY T3E.

Both computers are up-to-date. The Cray C98 is a parallel shared memory computer with vector processors while the Cray T3E is a massively parallel distributed memory computer with scalar processors. For the same application of electromagnetic scattering, algorithms, performances and CPU times are strongly dependent on the architecture. They are compared in the next sections.

A. The Electromagnetic Scattering Code

The *wave3d* program is used to model unbounded frequency domain problems such as microwave or electromagnetic scattering [12]. The Finite Element formulation is directly written in terms of the vector field. Because nodal-based finite elements are used, a penalty term is added to the formulation in order to avoid spurious reflections. The open boundary is modeled using a 3D second order vector Engquist-Majda absorbing boundary condition (A.B.C.). The Galerkin form of the global formulation for scattering problems is the following:

$$-\int_V \left[\frac{1}{\varepsilon_r} (\nabla N \times \nabla \times H) + k_0^2 \mu_r N \ H \right] dv + \int_V (\nabla N)(\nabla \cdot H) dv +$$

$$\int_S N \ T(H) ds - \int_S N n \nabla \cdot H ds = \int_S N \left[T(H^i) - n \times \nabla \times H^i \right] ds \qquad (1)$$

with the A.B.C.: $T(H) = jkH_t + \frac{j}{2k} \nabla_t^2 H_t,$

and H^i: incident field

This formulation leads to 3 complex unknowns per node. A good accuracy in the results is obtained with 10 nodes per wavelength and a distance between the device and the outer boundary equal to 1 wavelength. This high number of degrees of freedom justifies the use of parallel computers.

Boundary conditions on conductors and on symmetry planes are introduced by globally modifying the FE matrix. This one is approximately symmetrized by addition with its transposed. Because it is sparse, the matrix system is solved using the conjugate gradient algorithm.

B. Comparison Between Parallel Algorithm

Table I summarizes the algorithms used on both computers. These algorithms have been shown to give the best performances on each computer.

TABLE I
COMPARISON BETWEEN ALGORITHMS ON C98 AND T3E

	Cray C98	Cray T3E
matrix representation	sparse redundant	sparse regular
assembling	per element	per degree of freedom
preconditioning	incomplete Cholesky	block incomplete Cholesky
solving	conjugate gradient	conjugate gradient

On the Cray C98 [13], a redundant sparse row-wise representation is used to store the matrix. The access to the non-zero terms of a column are adjacent in memory, allowing good vector performances. On the other hand, this type of storage requires twice as much memory space as a regular sparse storage [9]. The assembling of the FE matrix is performed by elementary contributions. For large problems, the incomplete Cholesky preconditioning method is more efficient, due to the reduction of the number of iterations.

On the other hand, the assembling is performed by degree of freedom on the Cray T3E [14]. It allows an optimal speedup for this step to be obtained because no message passing is required between the processors. A block incomplete Cholesky preconditioning together with the conjugate gradient is used to solve the matrix system [15, 16]. Compared to a diagonal preconditioning, this method reduces the number of iterations by generating a good preconditioning. Compared to the usual incomplete Cholesky preconditioning, no message passing is required and the overhead introduced by the parallelism is negligible when the number of degrees of freedom is large. The speed up is then considerably better that the optimal linear gain [15].

C. Comparison Between Parallel Performances

Parallel performances are first analyzed on a 60000 degrees of freedom matrix (Fig. 3). This is the largest problem which can be solved on only one processor of the Cray T3E. This is essential to obtain the actual speedup. On the other hand, this size of problem is insufficiently large to obtain good performances on a large number of processors, because of the cost due to the overhead during the conjugate gradient iterations. This explains the breakpoint in the speedup at the 4 processor level (Fig. 4). The breakpoint is of course strongly dependent on the size of the problem size. Note that the parallel performances on the Cray T3E are
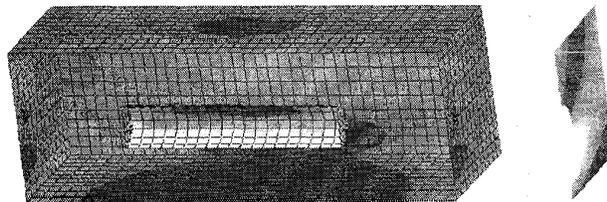

Fig. 3. 60000 degrees of freedom problem - scattering by a perfect electric conducting cylinder.
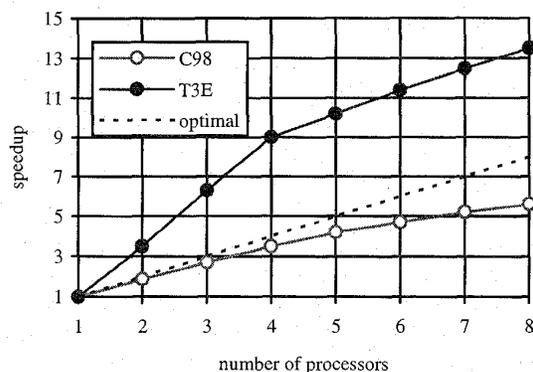
Fig. 4. Speedups for a 60000 degrees of freedom problem.

expressed in terms of superlinear speedup: the speedup is greater than the number of processors used. Indeed the block incomplete Cholesky preconditioning leads to a weakening of the convergence rate. But the number of preconditioning operations decreases when the number of processors increases, allowing better performances.

The difference between the CPU times on both C98 and T3E (Table II) is due to the different monoprocessor performances (average performances: 50 Mflops for the T3E and 375 Mflops for the C98). On the other hand the CPU time used on the C98 increases more slowly than on the T3E when the number of degrees of freedom increases. First, is in the increasing of the vector performances on the C98. Second, the overhead becomes more penalizing on the T3E because the size and the number of messages increase.

CONCLUSION

We have presented in this paper our experience in parallel computational electromagnetics. The implementation of a code on a Cray C98 is easy and the algorithms do not need to be rewritten. However, a thought intervention of the programmer is necessary to add manually compiler directives. Vector performances have also to be watched: a good compromise between vectorization and parallelization has to be found. On the other hand, the implementation on a Cray T3E is more difficult. Domain decomposition techniques may be used for existing codes, keeping the overall modifications made to the codes to a minimum. But

TABLE II
COMPARISON OF THE TOTAL SOLVING TIMES ON C98 AND T3E. COMPUTATIONS ARE
PERFORMED ON 8 PROCESSORS WITH ALGORITHMS PRESENTED IN TABLE I.

degrees of freedom	Cray C98		Cray T3E	
	CPU time (s)	iterations	CPU time (s)	iterations
45000	476	415	4643	426
120000	6650	10701	327531	13439
210000	15101	16586	751237	21829

this strategy is not really adapted to massively parallel computation. Better performances may be obtained by rewriting completely the code.

Like in many other disciplines, computational electromagnetics requires a computing power considerably higher than that offered by today's conventional supercomputers. Massively parallel computing is destined to play a great role in the next scientific developments. The use of such computers requires deep modifications of the sequential algorithms, so future developments will have to take this fact into account.

REFERENCES

[1] R.K. Agarwal, "Parallel computers and large problems in industry," Computational Methods in Applied Sciences, Elsevier Science Pub., pp. 1-11, 1992.
[2] K. Iwano, V. Cingoski, K. Keneda, H. Yamashita, "A parallel processing method in FE analysis using domain division," IEEE Trans. on Mag., vol. 30, no. 5, pp. 3598-3601, September 1994.
[3] Y.S. Choi-Grogan, K. Eswar, P. Sadayappan, R. Lee, " Sequential and parallel of the partitioning finite element method," IEEE Trans. on Ant. and Prop., vol. 44, no. 12, pp. 1609-1616, December 1996.
[4] T. Horie, H. Kuramae, T. Niyo, " Parallel electromagnetic-mechanical coupled analysis using combined domain decomposition method," IEEE Trans. on Mag., vol. 33, no. 2, pp. 1792-1795, March 1997.
[5] G. Mäder, F.H. Uhlmann, " A parallel implementation of a 3D-BEM-algorithm using distributed memory algorithms," IEEE Trans. on Mag., vol. 33, no. 2, pp. 1796-1799, March 1997.
[6] S.R.H. Hoole, K. Agarwal, " Optimization algorithms for magnetics and their parallelizability," IEEE Trans. on Mag., vol. 33, no. 2, pp. 1966-1969, March 1997.
[7] M. Dracopoulos, K. Parrot, G. Molinari, M. Nervi, J. Simkin, " Results of parallelisation of existing electromagnetic finite element codes on some different parallel architectures," Applied Computational Electromagnetics Society Journal, vol. 12, no. 2, pp. 107-112, 1997.
[8] S. Obayashi, A. Oyama, "Three-Dimensional Aerodynamic Optimization with Genetic Algorithm," Computational Fluid Dynamics '96, John Wiley & Sons Ltd., pp. 420-424, 1996.
[9] H. Magnin, J.L. Coulomb, R. Perrin-Bit, " Parallel and vectorial solving of finite element problems on a shared-memory multiprocessor," IEEE Trans. on Mag., vol. 28, no. 2, pp. 1712-1715, March 1992.
[10] K.M. Decker, R.M. Rehmann, Programming Environments for Massively Parallel Distributed Systems, Birkhaüser Verlag Basel, 1994.
[11] D. Zoist, "Parallel processing techniques for FE Analysis: Sstiffnesses, loads and stresses evaluation," Computers & Structures, vol. 28, no. 2, pp. 247-260, 1988.
[12] L. Nicolas, K. A. Connor, S. J. Salon, B. G. Ruth, and L. F. Libelo, "Three dimensional FE analysis of high power microwave devices," IEEE Trans. Mag., no. 2, pp. 1642-1645, March 1993.
[13] C. Vollaire, L. Nicolas, "Implementation of a finite element and absorbing boundary conditions package on a parallel shared memory computer," Dig. of COMPUMAG 97, 1997.
[14] C. Vollaire, L. Nicolas, A. Nicolas, "Finite element and absorbing boundary conditions for scattering problems on a parallel distributed memory computer," IEEE Trans. on Mag., vol. 33, no. 2, pp. 1448-1451, March 1997.
[15] C. Vollaire, L. Nicolas, "Parallel iterative solvers for large sparse system of equations on a distributed memory computer," Dig. of COMPUMAG 97, 1997.
[16] M.L. Barton, "Three-dimensional magnetic field computation on a distributed memory parallel processor," IEEE Trans. on Mag., vol. 26, no. 2, pp. 834-836, March 1990.