Numerical performance of the distributed vector finite-element time-domain algorithm

Boguslaw Butrylo, Christian Vollaire, Laurent Nicolas, Alain Nicolas

Numerical Performance of the Distributed Vector Finite-Element Time-Domain Algorithm

Boguslaw Butrylo, *Member, IEEE*, Christian Vollaire, *Member, IEEE*, Laurent Nicolas, and Alain Nicolas, *Member, IEEE*

***Abstract*—This paper deals with a distributed time-domain modeling of electromagnetic phenomena with the finite-element method. The model is approximated by edge elements. The constitutive equations and method of parallelization of the algorithm are presented. The properties of the distributed finite-element time-domain algorithm are discussed. Some typical performance metrics are studied for the parallel versions of the software. The presented algorithm is executed on a heterogeneous and a homogeneous clusters of workstations. Two different distributed memory environments (MPI and PVM) are used to evaluate the efficiency of the algorithm.**

***Index Terms*—Distributed computing, edge elements, finite-element time-domain (FETD) algorithm, high-frequency electromagnetic field.**

I. Introduction

In THE LAST years the various types of edge elements are widely used in computational electromagnetics. Flexibility of the finite-element technique and right physical sense of the edge elements make this formulation useful in modeling of electromagnetic phenomena. High spatial and temporal resolutions of the analyzed electromagnetic problem require high-performance computing resources.

The distributed implementation of the finite-element time-domain (FETD) algorithm enables to overcome some limitations of a sequential version of this method [1]. High-performance simulation of the time-domain problem enables to reduce either memory cost or time of computation but, as usual, some weaknesses of the distributed implementation are revealed [2]. In this paper, the properties of the vector FETD are evaluated in the known distributed multicomputer environments.

II. Constitutive Equations

The distribution of electric field $\mathbf{e}$ in the analyzed model is stated by time-dependent vector wave equation

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{e} + \sigma \frac{\partial \mathbf{e}}{\partial t} + \varepsilon \frac{\partial^2 \mathbf{e}}{\partial t^2} = 0 \tag{1}$$

Manuscript received July 1, 2003. This work was supported in part by the Development and Scientific Found of Region Rhone-Alpes, France, under Grant 96106550.

B. Butrylo is with the Bialystok Technical University, Department of Theoretical Electrotechnics and Metrology, Bialystok 15-950, Poland (e-mail: Butrylo@we.pb.bialystok.pl).

C. Vollaire, L. Nicolas, and A. Nicolas are with the Centre de Genie Electrique de Lyon (CEGELY) UMR CNRS 5005, Ecole Centrale de Lyon, Ecully 69134, France (e-mail: Christian.Vollaire@ec-lyon.fr; Laurent.Nicolas@ec-lyon.fr).

Digital Object Identifier 10.1109/TMAG.2004.825427

where μ, σ, and ε are the parameters of the linear and isotropic media. In the case of excitation by a plane wave, according to general Galerkin scheme, the weak form of (1) is given by

$$\int_V \frac{1}{\mu}(\nabla \times \mathbf{e})(\nabla \times \mathbf{W}) dV + \int_V \mathbf{W}\sigma \frac{\partial \mathbf{e}}{\partial t} dV + \int_V \mathbf{W}\varepsilon \frac{\partial^2 \mathbf{e}}{\partial t^2} + \int_{S_{ABC}} \mathbf{W} \frac{1}{\mu c} \left(\frac{\partial \mathbf{e}}{\partial t} \times \vec{\mathbf{n}} \right) dS = \int_{S_{ABC}} \mathbf{W} \left[\frac{1}{\mu c} \left(\frac{\partial \mathbf{e}^i}{\partial t} \times \vec{\mathbf{n}} \right) - \vec{\mathbf{n}} \times \nabla \times \mathbf{e}^i \right] \tag{2}$$

where $\mathbf{W}$ is the test vector function, S_{ABC} is the external surface of the model, and $\mathbf{e}^i$ is the incident field. Since the unbounded domain of computation must be limited, the domain of analysis is truncated, and the first-order Engquist–Majda absorbing boundary condition is assumed on the external surface S_{ABC} [3].

This equation is discretized in time domain and in space domain to yield a system of linear equations which must be solved. Space discretization is achieved using incomplete first-order tetrahedral edge elements $H(\mathrm{curl}, \Omega)$ [4]. Considering the central Euler difference approximations of the first- and the second-order derivatives, the final form of the equation is

$$\left(\mathbf{T} + \frac{\Delta t}{2} \mathbf{R} \right) \cdot \mathbf{e}_{n+1} = (2\mathbf{T} - \Delta t^2 \mathbf{S}) \cdot \mathbf{e}_n + \left(\frac{\Delta t}{2} \mathbf{R} - \mathbf{T} \right) \cdot \mathbf{e}_{n-1} - \Delta t^2 \mathbf{f}_n \tag{3}$$

where $\mathbf{T}$, $\mathbf{R}$, and $\mathbf{S}$ are mass, damping, and stiffness matrices, respectively [5]. These matrices are sparse, symmetric, and nondiagonal, therefore, an implicit solver must be used to calculate the time-dependent distribution of electric field $\mathbf{e}_{n+1}$. The $\mathbf{f}_n$ vector represents the dynamic load in the analyzed model and Δt is the assumed time step in the time integration scheme.

III. Distributed Implementation

The distributed version of the presented algorithm is based on the classical domain decomposition paradigm. Because the unknowns are connected with the edges, the set of edges is decomposed. There are no geometrical restrictions of the decomposition, because the $\mathbf{T}$, $\mathbf{R}$, and $\mathbf{S}$ matrices are assembled by degrees-of-freedom (DOF) [6], [7].

The elaborated FETD algorithm works in single process multiple data (SPMD) mode, whereas the configuration of the distributed memory environment is stable [8].

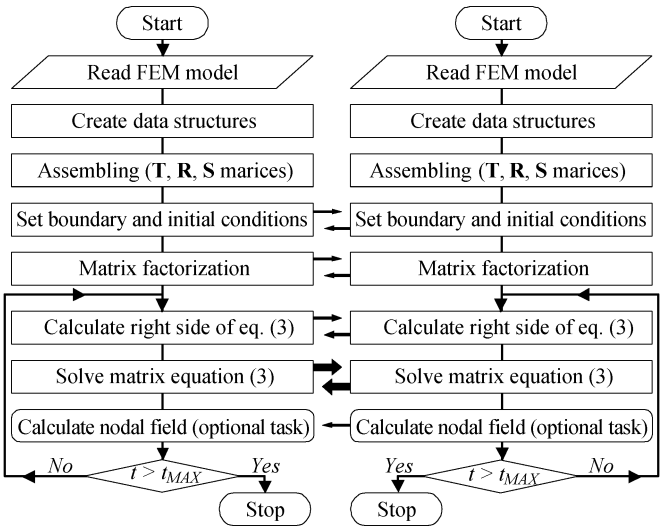

Fig. 1. Flow chart of the distributed FETD edge-based algorithm (two concurrent processes, t_{MAX} is a maximal time of simulation).

The FETD algorithm involves many types of computing tasks, ranging from two- and three-dimensional (2-D, 3-D) assembling, to extensive matrix operations. These computationally intensive operations are performed in two main stages: assembling of the matrix equation and time integration loop (Fig. 1).

The first stage starts from an input operation, and then the nodal form of the model is translated into an edge formulation. The entire edge description of the geometry is performed independently by each of computing units, because this task is not computationally expensive. Finally, the edge-based form of the analyzed model is decomposed and saved in the processing units. Only the data required on each processor are stored in memory.

Local, distributed assembling of $\mathbf{T}$, $\mathbf{R}$, and $\mathbf{S}$ matrices is performed in the next step. Each computing unit assembles the matrices only for its local subset of edges. The task allocation obtained in this stage is almost ideal. Any processing unit does not need to access data pertaining to the other processor. Only at the end of this stage, the information about boundary conditions (BC) is exchanged between workstations. As the result of the distributed assembling, the matrix (3) is distributed by rows among processing units of a cluster.

Traditional FETD scheme involves the repeated solution of system of sparse linear equations. According to Fig. 1, at least three different tasks can be distinguished in the time integration loop.

First, each processor assembles its local part of the right-hand side of matrix equation (incident wave or source antenna). These partial source terms are concatenated in SPMD mode. Then, the preconditioned conjugate gradient (PCG) is used to solve the matrix (3). Since the matrix system is well preconditioned (ten iteration per time step), diagonal preconditioning is used to avoid messages passing during this stage.

Partial matrix vector multiplications are performed in parallel [7]. Therefore, the basic matrix and vector operations are parallelized in the time integration loop. In order to reduce communication traffic, only nonzero terms of partial vectors are sent for the concatenation (SPMD mode). The communication pattern of the FETD algorithm is highly structured and fully predictable.

The transfer from edge to nodal results is the third step in the time integration loop. It is the most time consuming operation in the time integration loop, because the Gaussian quadratures must be calculated in each time step. This part of the algorithm is efficiently parallelized, but it is not taken into account as a specific form of the postprocessing task.

TABLE I
PROPERTIES OF THE MULTICOMPUTER SYSTEMS

	The A Cluster	The B Cluster
Type of the cluster	heterogeneous	homogeneous
Number of computing units	4×PC	8×PC
Type of processor	1×AMD 1 GHz, 3×AMD 800 MHz	8×Intel Celeron 1.2 GHz
RAM memory of a single PC	512 MB	256 MB
Communication network	Gigabit Ethernet (1Gbps)	Fast Ethernet (100Mbps)
Distributed memory environment	LAM 6.5.6 / MPI-2 or PVM 3.4.2	WMPI 1.5

IV. DISTRIBUTED ENVIRONMENT

Two different types of clusters of workstations are used to evaluate the presented FETD algorithm (Table I). The first cluster (A) is a heterogeneous system and the processing units are connected by the gigabit Ethernet hardware. The second cluster (B) has worse communication infrastructure (fast Ethernet) but it consists of eight equivalent processing units.

Two different distributed memory environments are implemented in the clusters. The interdependent threads communicate through either MPI or PVM message passing environment [8]. The communicational load of the presented FETD algorithm is proportional to the total number of edges in the finite-element method (FEM) model, and it depends on the number of processing units in the cluster. In the MPI version of the FETD algorithm, the network traffic is minimized by using broadcast communication rather than point-to-point data transfer. In the PVM version, the processing units communicate only in the individual mode because of the low performance of the implemented broadcast function.

V. VALIDATION OF THE ALGORITHM

The elaborated algorithm is scalable, because the number of computing units can be easily matched in the SPMD mode. The real enlargement of the FE model is limited by minimum size of memory of a single computing node, because the set of data is not totally decomposed. Fig. 2 gives the memory repartition for a 342 342 DOF problem, since the total size of local variables is related to the total size of global variables. Global variables are the set of data duplicated on each processor (e.g., some components of the edge-based FE model). Local variables concern data specific to each processor. The local parts of matrices and specific structures for PCG algorithm constitute this set of variables. The structure of the algorithm and form of data structures affect the relation between local and global variables. This relation does not depend on the hardware platform and/or type of distributed environment.

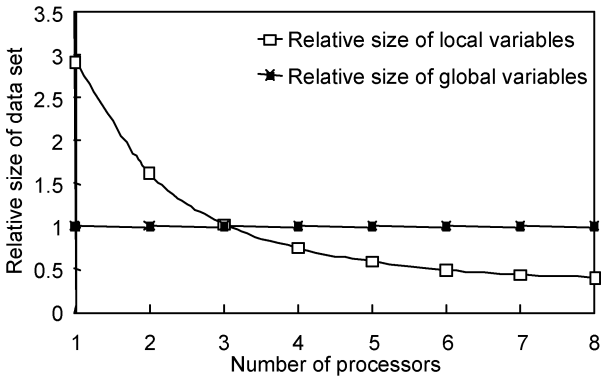

Fig. 2. Relation between local and global variables in the distributed edge-based formulation of the FETD algorithm.

TABLE II
LOAD BALANCING OF THE ALGORITHM

Number of processing unit	Assembling stage		Time integration loop	
	A-cluster	B-cluster	A-cluster	B-cluster
1	100%	100%	100%	100%
2	106%	112%	100%	100%
3	110%	121%	100%	100%
4	117%	132%	100%	100%
5	-	133%	-	100%
6	-	138%	-	100%
7	-	116%	-	100%
8	-	128%	-	100%

The benchmark problems [a plane wave propagating in free space, and diffraction of electromagnetic plane wave on a perfect electric conducting (PEC) body] are calculated to determine the performance of the presented algorithm. The input and output subtasks are not taken into account in this analysis because the elapsed time of I/O operations depends principally on the properties of hard disks.

The interdependence of the processes is different in the assembling stage and in the time integration loop. The assembling thread in the processing unit is loosely connected with the others concurrent threads. The subtasks in the distributed implementation of the PCG algorithm are tightly coupled. In this case, the workstations are loaded uniformly (Table II).

The nominal load balancing is ideal in the time integration loop, but it does not mean that the absolute time of computation is reduced. The maximum computation time of the worst workstation approximates the computation time for all of the processing units. When one processing unit is slow or highly loaded, the others processing units have to wait for a part of data from this unit.

The speedup of MPI implementation evolves with the size of the assembled FE model (Fig. 3). The interprocessor communication plays a significant role for small models. Once the mesh size increases, the computation becomes dominant. For the largest FE models, the speedups of MPI and PVM implementations are approximately equal. They increase linearly with respect to number of processing units, however, it is less than the ideal one (Figs. 4 and 5). This stage is highly parallelized, but inherently sequential nature of some operations and mutual data transfers of BC data slow the overall speedup down. The assembling and matrix factorization subroutines are the most time consuming operations in the first stage of the FETD algorithm.

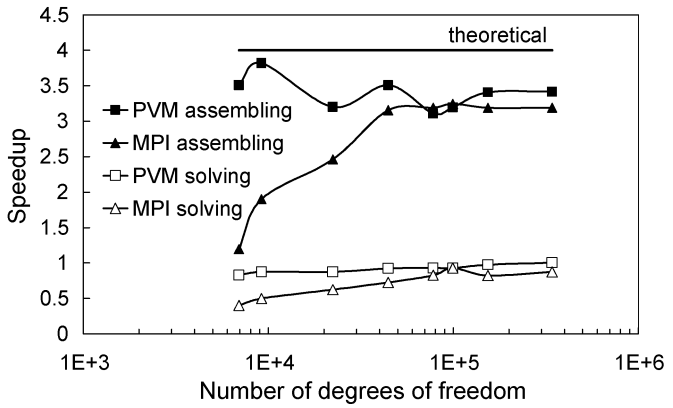

Fig. 3. Speedup of the assembling stage as a function of DOF (cluster A—four processors).

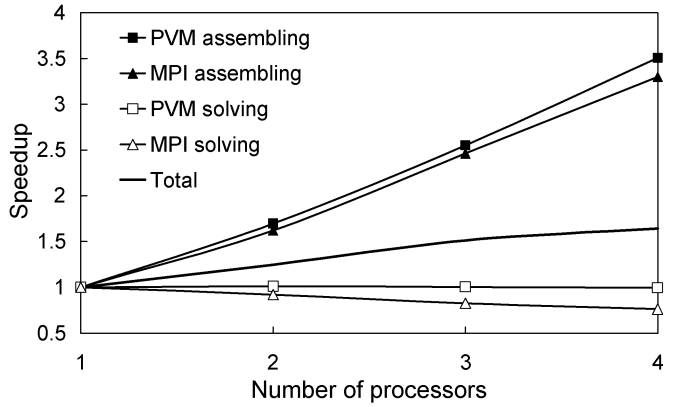

Fig. 4. Speedup of the assembling stage, one time step of the solving stage and for the total computation (cluster A—342 342 DOF problem).

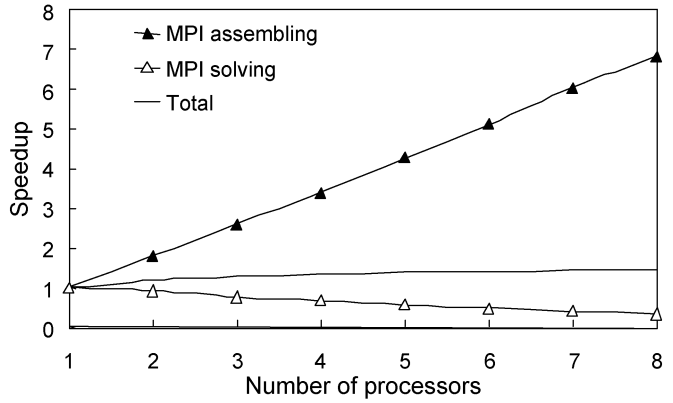

Fig. 5. Speedup of the assembling stage, one time step of the solving stage and for the total computation (cluster B—268 380 DOF).

The time integration loop is more computationally demanding. The crucial part of this stage is distributed version of the iterative solver subroutine. However, the speedup of this part is not good, and it slows down the computation stage. The elapsed time of the distributed solver is worse than the time of a sequential implementation (Figs. 3–5). The diagonal preconditioner is perfectly parallelized since distributed subtasks are executed independently. Therefore, the total performance of the

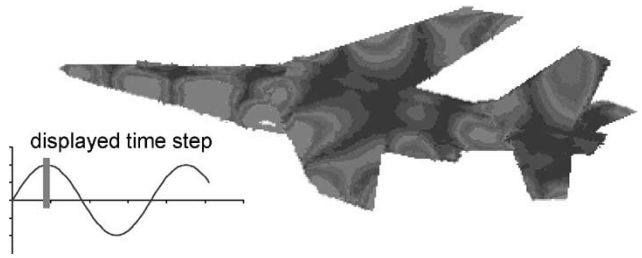

Fig. 6. Scattering by a PEC airplane in time domain (574 151 DOF).

distributed PCG solver is shaped by the matrix operations. The global matrix-vector product is a linear combination of locally calculated subvectors. These partial vectors must be transferred between computing units.

On the other hand, the speedup of the solver stage (one time step: source term assembling and PCG) is below 1 for any FE model. The implemented parallel PCG algorithm with distributed preconditioner degenerates into a form with a poorer performance than the sequential algorithm. This effect arises from nonoverlapped, intensive communication and interdependencies between distributed threads of the PCG subroutine. The distributed PCG solver can cope with extremely large FE-edge element model, and the maximum size of the model is limited only by the parameters of the hardware. The overall speedup of the implemented PCG algorithm does not depend on the number of DOF. It remains constant for a wide range of the FE models, and due to the communication load it decreases when the number of processing units is enlarged.

The total speedup of the FETD algorithm is given as

$$\text{Speedup} = \frac{\left(t_A + \sum_{n=1}^{N} t_{S,n}\right)_{NP}}{\left(t_A + \sum_{n=1}^{N} t_{S,n}\right)_{NP=1}} \tag{4}$$

where t_A is the time of assembling, $t_{S,n}$ the run time of a single time step, N the number of time steps, and NP is the number of computers in the cluster. According to the Amdahl's law, the speedup is limited by relation between sequential and parallel parts of the algorithm [8]. There is no clear rule to fix the speedup of the FETD algorithm, since the nature of the assembling stage differs from the solver. The upper limit of the speedup of the FETD algorithm is described by the performance of the assembling stage (i.e., it directly depends on the number of processing units), whereas the lower limit is equal to the speedup of the solver. In that case, the total speedup is a function of the time step Δt and the number of time steps N. It is greater than one in the presented benchmark problem (assembling and 300 time steps, Figs. 4 and 5).

The heart and bottleneck of the concurrent FETD algorithm is the distributed solver for large sparse matrix equation. The bandwidth of the communication network in the cluster essentially determines the speedup of the solver and FETD algorithm. However, the scalability and global speedup of the parallel algorithm are satisfactory. In this way, realistic problems can be solved (Fig. 6).

VI. Conclusion

Three-dimensional time-dependent simulation of electromagnetic phenomena requires a large amount of memory and processing time to find the solution of a realistic model. The elaborated distributed version of FETD algorithm enables to perform computationally and memory expensive simulations.

The presented distributed implementation of the FETD algorithm is a demanding application. The extensive parallelism is applied in assembling stage and time integration loop. The parallelization of the FETD algorithm improves performance of numerical simulation.

The run time of the algorithm is reduced considerably in the stage of matrix assembling. The overall speedups of the MPI and the PVM implementations in the assembling stage are very close. The presented results indicate slightly better numerical performance of PVM version.

The coherent form of the FEM model is constructed by the data transfer, since the data set is decomposed. The efficient data transfer is a critical issue in the distributed implementation of the FETD method. Communication latency limits the performance of the PCG solver and the total speedup of the time integration loop.

References

[1] J.-F. Lee, R. Lee, and A. Cangellaris, "Time-domain finite-element methods," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 430–442, Mar. 1997.

[2] U. Navsariwala and S. Gedney, "An unconditionally stable parallel finite element time domain algorithm," in *Proc. IEEE APS/URSI Symp.*, Baltimore, MD, July 1996, pp. 112–115.

[3] B. Engquist and A. Majda, "Absorbing boundary conditions for numerical simulation of waves," *Math. Comput.*, vol. 31, no. 139, pp. 629–651, 1977.

[4] A. Bossavit, "A rationale for edge-elements in 3-D fields computations," *IEEE Trans. Magn.*, vol. 24, pp. 74–79, Jan. 1988.

[5] N. M. Newmark, "A method of computation for structural dynamics," *J. Eng. Mechanics Div., ASCE*, vol. 85, pp. 67–94, July 1959.

[6] C. Vollaire, L. Nicolas, and A. Nicolas, "Parallel computing for the finite element method," *Eur. Physical J. Appl. Phys.*, vol. 1, pp. 305–314, 1998.

[7] C. Vollaire and L. Nicolas, "Preconditioning techniques for the conjugate gradient solver on a parallel distributed memory computer," *IEEE Trans. Magn.*, vol. 34, pp. 3347–3350, Sept. 1998.

[8] R. Buyya, *High Performance Cluster Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1999, vol. 2.