DCABES and ICPACE Joint Conference
On
Distributed Algorithms for Science and Engineering

AN OBJECT-ORIENTED DISTRIBUTED COMPUTING SOLVER IN JAVA FOR 3D COMPUTATIONAL ELECTROMAGNETISM APPLICATIONS

Laurent Baduel, Françoise Baude, Denis Caromel, Fabrice Huet and Stéphane Lanteri

INRIA, 2004 Route des Lucioles, BP 93
F-06902 Sophia Antipolis Cedex, France
First.Last@sophia.inria.fr

Abstract
Within the trend of object-based distributed computing, we present the design and implementation of a distributed, object oriented, numerical simulation tool, called Jem3D, for 3D electromagnetic wave propagation problems. JEM3D is fully programmed in Java and its distributed version relies on an active object pattern.

1. INTRODUCTION

The availability of powerful computers and high-speed network technologies as low-cost commodity components is changing the way we use computers today. These technological opportunities have led to the possibility of using distributed computing platforms as a single, unified resource, leading to what is popularly known as Grid computing. However this emerging grid computing concept also brings additional constraints on the development of large-scale distributed applications such as, heterogeneity (both in terms of CPUs and interconnection networks) and multi-localization. We present here the results of a collaborative effort between computer scientists and applied mathematicians towards the use of modern programming languages and libraries such as Java and *ProActive* for the design of a modular and extensible object-oriented version of a finite volume solver designed on tetrahedral meshes for the solution of the 3D Maxwell's equations [2].

2. FINITE VOLUME SOLUTION OF 3D MAXWELL' S EQUATIONS

Electromagnetic wave propagation is modeled by the Maxwell's equations. Recently, Piperno *et al.* [2] have proposed a new finite volume scheme for solving the three-dimensional time domain Maxwell's equations on irregular meshes. A cell-centered formulation is adopted meaning that the discrete unknowns are the average over a tetrahedron of the components of the electric and magnetic fields. This scheme combines the use of a centered numerical flux function for the calculation of the flux balance at a control volume boundary, with an explicit leap-frog scheme for time integration of the semi-discrete equations. It is proven to conserve a discrete electromagnetic energy which is a quadratic form of the unknowns (the electric and magnetic vectors) subjected to a CFL-like condition thus yielding a stability criterion for the overall scheme. In practice, this scheme has been implemented using unstructured tetrahedral meshes; however, it is potentially applicable to general, hybrid meshes, combining hexahedral, prismatic and tetrahedral elements. Higher-order formulations can be designed as well [4]. This variety of situations naturally motivates the specification of a general object-oriented framework that would facilitate the development of various numerical simulation tools. We have thus designed such an object-oriented model whose main features are: (1) the ability to deal with 2D and 3D problems, (2) the possibility of using different types of mesh elements (triangle, quadrangle, tetrahedron and hexahedron) and, (3) the inclusion of the two main classes of finite volume methods i.e. vertex-centered and element-centered formulations. In this model, several hierarchy of classes are used for the definition of geometric entities such as a tetrahedron, a control volume and a facet (a control volume is delimited by several facets), that are relevant to finite volume solvers. The basic numerical kernel of a finite volume solver such as the one described in [2] consists in the computation of a numerical flux between two control volumes sharing a facet. This model as been used to develop a sequential, object-oriented, version of an existing FORTRAN 77 code, called JEM3D, implementing the finite volume method introduced in [2]. The programming of JEM3D fully relies on Java.

3. PARALLEL AND DISTRIBUTED FINITE VOLUME SOLVER

This section briefly explains how, using the *ProActive* library, we have programmed a parallel and distributed version of Jem3D.

ProActive is an Open Source, high-level environment for the Grid, featuring object-oriented group communications, and interactive XML deployments [5]. As *ProActive* is built on top of the Java standard API, it does not require any modification to the standard Java execution environment, nor does it make use of a special compiler, pre-processor or modified virtual machine. A distributed or concurrent application built using *ProActive* is composed of a number of medium-grained entities called *active objects*. Each active object has one distinguished element, the *root*, which is the only entry point to the active object, its own thread of control and is granted the ability to decide in which order to serve the incoming method calls that are automatically stored in a queue of pending requests. Method calls sent to active objects are always asynchronous with transparent *future objects* and synchronization is handled by a mechanism known *as wait-by-necessity* [6]. The *ProActive* library provides a way to migrate any active object from any JVM to any other one through a primitive which can either be called from the object itself or from another active object through a public method call. *ProActive* also offers a group communication mechanism that achieves asynchronous remote method invocation for a group of remote objects, with automatic gathering of replies. We refer to [5]-[7]-[8] for more details on *ProActive*.

The underlying idea for the parallelization of JEM3D is to apply a standard and natural geometric decomposition of the 3D computational domain into sub-domains. As such, some facets will contribute to control volumes that may be located onto neighboring sub-domains. We introduce the VirtualBoderFacets (VBF) to represent those facets that belong to two sub-domains. In a couple of neighboring sub-domains, both have a reference to a VBF designating the shared facet. Each VBF contributes to the computation. Two neighboring VBFs which are copies of the same facet must exchange and combine their physical values (i.e. the components of the electric and magnetic fields) to compute the associated numerical flux. For the update access, it is the sub-domain's responsibility to trigger a remote method call onto the corresponding sub-domain, implemented as an *active object*, which itself sets values in the twin VBF. Thanks to polymorphism and dynamic binding, there is no need to explicitly deal with the effective real types of facets: internal or virtual. As a result, the control volumes that reference virtual border facet, as well as the loop that uses them, can execute unchanged. Our architecture features a totally decentralized approach. The application is fully *peer-to-peer*: each sub-domain communicates with the others without any centralized supervisor. As centralized points are usually bottlenecks due to overload problems, we achieve a better scalability.

5. PERFORMANCE RESULTS

We present here preliminary results that compare the performances of the original FORTRAN 77/MPI version of our finite volume solver (referenced as "EM3D" in the tables below) with those of the distributed object-oriented Java/*ProActive* version (referenced as "JEM3D"). For this purpose, a simple test case has been considered which consists in the simulation of the propagation of an eigenmode of a cubic metallic cavity. Timings are given for 100 iterations of the main time stepping loop using different mesh sizes and numbers of processing nodes. The computing platform is a cluster of dual-nodes PCs (Intel Pentium 4/2 GHz) with Gigabit interconnection. As might be predicted, the JEM3D version is slower by a factor 3 to 4. This is the combined result of several factors: JEM3D exploits object-oriented programming features such as polymorphism which is not at no cost, Java is an interpreted language and *ProActive* is using the standard RMI transport layer. Clearly, there is room for improvements on each of these aspects.

# procs	EM3D	JEM3D	JEM3D/EM3D
6	155 sec	510 sec	3.3
8	120 sec	340 sec	2.8
12	80 sec	270 sec	3.4
16	65 sec	225 sec	3.4

Table 1 - Parallel performance results for EM3D and JEM3D (# tetrahedra 3,072,000)

SUMMARY

We have presented preliminary results of an ongoing collaboration between computer scientists and applied mathematicians towards the use of modern programming languages and libraries such as Java and *ProActive* for the design of grid aware numerical simulation tool of 3D electromagnetic wave propagation problems. In order to take advantage of parallelism and distribution, we use *ProActive*, a library for parallel, concurrent and distributed computing in Java featuring additional characteristics compared to the standard Java RMI API.

As might be predicted, actual benchmarks show slower performances of about a factor 3 to 4 compared to those of a FORTRAN-MPI version, which is undoubtely a good achievement in a 100 % Java environment. More importantly, the aim of this work is to emphasize on the benefits we get on software engineering aspects (possible extension of the Java version, full portability, ease of deployment, etc.) through a complete rewriting of the FORTRAN version. We do not get the performance of executing native code resulting from FORTRAN or C++ programming (see for instance works that wrap MPI-based legacy codes as Java or CORBA components [3]). Even under these conditions, we demonstrate that an object-oriented SPMD programming approach entirely based on point-to-point or collective method invocations in Java, can still yield acceptable performances and parallel speedup.

REFERENCES

[1] M.S. Shields, O.F. Rana and D.W. Walker, "A collaborative code development environment for computational electro-magnetics", IFIP TC2/WG2.5 Working Conference on the Architecture of Scientific Software, Kluwer Academic Publishers, 119-141 (2001).
[2] S. Piperno, M. Remaki and L. Fezoui, "A nondiffusive finite volume scheme for the three-dimensional Maxwell's equations on unstructured meshes", SIAM J. Num. Anal., 39(6), 2089-2108 (2002).
[3] M. Li, O.F. Rana and D.W. Walker, "Wrapping MPI-based legacy codes as Java/CORBA components", Future Generation Computer Systems, 18(2), 213-223 (2001).
[4] S. Piperno and L. Fezoui, "A centered discontinuous Galerkin finite volume scheme for the 3D heterogeneous Maxwell equations on unstructured meshes", INRIA Research Report No. 4733 (2003).
[5] F. Baude, D. Caromel, F. Huet, L. Mestre and J. Vayssière, " Efficient, flexible, and typed group communications in Java", 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), IEEE, 93-102 (2002).
[6] D. Caromel, "Towards a method of object-oriented concurrent programming", Communications of the ACM, 36(9), 90-102 (1993).
[7] L. Baduel, F. Baude and D. Caromel, "Efficient, flexible, and typed group communications in Java", Joint ACM Java Grande ISCOPE Conference, ACM Press, 28-36 (2002).
[8] D. Caromel and W. Klauser and J. Vayssière, "Towards seamless computing and metacomputing in Java", Concurrency, Pract. Exp., 10, 1043-1061 (1998).