Object-Oriented parallel CFD with JAVA

Dominique Eyheramendy[*]

Jeune Equipe Modélisation Calcul Scientifique, Bât. ISTIL,
Institut des Sciences et Techniques de l'Ingénieur de Lyon, Université Lyon 1
15 Blvd Latarget, 69100 Villeurbanne, France

Key words: Multi-threading in JAVA, Domain Decomposition, Stabilized Finite Elements for Navier-Stokes.

Introduction

Today the object-oriented technologies are widely used in finite element computational mechanics. Many authors have shown the strength of the approach in different fields of mechanics, including parallel and/or CFD computations: e.g. a study of a transient model of fluid mechanics fully coupled to an electrochemistry model in [PES 96], some object-oriented techniques dedicated to CFD in [MUN 00], a finite element model for modeling heat and mass transfert using the Diffpack library in [SUN 98], a Arbitrary Lagragian-Eulerian stabilized formulation for hydrodynamics with shock capturing techniques in [KER 98], a use of the PVM library to parallelize explicit computations in structural dynamics in [KRY 98], a general framework for managing parallel simulations based on domain decomposition methods in [CHA 01], etc… Following a similar way, the author has developed concepts to fasten the design of finite element formulations and corresponding numerical codes by the way of symbolic concepts. All these works are entirely or partially implemented in C++, which is today commonly adopted by the engineer community. Using classical object-oriented approaches, e.g. using the C++ language, the code losses an important feature: its portability. The strength of the Java language over the traditional language lies in the platform independence. In [GIN 00], the problem of the utilization of Java for numerical computation in the industrial real life problems is raised up, and no definitive response is brought probably because of lack of experiments in the domain. One aim of the present work is to give an example of large scale development in finite elements for engineering application. As parallel programming is significantly more complex than sequential programming, the idea of this work is to develop a pure JAVA framework for finite elements parallel computations.

In this paper, we would like to describe some aspects of an application of Java environment to domain decomposition in CFD. Our main concern in this presentation is to show some pure performance comparison tests between JAVA and C, and to show two applications of multi-threading programming in JAVA. After a brief description of the basic features useful in JAVA, we introduce a simple way of multi-threading matrix/vector product in Java for solving large linear systems by the way of an iterative method. At last, we show a tentative development for an accelerated Schwartz overlapping domain decomposition method for the Navier-Stokes problem; we would like to try to show that classical fast development capabilities in the finite elements environment still exist in Java.

JAVA for scientific computing

Roughly speaking, we distinguish the Java programming language from the Java Virtual Machine (JVM). The JVM is an interpreter that executes the program compiled to Java byte-codes. The main consequence is that a program compiled on a system can be run on all systems. This very attractive aspect could hide a major drawback especially in CFD computation: the efficiency. Most computations in mechanics involve a large number of scalar products (elemental contributions computation, Crout reduction in direct linear systems solvers, matrix/vector prod-

[*] E-mail: eyheramendy@cdcsp.univ-lyon1.fr

ucts in iterative linear system solvers). Here, we test exactly the same code (Java has a C syntax, only memory allocation) for computation of matrix/vector products, with and whitout redirection, i.e. code respectively corresponding to *v[j] = A[i][j] * x[j]* and *v[j] = A[i][j] * x[table[j]]*. Results are similar on different platforms (Windows 2000, Linux, Tru 64 Dec-Unix) and shows roughly speaking that Java is from 72% to 85% within the C compiled code with maximal optimization options for direct memory access, and from 65% to 82% with redirections. The best results are obtained for large sized matrices. Our conclusion is that rather good performances rate can be achieved for computational tools in Java, enough efficient to develop a tool to easily design numerical algorithms for large application. One question remains still opened is the capabilities of the JVM to manage a large amount of memory.

Multi-threaded matrix/vector product in JAVA

In Java, multithreading programming is embedded into the language. The key point is the class Thread. The question is then to check the performance of this class. The test done here is to parallelize matrix/vector product in an iterative linear system solver, a conjugate residual method. For each matrix/vector product, the scalar products are separated in n groups, each group of scalar products is taken by a thread. The finite elements problem solved is a Stokes problem, using a classical Galerkin formulation. The interpolation is bilinear for velocity and constant for pressure. The cavity flow problem is solved up to 50000 degrees of freedom (Dof). The tests are conducted on a 4 processors Compaq machine. Beyond 1000 Dofs the results are stable and show a speedup of 60% for 2 threads, 50% for 3 and 4 threads (4 processors on the machine), and then slowly decreases for higher number of threads. These results are similar to the ones that could be obtained in the same context with the PVM library. This experience has shown the efficiency of threads programming in Java. Moreover we have remark that the time spent in threads managing can be neglected compared to the computations time.

Multi-threaded multiplicative Schwartz overlapping method for a stabilized Navier-Stokes formulation

After a simple application of threads programming in Java, we illustrate now the fast development capabilities of our Java FE environment for parallel computations and to evaluate the behavior of the Java VM. Multi-threading is applied to a Schwartz overlapping method. Each thread takes in charge the computation over a subdomain at each Schwartz iteration. We study a classical Galerkin Navier-Stokes formulation stabilized by adding least-squares type terms similar to the ones presented in [HUG 89] and [TEZ 92]. Linearization is introduced either in the Schwartz iterations scheme or in a Newton like scheme in the Schwartz scheme. A direct linear system solver based on a Crout decomposition is used to solve the linear system at each iteration. In order to decrease the number of iterations with a minimum overlap, we tested various Aitken type accelerations schemes (see [GAR 01]). The cavity flow problem is solved up to Reynolds 1000 over 16 subdomains on a four processors Compaq machine (4 EV6 processors – 500MHz). This experiment allows us to draw two main conclusions about the FE computation environment:

- As depicted in the past years in [DUB 93] [EYH 00] [EYH 01], a pure object-oriented approach can be applied to multi-threading programming of finite elements. The programming effort needed to parallelize a code is much cheaper to the one needed for an equivalent environment in C++ for exemple.
- At each iteration, a linear system is allocated, formed and solved. With a large number of iterations, we have remarked that the Java Virtual Machine manages efficiently memory, i.e. time needed for one iteration remains constant. Automatic memory management scheme, i.e. garbage collecting, is efficient enough to conduct computations.

Conclusion

In this paper, we have presented object-oriented techniques to parallelize a finite element CFD computations in Java. At the current point, we have shown the high potential of the Java

Virtual Machine in the design of numerical schemes on a multiprocessor machine in real situation for computational mechanics. In Java, multi-threading programming can be introduced in a natural way in classical finite elements codes. The key point of this study is that the object-orientedness of finite elements code in Java is conserved. The following step is the evaluation of communication scheme in Java in the CFD context. After that, large computations may be envisaged on clusters. Beyond this experiment, today around the world many computer scientists put a large amount of effort to fasten the Java Virtual Machine for parallel computation (e.g. see web sites www.javagrande.org and www.javahpc.org). Two trends are really hopeful from our point of view: the development of single Virtual Machine running on a cluster of machine and the development of communication over the network (RMI, sockets…). In this context, the framework presented in this paper will fully take advantage of these developments, and then enhance drastically the computations in the domain of interest.

References

1. [CHA 01] A. S. Charao, Multiprogrammation parallèle générique des méthodes de décomposition de domaine, PhD thesis report, Institut National Polytechnique de Grenoble, 2001.
2. [DUB 93] Y. Dubois-Pèlerin and Th. Zimmermann, Object-oriented finite element programming : III. An efficient implementation in C++, Comput. Methods Appl. Mech. Engrg., vol. 108 (1993) pp. 165-183.
3. [EYH 00] D. Eyheramendy, An object-oriented hybrid Symbolic/Numerical Approach for the Development of Finite Element Codes, Finite Element Analysis and Design, vol. 36 (2000) pp. 315-334.
4. [EYH 01] D. Eyheramendy and Th. Zimmermann, Object-oriented finite elements : IV. Application of symbolic derivations and automatic programming to nonlinear formulations, Computer Methods in Applied Mechanics and Engineering, vol. 190 n° 22-23 (2001) pp. 2729-2751.
5. [GAR 01] M. Garbey and D.Tromeur-Dervout, Two level domain decomposition for multiclusters, In T. Chan and all editors, Domain Decomposition in Sciences and Engineering, pp. 325-339, 2001.
6. [GIN 00] M. Ginsberg, J. Hauser, J. E. Moreira, R. Morgan, J. C. Parsons and T. J. Wielenga, Panel session: future directions and challenges for Java implementations of numeric-intensive industrial applications, Advances in Engineering Software, Vol. 31,2000, pp. 743-751.
7. [HUG 89] T.J.R. Hughes, L.P. Franca and M. Balestra, A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/Least-squares method for advective-diffusive equations, Comput. Methods Appl. Mech. Engrg., vol. 73 (1989) pp. 173-189.
8. [KER 98] D. S. Kershaw, M. K. Prasad, M. J. Shaw and J. L. Milovich, 3D Unstructured mesh ALE hydrodynamics with the upwind discontinuous finite element method, Computer Methods in Applied Mechanics and Engineering, Vol. 158, 1998, pp. 81-116.
9. [MUN 00] O. Munthe and H. P. Langtangen, Finite elements and object-oriented implementation techniques in computational fluid dynamics, Computer Methods in Applied Mechanics and Engineering, Vol. 190, 2000, pp. 865-888
10. [KRY 98] P. Krysl and T. Belytschko, Object-oriented parallelization of explicit structural dynamics with PVM, Computers & Structures, Vol. 66,1998, pp. 259-273
11. [PES 96] A. P. Peskin and G. R. Hardin, An object-oriented approach to general purpose fluid dynamics software, Computers & Chemical Engineering, Vol. 20, 1996, pp. 1043-1058.
12. [SUN 98] S.-H. Sun and T. R. Marrero, An object-oriented programming approach for heat and mass transfer related finite element analyses, Computers & Chemical Engineering, Vol. 22, 1998, pp. 1381-1385
13. [TEZ 92] T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Advances in applied Mechanics, vol. 28 (1992) pp. 1-44.