An Object-Oriented Framework for Parallel Incompressible Flow Simulations

Freddy Perez Ramirez[1]
Raul Perez Acosta[2]
Advisor: Dr. Wilson Rivera[3]

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681

Abstract

The coupling and domain decomposition techniques combined with extensive use of object-oriented programming techniques will result in an efficient, flexible, and systematic process for producing parallel partial differential equations codes. This paper discusses important issues regarding the implementation of an object-oriented framework for simulating incompressible flow using parallel computers.

1. Introduction

Computational fluid dynamics (CFD) is increasingly becoming an integral part of the engineering development process. Consequently, CFD must be capable to satisfy real world requirements in a timely manner. Parallel processing on scalable distributed systems is the current trend to provide fast turn around time for CFD simulations. However, the software design in parallel environments is particularly complicated.

Hammond and Barth [1] were one of the first authors to present a parallel two-dimensional solver, which was implemented on a Connection-Machine CM-2. Drikakis and Schreck [2] presented parallel implicit numerical methods for compressible flows. Pankajakshan and Briley [3], on the other hand, presented a parallel multiblock incompressible flow solver. The common features of these representative works is the addressing of parallel processing at the level of lineal algebra and their implementations using procedural programming languages.

We propose a high-level parallelization of simulation codes through an extensive use of object-oriented programming techniques. The possibility of a modular implementation of mathematical abstractions, which is a direct advantage of object-oriented programming, gives rise to the generalization of computational kernels, which are reusable in many simulation applications from different disciplines of scientific computing. This approach makes it possible to hide computational details when needed as well as produce sequential simulators with unified generic interfaces, which easily allow modification and extension.
This paper, which is an early stage of our research, focuses on the practical issues

[1] Graduate student (freddy_peru@hotmail.com)
[2] Undergraduate student (ptta@yahho.com)
[3] Assistant Professor (wrivera@ece.uprm.edu)

encountered in the analysis and design of a parallel incompressible flow simulator. Next session describes the governing equations of incompressible flows. Session 3 discusses the parallel processing requirements. Finally, session 4 depicts the object-oriented framework.

2. Incompressible Flow

The unsteady incompressible Navier-Stokes equations are widely used for modeling incompressible flow in biomedicine, hydrodynamics and computational oceanography. An artificial time derivative of pressure is added to the continuity equation to cast the complete set of governing equations into a time-marching form. The resulting set of equations in integral form represents a system of conservation laws, which can be written as

$$\frac{\partial}{\partial t}\int_{\Omega} Q dV + \oint_{\partial\Omega} F dA = \frac{1}{\mathrm{Re}}\oint_{\partial\Omega} G dA$$

The algorithm applied for solving the Navier-Stokes equations is an implicit finite volume formulation, second order in time with an approximate Riemann solver based on Roe flux approximation to achieve up to third order spatial accuracy [4]. We will modify a public code referred to as MOUSE [5] to deal with domain decomposition and parallel processing.

3. Parallel Processing

Due to enormous computational requirements of many challenges in computational fluids dynamics, the use of parallel processing is fundamental. Parallel processing has a potential for not only reducing the computational time, but also concentrating memories belonging to different processors to carry out larger calculations. Thus, the migration of sequential partial differential equations (PDE) simulators to multiprocessor platforms is well justified.

In general, the parallelization of flow solvers should satisfy several constraints. First, the accuracy of the overall numerical scheme must no be compromised; meaning the solution computed in parallel must have a one-to-one correspondence with the solution computed in serial mode.

When solving PDEs using non-overlapping domain decomposition methods, one often needs numerical boundary conditions on the boundaries between subdomains. These numerical boundary conditions can significantly affect the stability and accuracy of the final algorithm. A new approach based on explicit predictor and implicit correctors has been proposed [6]. This method, which has demonstrated a significant improvement in accuracy when calculating transient solutions, will be used in our parallel implementation. Second, the consequences of the inevitable domain decomposition should not seriously compromise the convergence rate of the iterative algorithm. It is necessary to investigate the effect of a variety of algorithms on convergence, so that one can choose the best combination of options for a specific problem type. Finally, the implementations should use efficiently computational resources. Scalability metrics will be integrated into our parallel implementation [7].

The problem of numerical boundary conditions is more complicated if intercomponents and multidisciplinary computational simulations are considered. As a consequence, further research is needed to develop numerical coupling algorithms for interchange boundary information between

codes at different levels of complexity. Our long-term goal is to produce a family of multidisciplinary solvers.

The current trend in high performance computing hardware is towards clusters of symmetric multiprocessors (SMP). In principle, we will use OpenMP for the parallel implementation but further research will address the effect of having MPI/OpenMP implementations. In fact, the object-oriented framework will help to this hybrid approach.

4. Object-Oriented Framework

Three hierarchical classes compose the object-oriented framework (see Figure 1). The base class *SubdomainSolver* is a generic representation of any subdomain solver. This class consists mainly of a group of pure virtual member functions implementing different algorithms and formulations for solving subdomain problems. These virtual members need to be overridden in derived subclasses so that the user can either take ready-built class objects or incorporate new methods. The *Communicator* class controls the exchange of information between processors. Subclasses of Communicator implement different methods for dealing with boundary conditions between subdomains and communication protocols. Finally, the *GlobalAdministrator* class governs parallelization features such as partitioning, load balancing, and synchronization as well as a set of user options including numerical algorithms selection and level of accuracy and convergence. At run-time on each processor, instances of the subclasses *PDESolver* and *PDEComm* are created according a set of selected parameters defined in the *RunTime* subclass. These subclasses are the interfaces among the different hierarchies.

With this object-oriented approach we expect to produce, reliable, flexible, and extensible multidisciplinary. The practical issues that we have to deal with in this work include programming abstractions, portability, numerical accuracy, and computational efficiency.

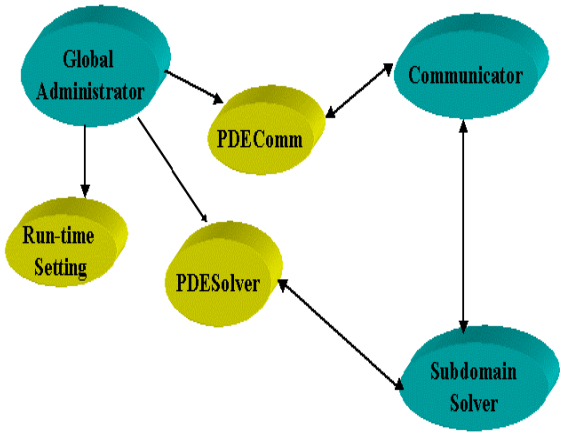

Figure 1: Object-Oriented Framework

References

[1]. S. W. Hammond and T. J. Barth, "Efficient massively parallel Euler solver for two-dimensional unstructured grids." *AIAA Journal*, 30(4): 947-952, 1992.

[2]. D. Drikakis and E. Schreck, "Development of parallel implicit Navier-Stokes solverson MIMD multi-processor systems." *AIAA 93-0062*.

[3]. R. Pankajakshan and W. R. Briley, "Parallel solutions of viscous incompressible flow on multiblock structured grids using MPI." In *Parallel Computational Fluid Dynamics*, pp. 601-608, Elsevier Science, 1996.

[4]. W. R. Briley, S. S. Neerarambam, and D. L. Whitfield, "Implicit lower-upper/approximate factorization schemes for incompressible flow." *Journal of Computational Physics*, 128: 32-42, 1996.

[5]. The Institute of Combustion and Gasdynamic at the University of Duisburg. http://fire8.vug.uni-duisburg.de/Mouse

[6]. W. Rivera, J. Zhu, and D. Huddleston, "An efficient parallel algorithm with application to computational fluid dynamics." To appear in *Computers & Mathematics with Applications*, 2001.

[7]. W. Rivera, "Scalable parallel genetic algorithms." To appear in *Artificial Intelligence Review*, 2001.