Comparison of Hybrid Programming Models for the GMRES Method

Joanna Płażek¹, Tomasz Jurczyk², Jacek Kitowski³

¹ Institute of Teleinformatics, Cracow University of Technology, ul. Warszawska 24, 31-155 Cracow, Poland
plazek@pk.edu.pl

² Institute of Computer Science, AGH-UST, al. Mickiewicza 30, 30-059 Cracow, Poland

³ ACC CYFRONET-AGH, ul. Nawojki 11, 30-950 Cracow, Poland

Abstract. In this paper we compare parallel efficiency for two implementations of a heterogeneous parallel algorithm for finite element compressible fluid flow simulations on unstructured grids. Both incorporate two kinds of programming models: the explicit model (with message-passing paradigm between nodes) and the implicit one (with data-parallel programming inside nodes). Message passing is based on PVM or MPI system while data-parallel is based on OpenMP. The algorithms are dedicated for multiprocessors and for clusters with SMP nodes. Time discretization of the compressible Euler equations is organized with a linear, implicit version of the Taylor-Galerkin time scheme, while finite elements are employed for space discretization of one step problems. The resulting nonsymmetric system of linear equations is solved iteratively with the preconditioned GMRES method for a chosen ramp problem of 2D inviscid flow simulations.

AMS Subject Classification: 65Y05, 65F10, 76M10
Keywords and phrases: shared memory and message-passing programming, finite element method, iterative solvers, PVM, MPI, OpenMP

1 Introduction

Two classical programming models can be adopted for parallel computing: the explicit model (with message-passing paradigm) and the implicit one (with data-parallel programming). Since the advanced multiprocessors and clusters are constructed with SMP nodes the choice between the programming models is not obvious and some integration of multiprocessing and multithreading is profitable [1, 2]. Due to faster communication within the SMP nodes than between the nodes, it would be probably not the best choice to implement distributed memory organization (often called Distributed Shared Memory, DSM) in the entire program. The better way is to adopt this kind of programming within the SMP nodes, while using the explicit programming between the nodes. Such an approach is consistent with present trends in high performance computing, in which the dominant architecture will be clusters of SMPs in its many variants.