FORTRAN 95 / HPF PARALLEL IMPLEMENTATION OF A WEIGHTED-AVERAGE-FLUX BLAST CODE WITH TVD FLUX LIMITERS AND DIMENSIONAL SPLITTING

Dr. Kirk Mathews, Captain Mark Wittig USA
Captain Mark Suriano, USAF
Air Force Institute of Technology, AFIT/ENP, 2950 P Street
Wright-Patterson, AFB, OH 45433-7765, USA

Our philosophy for writing computer programs, and especially parallel programs, is to write readable, sound, maintainable, portable, high-level source code and to let the compiler deal with the tedious details: memory management, array partitioning, interprocessor communications, and architecture- and hardware-dependent optimizations. If a code isn't fast enough, it will be. Time brings faster hardware and smarter compilers that optimize more effectively. Real speedups are achieved by developing better algorithms, not by tweaking existing code.
Sadly, ours is not the usual philosophy. Routinely, the opposite approach is taken. Code is made unreadable by tweaks that are intended to optimize it for a specific hardware and operating system. Critical (or legacy) code segments are written in assembler. Parallelization is achieved using calls to low-level routines from libraries such as PVM or MPI. This gives the programmer complete control at the cost of attending to every detail. The results of such efforts are all too familiar. Antiquated computer hardware and software is retained long after the vendor stopped supporting it because that is perceived to be easier than porting to a new machine. Programmers spend their careers patching old codes and rewriting assembly language routines for new machines. Codes are opaque and unmaintainable, with persistent bugs. The patch that fixes one error introduces more errors. And so on.
In many areas of nuclear engineering, programs originally written in the 1960's, with the marginally-adequate approximations and algorithms (written in FORTRAN IV and assembler) that were all that the machines of the day could do, are still with us. Portions have been rewritten in FORTRAN 77. Pretty user interfaces written in C++ have been added. But the underlying modeling has not changed in thirty or forty years. These codes lack robustness and reliability. The user needs years of apprenticeship to learn the art of knowing when the results are right and when they aren't. In our view, the community needs to start over with new algorithms and write new source code. But funding agencies remember the years and careers spent (with primitive, hostile tools) getting those codes to work at all and conclude that they are good enough and that reliable codes would be too expensive.
Our purpose here is to present some of the features of Fortran 90, together with its extensions Fortran 95 and High Performance Fortran (HPF), that make our philosophy practical, and to do so in the context of state-of-the-art algorithms for hydrodynamic modeling of air blast. With modern development environments and this combined language, which we call Modern Fortran, writing a new code can be less effort and more effective than porting many currently-used codes to new machines.
We have implemented an air blast code that uses Godunov's method with Toro's TVD-limited, weighted-average fluxes and dimension splitting. The code uses many of the techniques described here. It was developed using Compaq Visual Fortran© in the Microsoft Developer Studio© under Microsoft Windows©, and then ported to a Compaq ES-40 parallel computer and parallelized using HPF directives in Compaq Fortran©. Some computational results are presented at the end of this paper.