Experiences with HPF for Scientific Applications *

G. Cabitza, C. Nardone, C. Bagaini, A. Balzano, E. Bonomi, L. Brieger, M. Ennas, E. Garau **, D. Greco, G.Lecca, E. Pieroni, C. Rossi

CRS4, Center for Advanced Studies, Research and Development in Sardinia,
via N.Sauro 10, I–09123 Cagliari, Italy
Email: cmn@crs4.it

Abstract. The data-parallel programming paradigm emerges as the natural choice for a large class of scientific applications. One of its most significant features consists in hiding the details of the communications and synchronizations among processors, leaving them to the compiler. The advent of the new standard data-parallel language, High Performance Fortran (HPF), promises to bring portability of data-parallel codes across different parallel architectures.
A wide spectrum of activities around HPF has been carried out and planned for the future at CRS4, ranging from basic tools to parallel libraries, kernel codes and time-consuming real applications. Examples are the `HPParLib++` project (a run-time system for HPF), codes for seismic migration and modeling, a molecular dynamics code of liquid water and a shallow water transport code for the simulation of lagoon systems. The strategy behind this work is to advance both in basic computer science and in the development of end-user applications in order to benefit from the synergetic exchange of ideas and solutions.

1 Introduction

The current trend in high-performance computing (HPC) favours distributed memory (DM) parallel machines, because of their advantage in terms of scalability and cost over traditional vector supercomputers. Moreover, DM machines are more easily scalable than shared memory (SM) machines when the number of nodes is larger than about ten. Actually, DM machines are more difficult to program than SM multiprocessors or vector supercomputers. In fact, DM computers (or clusters of workstations) are usually programmed using a sequential language for the nodes and a message-passing library (proprietary or portable), using task-parallelism. But they can also be programmed using a high level parallel language such as High Performance Fortran (HPF) [1] using data-parallelism.

The message-passing (MP) programming style is more efficient than the data-parallel (DP) one for an important class of problems (problems with irregular

* Work carried out with the support of the Sardinian Regional Authorities.
** Supported by a grant from CNR, Gruppo Nazionale per l'Informatica Matematica.

distributions – particle-in-cell codes and unstructured mesh solvers for instance), but it is error prone because the user must explicitly manage all data and task distribution and communication between processors. Software maintenance and re-usability are also more difficult to achieve because a message-passing code tends to be difficult to read (for example, it has more than one thread) and the development environment is generally quite poor.

In constrast, a high-level parallel language such as HPF, ensures that the machine can be viewed by the programmer as a single entity with a single thread of computation and a global name space, leading to obvious advantages in terms of "programmability". Such features also allow an easier design of the development environment (debugger, profiler, etc...), contributing to the productivity of the programmer. The user must define the data distribution, but the implementation, including detailed communications and synchronization among processors, is left to the compiler. HPF also provides an interface to "local" external routines which may be written in any language. This feature allows the optimization of hot spots of the code by using specialized MP routines.

A significant fraction of scientific applications consists of problems that are intrinsically DP; codes written in a DP language for such applications, will be efficient on most parallel machines. It is often stated that MP only can squeeze out the highest efficiency for a code running on a DM architecture, but painstaking hand-tuning of an MP code may not be a cost-effective solution, given the rapid evolution of compiler technology.

We focus on HPF because it is the recent, emerging standard in DP languages. Previous experience with CMFortran was invaluable as an introduction to the DP concept for real applications.

At the present time, there are several providers of HPF products, including computer vendors, independent software companies and research institutions. In the following we will present results obtained by using the Portland Group `pghpf` (version 1.3) and the GMD public domain `Adaptor` [2] (version 3.0) compilation system. All the timings reported are obtained on a IBM SP2, equipped with "thin" Power2 nodes at 66 MHz, high performance switch and POE 1.2. Compilations were always performed with IBM `xlf90` 3.2 as target compiler.

2 Basic Data-Parallel Tools

The aim of this effort is to build basic tools for the DP approach and to improve CRS4 know-how on parallel algorithms. In this context we have designed `HPParLib++` [3, 4], a library of C++ classes supporting data-parallel computation on a distributed memory environment. `HPParLib++` provides the user with classes of multi-dimensional arrays, array sections, and a mechanism to distribute them across the processors.

The overall design of the `HPParLib++`, based on the object-oriented style of programming, achieves portability, modularity and openness. Portability is obtained by isolating the system-dependent part of the library from the rest of the system (the Physical Processors class), assuming the existence of an MP

library for the system target and by using only simple communication primitives (send, receive, barrier and broadcast). Actually we already support PVM, MPI and IBM SPx MPL. The library implements all HPF data distribution model features. Furthermore, modularity and openness, intrinsic features for a well designed object-oriented software package, allow us to use `HPParLib++` as a "laboratory" for the exploration of new features within the HPF data distribution model.

Another important aspect is that the library is totally dynamic. Namely, all the information about the actual array distribution is collected during the distribution and alignment phases and is stored in the mapping descriptor giving a complete representation of the array distribution onto the grid of abstract processors. Therefore, once we have built this descriptor, we need no further information about the template or the processor grid related to the array. For instance, all operations involving array or section movements between processors are managed exclusively using the information of the mapping descriptor.

We have also provided the package with a complete Fortran interface and we have integrated `HPParLib++` as the run-time system for the portable compilation system `Adaptor` [2], resulting in a source-to-source translator which converts an HPF code into a SPMD (Single Program Multiple Data) program, composed of a Fortran 77 source with calls to `HPParLib++`.

3 Applications

3.1 Geophysical Computing

The characterization of the subsurface structure for oil prospecting is the most important industrial application of geophysical exploration methods today. A key role in such a task is the migration of seismic data. It consists in the imaging of the earth subsurface structure from the data gathered at the surface after a pressure field has been generated.

We developed a self-contained HPF code (`cnv3d`) [5], performing post-stack migration in a 3D non-homogeneous medium. This code extrapolates the pressure field P(x,y) in depth convolving the initial data with a carefully designed symmetric finite length filter ($\sim 40^2$ lattice units). Following McClellan approach, this is obtained by iteratively (~ 20 times) convolving the data with a smaller (7^2) operator.

We also implemented a pure spectral method, known as PSPI (Phase Shift Plus Interpolation), for post-stack migration [6]. The advantage of computing in the Fourier domain is the unconditional stability of the extrapolation and, in contrast with usual finite difference approximations, the exactness of the dispersion relation implemented by the phase shift formula. The full wavefield is computed by interpolating different extrapolated fields, each one with a reference velocity.

For both algorithms the extrapolation phase is completely concurrent over the N_ω time-frequency components, distributed among nodes. This corresponds

to the directive `(*,*,BLOCK)` for P(x, y, ω). Only the imaging phase, obtained by summing the pressure field over all frequencies, requires a global communication. The results obtained are excellent, giving rise to an approximately linear relative speed-up in the number of processors, as shown in Fig.1. The figure refers to a benchmark performed by using a constant velocity. Since the computational time for `pspi` depends on the complexity of the velocity model, `cnv3d` may become more competitive than shown in the figure for real seismic data in complex geological structures.

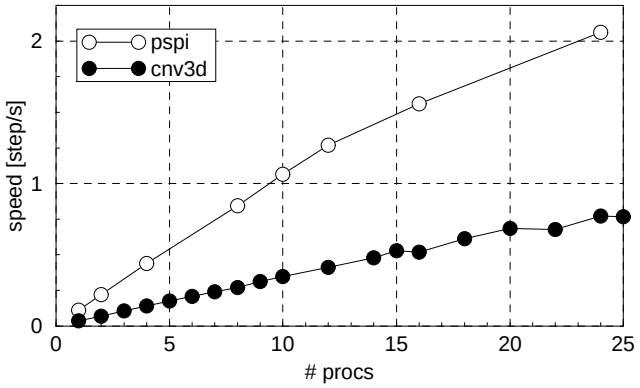

Fig. 1. `pspi` and `cnv3d` speed performances on IBM SP2 in a constant velocity medium with computational grid sizes $N_x = 672$, $N_y = 4$, $N_\omega = 180$.

Another aspect of the geophysics project is the modeling phase, i.e. the simulation of the acoustic wave propagation in the non-homogeneous earth subsurface. This provides accurate synthetic data for the validation of migration codes as well as being of intrinsic interest. Indeed, we want to explore the use of the full wave equation for migration purposes. At the moment, we have a simple modeler in 2D (`mod2d`), in both spectral- and finite-differences version.

3.2 Molecular Dynamics

The N-body problem, i.e. the interaction of N particles through a suitable potential, can be cast naturally in a DP form. Indeed, the kernel of Molecular Dynamics (MD) [7], a simulation technique widely used in the physics of liquids, solid-state physics, bio-chemistry, etc. is a N-body problem.

When long-range interactions are involved, the simpler algorithm combines every molecule with each of the other N-1 *(all-pairs interaction)*. Such combinations are achieved in parallel by using a regular loop communication pattern among processors, obtained by the Fortran 90 `cshift` intrinsic repeated (N-1) times [8].

This approach is embodied in our HPF MD code for the simulation of SPC water with the Ewald sum method. Timings with `pghpf` on the IBM SP2 and a

reasonable fit for them (indicated by the continuous lines) are shown in Fig.2. The speed diagram in log-log scale demonstrates the N^2 scaling. The scalability is acceptable with a sizeable number of molecules (which depends mostly on the computational load – latency ratio).

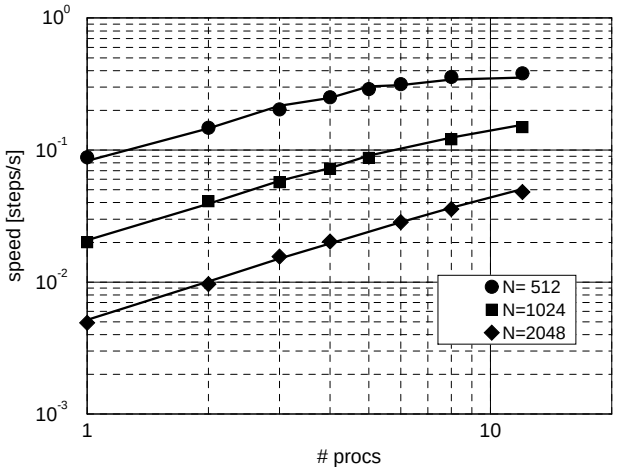

Fig. 2. Speed of MD water code vs number of IBM SP2 processors for various problem sizes. Continuous lines indicate the fit described in text.

The management of the irregular structures arising from short-range interaction models is more problematic in HPF. Implementations using indirect addressing can be quite inefficient depending on the quality of the compiler. We expect an improvement with the advent of HPF-2, the next phase of the HPF Forum, which will address this kind of problems.

3.3 Transport Modeling in Lagoons

Ecosystem modeling typically requires long-term simulations of advection-diffusion of several chemically reactive species, calling for HPC implementations. In the framework of the MMARIE project (a Concerted Action in the field of numerical modeling of marine ecosystems funded by EU) we are developing codes for the simulation of lagoon ecosystems [9]. Water circulation and solute transport due to tide and wind forcing is modeled by the shallow water equations, which we solve by 2D finite difference (FD) schemes.

We planned to assess both the relative performances and the costs of code development in MP and DP styles for running on DM machines. We started with the HPF implementation of the multispecies transport kernel of our codes because a suitable practical approach consists in assuming a representative periodic hydrodynamic pattern as the input for the continuous simulation of transport

(even though the optimal strategy should be to simultaneously simulate both hydrodynamics and transport).

An explicit `QUICKEST-type` FD scheme has been implemented, taking advantage of its favorable performance with respect to standard implicit schemes as well as its natural extension to a parallel algorithm.

The initial porting of the original code (`TransExp`), written in Fortran 77, was quite straightforward and achieved good relative parallel efficiency. However, the absolute performance in comparison with the sequential version was not satisfactory, also because of the peculiar structure of the FD stencil (varying locally depending on the sign of the velocity components). Much better performance was obtained by a careful redesign of the core portion of the code.

A key optimizing feature of HPF compilers is the ability to recognize overlapping areas in the array distribution, so that most stencil computations on the grid do not necessarily involve communications. With `Adaptor` we were able to switch this feature explicitly on or off, obtaining striking results (Fig.3).

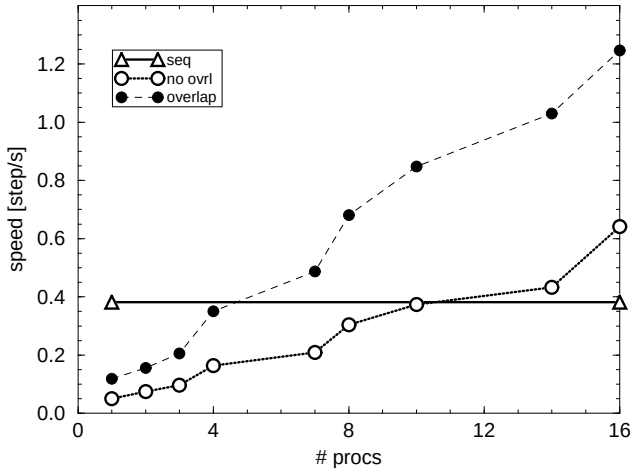

Fig. 3. Speed of `TransExp` kernel routine on a 256^2 grid running on the IBM SP2, with and without overlapping arrays. The speed of the sequential version of the code is also indicated.

4 Conclusions and Outlook

The most attractive HPF features are two: *programmability* (HPF is a high level language) and *portability* (HPF is a standard). One more advantage of HPF is that it is semantically just Fortran 90 augmented by compiler directives, which are skipped for sequential execution. Unfortunately, since the current versions of HPF compilers do not directly support irregular data distributions, HPF alone is hardly effective for problems involv ing such distributions.

However, the HPF support for interfacing with external routines, written in any sequential language and running locally on each processor, can be exploited to integrate DP modules and optimized MP procedures into a parallel library. This feature can be used to build a sort of mixed DP-SPMD programming style that is particularly important for irregular computations where data-access patterns and workload are usually known only at run-time.

Thus, we intend to reuse the considerable experience acquired at CRS4 on the numerical solution of large sparse linear systems of equations (in both DP [10] and MP [11] paradigms) by exploiting HPF interfacing. Currently, there is an effort to integrate most of the parallel software for linear algebra developed at CRS4 or acquired elsewhere, such as the public domain library Scalapack.

In conclusion, even though the current status of the compilers does not permit performances as high as with MP coding, the use of HPF can be attractive, especially for new projects. On the basis of our experience at CRS4, we believe that adopting HPF is a good medium-to long-term investment.

References

1. High Performance Fortran Forum: High Performance Fortran Language Specification Version 1.0. *Scientific Programming* **2** (1993) 1–170.
2. Brandes, T.: ADAPTOR - A Transformation Tool for HPF Programs. In Decker, K.M., Rehmann, R.M. (eds.): Programming Environments for Massively Parallel Distributed Systems. Birkhauser Verlag, 1994, p. 91–96.
3. Greco, D.: Parlib++: A C++ library for data-parallel programming. *AICA Rivista di Informatica* **24** (1995).
4. Greco, D., Cabitza, G.: HPParLib++: A run-time system for HPF. Proc. Annual Conf. AICA (Chia, Sardinia, Sept. 1995). Vol. I, p. 153.
5. Bagaini, C., Bonomi, E., Pieroni, E.: Split Convolutional Approach to 3D Depth Extrapolation. Proc. 65^{th} Annual Meeting SEG (Houston, Oct. 1995), p. 195.
6. Bagaini, C., Bonomi, E., Pieroni, E.: Data parallel implementation of 3-D PSPI. Proc. 65^{th} Annual Meeting SEG (Houston, Oct. 1995), p. 195.
7. Allen, M. P., Tildesley, D. J.: *Computer Simulation of Liquids*. Clarendon Press, 1987.
8. Nardone, C., Rossi, P., Valentini, M.: Data-parallel Molecular Dynamics of Complex Molecules by a Flexible, Multiple Time Scales Approach. In Alimi, J.-M., Serna, A., Scholl, H. (eds.): Science on the Connection Machine System. Proc. 2nd European Connection Machine Users Meeting (Paris Oct. 1993), p. 325.
9. Balzano, A., Nardone, C., Rossi, C.: HPF parallelization strategies for explicit shallow water transport models on distributed memory machines. Proc. MMARIE Annual Meeting (Cagliari, Jan. 1996).
10. Brieger, L., Lecca, G.: Data parallelism in finite element computation. In Peters, A. *et al* (eds.): Proc. of the X Int. Conf. on Computational Methods in Water Resources. Kluwer Academic, 1994.
11. Brieger, L., Lecca, G.: Parallel multigrid preconditioning for finite element models of groundwater flow. In Alvaro, A. *et al* (eds.): Proc. of the XI Int. Conf. on Computational Methods in Water Resources. Kluwer Academic, to appear, 1996.

This article was processed using the LaTeX macro package with LLNCS style