Evaluation of High Performance Fortran on some Real Applications *

Thomas Brandes
German National Research Center for Computer Science (GMD),
P.O. Box 1316, 53731 Sankt Augustin, FRG
e-mail: brandes@gmd.de

Abstract

Due to the existence of a global name space, the data parallel programming model is very comfortable to use. All communication requirements are implicit and therefore left to the compiler. To support this model even to general classes of parallel architectures, i.e. MIMD machines with distributed memory, High Performance Fortran was developed. Beside a data parallel point of view the programmer has the ability to specify data distributions explicitly via directives.

Some real application programs have been investigated to show the usefulness of High Performance Fortran. Therefore these programs have been transformed into the data parallel style and a prototype compilation system has been used to translate them to message-passing programs running on different platforms.

The results verify that the generated message-passing programs scale and can achieve nearly the same performance as their hand-written counterparts. The experiences have also shown where the data parallel programming model has difficulties and where additional optimizations are required.

1 Introduction

Within the ESPRIT-6643 project PPPE (Portable Parallel Programming Environment) some real application programs have been evaluated. The goal of this evaluation was to investigate how partitioning tools and future High Performance Fortran (HPF) compilers can actually help for the parallelization of more complex application codes and how far HPF programs can be competitive with hand-written message-passing programs.

For this purpose some programs of the RAPS (Real Applications on Parallel Systems) benchmark suite have been chosen. As there are also hand-written message passing programs of these codes available, a comparison of data parallel programs and message passing programs was possible.

The sequential versions of the codes written in Fortran 77 needed some changes to become HPF programs. Some situations have been identified where the current HPF definition is not sufficient enough as the current compiler technology still requires more information about the code to generate efficient programs.

Performance results are available by using a prototype HPF-like system. Though hand written message-passing programs will be faster in any case, current compilation systems can achieve nearly the same performance.

*This work was funded by the Esprit-6643 project PPPE (Portable Parallel Programming Environment)

2 The Application Codes

2.1 AVL FIRE Code

FIRE [1] is a fully interactive fluid dynamics package, including powerful, menu-driven pre- and post-processing, developed specially for computing compressible and incompressible turbulent fluid flow as encountered in engeneering environments. 3D and 2D unsteady or steady simulations of flow and heat transfer within arbitrary complex geometries with moving or fixed boundaries can be performed.

Some solvers of this code are part of the RAPS benchmark suite (every solver has about 350 lines of code). These solvers are iterative ones working on unstructured grids. Structured grids are represented by two- or three-dimensional arrays, whereas unstructured grids are represented by one-dimensional arrays to reduce memory requirements. But then neighborhood information must be represented in integer arrays that will be used for indirect addressing within all programs.

Though for the evaluation all three benchmark solvers have been considered, only the results for one solver (GCCG, orthomin with diagonal scaling) will be presented.

2.2 IFS Code

In cooperation between GMD and ECMWF (European Center for Medium-Range Weather Forecasts) the ECMWF's production code for medium-range weather forecasts, the IFS (Integrated Forecasting System), has been parallelized. The solution method employed in the IFS is the spectral transform technique, in which three different function spaces are involved in the computations, namely grid point space, Fourier space and spectral space.

For the evaluation a sequential program of the 2D model of the IFS has been investigated (about 22.000 lines of code). This version contains already all relevant data structures and algorithmic components of the corresponding 3D model. It should be mentioned that the same code has already been parallelized successfully by hand [GJS93].

3 Rewriting Applications in HPF

HPF is a language that supports the data parallel programming model [Hig93]. This model avoids explicit communication and stands for a single thread of control, global name space, and loosely synchronous parallel computation. It is easy to use and data-parallel programs usually scale very well.

For transforming sequential Fortran 77 programs into HPF it is necessary to choose a suitable data layout and to identify the data parallelism by using array operations or parallel loops.

A tool which can greatly simplify the task of porting serial Fortran 77 programs to the data parallel model is the CMAX Converter [TMC92]. The CMAX tool has been derived from the FORGE tool of APR (Applied Parallel Research) [APR93]. It converts sequential programs into data parallel CM Fortran programs [TMC91] by considering the serial loops with its dependences and transforming them either to array operations or forall loops if

[1] FIRE is a registered trade-mark of AVL LIST Gmbh

possible. Furthermore, the layout or distribution of all array variables is determined by using interprocedural analysis.

CM Fortran programs can be rewritten straightforwardly to HPF. This requires in most cases only little syntactical changes.

3.1 Rewriting the AVL FIRE Code

The sequential benchmark solver has been translated to Connection Machine Fortran by using the CMAX Converter. Some small modifications of the original code were necessary to get the solver code successfully translated. Some cleaning up and hand optimizations of the original code have been done to get much more better performance results on a CM-2 and on a CM-5.

The main loop within one iteration computes a new value for every cell by using the old values of this cell and of the six neighbored cells. This loop has no dependences and is directly translated by CMAX to a forall loop that is equivalent to the following vector statement.

```
      DOUBLE PRECISION DIREC1(NNCELL), DIREC2(NNINTC)
      INTEGER LCC (6,NNINTC)
      ...
      DIREC2(1:NNINTC) = BP(1:NNINTC)*DIREC1(1:NNINTC)
     x        - BS(1:NNINTC)*DIREC1(LCC(1,1:NNINTC))
     x        - BW(1:NNINTC)*DIREC1(LCC(4,1:NNINTC))
     x        - BL(1:NNINTC)*DIREC1(LCC(5,1:NNINTC))
     x        - BN(1:NNINTC)*DIREC1(LCC(3,1:NNINTC))
     x        - BE(1:NNINTC)*DIREC1(LCC(2,1:NNINTC))
     x        - BH(1:NNINTC)*DIREC1(LCC(6,1:NNINTC))
```

This statement is very inefficient as it requires six indirect accesses to the array `DIREC1`. Transformations by hand were necessary to reduce them to one indirect access.

3.2 IFS Code

For the 2D model of the IFS, a 1D partitioning of the data in each of the different data spaces is adequate [GJS93]. During the computations in grid point space and in Fourier space, the partitioning is over latitudes, i.e. different processes are responsible for the calculations of different latitudes. The spectral coefficients are partitioned with respect to wave numbers m such that for a given m all corresponding values are mapped onto the same process. Because of the different distributions there is a redistribution of the data in the Fourier space required. One distribution is with respect to latitudes and the second one is with respect to wave numbers m.

Using HPF layout directives for this distribution of the different spaces was impossible as the corresponding data structures in the code are not two- or three-dimensional arrays but one-dimensional arrays that are used as working spaces with additional integer arrays for indexing. The advantage of these data structures is of course the optimal memory requirements. But it is not possible to use HPF layout directives in such a way that the partitioning is conform to the distribution of the spaces in the logical view.

Therefore a restructuring process was necessary to transform the work spaces into two or three-dimensional arrays (see figure 1). This work has to be carried out by hand. Afterwards HPF layout directives can be set that specify exactly the wanted distribution.

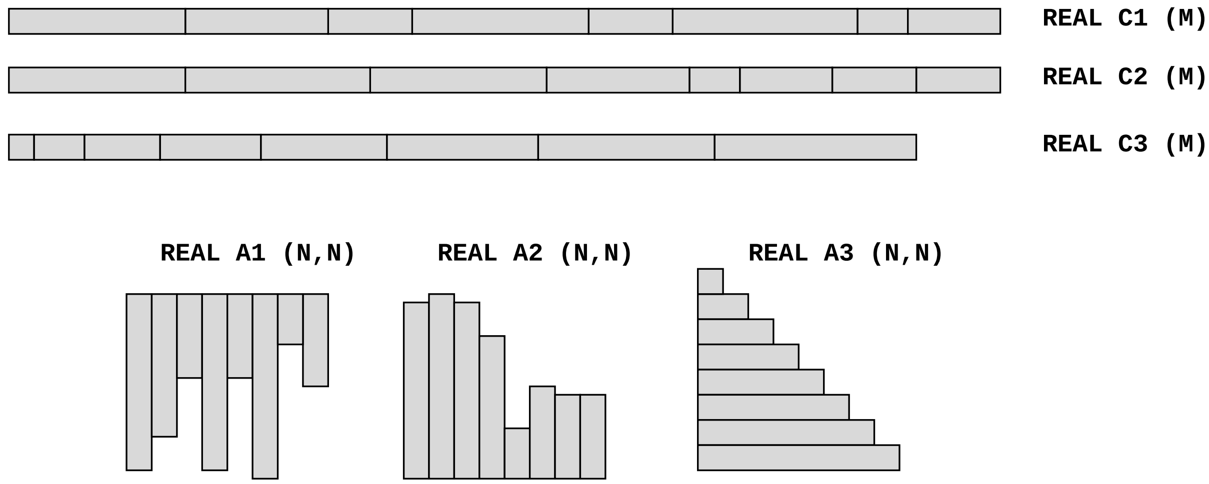

Figure 1: Restructuring of Work Spaces to Structured Arrays

In HPF, an assignment between two arrays with different layout directives results in a redistribution of data. By this way the redistribution of the Fourier space can be realized. HPF contains also a redistribution directive, but this directive is not part of the subset. Some other smaller changes were necessary to identify the parallel loops and the reductions for calculating the spectral norm.

Using the CMAX tool was not possible for this code as this tool would only detect the fine-grained parallelism, but the given parallelism is coarse-grained. In the most relevant parts, the code is composed of loops calling complex subroutines that can be executed independently. This parallelism cannot be detected automatically and must be specified by hand using the HPF INDEPENDENT directive.

The current HPF definition has no directives for specifying private arrays and independent loops that require absolutely no communication. As compiler technology is far away from detecting this automatically, this information must also be given by additional directives to achieve better performance. Such directives are not part of HPF.

4 Compiling the Data Parallel Programs

The data parallel versions of the application programs have been translated to message passing programs by using the Adaptor tool. Adaptor (Automatic Data Parallelism Translator) is a system developed at the GMD for compiling data parallel Fortran programs to equivalent Fortran 77 message passing programs [Bra93]. Only some small modifications were necessary to get the codes successfully translated by Adaptor with its current restrictions.

Some special directives of Adaptor have been used to identify private arrays and parallel loops without communication in the IFS code.

The runtime system of the Adaptor tool is available for many parallel architectures and the programs did actually run on all of these parallel machines. In the following only results for

tjunc (13845 cells)	NP = 1	NP = 2	NP = 4	NP = 8	NP = 16	NP = 32
Adaptor	–	–	71 s	37 s	25 s	16 s
Adaptor (optimized)	74 s	41 s	22 s	13 s	9 s	7 s
Hand-Coded	74 s	39 s	19 s	11 s	7 s	5 s

Table 1: Results for AVL benchmark solver on CM-5 (no vector units)

the Connection Machine CM-5 are presented. The vector units have not been used as the node programs of the message passing programs could not take advantage of them.

5 Results

The performance of the automatically generated program for the AVL benchmark solver was very low due to the high overhead for indirect addressing (see first row in table 1).

The hand-written message passing programs take advantage of the fact that the data decomposition is also a domain decomposition and that the communication pattern does not change between the iterations. This implies that there are only small boundaries between processors (see figure 2) and the necessary communication is reduced to a minimum.

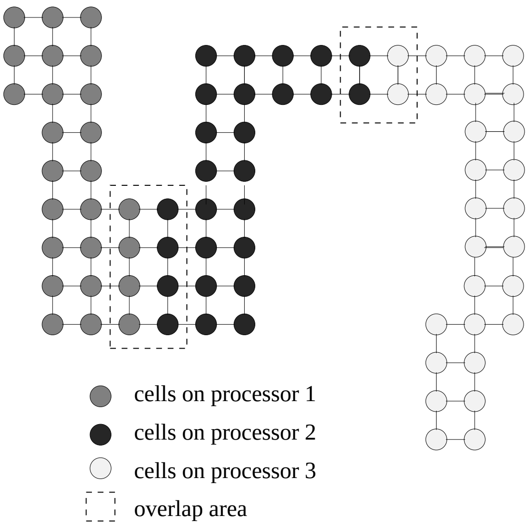

Figure 2: Domain Decomposition of an Unstructured Grid

Adaptor supports also this kind of decompositions, but some small modifications of the generated code were necessary to get similar results as the hand coded version of the solver (see second row of table 1).

Table 2 shows for a typical problem size of the IFS code the runtimes in seconds. The automatically generated message passing program has nearly the same performance as the hand-coded message passing program. Indeed both codes use the same parallelization strategy, but the data parallel version uses nearly the double space of memory. It has also to be mentioned that there is still a high potential for optimizations in the Adaptor compilation system.

CM 5, T63	NP = 1	NP = 2	NP = 4	NP = 8	NP = 16	NP = 32
Adaptor	71.0	35.9	19.9	10.0	5.5	3.2
Hand-Coded	64.5	34.0	17.5	9.0	4.6	2.6

Table 2: Results for T63 on CM 5 (no vector units), Adaptor 2.0

6 Conclusion

High Performance Fortran is on the way to become a useful alternative to writing message passing programs. In contrary to message passing programs, less straightforward work is to be done and data layout can be changed very easily.

If some users still expect to get their old unchanged codes running efficiently on parallel systems, they will be disappointed. Also HPF requires parallel thinking and the employment of a good parallelization strategy at all. Also some knowledge of HPF compiler techniques might be necessary.

While hand-written message passing programs can take advantage of nearly every additional information concerning the data and program structure, compiler for data parallel programs are restricted to the information that they can detect in the program at compile time. Especially better support for irregular data structures is needed before HPF and HPF compilers can be actually competitive.

References

[APR93] FORGE 90: xHPF 1.0 Automatic Parallelizer for High Performance Fortran on Distributed Memory Systems - User's Guide. Technical report, Applied Parallel Research, Inc., April 1993.

[Bra93] T. Brandes. Adaptor: A Compilation System for Data Parallel Fortran Programs. In *International Workshop on Automatic Distributed Memory Parallelization, Automatic Data Distribution and Automatic Parallel Performance Prediction*, Saarbrücken, March 1993. Vieweg.

[TMC92] Thinking Machines Corporation. Using the CMAX Converter. Manual, TMC, September 1992.

[GJS93] U. Gärtel, W. Joppich, and A. Schüller. Parallelizing the ECMWF's Weather Forecast Program: The 2D Case, Technical Documentation and Results for the IFS-2D Model. Arbeitspapiere der GMD 740, Gesellschaft für Mathematik und Datenverarbeitung mbH, March 1993.

[Hig93] High Perforamnce Fortran Forum. High Performance Fortran Language Specification. Final Version 1.0, Department of Computer Science, Rice University, May 1993.

[TMC91] Thinking Machines Corporation. CM Fortran Programming Guide, Version 1.0. Manual, TMC, January 1991.