HIGH PERFORMANCE FORTRAN

Dr. G. M. van Waveren
member of ISO/WG5 and HPFF
e-mail: <waveren@xs4all.hacktic.nl>

1. Introduction

High Performance Fortran consists of a set of extensions to Fortran 90, which make it possible to develop efficient data-parallel applications for massively parallel computer systems. Typical data-parallel applications are seismic applications, the Fast Fourier Transform, computational fluid dynamics and weather simulation.

The advantages of using parallel processing technology in industrial applications lie in the field of cost reduction and turnaround time improvement. For example in seismic production, the improvement in turnaround time can lead to lower costs and quicker results for clients. Also in product development where computer simulations play a significant role, for instance drug design, aircraft design and flow meter design, the use of parallel processing technology leads to a decrease in the time-to-market.

In the development of parallel applications, the use of the emerging software development standards has the advantage that the applications are portable between platforms, making the user less dependent upon the vendors. The user can then switch relatively easily between vendors.

In this lecture, firstly an overview of High Performance Fortran 1.0 will be given. Secondly the incorporation of High Performance Fortran 1.0 into the ISO standards will be discussed. Thirdly the experience of vendors and users gained in 1993 will be discussed. This lecture will end with a discussion of the plans of the High Performance Fortran Forum for 1994.

2. Overview of High Performance Fortran, version 1.0

High Performance Fortran (HPF) [1] is a set of extensions to Fortran, which make it possible to write efficient data parallel programs for massively parallel systems. It was developed between March 1992 and March 1993 by the High Performance Fortran

Forum (HPFF). HPF seems likely to become the language standard for massively parallel systems. In this paragraph, the main features of HPF will be discussed.

A data distribution model has been introduced in the language to allow the user to advise the compiler on the allocation of data objects to processor memories, with the assumption that operations on two data objects will complete faster if the objects are on the same processor. This is specified by specially formatted comments called directives. The distribution directive can divide an array into equal-sized pieces (BLOCK distribution) or assign elements to processors in round-robin fashion (CYCLIC distribution). Additional directives allow more complex patterns, dynamic data remapping, and passing special information about procedure arguments. In the model the data objects are mapped to abstract templates and distributed on abstract processors. The compiler subsequently maps the abstract processor distribution to physical processors. In figure 2.1 this is graphically illustrated.

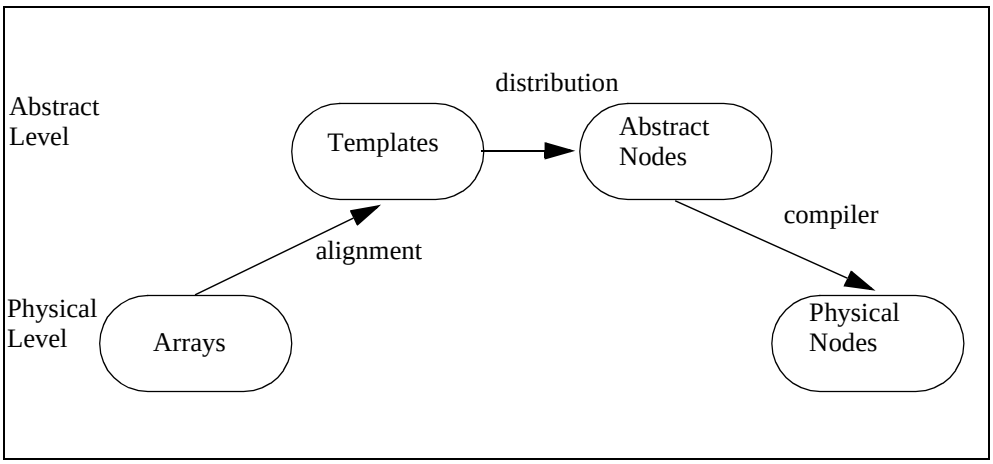

Fig. 2.1: Graphical illustration of data distribution model. Firstly a set of abstract nodes is defined in the software. Secondly templates of arrays are defined and distributed in block or cyclically along the abstract set of nodes. Thirdly the arrays in the software are aligned along the corresponding template. The translation of the abstract set of nodes to the physical nodes is left up to the compiler.

Simultaneous operations on large groups of array elements can be assigned using the Fortran 90 array notation and the FORALL statement and construct. In the Fortran 90 array notation, array sections are treated as objects and simultaneous assignments can be made for these objects. The FORALL statement and construct also allow such parallel assignments, but also provide a wider range of array section shapes and the ability to apply user-defined functions elementally.

Other HPF features include the HPF library, PURE functions, and EXTRINSIC procedures. Space does not allow discussion of these features and the reader is referred to the HPF Language Specification [1].

3. ISO standardization

The ISO working group on Fortran met in 1993 and agreed upon the principle that the ISO Fortran standard should be extended in such a manner, that most HPF programs will conform to it. For ISO standardization purposes, the HPF extensions were subdivided into language constructs and non-language constructs.

The language constructs will be included in the next Fortran revision, i.e. Fortran 95. These consist of the FORALL statement and construct and some minor extensions to intrinsic functions. Fortran 95 addresses itself only to pure language constructs. Thus, the non-language constructs, i.e. the directives and the HPF_LIBRARY module, will be included into another standard document than Fortran 95.

4. Vendor- and User experience with HPF 1.0

During a meeting in january 1994 in Houston, Texas, USA, vendors and users gave presentations on their experiences with HPF 1.0. This paragraph is meant to give to the reader an account of these discussions.

4.1. Vendor experience

The vendors Thinking Machines Corporation, DEC, IBM, the Portland Group, Applied Parallel Research and Convex gave presentations at this meeting. Some of their experiences are listed below:

- The development of a compiler for HPF is an incentive for the development of a compiler for Fortran 90.
- HPF is efficiently implementable.
- The language is expressive.
- The language must be used carefully. Small syntactic changes can cause enormous differences in performance.

4.2. User experience

Fox classified user applications with respect to their parallelization properties. The following four classes emerged:

- *Synchronous applications:* Applications where the data structure is simple to parallelize. Examples are crystalline molecular dynamics or Monte Carlo and Fast Fourier Transform. These applications are typically data-parallel and can be efficiently implemented using HPF.
- *Embarrassingly parallel applications:* Essentially independent execution of disconnected components. The independent components can be quite complicated. Examples are the calculation of matrix elements of an Hamiltonian and quantum Monte Carlo calculations. These applications are also efficiently implemented in HPF.
- *Loosely synchronous applications:* Applications where the data structure is hard to parallelize. Examples are direct methods for sparse matrix solvers and multipole methods for particle dynamics. The expression of these applications in HPF 1.0 is uncertain.
- *Asynchronous applications:* Applications with functional and/or data parallelism that is irregular in space and time. Examples are event driven simulations and transaction analysis. These applications are currently outside the scope of HPF.
- *Metaproblems:* Asynchronous collection of (loosely) synchronous components. The different components are data-parallel and can be expressed in HPF.

5. Plans of the High Performance Fortran Forum for 1994

The High Performance Fortran Forum has set itself the following four tasks for 1994:

- HPFF wants to encourage ‘industrial strength’ implementations of HPF by vendors. This means that the HPFF will develop more notes to implementors, collect and publish practical programming kernels and support the establishment of a validation suite.
- HPFF will turn out clarifications, corrections and interpretations of HPF 1.0. This includes clarifying the ways that HPF 1.0 already supports coarse-grained parallelism. Any individual or organization needing a clarification or interpretation, can make his or her request known to the HPFF. The electronic mail addresses for sending in requests are <hpff-interpret@cs.rice.edu> for interpretations and <hpff-comments@cs.rice.edu> for comments.
- HPFF will identify and flesh out requirements for HPF 2.0. However these will not be developed in 1994. During 1994 a decision will be taken whether to develop them

in 1995. Candidates for requirements are parallel I/O, irregular grids and additional tasking.
- An HPF 1.x document will be produced during 1994.

A general meeting will take place in early 1995 to discuss the results and experiences reached in 1994 and to make plans for 1995.

Acknowledgements

The author wants to thank Schlumberger Ltd, the Netherlands Fortran Specialists Group and the Nederlands Normalisatie Instituut for help towards the trips to the standardization meetings, necessary for writing this article. He thanks Chuck Koelbel of Rice University for reviewing the article. He also wishes to thank Marisa do Nascimento for her support during the writing of this article.

References

[1] High Performance Fortran Forum: High Performance Fortran Language Specification, May 3, 1993. This document is available via anonymous ftp from the file servers located in the US at Rice University, Oak Ridge National Laboratory and in Europe at GMD-I1.T (Sankt Augustin, Germany).