High Performance Fortran: Implementor and Users Workshop

Alok Choudhary *
Syracuse University
Syracuse, NY

Charles Koelbel
Rice University
Houston, TX,

Mary Zosel
Lawrence Livermore
National Lab

Abstract

High Performance Fortran (HPF) is anticipated to be an industry-wide standard language portable from workstations to massively parallel supercomputers, implemented by many vendors. Functions needed on the highest-performance parallel machines are particularly well-supported in HPF. The definition of HPF, (version 1, completed May 1993) incorporated feedback from a large number of users during the language design phase.

The purpose of this workshop is to continue the interaction among developers of compilers and software tools for HPF, potential users of HPF and hardware developers. Specifically, this workshop focuses on the feedback from the implementors of HPF as well as from users of the initial implementations. The reports will be followed by a discussion for the purpose of gathering user and implementor input about how the HPF specification might be changed to strengthen the language and make it useful for a broader class of applications.

1 Background

The annual SuperComputing conferences have played an important part in the development of High Performance Fortran, starting with SC91 when Ken Kennedy and Geoffrey Fox described the initial motivation behind the development of Fortran- D in a panel discussing problems related to portability for parallel computing. There was a wide response to this work, and by the time that SC91 was here, there was already pressure to standardize the language associated with this research project. This was discussed in a birds-of-a-feather session at SC91, and resulted in the creation of the High Performance Computing Forum (HPFF). A specific goal of HPFF was to have a draft language specification ready in time for workshop presentation calling for public feedback at SC92. It is now appropriate that a workshop at SC93 explore the initial experiences with the first version of HPF.

Version 1.0 of HPF language specification was released in May 93 after incorporation of changes suggested during the 3 month comment period [HPF-Spec92].

*Supported by ARPA contract # DABT63-91-C-0028 and NSF Young investigator award CCR-9357840.

2 Motivation for HPF

Since its introduction in the 1950s, Fortran has been the language of choice for scientific and engineering programming. Support for Fortran is available on virtually all computers ranging from personal computers and workstations to servers, mainframes, and supercomputers. Fortran itself has evolved over this period, most recent version being Fortran 90 which incorporates many new features including Array Constructs, Elemental Intrinsic Functions, Reduction Intrinsic Functions, Dynamic Storage Allocation, enhanced support for Modular Programming etc.

Exploiting the full capability of modern parallel architectures, however, increasingly requires more information than that can be expressed even in Fortran 90. Such information includes indicating

- opportunities for parallel execution,
- types of available parallelism,
- data mapping among processors,
- placement of data within a single processor, and
- specification of control parallelism.

Over the last year, High-Performance Fortran Forum, an informal coalition of industrial, academic, and government groups defined extensions to Fortran 90. These extensions in conjunction with the base Fortran

90 language are termed as "High-Performance Fortran (HPF)".

The basic goal of the HPFF was to define a language which will serve as a platform for portable parallel programming. It is believed that most vendors of parallel computers and third party compiler and system software developers for parallel computers will adopt HPF.

3 An Overview of HPF

HPF is designed to provide a portable extension to Fortran 90 for writing data parallel applications. It includes features for mapping data to parallel processors, specifying data parallel operations, and methods for interfacing HPF programs to other programming paradigms. HPF uses compiler directives where ever the extensions do not cause a potential change to the meaning of the program and explicit language extensions where the semantic meaning is changed.

The parallelism in a program, expressed by constructs like array assignment, FORALL statements, DO INDEPENDENT loops, intrinsic and standard library procedures, and EXTRINSIC procedures, determines how many operations a computer can possibly do at one time. It is then the compiler's responsibility to schedule those operations on the physical machine so that the program runs as fast as possible.

Communication in a program is an overhead that opposes parallelism. HPF puts much of the burden of communication on the compiler; the user supplies a very high-level data mapping strategy and the system generates the details of the communication it implies.

The data mapping directives are probably the most publicized features in HPF. They describe how data is to be divided up among the processors in a parallel machine. The presumption is that the processor responsible for some data (also called the processor that owns the data) can read or write it much faster than another processor. HPF describes the data-to-processor mapping in two stages using the DISTRIBUTE and ALIGN operations. DISTRIBUTE is an HPF directive that describes how an array is divided into even-sized pieces and distributed to processors in a regular way. ALIGN is an HPF directive that describes how two arrays "line up" together. A group of arrays that are usually used together in a predicable manner may be aligned together to some base array, or to some abstract TEMPLATE and then distributed with a single DISTRIBUTE directive.

Additional data mapping directives allow redistribution (REDISTRIBUTE) and realignment (REALIGN) of data arrays and allow specification of abstract processor arrangements (PROCESSORS) for the purpose of load balance and memory balance across the physical processors of a machine.

The interface between subprograms requires special features for conveying the expectations about the distribution and alignment of the subprogram arguments. Three basic forms are supported: a prescriptive specification, where the programmer gives a specific mapping and the compiler coerces the argument appropriately; a descriptive specification, where the programmer alleges that the argument already has certain mapping properties, and a transcriptive specification where the compiler is instructed to adapt to whatever mapping the incoming argument may have. When an explicit Fortran 90 interface is specified for a subprogram, the compiler will assure conformance for all arguments in the subprogram invocation.

Traditional FORTRAN features for storage association (COMMON and EQUIVALENCE) have a strong interaction with data mapping directives. Some uses of storage association make data mapping either highly inefficient or even impossible, e.g. two different distributions for arrays associated by EQUIVALENCE. HPF gives a set of rules for use of COMMON and EQUIVALENCE and supplies a SEQUENCE directive to allow the programmer to specify that storage association is required. A similar set of restrictions is applied to passing subprogram arguments when the shape of the argument is not the same on both sides of the call.

The new HPF features for parallelism are primarily syntax extensions, along with one additional directive. The FORALL construct extends Fortran 90 array operations by supplying a multi-dimensional set of index values that are used to evaluate the results of each statement in the construct. HPF introduces PURE functions . These are functions that have no side-effects and thus may be used inside a FORALL statement and still provide deterministic results. The HPF INDEPENDENT directive may be applied to either the FORALL construct or to the Fortran DO loop. A programmer can use it to promise the compiler that the FORALL or DO does not make any "bad" data accesses that force the loop to be run sequentially. With this information, a compiler knows it is safe to produce parallel code.

Additional HPF language extensions introduce a few new intrinsic procedures and an extensive library. These allow for interrogation of the machine and mapping characteristics and also define many data parallel operations such as combining-scatter operations.

These data parallel operations allow these common operations to be optimized by the HPF vendor for maximum efficiency on a given machine, while still maintaining the portability of the user's code.

The last new feature of HPF is a recognition that many programs need to use more than a single programming paradigm. The EXTRINSIC program interface definition allows a programmer to give explicit notice to the compiler that the program is entering a different model of execution. For machines that support MIMD type operations, HPF defines one additional programming paradigm called HPF_LOCAL where data references are specifically local to a processing node rather than global to the entire processor set.

Finally, HPF recognizes that efficient implementation of all of the HPF features in addition to full Fortran 90 is potentially incompatible with timely productions of compilers that allow the programmer portable data parallel programs. As a result, a subset of the language is defined for the purpose of encouraging early access to basic features.

4 Vendor Endorsement of HPF

Many vendors have already announced efforts to develop software tools to support HPF. These vendors include TMC, MasPar, DEC, Intel, APR, KAI and PGI. Other active participants from the industry who have shown considerable interest in supporting HPF (but have not made any formal announcements yet) include IBM, Cray Research, Convex, HP, Ncube, Meiko, Fujitsu and Archipel.

5 Goals of the Workshop

The main goal of this workshop is to continue the interaction among developers of compilers, potential users of HPF and hardware developers. Specifically, the goals are to

- stimulate discussion among implementors of HPF so that they can share their experiences,
- to provide feedback to the implementors from the users and other workshop attendees, and
- identify potential problem areas, limitations of the language and identify features that may be desirable in the newer versions of HPF.

6 WHat next for HPF?

In the interest of getting a specification completed so that implementations could begin, HPFF knowingly limited discussion in some important areas. The current plan for HPFF is to reconvene with a meeting in early 94 calling for proposals of features that should be included, clarified, (or even eliminated) in a second version of HPF. This would be followed by a set of meetings similar to the first set of HPFF meetings. Important features for discussion include support for unstructured arrays and parallel I/O, in addition to areas of the language that have been identified as needing clarification.

Acknowledgements

We would like to thank Ken Kennedy and Geoffrey Fox who initiated the HPF effort. Special thanks to all our colleages in the HPFF for their strong commitment to the entire process without which HPF would not have been defined in time. Finally, on behalf of HPFF, we would like to express our gratitude to hundres of people from all over the world who provided constant input and feedback during and after the language definition process.

Bibliography

[ABMSW92] J. C. Adams, W. S. Brainerd, J. T. Martin, B. T. Smith , J. L. Wagener , Fortran 90 Handbook, Intertext-McGraw Hill, New York, NY, 1992

[ALS91] E. Albert, J. Lukas and G. Steele, Jr. Data Parallel Computers and the FORALL Statement , Journal of Parallel and Distributed Computing, October 1991.

[BCFH93] Z. Bozkus, Alok Choudhary, Geoffrey C. Fox, T. Haupt, and S. Ranka, *Fortran 90D/HPF Compiler for Distributed Memory MIMD Computers: Design, Implementation, and Performance Results*, Supercomputing '93, November 1993 (to appear), Portland, OR. (R)

[CMZ92] B. Chapman, P. Mehrotra and H. Zima , Programming in Vienna Fortran , Scientific Programming 1,1, August 1992.

[CMZ93] B. Chapman, P. Mehrotra and H. Zima , High Performance Fortran Without Templates: An Alternative Model for Distribution and Alignment , In Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP) , San Diego, CA, May, 1993.

[CW92] M. Chen and J. Cowie , Prototyping Fortran-90 Compilers for Massively Parallel Machines , Proceedings of the SIGPLAN '92 Conference on Program Language Design and Implementation, San Francisco, CA., June, 1992.

[Fox86] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon and D. Walker, Solving Problems on Concurrent Processors,, Prentice-Hall, Englewood Cliffs, NJ, 1988

[FHKKKTW91] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C. Tseng and M. Wu , Fortran D Language Specification , report COMP TR90-141 (Rice) and SSCS-42c (Syracuse), Department of Computer Science, Rice University, Houston, Texas, and Syracuse Center for Computational Science, Syracuse University, Syracuse, New Your, April 1991.

[F9091] International Organization for Standardization and International Electrotechnical Commission , Fortran 90 [ISO /EC 1539: 1991 (E)], May 1991. Also ANSI X3.198-1992.

[HPFSpec92] High Performance Fortran Forum , High Performance Fortran Language Specification, version 1.0, Center for Research on Parallel Computation, Rice University, Technical Report CRPC-TR92225, Revised May 1993. To appear in Scientific Programming , vol. 2, no. 1.

[HPFJoD92] High Performance Fortran Forum , High Performance Fortran Journal of Development, Center for Research on Parallel Computation, Rice University, Technical Report CRPC-TR933300, May 3, 1993. To appear in Scientific Programming , vol. 2, no. 1.

[HKK:Overview] S. Hiranandani, K. Kennedy, C. Koelbel and U. Kremer and C. Tseng , An Overview of the Fortran D Programming System , In Languages and Compilers for Parallel Computing, Fourth International Workshop, ed. U. Banerjee, D. Gelernter, A. Nicolau and D. Padua , Springer-Verlag, Santa Clara CA., Aug. 91.

[KLS90] K. Knobe, J. Lukas and G. Steele, Jr. , Data Optimization: Allocation of arrays to Reduce Communication on SIMD Machines,, Journal of Parallel and Distributed Computing, Vol 8 No. 2, Pages 102-118, Feb. 1990.

[KLSSZ93] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, Jr., and M. Zosel, The High Performance Fortran Handbook, MIT-Press, Cambridge, MA., To appear 1993.

[KM92] C. Koelbel , An Overview of High Performance Fortran , Fortran Forum , Vol. 11 No. 4, Dec. 1992.

[DBL93] D. B. Loveman , High Performance Fortran, IEEE Parallel and Distributed Technology, Vol. 1, No. 1, Feb. 1993.

[DBL92] D. B. Loveman , Elemental Array Assignment – the FORALL Statement , Proceedings of the Third Workshop on Compilers for Parallel Computers, Vienna, Austria, July 1992.

[MR90] M. Metcalf and J. Reid, Fortran 90 Explained, Oxford Science Publications, 1990.

[GLS93] G. L. Steele Jr., High Performance Fortran : Status Report , ACM SIGPlan Notices ,Vol. 28 No. 1, Jan. 1993.

[TMC91] Thinking Machines Corporation , CM Fortran Reference Manual , Cambridge, Massachusetts , July 1991.

[MilSt1753] US Department of Defense , Military Standard, MIL-STD-1753: FORTRAN, DoD Supplement to American National Standard X3.9- 1978 , November 9 , 1978.

[ZBCM92] H. Zima, P. Brezany, B. Chapman, P. Mehrotra and A. Schwald, Vienna Fortran — A Language Specification, Version 1.1, ICASE Interim Report No. 21, Hampton VA., Mar. 1992.

[ZC91] H. Zima and B. Chapman , Supercompilers for Parallel and Vector Computers , Addison-Wesley, New York, NY, 1991.