SHORT COMMUNICATION

High Performance Fortran Implementations: A Survey

R.H. PERROTT,[1] E. AYGUADE,[2] J. GARCIA,[2] AND J. TORRES[2]

[1] *Department of Computer Science, Queen's University, Belfast BT7 1NN, UK; e-mail: r.perrott@qub.ac.uk*
[2] *European Center for Parallelism at Barcelona (CEPBA), Universitat Politecnica de Catalunya, Gran Capita, s/n Modul D-6, 08071, Barcelona, Spain; e-mail: {eduard,jordig,torres}@ac.upc.es*

1 INTRODUCTION

The Universitat Politecnica de Catalunya, Barcelona (UPC) and The Queen's University of Belfast (QUB) have been engaged in language design and implementation projects for many years with particular emphasis on languages for scientific and engineering applications. Currently in this language area, the data parallel approach is receiving great attention from users, language designers and computer vendors. One of the major innovations in data parallelism has been the advent of High Performance Fortran (HPF). HPF has been the result of the deliberations of a group of academic, commercial and industrial organizations with interests in high performance computers – the HPF Forum. HPF is intended to provide a 'standard' for parallel programming on a wide range of machines.

In the spring of 1993 the HPF Forum produced a report of the HPF language which is described in *Scientific Programming* [1]. HPF contains directives which are structured comments that suggest implementation strategies or assert facts about a program to the compiler. A program should generate the same results whether the directives are processed or not. In addition, there are a few new features of the language which are direct extensions of Fortran 90. For example, the FORALL construct enables the expression of general array assignments to sections of arrays. To ease the compiler problems and speed up the implementation of HPF, the Forum defined an official subset of HPF as well as the full language. The official subset includes all HPF directives and language extensions but excludes features such as

1. the directives REALIGN, REDISTRIBUTE, DYNAMIC, VIEW and PURE;
2. the full FORALL construct (the simple FORALL element array assignment is available);
3. the HPF library.

Following the first report there has been further discussion within the Forum on other topics such as task parallelism and irregular data structures, these features may be included in a next language, commonly referred to as HPF 2.

In order to ascertain the current status of the implementation of data parallel languages with particular emphasis on HPF a survey of proposed or existing implementations has been carried out. The results of the survey are presented in this article.

The main objectives of the survey were to determine

1. what is currently available;
2. the current status of the implementations;
3. how they adhered to the proposed 'standards';
4. what future developments are planned.

The survey was carried out over the period February 1996 to May 1996. It is a snapshot of the state of HPF implementations. The authors are indebted to the contributors for providing the information. The following summary has been collated on the basis of the information provided and the authors hope that they have been accurate in their interpretation of the provided information. A Web site at **http://www.ac.upc.es/HPFSurvey** contains the main details of the survey. We hope to update the Web site at regular intervals as progress is made in the area of HPF implementations and trust that the data parallel community will find this useful and use it as a point of reference for developments in this area.

© 1997 IOS Press
ISSN 1058-9244/97/$8
Scientific Programming, Vol. 6, pp. 243–248 (1997) .

Table 1. List of HPF Compilers

	Product	Support	Vendor/developer	URL
Prototypes	ADAPTOR	F−+	GMD	www.gmd.de/SCAI/lab/adaptor/adaptor_home.html
	Annai/PST	O+	CSCS	www.cscs.ch/~wylie/CSCS-NEC/Tools.html
	HPFC	S+	Ecole de Mines de Paris	www.cri.ensmp.fr/pips/hpfc.html
	PANDORE	S	IRISA	www.irisa.fr/pampa/PANDORE/pandore.html
	shpf	O+	University of Southampton	www.ccg.ecs.soton.ac.uk/shpf/shpf.html
	vfcs	S+	Institute for Software Technology and Parallel Systems	www.par.univie.ac.at/project/vfcs.html
Commercial products	Expert HPF	O	ACE	www.ace.nl/pressrel/hpf.html
	Fortran 90 HPF	F+	Digital Equipment Corporation	www.digital.com/info/hpc/fortran/hpf.html
	HPF Mapper	O	N.A. Software	www.connect.org.uk/merseymall/NASoftware/fortran/toolhpf.html
	pghpf	F−	Portland Group	www.pgroup.com/HPF.src.html
	TM/HPF	O+	Thinking Machines Corporation	www.think.com
	VAST-HPF	F−	Pacific Sierra Research Corporation	www.psrv.com/vast/vasthpf.html
	XL HPF	S	IBM	www.software.ibm.com/ap/fortran/xlhpf/index.html

Support: F: full HPF definition; O: official HPF subset; +: also includes own features; −: with a few exceptions; S: some features only.

2 LIST OF COMPILERS

The responses to the questionnaire have been divided into two groups, namely, those implementations which are regarded as prototypes and are taking place at research institutions or universities, and those implementations that are being produced by commercial companies. Table 1 contains the implementations considered, the name of the developer and contact details.

F indicates that the full language features are included, O the official subset language features. Some implementations fall between the full and official subset 'standards' in that they have included some HPF features only, that is, features selected from both the full and official 'standards'; these implementations are labelled S. Unfortunately may implementations have introduced variations on the full and official subset 'standards'. This is marked by using + (plus) to indicate that extra language features not part of the 'standards' have been included while − (minus) indicates that some features have been omitted. The third column in Table 1 records the adherence of each compiler to the 'standards'.

In summary, only one implementation is available for the full language but also include extra features, namely, DEC Fortran 90 HPF. The same situation occurs for the official subset with Southampton's shpf, Thinking Machines TM/HPF, ACE EXPERT HPF, Annai/PST and NA Software's HPF Mapper, all offering at least the official subset language. The picture becomes less clear when the other implementations are considered as they have yet to implement all the features of either of the two 'standards'.

3 GENERAL DESCRIPTION OF SYSTEMS

This section gives a brief overview of each of the considered systems. It highlights the broad characteristics of each implementation and gives brief background information, where appropriate.

3.1 Prototype Systems

ADAPTOR: GMD, Germany

ADAPTOR (Automatic DAta Parallelism TranslaTOR) is a source to source transformation system that transforms data parallel programs written in Fortran with array extensions, parallel loops and layout directives into parallel programs with explicit message passing. By means of the source-to-source transformation ADAPTOR translates a data parallel program into an equivalent SPMD (Single Program Multiple Data) program that runs on all available nodes. The essential idea of the translation is to distribute the arrays of the source program onto the node processors. Thus, parallel loops and array operations are restricted to the local data belonging to a processor (owner computes rule). The necessary communication statements for exchanging non-local data are inserted automatically. The control flow and statements with scalar codes are replicated onto all nodes. ADAPTOR supports both CM Fortran and HPF and generates programs in Fortran 77 or Fortran 90.

Annai/PST: CSCS-NEC, Switzerland

Annai is an HPF/MPI tool environment consisting of a compilation system, debugger, performance monitor and analyzer where MPI is the Message Passing Interface. Annai/PST (Parallelization Support Tool) specifically refers to the compilation and parallelization component that supports an extended HPF model and allows the combination of HPF and extrinsic SPMD HPF or C. Annai/PST provides support for user-defined data distributions and loop distributions, run-time data consistency analysis and high level program analysis. Annai/PST performs translations from extended HPF to portable ANSI C plus calls to a run-time library. The PST library, written in C++, provides run-time communication pattern generation and management. MPI is used as the basic communication library.

HPFC: Ecole de Mines, Paris, France

The HPFC compiler is a prototype compiler being build to test new optimization techniques for compiling HPF. It does not take full HPF as input rather the syntax is restricted to Fortran 77 with certain static and dynamic HPF mapping directives. The target programming model is a host node distributed memory message passing model. The low level issues of process management and basic communication are handled by PVM. HPFC is integrated into PIPS (Scientific Programs Interprocedural Parallelizer) which provides program analysis functions.

PANDORE: IRISA, France

PANDORE is an environment for programming distributed memory computers using a subset of HPF or the C Pandore language. The environment comprises a compiler, machine independent run-time and execution analysis tools including a profiler and a trace generator. The PANDORE compiler includes a run-time resolution technique that introduces masks and potential communications at the statement level. An optimized compilation scheme handling reductions and parallel loops with one statement is also included. A suitable run-time system completes these compilation techniques and obtains efficient execution for a large class of programs. The compilation scheme is based on the decomposition of the arrays into rectangular blocks and supports HPF BLOCK and CYCLIC distributions.

shpf: Southampton University, UK

The Southampton HPF (shpf) system is a translator from HPF to standard Fortran. The generated Fortran contains calls to an HPF run-time system which, in turn, makes calls to an underlying message passing library. Concurrency and data parallel execution are obtained from the use of array syntax such as array assignments, array external and intrinsic functions and the WHERE statement and construct. More specifically, the system translates HPF to standard Fortran 90 with calls to a run-time system written in C++, the run-time library is layered on MPI.

VFCS: Vienna University, Austria

The Vienna Fortran Compilation System (VFCS) is an integrated compilation environment that includes a parallelizing compiler for Vienna Fortran and HPF, program analysis and transformation tools, and tools for sequential profiling, performance predication and measurement. The input languages for VFCS are Fortran 77 and subsets of Vienna Fortran, Vienna Fortran 90 and HPF, including Fortran 90 array syntax. VFCS supports most of Vienna Fortran and HPF data parallel extensions, it currently provides regular block distributions, general block distributions, distributions to processor subsets, distribution of workspaces, distribution by alignment and indirect distributions using mapping arrays. The code generator generates Fortran message passing code for various parallel target architectures including the Intel iPSC 860 Hypercube, the Intel Paragon, the Meiko CS 2 as well as the PARMACS and MPI standards.

3.2 Commercial Systems

EXPERT HPF: Associated Compiler Experts, Amsterdam

The EXPERT HPF compiler is based on ACE's CoSy compilation system which allows the reuse and integration of existing and new compiler functionality. This technology has been used to provide a compiler for the official subset of HPF. The compiler translates HPF code to either PowerPC code or Sparc code or standard C. The system utilizes some run-time system routines and MPI. The EXPERT HPF compiler compiles both Fortran 90 and HPF code directly to PowerPC code. The system is being used in both the Parsytec GC PowerPlus and PowerXplorer parallel systems. Future plans include the implementation of HPF 2 as soon as it is approved.

Fortran 90 HPF: DEC

The DEC HPF compiler is a native alpha microprocessor compiler that generates code for a single alpha workstation, for a symmetric multiprocessor (SMP) alpha server, for clusters of alpha workstations or clusters of SMP alpha servers. Portable executable code can be constructed to run on any of these configurations without recompilation or relinking. The full version of the language is supported as well as features proposed for the next version of HPF. DEC's HPF includes the full Fortran 90 language. A number of HPF specific optimizations are included such as message vectorization, nearest neighbour optimizations and parallelized reductions. DEC expects that the next release of the compiler will support many of the remaining features of HPF 2, the release after that is expected to support all of the remaining HPF 2 language features.

HPF Mapper: NA Software Products, UK

The HPF Mapper translates HPF to Fortran 90 plus message passing calls to one of PVM, MPI or Parmacs message passing subsystems. It provides access to parallelism on distributed memory systems. It produces SPMD Fortran 90 message passing code which will run on any system supporting a suitable message passing library and a Fortran 90 compiler. Implementations currently available include Sun workstation clusters, Meiko CS 2 and T800 Transputer systems. Versions for the IBM SP 2 and Silicon Graphics are under development. The HPF Mapper accepts subset HPF programs and the following extensions, namely, free form source, CASE constructs, modules and full Fortran external I/O. Future plans include the addition of features to complete the full HPF language.

pghpf: Portland Group

The Portland Group's HPF compiler, pghpf, is a retargetable compiler for HPF. It is designed for compilation on a wide range of systems, including workstation clusters, shared memory and distributed memory systems. The key to this portability is a transport independent interface which acts as the kernel of the pghpf run-time system, it has been implemented on top of PVM, MPI and Parmacs and multi-threaded operating systems. The compiler performs global optimization, vectorization, communication optimization, and interprocedural analysis in support of parallelization for the HPF paradigm. A few features are missing from the system, namely, pointer/target (F90), character array (F90), PURE (HPF), internal procedures (F90) and recursion (F90) these are scheduled to be added in the near future.

TM/HPF: Thinking Machines Corporation, USA

TM/HPF 1.0 is an official subset HPF compiler, which includes the HPF library. TM/HPF also supports a switch mode that allows the processing of both CM Fortran layout directives and specific HPF syntax. The compiler produces nodal Fortran 77 code and it is layered on top of a ported version of the CM5 run-time system. Because of the limitations of this run-time system TM/HPF does not implement some mapping features such as CYCLIC and BLOCK_CYCLIC distributions and some alignment such as transpose, collapse and replicated axes.

VAST-HPF: Pacific-Sierra Research Corporation

VAST-HPF translates the input HPF source to Fortran 77 and then passes it to a Fortran 77 compiler. It generates calls to message passing routines in place of the original HPF code and supports PVM, P4 and MPI. It is planned for platforms such as SUN/Sparc, IBM RS/6000, HP 9000/7000, SGI and DEC alpha systems. VAST-HPF optimizes both communication between nodes in distributed systems and performance of code on a single node. Features include optimized generation of data motion calls to avoid redundant send and receives, optimization of generated node programs for superscalar processors. VAST-HPF supports all of Fortran 90 but currently omits several features of HPF such as REALIGN, REDISTRIBUTE, DYNAMIC, inherit and transcriptive distributions and some HPF intrinsics. Future plans include completing the HPF implementation and adding new optimizations to take advantage of HPF 2 features.

XL HPF: IBM

The XL HPF is a native compiler, not a preprocessor. It supports parallel execution through the addition of compiler directives to Fortran 77 and Fortran 90 programs and exploits the standard MPI communication protocol. It is an implementation based on subset HPF for RS/6000 SP systems and clusters of RISC system/6000 systems. It also includes extensions from outside the HPF 'standard'. It provides a number of parallelization and optimization features such as message vectorization, collective communication, elimination of redundant communication, optimization of nearest neighbour shift communication and transformation loops for communication scheduling. XL HPF also has an optimizing back end designed to exploit the systems architecture of POWER, POWER2 and PowerPC.

4 RESULTS

The technical results of the survey have been condensed into a single table, Table 2. It contains details of each implementation, in particular data layout features, data parallel statements and code generation which are detailed in the following columns.

The data layout features have been classified first according to static or dynamic characteristics and then with respect to features within that broad classification.

Under the static heading the features considered were intradimensional alignments and the data distributions, namely, BLOCK, CYCLIC and BLOCK_CYCLIC and whether these distributions were applied to a single dimension or across many dimensions. The picture is variable across the implementations. The following implementations provide all the features mentioned, namely,

Table 2. Main Features

Product	Static layout						Dynamic layout			Parallel statements			Code generation	
	O	B	C	BC	1D	nD	P	L	A	FORALL	INDEP	Other	Library	Low-level
ADAPTOR	+	+	+		+	+	+	+	+	+	+	+	+	
Annai/PST		+	+	+	+	+	+			+	+		+	
HPFC	+	+	+	+	+	+		+	+		+		+	
PANDORE		+	+	+	+	+	+				+		+	
shpf	+	+	+	+	+	+	+						+	
vfcs	+	+			+	+				+	+	+	+	
Expert HPF	+	+	+	+	+	+	+			+			+	+
Fortran 90 HPF	+	+	+	+	+	+	+			+	+			+
HPF Mapper	+	+		+	+	+	+			+			+	
pghpf	+	+	+	+	+	+	+	+	+	+	+	+	+	
TM/HPF	+	+			+	+	+			+		+	+	
VAST-HPF	+	+	+	+	+	+	+			+	+		+	
XL HPF	+	+		+	+	+	+			+	+			+

Static layout:	O: intra-dimensional alignment (offset and/or stride); B: BLOCK; C: CYCLIC; BC: BLOCK_CYCLIC distribution; 1D: one-dimensional; nD: multi-dimensional distribution.
Dynamic layout:	P: at procedure boundaries; L: at loop boundaries; A: at any point in progam.
Parallel stat.:	FORALL statement; INDEPENDENT directive; Other features, such as ON HOME, REDUCTION, ...
Code generation:	Library: Fortran/Fortran-90/C plus calls to a message-passing library; Low-Level: specific target machine code.

HPFC, Southampton shpf, PGI pghpf, VAST-HPF, ACE EXPERT HPF and DEC Fortran 90 HPF. Outside these implementations the feature which is missing most frequently is that of the BLOCK_CYCLIC distribution.

For dynamic distributions information on three features was requested, namely, redistribution at procedure or loop boundaries or at any point in the program. Only ADAPTOR and pghpf reported that all three distributions were possible in their system. In general most systems provided distribution at procedure boundaries only.

In the case of statements, information on the FORALL and INDEPENDENT statements as defined in HPF was requested. Several systems have included both features, namely, Vienna VFCS, ADAPTOR, PGI pghpf, VAST-HPF, Annai/PST, IBM XL HPF and DEC Fortran 90 HPF. Other features such as the ON clause or the REDUCTION clause have been included in some compilers.

The type of code generated by each of the systems is divided into two groups, namely,

1. systems which generate high level languages such as Fortran plus calls to message routines;
2. systems which generate low level code for particular target machines.

The first category was the most popular and reflects the use of preprocessor systems to implement code generation for HPF systems. Those systems generating low level specific code included Vienna VFCS, ACE EXPERT HPF, IBM XL HPF and DEC Fortran 90 HPF.

There are a larger number of preprocessor systems than systems generating specific low level code and the latter tend to be commercial systems.

5 Summary

The HPF 'standards' became publicly available in the spring of 1993. In the intervening years substantial progress has been made in finding solutions to many of the problems of implementing the language and data features defined in HPF. A wide range of systems is currently being implemented by various groups. In particular, in research environments preprocessor systems based on Fortran with calls to a message passing system have been developed to provide data parallel systems. In the commercial environment there is more emphasis on systems which generate low level code. Both approaches are contributing to the solution of the substantial compiler implementation problems which are inherent in the provision of data parallel language implementations, however, a disappointing feature of the systems that we have considered is the variance in the adherence to the HPF 'standards'.

Most of the developers have put a big effort in performing optimizations such as redundant communication elimination, message vectorization, collective communication, optimization of nearest neighbour shift communication and reductions. These optimizations reduce the data movement overhead in parallel programs making possible noticeable performance gains. Future developments for most of the developers and companies are based on the emerging HPF 2 proposal.

It is our intention to try and maintain the Web site as a repository for developments in this area and we would welcome input from those actively involved in this field in the pursuit of this task.

REFERENCES

[1] High Performance Fortran Forum, *Sci. Prog.*, vol. 2, no. 1–2, pp. 1–170, 1993.