110063 **N95-16478**

1994 Workshop on The Role of Computers in LaRC R&D

High Performance Fortran

Piyush Mehrotra

Institute for Computer Applications in Science and Engineering
pm@icase.edu

Introduction

Recently an international group of researchers from academia, industry and government labs formed the High Performance Fortran Forum aimed at providing an intermediate approach in which the user and the compiler share responsibility for exploiting parallelism. The main goal of the group has been to design a high-level set of standard extensions to Fortran called, High Performance Fortran (HPF), intended to exploit a wide variety of parallel architectures [2, 4].

A major performance issue of most parallel machines including distributed memory machines and non-uniform access shared memory machines, is the locality of data. The HPF extensions allow the user to carefully control the distribution of data across the memories of the target machine. However, the computation code itself is written using a global name space independent of the distribution of the data. As HPF is targeted towards data parallel algorithms, it supports forall loops and array statements to specify the data parallelism. However, there are no explicit constructs for management of tasks or for communication of data. It is the compiler's responsibility to analyze the distribution annotations and generate parallel code, generally SPMD, inserting synchronization where required by the computation. Thus, using this approach the programmer can focus on high-level algorithmic and performance critical issues such as load balance while allowing the compiler system to deal with the complex low-level machine specific details.

The HPF design is based on language research done by several groups, in particular, Kali [5, 6], Vienna Fortran [1, 7] and Fortran D [3], the first two of these were partially developed at ICASE.

HPF Overview

High Performance Fortran is a set of extensions for Fortran 90 designed to allow specification of data parallel algorithms. The programmer annotates the program with distribution directives to specify the desired layout of data. The underlying programming model provides a global name space and a single thread of control. Explicitly parallel constructs allow the expression of fairly controlled forms of parallelism, in particular data parallelism. Thus, the code is specified in high level portable manner with no explicit tasking or communication statements. The goal is to allow architecture specific compilers to generate efficient code for a wide variety of architectures including SIMD, MIMD shared and distributed memory machines.

Fortran 90 was used a base for HPF extensions for two reasons. First, a large percentage of scientific codes are still written in Fortran (Fortran 77 that is) providing programmers using HPF with a familiar base. Second, the array operations as defined for Fortran 90 make it eminently suitable for data parallel algorithms.

Features of High Performance Fortran

In this subsection we provide a brief overview of the new features defined by HPF.

- *Data mapping directives:* HPF provides an extensive set of directives to specify the distribution and alignment of arrays. The distribution directives can be used to specify the layout of data arrays on an underlying set of abstract processors. The alignment directives allow the arrays to be aligned so that specified elements are placed on the same abstract processors.

- *Data parallel execution features:* The **FORALL** statement and construct and the **INDEPENDENT** directive can be used to specify data parallel code. The concept of *pure* procedures callable from parallel constructs has also been defined.

- *New intrinsic and library functions:* HPF provides a set of new intrinsic functions including system functions to inquire about the underlying hardware, mapping inquiry functions to inquire about the distribution of the data structures and a few computational intrinsic functions. A set of new library routines have also been defined so as to provide a standard interface for highly useful parallel operations such as reduction functions, combining scatter functions, prefix and suffix functions, and sorting functions.

- *Extrinsic procedures:* HPF is well suited for data parallel programming. However, in order to accommodate other programming paradigms, HPF provides *extrinsic* procedures. These define an explicit interface and allow codes expressed using a different paradigm, such as an explicit message passing routine, to be called from an HPF program.

Full details of the language can be found in the HPF Language Specification document [2] which is also available via anonymous ftp from public/HPFF/draft at `titan.cs.rice.edu`.

There is a second round of meetings of the *High Performance Fortran Forum* being currently held in Chicago to consider clarifications of HPF 1 and to chart out requirements for future extensions to HPF. Further information about these meetings and HPF in general can be found on Mosaic through the URL *http://www.erc.msstate.edu/hpff/home.html*

References

[1] B. Chapman, P. Mehrotra, and H. Zima. Programming in Vienna Fortran. *Scientific Programming*, 1(1):31–50, 1992.

[2] High Performance Fortran Forum. High Performance Fortran Language Specification Version 1.0. *Scientific Programming*, 2((1-2)):1–170, Spring and Summer 1993.

[3] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C. Tseng, and M. Wu. Fortran D language specification. Department of Computer Science Rice COMP TR90079, Rice University, March 1991.

[4] C. Koelbel, D. Loveman, R Schreiber, G. Steele, and M. Zosel. *The High Performance Fortran Handbook*. The MIT Press, 1994.

[5] P. Mehrotra. Programming parallel architectures: The BLAZE family of languages. In *Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing*, pages 289–299, December 1988.

[6] P. Mehrotra and J. Van Rosendale. Programming distributed memory architectures using Kali. In A. Nicolau, D. Gelernter, T. Gross, and D. Padua, editors, *Advances in Languages and Compilers for Parallel Processing*, pages 364–384. Pitman/MIT-Press, 1991.

[7] H. Zima, P. Brezany, B. Chapman, P. Mehrotra, and A. Schwald. Vienna Fortran – a language specification. Internal Report 21, ICASE, Hampton, VA, March 1992.

HIGH PERFORMANCE FORTRAN

1994 Workshop on The Role of Computers in LaRC R&D

June 16, 1994

548 [illegible]

Piyush Mehrotra

Institute for Computer Applications in Science and Engineering

pm@icase.edu

High Performance Fortran Forum

Primary goal:

to define a set of "standard" *Fortran 90* extensions for *data parallel codes* to extract top *performance* on **SIMD and MIMD** non-uniform memory access (NUMA) machines.

Most scientific codes, including codes of interest to NASA researchers, exhibit data parallelism:

Data parallelism: *similar operations performed in parallel on different parts of structured data* (represented by arrays).

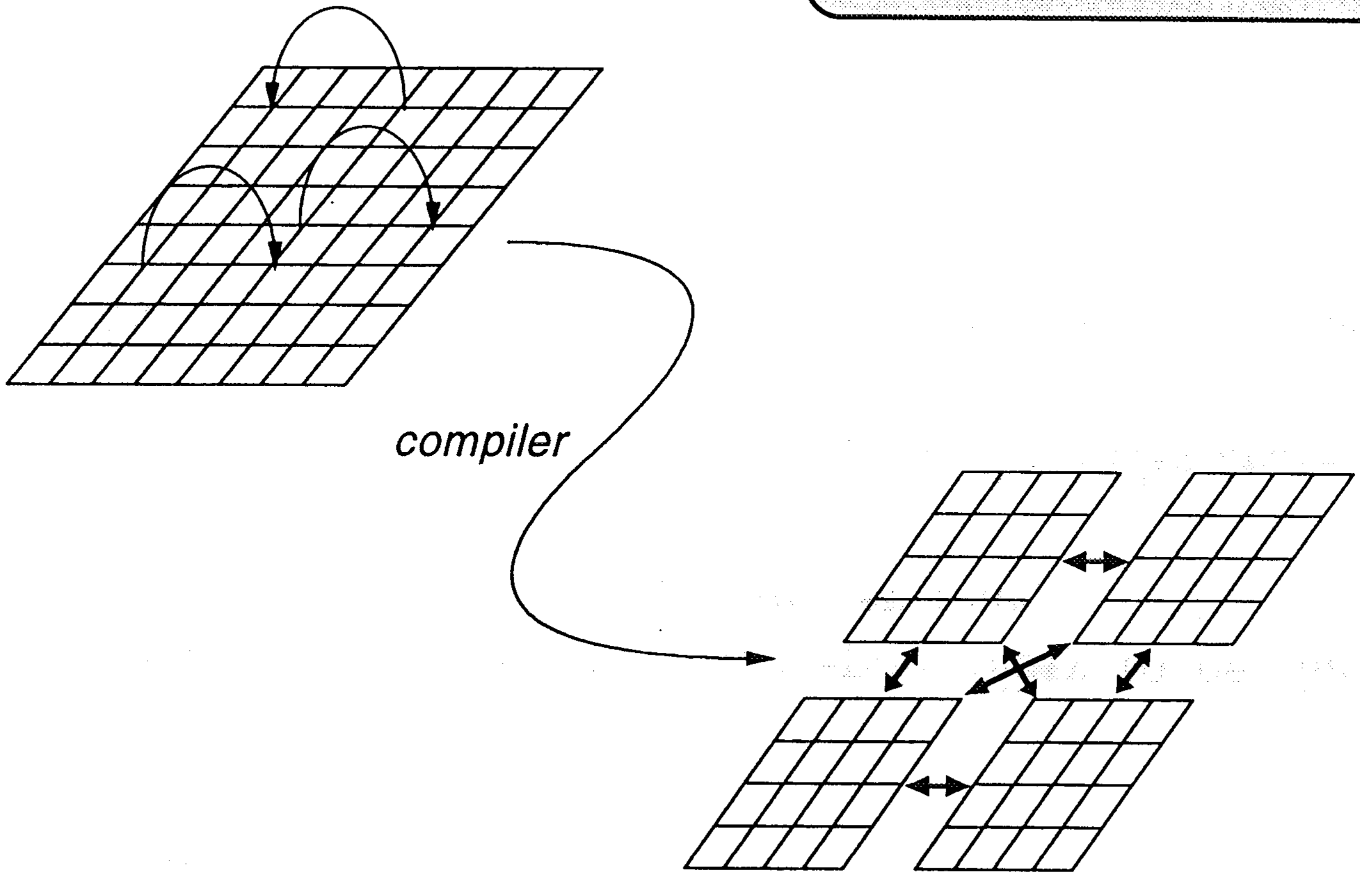
Motivation

- Porting codes to fragmented memory machines requires:
  - distribution of data, and
  - insertion of communication.
- Explicitly parallel approach allows effective exploitation of underlying architecture.
- Disadvantages:
  - user has to cope with low level details
  - programs are difficult to design and debug
  - hardwires distribution choices

Approach:

Allow programmers to use a global name (index) space while controlling the distribution of code and data.

User code - parallel operations on shared data.



Generated code - SPMD code with embedded communication.

Approach

- Exploiting parallelism is a responsibility to be shared between the user and the compiler/runtime system.

- User controls data layout using a *global index space.*

- There are no explicit statements for process management, synchronization or communication.

- Compiler produces SPMD code with embedded statements for process interactions.

Previous work

IVTRAN - data distributions in an SIMD language for ILLIAC IV (Mass. Comp. Assoc., 1973)

Kali - user specifiable data distributions for distributed memory machines (ICASE, 1988)

. . . - *Dino, DPC, Superb, Crystal, ID Noveau, CM Fortran,* . . .

FortranD - allowed alignment to decompositions (Rice Univ., 1991)

Vienna Fortran - comprehensive set of extensions for Fortran (Univ. of Vienna/ICASE, 1992)

HPF Features

- Fortran 90 as base language - new data structures, array operations, modules, etc.
- User directives for data layout
- Data parallel constructs: forall statement and construct
- New intrinsics/library functions
- Extrinsic procedures
- *No* new I/O features
- Restrictions on sequence and storage association
- HPF subset

Data Layout Rationale

Memory of parallel architectures is generally fragmented. Thus:

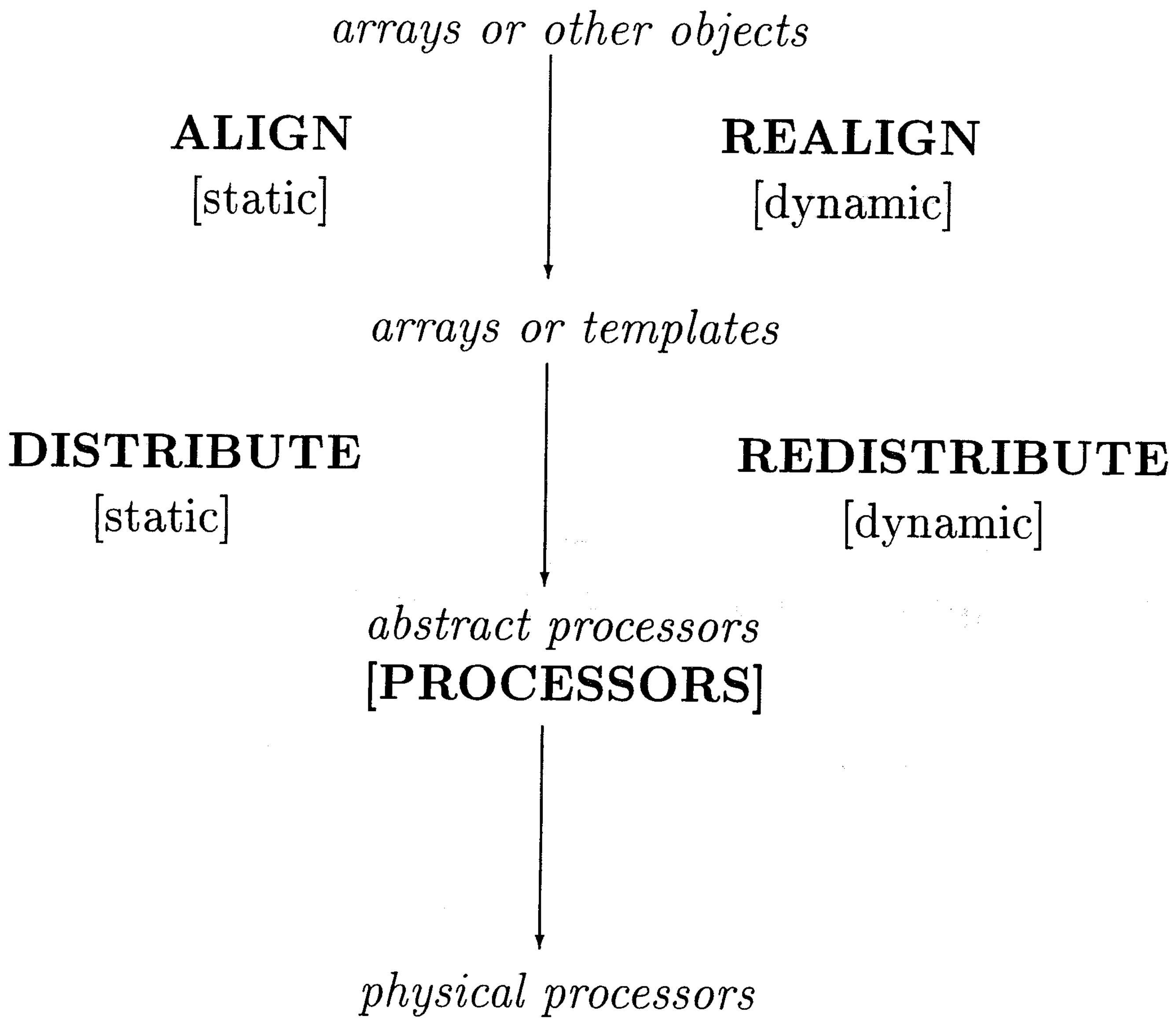
- if unrelated data is properly *distributed*, i.e., in different memories, operations can be done in parallel:

```
DO i = 1, N
   A(i) = A(i) + 1
ENDDO
```

- if related data is properly *aligned*, i.e., in the same memory, communication costs can be reduced:

```
DO i = 1, N
   A(i) = B(i) + C(i+1)
ENDDO
```

HPF Mapping Model



Example - Jacobi

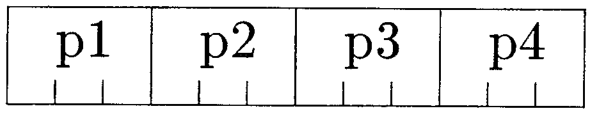
```
!HPF$  PROCESSORS  p(number_of_processors())
       REAL  u(n, n),  f(n, n)
!HPF$  ALIGN  WITH  u :: f
!HPF$  DISTRIBUTE  (*, BLOCK)  ONTO  p :: u
           ...
       FORALL  (i = 2:n-1, j = 2:n-1)
            u(i, j) = 025*( u(i+1, j) + u(i-1, j)
     +                      u(i, j+1) + u(i, j-1)) - f(i, j)
       END FORALL
```

- *p*: a one-dimensional array of abstract processors
- array *f* identically aligned with array *u*
- columns of array *u* distributed blockwise
- code is written using a *global index space*
- compiler introduces required communication

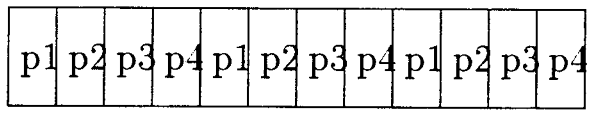
Distribution of Data

A 12 element array on 4 processors

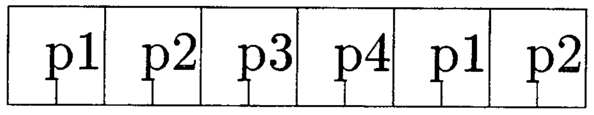
- **BLOCK :**



- **CYCLIC :**



- **CYCLIC** (2):



Example - Jacobi 2D distribution

```
!HPF$   PROCESSORS  p(4,4)
        REAL  u(n, n),  f(n, n)
!HPF$   ALIGN  WITH  u :: f
!HPF$   DISTRIBUTE (BLOCK , BLOCK) ONTO p :: u

         FORALL (i = 2:n-1, j  = 2:n-1)
              u(i, j) = 025*( u(i+1, j) + u(i-1, j)
       +                      u(i, j+1) + u(i, j-1)) - f(i, j)
         END FORALL
```

Only the distribution directives are changed - the code remains the same.

Conclusions

- A large percentage of scientific codes, including codes of interest to NASA researchers, are data parallel in nature.
- HPF provides a high level approach for porting data parallel algorithms to both shared and non-shared memory machines.
  - Add distribution directives to specify layout.
  - Use array statements and forall constructs to specify data parallelism.

- Porting of existing Fortran 77 codes may require extensive rewrites to eliminate dependence on storage and sequence association.
- Extrinsic procedures allow escape from the HPF model.

Further Information:

- Compilers with basic capabilities should be available soon.

Announced Products	**Announced Efforts**	**Interested**
Applied Parallel Research	TMC	Cray
Kuck and Associates	IBM	HP
Portland Group	nCube	Fujitsu
Intel	NEC	Silicon Graphics
Meiko	Maspar	Hitachi
Digital	Convex	Sun
	. . .	

- HPF document available by anonymous ftp from `titan.cs.rice.edu` in /public/HPFF/draft. Also as *Scientific Programming*, Vol. 2, Nos. 1-2, pages 1-170, Spring and Summer 1993.
- Mosaic URL: *http://www.erc.msstate.edu/hpff/home.html*
- Another round of HPFF meetings being held in Chicago currently for HPF 1 clarification and HPF 2 requirements.

SESSION 11 Advanced Topics

Chaired by

Susan Voigt