HPFF Meeting Notes for the March 9-10, 1992 Meeting

High Performance Fortran Forum

CRPC-TR93311
May 1993

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

HPFF Meeting Notes
March 9-10, 1992

Notes taken by Chuck Koelbel, with input from J. Ramanujam

Executive Summary:

This was the first working meeting of the High Performance Fortran Forum. As a first meeting, the most concrete results were procedural decisions, most notably rules for future meetings and working groups dedicated to specific language features. A list of goals was also adopted (albeit less formally).

Rules for future meetings

Attendance at meetings is open to all interested parties. Each organization (company, school, etc.) is limited to two representatives, however, in order to keep the meeting size manageable.

Every organization may have one voting representative. The organization should name a single individual for this task; if that person cannot attend, however, another can cast votes instead.

To be eligible to vote, a representative must have attended 2 of the last 3 meetings, counting the current meeting. (That is, if you miss 2 consecutive meetings you can't cast a vote.) Eligibility is decided based on organization. This rule, of course, would not be applied before the third meeting.

Future meetings will be 1.5 day gatherings. The first day will start around 9:00 and work into the evening, while the second day will end at about noon. In general, meetings during cold months will be held in Dallas, and meetings in hot months will be held in Chicago.

Working groups

Working groups were formed to discuss specific parts of High Performance Fortran. These groups will meet between full meetings to develop proposals for presentation at full meetings. The current set of working groups is

1. Fortran 90 (i.e. How much of Fortran 90 will be required by HPF?)
   Convener - Mary Zosel
2. Computational Model of ALIGN/DISTRIBUTE statements
   Convener - Guy Steele
3. Subroutine interfaces & dynamic redistribution
   Convener - Joel Williamson
4. FORALL, DOALL, and ON clause
   Convener - Chuck Koelbel
5. I/O and odds & ends
   No convener

It is expected that much of the technical discussion and work will happen in the group meetings. The group meetings will be open to any interested party; contact the convener for more information.

Goals

Language extensions and feature selection for Fortran supporting:

- Data parallel programming (defined as single thread, global name space, and loosely synchronous)
- Top performance on MIMD and SIMD computers with non-uniform memory access costs
  - But should not impede performance on other machines
  - Code tuning for various architectures
- Minimal deviation from other standards
- Minimal direct conflicts with Fortran 77 and 90
  - If possible, no direct conflicts
- Open interfaces to other languages & programming styles
- Compiler availability possible in near term (1993) with demonstrated performance on HPF test suite
- Produce validation criteria
- Provide input to future standards activities for Fortran and C
- Maximal simplicity
- Involvement (input) of high performance computing community

        Widely distributed language drafts
    Present final proposal in November '92
        Final draft in January '93
    Leave an evolutionary path for research

Future Meeting Dates -

April 23-24 Dallas
June 8-9 Chicago
July 23-24 Washington, DC
September 10-11 Chicago (tentative)

Detailed meeting notes:

Attendees

Ralph Brickner (Los Alamos National Laboratory), Alok Choudhary (Syracuse University), Marina Chen (Yale University), Peter Highnam (Schlumberger), Barry Keane (nCUBE), Ken Kennedy (Rice University), Chuck Koelbel (Rice University), Bob Knighten (Intel Supercomputer Systems Division), John Levesque (Applied Parallel Research), David Loveman (DEC Massively Parallel Systems Group), Tom MacDonald (Cray Research, Inc.), Piyush Mehrotra (ICASE), Andy Meltzer(Cray Research, Inc.), Rex Page (Amoco Production Research), J. Ramanujam (Louisiana State University), David Reese (Alliant Computer Systems Corp.), P. Sadayappan (The Ohio State University), Rony Sawdayi (Applied Parallel Research), Randy Scarborough (IBM Scientific Center, Palo Alto), Rob Schreiber (Research Institute for Advanced Computer Science), Vince Schuster (Portland Group), Henk Sips(ITI-TNO), Marc Snir (IBM T. J. Watson Research Center), Guy Steele(Thinking Machines Corporation), Swift Richard (MasPar Computer Corporation), Clemens-August Thole (GMD), Douglas Walls (Sun Microsystems), Joel Williamson (CONVEX Computer Corporation), Mary Zosel (Lawrence Livermore National Laboratory)

Some memorable self-introductions

Bob Knighten - “Our customers tell us we are going to be implementing HPF”
Rob Schreiber - “I'm the only applications user here”
Peter Highnam - “I'm also the only applications programmer here”

Principles

The meeting started with a presentation by Chuck Koelbel outlining issues in the areas of HPFF operating principles and procedures. The following copies his slides; comments are given later.

- Principles
- [original slide by Ken Kennedy]
    - Maximum Compatibility
    - Minimum Requirements
    - Machine Independence
    - Consider near-term first
    - Leave an evolutionary path
- Principles Questions (2 slides)
    - Implementation ease vs. Advanced features?
    - Existing practice vs. advanced features?
    - Hooks for future advanced features?
    - Portability vs. low-level tuning?
    - What about implementation efficiency?
    - What is the scope of HPFF?
        - What architectures?
            - SIMD? MIMD? Shared-memory? Distributed-memory? Vector processors? RISC processors? VLIW processors?
            - Caches?
            - Input/Output?
        - What problems?
            - Data-parallel, Embarrassingly parallel
            - Loosely synchronous
            - Regular? Irregular?
    - Standards Issues
        - Should HPFF be an ANSI standard? [NO]
        - Relationship to existing standards
            - Fortran 77?
            - Fortran 90?
            - X3H5 (formerly PCF)?
            - Vendor extensions?
- Procedures
- [original slide by Ken Kennedy]
    - One-year project
    - People committed to multiple meetings
    - One person per organization
    - Meeting costs
        - Materials fee for copying

        Meeting fee for actual attendees
    Test suite built separately
Layered Support
    1. Full Support
        [Must have this to call yourself HPF]
    2. Partial Support
        [Not required, but reserved against other use;
            upgrade to full support as they evolve]
    3. Machine-dependent Support
        [includes features primarily useful for one
            machine, vendor extensions]
    4. Hooks
        [Reserved syntax for advanced features]

Discussion of Standards

Clemens Thole added the now-forming standards effort in message-passing to the list of standards to consider; Ken Kennedy compared message-passing to assembly language. Henk Sips brought up the need for multi-paradigm support; Piyush Mehrotra pointed out the need to be careful about this interface.

Discussion of Procedures:

Ken Kennedy outlined what went wrong with PCF (organizationally).

An open process was seen as very important. Some conflict was felt between having an open process and limiting meeting attendance to a workable size. Remedies that were supported included: forming working groups for language sub-parts, carrying on electronic discussions, and wide distribution of documents.

Mary Zosel spoke strongly in favor of having widely distributed minutes, explaining decisions taken. This would cut down on rehashing dead points. Chuck promised to widely disseminate drafts and minutes electronically. Alok Choudhary offered to disseminate news via the NPAC newsletter. Ken suggested a workshop, tutorial, or birds-of-a-feather session at Supercomputing '92. In summary, it doesn't look like the draft will be hard to get.

After some discussion, it was decided that each company would receive one vote. The company would have to name one person to be its voting representative, thus minimizing turnover at meetings. Up to 2 representatives of a company could attend each meeting (including the voting delegate).

In order to vote, the representative must have attended 2 of the previous 3 meetings, counting the current one (this rule would be applied to companies, not individuals). This is the same as the ANSI committee rules. This rule would be applied uniformly, starting with the third meeting; one consequence of this is that an attendee could not vote at his/her first meeting.

Meeting locations will be in Dallas during the cold months, in Chicago in summer. This is based on airport accessibility. (Actual planning later modified this rule, because of multiple conflicts on one meeting.)

A procedural mechanism for rapidly converging to a straw man document was discussed. Temporarily “freezing” parts of the language from further discussion was the preferred method.

Discussion of Layering

There was a lively discussion here. However, this proposal lost support as the meeting wore on. See the discussion of Fortran 90 below for the final status of support levels.

Goals

After a break, Ken Kennedy started discussion to define HPFF's goals. The final list was

- Language extensions and feature selection for Fortran supporting data parallel programming (defined as single thread, global name space, and loosely synchronous)
- Top performance on MIMD and SIMD computers with non-uniform memory access costs
    - But should not impede performance on other machines
    - Code tuning for various architectures
- Minimal deviation from other standards
- Minimal direct conflicts with Fortran 77 and 90
    - If possible, no direct conflicts
- Open interfaces to other languages & programming styles
- Compiler availability possible in near term (1993) with demonstrated performance on HPF test suite
- Produce validation criteria
- Provide input to future standards activities for Fortran and C
- Maximal simplicity
- Involvement (input) of high performance computing community
    - Widely distributed language drafts
- Present final proposal in November '92
    - Final draft in January '93
- Leave an evolutionary path for research

This was achieved after nearly 2 hours of discussion. Some of the points made are given below.

David Loveman gave the first proposed goal: Create an industry-wide standard language portable from workstations to massively parallel supercomputers, able to express algorithms needed to achieve high performance on specific such architectures.

Guy Steele defined the purpose of the HPFF extensions as a language in which the programmer can express things that the compiler can map onto machines with parallel CPUs, parallel memory, or both. This is done by giving more information to compiler

Ken Kennedy defined shared memory as all memory having the same access time, as in the Tera machine, while shared name-space means all memory can be accessed by the same mechanism possibly with different access times, like essentially any other “shared memory” machine. Henk Sips added the requirement that access cost should be related to locality, and not simply be a random variable.

Several attendees stated the goal of allowing the programmer to reprogram his or her application only once for multiple machines.

Marc Snir divided this into 2 goals: high performance on a particular machine and good support for porting between platforms. Mary Zosel replied that there is no magic bullet for giving high performance everywhere.

Clemens Thole suggested HPF be not a new language, but rather new extensions and support. Peter Highnam advocated directives rather than new language features. David Loveman pointed out that selecting unsupported Fortran 77 features (everybody's favorite example: sequence association) could be equally important.

Rob Schreiber gave the above definition of data parallel. He asked whether this was all there was in the world of parallelism, and received a resounding “no”. It was agreed that HPF could not be all things to all people, however. Snir later pointed out that external interfaces could give a great deal of added capability.

Clemens Thole asked two questions: “Does HPF accept F77 and/or F90?” and “Does F77 accept HPF (similar for F90)?” This led to the minimal conflicts goal.

Clemens Thole stated that there were two types of programmers, those who insisted on using a particular programming style and those who would rewrite their codes. The group felt it was more important to support the second group, if the support could be made portable.

Marc Snir wanted to use “Don't stretch compiler technology” as a goal, meaning that major new compiler research should not be needed for the basic level of support. This was greeted with a chorus of “Oooooh”s from the audience. It was generally accepted that there should be regard for the difficulty of feasible implementations, however.

Barry Keane(?) suggested that we could demonstrate implementation with application codes in the near term to demonstrate feasibility. This suggested that Geoffrey Fox's test suite would be a good idea. Other responses suggested a correctness suite would also be needed.

Randy Scarborough made the point that a stable base for HPF is needed, and layers of support weakens that. Therefore, this sort of layering should not be a goal. Instead, HPFF should identify a single factor beyond Fortran 77 and 90 to lay claim to.

The group broke for lunch after this discussion.

Defining Features

Ken opened the session after lunch as a discussion of features that should and should not go into High Performance Fortran. David Loveman then presented a “straw person” proposal. His slides are presented below (although the best slide - a sort of Venn diagram of the Fortran standards - is too complex to reproduce here).

- What is Fortran?
  - Old standards
    - F77
      - “Bad” stuff
      - “Non-politically correct” stuff
      - “Good” stuff
    - MIL-STD-1753
    - SPECTRAN and other de facto standards
    - Vendor-specific extensions
  - New standards
    - Fortran D
      - “important” stuff
      - “not-so-important” stuff
  - Parallel extensions and directives
    - Fortran D
    - Current compilers from vendors
    - Other good ideas
- Language Components
  - [trust me, it's a great slide]
- Outline of Major Language Areas
- Orthogonal problems
  - Standard language

        - statements, expressions
        - data types
        - I/O specifiers
        - intrinsics (especially Fortran 90 intrinsics)
    - Extensions
        - statements, expressions
        - data types
        - I/O specifiers
        - intrinsics (especially Fortran 90 intrinsics)
        - directives
- Proposed Required Fortran 90 Features
    - F77 standard conformant except sequence & storage association
    - MIL-STD-1753 conformance (DO WHILE, END DO, bit operations, etc.)
    - Arithmetic array features
        - array sections, triplet notation, vector-valued subscripts
        - array constructors
        - array arithmetic operations
        - WHERE and block WHERE
        - array-valued external functions
        - automatic arrays
        - allocatable arrays, allocate, deallocate
        - assumed-shape arrays
    - Control statements: CASE, DO forever, CYCLE, EXIT
    - Declarations
        - object oriented syntax
        - attribute specification statements
    - Procedure features
        - INTERFACE blocks
        - optional arguments
        - keyword parameter passage
    - Syntax improvements (long identifier names, etc.)
- Proposed Non-required Fortran 90 features
    - sequence and storage association
    - assumed-size arrays
    - free form source input
    - CHARACTER arrays
    - recursion
    - Fortran 90 pointers
    - modules and USE statement
    - user derived types (RECORD)

- internal procedures
- generic operators

As one might expect, there was a LOT of discussion on this. The result of this was a list of issues for further discussion. These were referred to the smaller working groups as mentioned below.

- Fortran 77 vs. Fortran 90
  - Should we require some F90 extensions to F77 to get this to work
- What F90 features do we need?
- What F77 features do we deprecate?
- Boundary between “full” & “simple” distribution features?
  - static vs. dynamic
  - subroutine-oriented features
  - alignment
  - executable (conditional) distributions (i.e. unknown at compile time)
  - decomposition / processor mapping style?
  - distribution query features
  - user distributions (invertability)
- FORALL & ON clause & owner computes?
- Are distributions directives or commands?
- What are the defaults for distribution?
- Should we extend Fortran I/O statements?
- Should we have local blocks & subroutines?
- Should we have control parallelism?
  - interface with PCF parallel sections
- Local data?
- Multi-statement FORALL?
- Explicit processor control?
- Should we define frills (e.g. extra intrinsics)?
- Number of processors intrinsics & its friends and enemies?

After creating this list, some straw polls were taken. Note that there were many abstentions on some votes; these were taken as signs that the topic was very unsettled. (All “requirements” were requirements for the minimal level of HPF, assuming some multi-layer scheme were adopted.)

Poll Results:

- Should HPF require (some) array syntax? yes 22, no 5
- Should HPF require dynamic storage allocation? yes 22, no 3
- Should HPF require interface blocks? yes 10, no 3
- Should HPF require full Fortran 90? yes 0

Two interesting comments came out of this series of votes:

Randy Scarborough: HPFF should require all of F90 or none of it, not just parts

Guy Steele (replying to Clemens Thole): Note: This doesn't rule out Fortran 77 as a useful subset of HPF!

Should HPF require executable (conditional) distributions? yes 10, no 7

Should HPF require single statement FORALL? 22 yes, 0 no

Should HPF require some kind of multi-statement FORALL? 17 yes, 2 no

Should HPF also require a parallel DO along lines of PCF (i.e. no data sharing between iterations)? yes 3 no 10

Should HPF require control parallelism (parallel sections)? yes 2 no 13

Should HPF require local subroutines? yes 6 no 6

Special note: Guy Steele voted “no”, saying local sections were not portable to the CM-2

The issues were then divided into subsets. A working group will examine each subset between regular HPFF meetings and form recommendations. These will then be considered by the entire group at the next two full meetings.

1. Fortran 90
   - What are the minimal F90 features to require in HPF?
   - Fortran 77 vs. Fortran 90
   - What F90 features do we need?
   - What F77 features do we deprecate?
   - Consider storage association in particular
2. Model of align/distribute
   - What is the general model HPF should use for alignment, distribution, decompositions, and processor mappings?
   - Boundary between “full” & “simple” distribution features?
     - alignment
     - decomposition / processor mapping style?
     - user distributions (invertability)
   - FORALL & ON clause & owner computes?
   - Are distributions directives or commands?

- Consider storage association if needed for other decisions.

3. Subroutine interfaces & dynamic redistribution
   - Boundary between “full” & “simple” distribution features?
     - static vs. dynamic
     - subroutine-oriented features
     - executable (conditional) distributions (i.e. unknown at compile time)
     - distribution query features
4. FORALL, DOALL, and ON clause
   - Should we have local blocks & subroutines?
   - Should we have control parallelism?
     - interface with PCF parallel sections
   - Local data?
   - Multi-statement FORALL?
   - Explicit processor control?
   - SCAN intrinsics
5. I/O and odds & ends
   - Peter Highnam opined that parallel I/O is not needed, users can roll-their-own like they do now

The initial membership of each group is given below. “Conveners” are responsible for setting working group meeting times and places. Chuck Koelbel will set up the appropriate email lists at Rice; people can join as many lists as they want by sending him mail.

1. Fortran 90
   - Convener - Mary Zosel
   - Vince Schuster, John Levesque, Rex Page, David Loveman, Barry Keane [results to be discussed next meeting]
2. Model of align/distribute
   - Convener - Guy Steele
   - Chuck Koelbel, Marina Chen, Henk Sips, Alok Choudhary, P. Sadayappan, J. Ramanujam, Doug Walls, Andy Meltzer, Ralph Brickner, Peter Highnam, Joel Williamson, David Reese, Richard Swift, Piyush Mehrotra, David Loveman Tom MacDonald [results to be discussed next meeting]
3. Subroutine interfaces & dynamic redistribution
   - Convener - Joel Williamson
   - Marina Chen, P. Sadayappan, Randy Scarborough, Tom MacDonald, Mary Zosel, Rony ???, Rob

Schreiber, David Reese, Richard Swift , Piyush Mehrotra, David Loveman [results to be discussed at second meeting]

4. FORALL, DOALL, and ON clause
   - Convener - Chuck Koelbel
   - Randy Scarborough, Alok Choudhary, Guy Steele, Bob Knighten, Marc Snir, J. Ramanujam, Clemens Thole, Tom MacDonald, John Levesque, Andy Meltzer, Rob Schreiber, Rex Page, David Loveman [results to be discussed at second meeting]
5. I/O and odds & ends
   - No convener
   - Barry Keane, Alok Choudhary, Bob Knighten, Marc Snir, Peter Highnam, David Loveman

The last order of business before dinner was choosing the dates of the next several meetings. After some discussions of conflicts, the following were chosen:

- April 23-24 Dallas
  - Suffice to say that the weeks of April 16-17 and April 27-May 1 were full of conflicts for attendees. The suggestion was made and agreed that the Dallas meetings continue to be at the Harvey Suites. Unfortunately, we later discovered that the hotel could not accommodate us on this date. Ann Redelfs is now searching for another hotel.
- June 8-9 Chicago (no hotel yet)
- July 23-24 Washington, DC (no hotel yet, but possibly the ICS '92 conference hotel)
  - The meeting site was chosen for convenience with ICS '92, which several HPFF attendees will be at. The rest of July is a minefield of potential conflicts.
- September 10-11 Chicago (tentative)

After dinner, the group met again to discuss what features of Fortran 90 would be required in HPF. To put it mildly, the discussion was spirited. The first step was to list Fortran 90 features that should be considered as required, based on the results of the earlier poll. The set that finally emerged was

- Dynamic Storage Allocation
  - Automatic arrays on subroutine boundaries
  - Full ALLOCATE/DEALLOCATE statements
  - Pointers

        - Pointers to array sections
    - Interface Blocks
    - Array Language
        - Array section assignments
        - Array intrinsics
        - WHERE
        - Array section passing *
        - Assumed shape parameters *
        - Pointers to array sections *
        - Array-valued functions *
        - CHARACTER arrays

Features marked with an asterisk (*) are those that require dope vectors for efficient implementation; they were considered separately because of concern from some compiler implementors.

While this list was being built, a meta-discussion started (in fact, it started several times). The basic topic of discussion was how parallelism should be expressed in HPF. One group claimed that it should be expressed by Fortran 90 vector operations, while the other claimed that data distributions in conjunction with Fortran 77 statements sufficed. The debate was not definitively resolved.

Another meta discussion concerned the wisdom of choosing single features from Fortran 90, but not the entire language. Randy Scarborough and David Loveman argued that HPF should require all of Fortran 90 or none of it, on the grounds that half-hearted support of a standard was not acceptable. The danger of having large parts of the HPF specification devoted to explaining “required” and “optional” features was also mentioned. The argument against this point of view was that including large parts of Fortran 90 would delay implementation and acceptance of HPF. Ken Kennedy also stated that every Fortran 90 feature that HPF required would delay acceptance by some amount.

At this point several polls were taken to clarify the group's positions on the various features.

- Should HPF require things that require dope vectors? Yes 15, No 6
- Should HPF require the array language (not including pointers to arrays and array sections)? Yes 14, No 6
- Would there be interest in HPF if it did not require vectors? Yes 26, No 0

The apparent ambivalence of the last two votes led to much more discussion. The first topic of discussion involved portability. Guy Steele stated, to the general agreement of those present, that HPF's impact would be greater if performance were portable across vendors as well as code. The question was raised of whether this would be best accomplished by Fortran 90 array syntax or by extracting parallelism from Fortran 77 constructs. Ken Kennedy said both types of optimization would be selling points for compilers for some time to come; the competitive advantage would go to the vendors who could do both. Guy Steele accepted this, but called it a disaster if the only path to portable performance was to avoid the array language. John Levesque claimed that performance portability would not be achieved in the short term anyway, since vendors will divide into two groups based on their existing compiler technology (i.e. Thinking Machines will champion array constructs, while Cray will emphasize Fortran 77 optimization). Automatic vectorizers and Fortran 90 to Fortran 77 translators were mentioned as a possible solution, but Vince Schuster and others were dubious of their efficiency.

The question was raised of how long a Fortran 90 requirement would set vendors back in implementing HPF. Tom MacDonald of Cray estimated 1 year (and was the only vendor to make such an estimate), but also stated that Cray was committed to a Fortran 90 compiler in any case. David Loveman of DEC claimed that the delay for requiring other vendors to implement parallelizing Fortran 77 compilers would be about the same. Bob Knighten of Intel stated that their customers were very interested in arrays for the software engineering benefits. Marina Chen suggested having standards based on Fortran 77 and Fortran 90 and letting the market decide which was preferable.

Ken Kennedy asked the users present which was better, two languages or one nonportable language? All the users (Rob Schreiber, Rex Page, and Peter Highnam) agreed that Fortran 90 was a preferable language to work in, but differed on how long they would wait for it. They all would love to have excellent optimization of DO loops, since there is a lot of existing code. Mary Zosel stressed the need for a minimal subset useful for portability.

Perhaps the final word came from Ken Kennedy. He said that, regardless of what this committee decided to require from Fortran 90, vendors who didn't implement it all would advertise “HPFF extensions” in their Fortran 77 compiler. Vince Schuster opined that picking extensions to implement was a strategic marketing decision.

Before breaking into working groups, two final polls were taken.

Should HPF require automatic arrays and ALLOCATE/DEALLOCATE statements? Yes 19, No 1

Should HPF require the above and pointers? Yes 6, No 8

The next morning started with a short presentation by John Levesque trying to swing opinion away from requiring Fortran 90 in HPF.

His first slide showed vendors supporting Fortran 77 for parallel computing by a margin of 14-2 over Fortran 90:

1. Vendors who don't really efficiently support Fortran 90 in a parallel or vector implementation
    - Cray Research (vector)
    - NEC (vector)
    - Convex (vector)
    - Cray Research (shared memory)
    - NEC (shared memory)
    - Convex (shared memory)
    - Sequent
    - Intel
    - Meiko
    - Networked Workstations (several)
2. Vendors who don't really support F77 in a parallel or vector implementation
    - TMC
    - MasPar

This slide was roundly criticized from the floor. The main disagreements were that the “Fortran 77” companies listed were, for the most part, not doing the type of parallelization that HPF-77 would require. John then stated that his point was that many people wanted Fortran 77 over Fortran 90, which also provoked disagreement.

Levesque's next slide was titled “Why I don't like HPF on top of F90”

- Destroys important portability (Workstations, PCs)
- F90 is difficult to optimize for
    - vector register assign
    - cache utilization
- F77 DO is a good form for specifying a code block around which one can
    - strip mine for vector registers
    - tile for effective cache utilization
    - organize pre- and post-loop message passing

Rex Page made the point that we should realize Fortran 90 is not widely accepted. However, he stated one of his motives as forcing vendors forward - i.e. from Fortran 77 to Fortran 90. David Loveman urged the group to think about state of Fortran 90 compilers next year, when HPF would be adopted. Ken Kennedy, while admitting that there were compiler challenges, repeated his argument that vendors could use “HPF extensions” in their marketing descriptions. The users could be expected to vote with their feet and wallets.

The last slide was a modestly complicated graphic entitled “Ease of Efficient Compilation”, showing “Easy” and “Hard” implementations of Fortran 77-based and Fortran 90-based compilers on various machines. This was also somewhat controversial. To give a flavor of the discussion, Ken stated that he had a research project to examine all four of the connections John had drawn, including two that were marked “Easy”.

As there was obviously still a great deal of controversy, Ken Kennedy suggested putting off any decisions on Fortran 90 until the next meeting. Before then, he asked that all vendors (and others with access to many users) poll their customers on what they really want in the way of Fortran 90 support. It is important to do the right thing from their perspective, if HPF is to be successful.

Joel Williamson questioned why there was a debate on this topic at all. His suggestion was that we simply define a set of directives that could be applied to either Fortran dialect. Mary Zosel took this a step further and suggested that HPFF define a model for data distribution (and any other topics we wanted) and then describe language bindings for both Fortran 77 and Fortran 90.

Ken Kennedy then tried to start a new discussion on “What can we agree that we don't need in HPF?” Clemens Thole immediately suggested Fortran storage association. He then asked where storage association was actually needed, if HPF required Fortran 90 dynamic memory allocation. The major use was thought to be passing array sections without explicit syntax for the section in the caller. David Loveman and Piyush Mehrotra went on record for requiring explicit sections to be passed; somebody retorted that this would not be popular with users. The Cray proposal for canonical distributions was suggested as solving this, and some discussion followed. It quickly became obvious that in many cases the Cray proposal also disallowed storage association, but it did provide more compatibility than Fortran D. Piyush Mehrotra stated that similar constructs were available in Vienna Fortran. David Loveman noted that “middle ground” like this is often difficult to describe, but users may want some capability.

Among other Fortran 90 features Andy Meltzer suggested we should have defined meanings for any (interacting) F90 constructs, even if we don't require them in HPF. There was general agreement on this principle. The discussion then turned to yet another Fortran 77 versus Fortran 90 debate. Many of the arguments had been heard before. In this discussion, however, there was stronger support than previously for having two language bindings, one each for Fortran 77 and 90. It was decided to defer final action on this question until the next meeting. The Fortran 90 group, until that decision was made, was asked to isolate those aspects of F90 impacted by data distributions.

Before adjourning, a final poll was taken.

Should HPF define distributions + two language bindings (F77 & F90)? 15

Should HPF define a single language? 5

Abstained: 5

It was noted that this was much different from the poll results the night before.