High Performance Fortran: Achievements, Problems, Prospects

Ken Kennedy
Center for Research on Parallel Computation
Rice University

1. Achievements
   Implementations and usage

2. Problems
   Performance, missing features, tools, public perception

3. Prospects
   Will HPF survive?
   What must we do to improve public perception?
Current State

• Good news
  - Compilers are now widely available
    APR, DEC, IBM, PSR, PGI, Japanese vendors
    coverage of every target machine

• Concerns
  - Language not widely used
    good compilers slow to emerge
  - Vendors
    language implementability
    efficiency
  - Users
    coverage of application needs
    efficiency
Why Fortran Has Succeeded

- **Focus (HPF: yes)**
  - on scientific and engineering calculations
  - high-level data structure: multidimensional array

- **Portability (HPF: yes, with caveats)**
  - available on every viable machine

- **Efficiency (HPF: not quite yet)**
  - optimizing compiler a prerequisite

- **Software Base (HPF: not yet, but improving)**
  - libraries and applications

- **Educational Base (HPF: not yet)**
  - widely taught in science and engineering departments
HPF Commercial Interest

- Announced HPF Products
  - ACE
  - Applied Parallel Research
  - CDAC
  - Cray Research
  - Digital Equipment
  - EPC
  - Fujitsu
  - Hitachi
  - HP
  - IBM
  - Intel
  - Meiko
  - Motorola
  - NA Software
  - NEC
  - Pacific Sierra Research
  - Parsytec
  - Portland Group
  - Sun
  - Transtech

- Announced HPF Efforts
  - Lahey
  - NAG

- Interested
  - SGI
  - Tera
HPF Usage

- Installations
  - PGI reports over 125 site licenses
  - NCSA: PGI on SGI Origin is migration platform for CM-5

- Benchmarks
  - NAS Benchmarks: PGI compiler within factor of 2 (or better) of MPI
  - Tensrus Tensor Product Benchmark: 85 GF on 256 processor T3E

- Real Applications
  - NOAA, Princeton: Modular Ocean Model (100K lines)
  - Amoco: Falcon Reservoir Model (20K lines)
  - MATRA BAe Dynamics: AEROLOG (10K lines)
  - Quetzal: EPIC Crash Model (125K lines)
  - NCSA, UIUC: Riemann TVD and ENO Flow Code (50K lines)
  - U New Mexico: DFT Density Function Theory Code (6K lines)
  - CEA-DI CISI, Grenoble: Parallel Poisson Solver (7.7K lines) -- 23 GF on 128-processor T3E
  - MIT Earth Atmospheric and Planetary Science: Eulerian Ocean Model (3K lines)
  - Delft Univ TNO: Wish3D CFD code (125K lines, in progress)
  - U Houston: N-body Simulation (4K lines)
  - UIUC, Civil Eng: Nonlinear Multigrid Code (1K lines)
  - CRS4: GeoComp Seismic Migration Code
HPF Problems

- Compilers slow to mature
  - Fortran 90 features supported inconsistently
  - compilation for highest efficiency complex
  - initially, efficiency of object programs unsatisfactory
  - early users became discouraged, then negative (see below)
  - need to support emerging architectures

- Needed features missing
  - parallel math libraries
  - support for irregular problems
  - task parallelism
  - high performance input/output

- Programming support tools not widely available
  - debugging and performance tuning

- Public perception negative
  - HPF viewed as a failure, particularly by comparison with MPI
HPF Enhancement Activities

- HPF 2 Language Standard
  - Core Language: emphasis on implementable performance redistribution and realignment moved
  - Approved Extensions: future features
    support for irregular distributions, task parallelism, ON clause
    new extrinsic modules and interoperability components

- Compiler Enhancement
  - commercial compilers continue to improve
  - research on compilation (e.g., dHPF) has made impressive strides

- Library Development
  - effort to build interfaces to scientific libraries
    linear algebra (Dongarra) and FFT (Johnsson)
    definition effort underway, implementation funding sought

- HPF User Group
  - First meeting: Albuquerque, February 1997 — User Survey
<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>4.72</td>
<td>Full Fortran 90 Support</td>
</tr>
<tr>
<td>28</td>
<td>4.71</td>
<td>Tools</td>
</tr>
<tr>
<td>30</td>
<td>4.23</td>
<td>Math libraries</td>
</tr>
<tr>
<td>27</td>
<td>4.07</td>
<td>HPF_LOCAL</td>
</tr>
<tr>
<td>26</td>
<td>4.00</td>
<td>Full, efficient HPF 2.0 core</td>
</tr>
<tr>
<td>27</td>
<td>3.96</td>
<td>Efficient extrinsic interface</td>
</tr>
<tr>
<td>29</td>
<td>3.93</td>
<td>Efficient HPF Library</td>
</tr>
</tbody>
</table>
## User Survey Results 2: Desirable

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>3.55</td>
<td>Parallel I/O libraries</td>
</tr>
<tr>
<td>14*</td>
<td>3.5</td>
<td>F77_LOCAL</td>
</tr>
<tr>
<td>24</td>
<td>3.41</td>
<td>HPF_SERIAL</td>
</tr>
<tr>
<td>27</td>
<td>3.37</td>
<td>DO loop parallelization</td>
</tr>
<tr>
<td>25</td>
<td>3.32</td>
<td>Distribution of ptr-based data structures</td>
</tr>
<tr>
<td>25</td>
<td>3.24</td>
<td>Generalized BLOCK distribution</td>
</tr>
<tr>
<td>21</td>
<td>3.10</td>
<td>INHERIT</td>
</tr>
<tr>
<td>24</td>
<td>3.0</td>
<td>Task parallelism</td>
</tr>
<tr>
<td>25</td>
<td>3.0</td>
<td>Asynchronous I/O</td>
</tr>
<tr>
<td>18</td>
<td>2.94</td>
<td>ON</td>
</tr>
<tr>
<td>18</td>
<td>2.89</td>
<td>RESIDENT</td>
</tr>
<tr>
<td>24</td>
<td>2.83</td>
<td>CYCLIC(k) distributions</td>
</tr>
<tr>
<td>25</td>
<td>2.72</td>
<td>C interoperability</td>
</tr>
</tbody>
</table>
### User Survey Results 3: Not Needed?

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>2.38</td>
<td>INDIRECT distribution</td>
</tr>
<tr>
<td>25</td>
<td>2.36</td>
<td>Dynamic features (e.g., redistribution)</td>
</tr>
<tr>
<td>21</td>
<td>1.43</td>
<td>HPF_CRAFT</td>
</tr>
</tbody>
</table>

**Other Potential Features (not discussed in 1997)**

- Interoperability with OpenMP
- Interoperability with JAVA
Questions 1: Target Platforms

- Will HPF work well on distributed shared memory (DSM)?
  - locality is clearly an issue
  - so is minimization of off-processor memory references
  - granularity of off-processor accesses may be a problem

- Does HPF port to networks of shared-memory parallel workstations?
  - two level hierarchy
  - are new language features necessary?

- Is HPF useful for shared memory?
  - what about the SUN and Tera machines?
  - does it help with locality in uniprocessors?

- Can HPF be extended to provide useful support for heterogeneous distributed systems?
  - need explicit support for load matching
Questions 2: Competitive Paradigms

• Has MPI eliminated the need for HPF?
  - MPI is clearly machine-independent
  - MPI has achieved widespread acceptance

• Has C++ (or Java) eliminated the need for HPF?
  - majority of US grand challenges are being coded in C++
  - HPF and Fortran 90 lack support for object-oriented programming

• Has OpenMP eliminated the need for HPF?
  - are DSM (hardware or software) + OpenMP enough?
  - distributions still a significant issue

• Has Fortran 90 eliminated the need for HPF?
  - is data distribution essential?
Questions 3: Software Support

• What tools do we need to support HPF programming?
  - performance tuning without forcing the user to learn message-passing

• Can we improve the performance of node memory hierarchies?
  - performance on single nodes significantly degraded with HPF

• How does HPF relate to automatic parallelization?
  - does HPF assist implicit parallelization?
  - will explicit features like ON help?

• Do we really need the data mapping directives—can compilers map the data automatically?
  - can a tool do better by using large amounts of computation?

• Will interprocedural compilers simplify the language design and substantially improve performance?
  - much of the language complexity resides in features and machinery for handling subroutine calls
Public Perception: What Can Be Done?

- **Document Application Successes**
  - Contribute to the HPF Application Survey
  - Publicize successful applications

- **Improve Compilers and Tools**
  - Focus on reliability and efficiency of HPF 2.0 core

- **Implement New Features**
  - Focus on those that add key functionality with high efficiency
    - Generalized BLOCK
    - Distribution of pointer-based data structures

- **Add Library Support**
  - Efficient implementation of the HPF Library
  - Math and I/O libraries

- **Ensure Interoperability**
  - C, OpenMP
Summary

• HPF compilers are maturing
  - language available on every parallel system
  - impressive performance has been achieved on some applications

• HPF has significant problems
  - performance
  - needed features: irregular support, task parallelism, I/O
  - need for tools, libraries
  - public perception: we need success stories!

• HPF 2.0 addresses some of these problems
  - focus on implementable language for performance
  - advanced features as Approved Extensions
  - implementations slow to emerge

• Questions concerning language viability, applicability
  - emergence of MPI, OpenMP, C++, Java
  - new target architectures: workstation clusters, DSM