Thanks to…

- The High Performance Fortran Forum
- The Fortran D Group (Rice University)
- The Fortran 90D Group (Syracuse University)
- Ken Kennedy (Rice)
- David Loveman (DEC)
- Piyush Mehrotra (ICASE)
- Rob Schreiber (RIACS)
- Guy Steele (Sun)
- Mary Zosel (Livermore)

HPF Commercial Interest

Announced HPF Products

Applied Parallel Research	Motorola
Digital Equipment Corp.	NA Software
Fujitsu	NEC
Hitachi	Pacific Sierra Research
IBM	Portland Group
Intel	Thinking Machines
Meiko	Transtech

Announced HPF Efforts

ACE	NAG
Lahey	nCUBE

"Interested"

Convex/Hewlett Packard	Silicon Graphics
Cray Research	Sun Microsystems
Edinburgh Portable Compilers	Tera

High Performance Fortran in Practice

Charles Koelbel

High Performance Fortran Background

- **Defined by the High Performance Fortran Forum (HPFF) as a portable language for data-parallel computation**
- **History:**
  - Proposed at Supercomputing '91 and HPFF Kickoff Meeting
  - Meetings every 6 weeks during 1992 in Dallas
  - Final draft of HPF, version 1.0 in June 1993
  - New meetings in 1994 and 1995 to make corrections, define further requirements
- **Influences:**
  - Industry: CM Fortran, C*, MasPar Fortran, DEC
  - Academia: ADAPT, Fortran D, Kali, Vienna Fortran, CRPC
  - Government: Applications experience, $$$

For More Information

- **World Wide Web**
  - `http://www.crpc.rice.edu/HPFF/home.html`
  - `http://www.vcpc.univie.ac.at/HPFF/home.html`
- **Mailing Lists:**
  - Write to `majordomo@cs.rice.edu`
  - In message body: `subscribe <list-name> <your-id>`
  - Lists:
    - `hpff`
    - `hpff-interpret`
    - `hpff-core`
- **Anonymous FTP:**
  - Connect to `titan.cs.rice.edu`
  - Full draft in `public/HPFF/draft`
  - See `public/HPFF/README` for latest file list

Parallel Machines

- **Parallel computers allow several CPUs to contribute to a computation simultaneously.**
- **For our purposes, a parallel computer has three types of parts:**
  - Processors
  - Memory modules
  - Communication / synchronization network
- **Key points:**
  - All processors must be busy for peak speed.
  - Local memory is directly connected to each processor.
  - Accessing local memory is much faster than other memory.
  - Synchronization is expensive, but necessary for correctness.

HPF Features

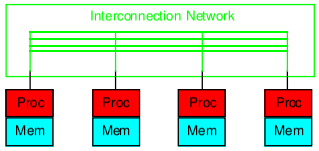
- **All of Fortran 90**
- **FORALL and INDEPENDENT**
- **Data Alignment and Distribution**
- **Miscellaneous Support Operations**
- **But:**
  - No parallel I/O
  - No explicit message-passing
  - Little support for irregular computations
  - Little support for non-data parallelism

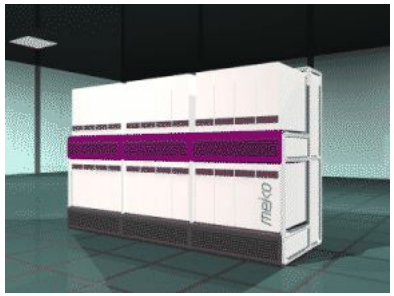
Outline

1. Introduction to Data-Parallelism
2. Fortran 90 Features
3. HPF Parallel Features
4. HPF Data Mapping Features
5. Parallel Programming in HPF
6. HPF Version 2.0

Distributed Memory Machines

- **Every processor has a memory others can't access.**
- **Advantages:**
  - Can be scalable
  - Can hide latency of communication
- **Disadvantages:**
  - Hard to program
  - Program and O/S (and sometimes data) must be replicated

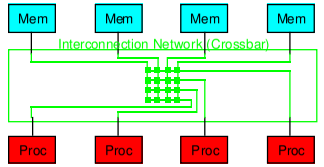




Meiko CS-2

Shared-Memory Machines

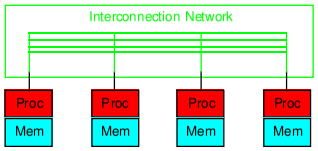
- **All processors access the same memory.**
- **Advantages:**
  - Easy to program (correctly)
  - Can share code and data among processors
- **Disadvantages:**
  - Hard to program (optimally)
  - Not scalable due to bandwidth limitations

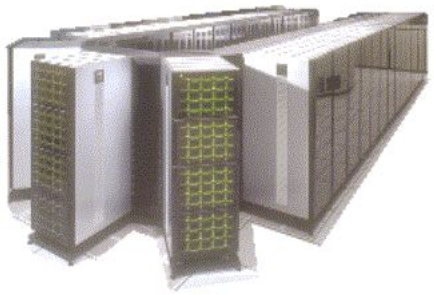


Fujitsu VP-500

Distributed Memory Machines

- **Every processor has a memory others can't access.**
- **Advantages:**
  - Can be scalable
  - Can hide latency of communication
- **Disadvantages:**
  - Hard to program
  - Program and O/S (and sometimes data) must be replicated

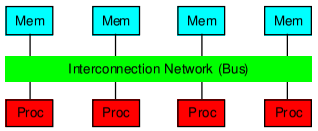




Intel Paragon At Sandia National Labs

Shared-Memory Machines

- **All processors access the same memory.**
- **Advantages:**
  - Easy to program (correctly)
  - Can share code and data among processors
- **Disadvantages:**
  - Hard to program (optimally)
  - Not scalable due to bandwidth limitations



SGI Power Challenge XL

Networks

- **High-speed networks are increasingly being used as single computing resources.**
  - Note: Not all machines have to be the same!
  - Note: Network nodes may themselves be parallel processors!
- **Advantages:**
  - Cheap
  - Can be very effective
  - Can upgrade system incrementally
- **Disadvantages:**
  - Heterogeneous computing is still research
  - Some people won't let you use their machines
  - Dedicated machines will always have better latency

Data-Parallel Languages

- **Data-parallel languages provide an abstract, machine-independent model of parallelism.**
  - Fine-grain parallel operations, such as element-wise operations on arrays
  - Shared data in large, global arrays with mapping "hints"
  - Implicit synchronization between operations
  - Partially explicit communication from operation definitions
- **Advantages:**
  - Global operations conceptually simple
  - Easy to program (particularly for certain scientific applications)
- **Disadvantages:**
  - Unproven compilers

Conventional Vector Supercomputers

- **Vector computers apply vector pipelines to sequences of operands.**
- **For our purposes, (uniprocessor) vector computers have three parts:**
  - Vector pipelines
  - Vector registers
  - Memory
- **Key points:**
  - All pipeline stages must be filled for peak speed.
  - Do this by creating long vectors and matching number of operations to number of pipelines.
  - Loading vectors from memory is slow, especially for non-unit strides.

Parallel Programming

- **A parallel program is a collection of tasks and a partial ordering between them.**
- **Goals for parallel algorithms:**
  - Match tasks to the available processors (exploit parallelism).
  - Minimize ordering (avoid unnecessary synchronizations).
  - Utilize remaining ordering (exploit locality).
- **Sources of parallelism:**
  - Data parallelism: updating array elements simultaneously.
  - Functional parallelism: conceptually different tasks which combine to solve the problem.
  - Speculative parallelism: temporarily ignore partial ordering, repair later if this causes problems.
  - This list is not exhaustive…

Object-Oriented Languages

- **Programs are collections of interacting objects**
  - Objects can be active concurrently
- **Objects usually have private data**
  - Other sharing depends on the base language and preferences of the language designers
- **Synchronization explicit on method invocation**
  - Lower-level synchronization sometimes also possible
- **Examples: C++ classes, pC++, CC++**
- **Advantages:**
  - Convenient, once you have the base classes
- **Disadvantages:**
  - Perceived as slow

How HPF Is Implemented

- **Analysis**
  - Traditional dataflow and dependence analysis
  - Data mapping analysis
- **Computation Partitioning**
  - Use data mapping info to create locality
  - Transform code to enhance locality
- **Communication Introduction**
  - Communicate (move data) if data mapping and computation partitioning don't agree
  - Lots of optimization to minimize/package communication
- **Code Generation**
  - The really ugly details
  - Lots of optimization possible here, too

Data-Parallel Languages, cont.

- **Abstractions like data parallelism split the work between the programmer and the compiler.**
- **Programmer's task: Solve the problem in this model.**
  - Concentrate on high-level structure and concepts
  - Aggregate operations on large data structures
  - Data in global arrays with mapping information
- **Compiler's task: Map conceptual (massive) parallelism to physical (finite) machine.**
  - Fill in the grungy details
  - Guided by user "hints" like mapping information
  - Optimizing computation and communication

Message-Passing Systems

- **Program is based on relatively coarse-grain tasks**
- **Separate address space and a processor number for each task**
- **Data shared by explicit messages**
  - Point-to-point and collective communications patterns
- **Examples: MPI, PVM, Occam**
- **Advantages:**
  - Close to hardware.
- **Disadvantages:**
  - Many low-level details.

The High Performance Fortran Model

- **Performance is determined by**
  - Basic operation count
  - Degree of parallelism
    - Vector operations
    - Independent loop iterations
    - Compiler-detected parallelism(?)
  - Communication
    - Data mapping
    - Unaligned variable references
- **HPF gives a high-level description of parallelism and data distribution.**
- **HPF deemphasizes low-level communications details.**

What Compilation Means for Programmers

- **Help analysis with assertions**
  - `ALIGN` and `DISTRIBUTE`
  - `INDEPENDENT`
- **Distribute array dimensions that exhibit parallelism**
  - Conflicts require complex compilers, `REDISTRIBUTE`, or new algorithms
- **Consider communications patterns**
  - `BLOCK` generally good for local stencils and fully-filled arrays
  - `CYCLIC` and `CYCLIC(K)` generally good for load balancing and triangular loops
- **Don't hide what you are doing**