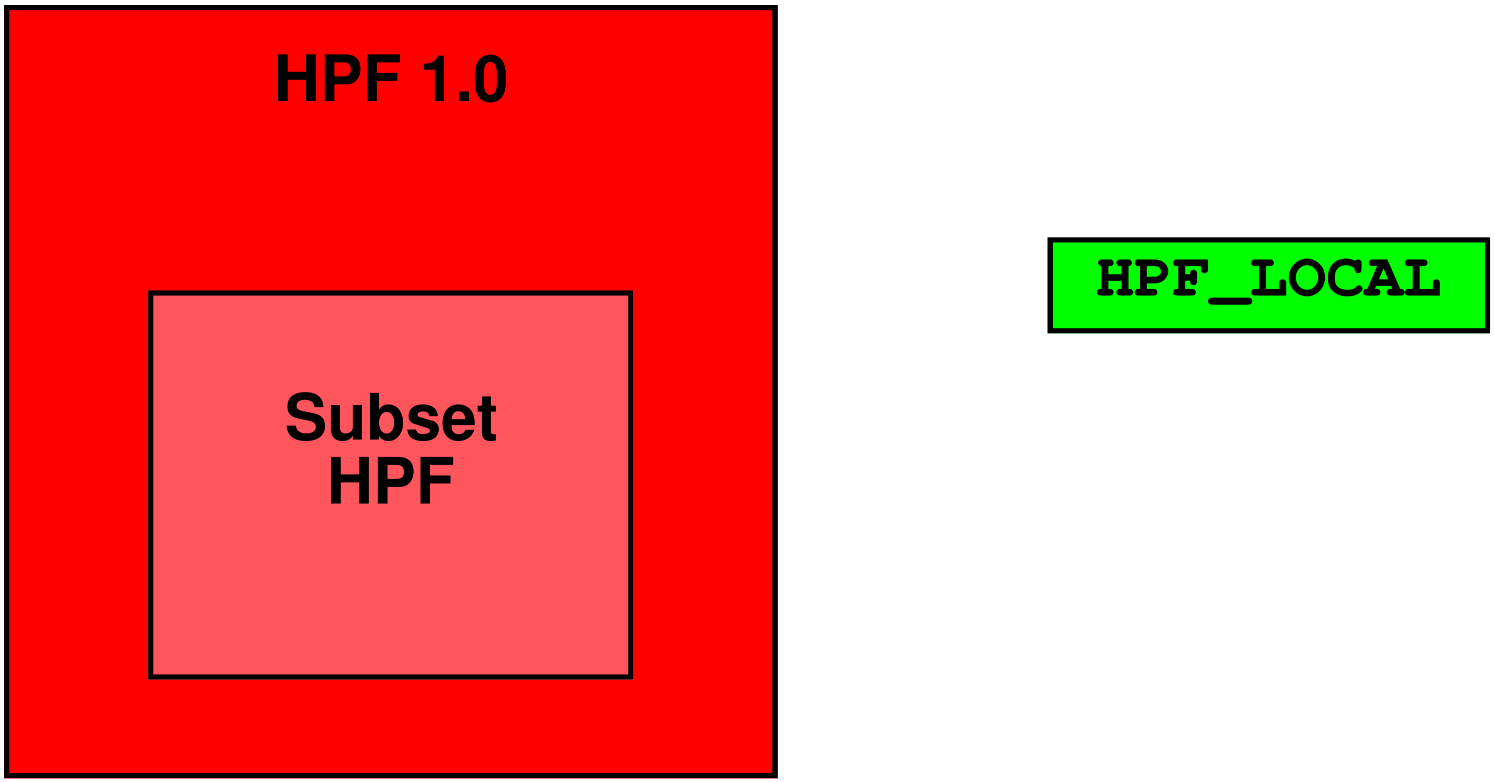
Outline

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

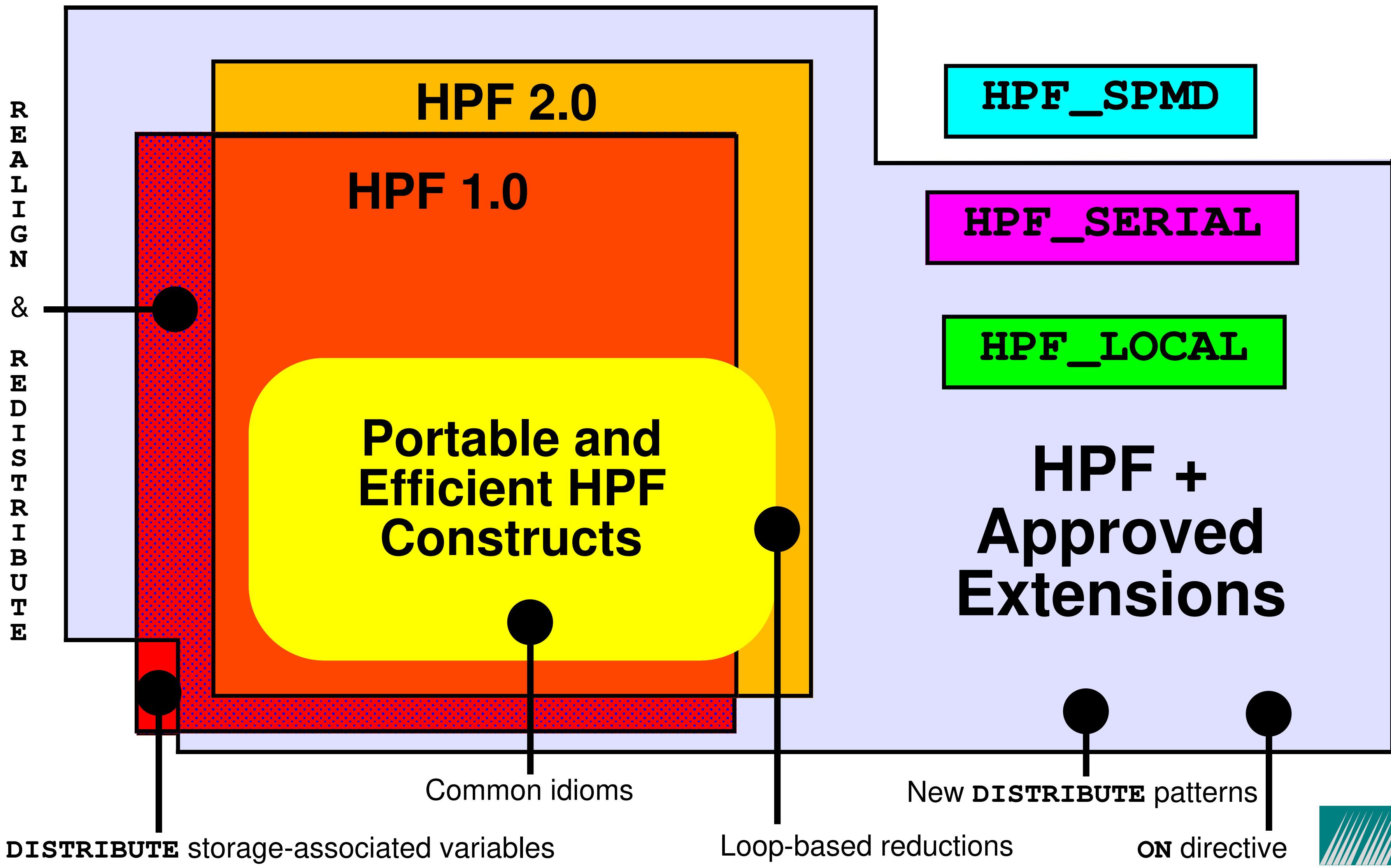
HPF 2

- **New HPFF meetings are being held to develop extensions to HPF**
  - Meetings began January 1995
  - Preliminary presentations at Supercomputing ’95
  - Complete draft at Supercomputing ’96
- **Areas for extensions**
  - Control parallelism (**`hpff-task@cs.rice.edu`**)
  - Data distribution (**`hpff-distribute@cs.rice.edu`**)
  - External interfaces (**`hpff-external@cs.rice.edu`**)
- **Input is welcome!**
  - Send mail to `majordomo@cs.rice.edu` to get on a list

HPF 2.0 Features



HPF 2.0 Features



HPF 2.0 Deletions and Simplifications

- **`DYNAMIC`, `REALIGN`, and `REDISTRIBUTE`**
  - Now “just” approved extensions
  - Removed due to complexity of implmentation
- **Mapping in the presence of sequence association**
  - Now forbidden in all cases
  - Removed due to complexity, and lack of user demand
- **Subroutine interfaces**
  - Explicit interface required if mapping changes, or **`INHERIT`** used
  - Descriptive (“star”) syntax retained, primarily for error checking
  - Changed to match spirit of Fortran 95, and simplify user model

New Parallel Execution Features

- Computation Placement

```
!HPF$ INDEPENDENT
DO i = 1, n
    !HPF$ ON HOME( ix(i) )
    x(i) = y(ix(i)) - y(iy(i))
END DO
```

- Loop Reductions

```
!HPF$ INDEPENDENT, NEW(xinc), REDUCTION(x)
DO i = 1, n
    CALL sub(i, xinc)
    x = x + xinc
END DO
```

- Task Parallelism

```
!HPF$ TASKING
    !HPF$ ON HOME(p(1:8))
    CALL foo(x,y)
    !HPF$ ON HOME(p(9:16))
    CALL bar(z)
!HPF$ END
```

Example of New Parallelism

```
!HPF$ INDEPENDENT
DO i = 1, 12
   !HPF$ ON HOME( ix(i) )
   x(i) = y(ix(i))-y(iy(i))
END DO
```

ix	1	2	3	4	5	6	7	8	9	10	11	
iy	2	2	4	4	6	6	8	8	10	10	12	12
x												
y												

With ON HOME(ix(i))

```
x(1)=y(1)*y(2)
x(2)=y(2)*y(2)
x(3)=y(3)*y(4)
```

```
x(4)=y(4)*y(4)
x(5)=y(5)*y(6)
x(6)=y(6)*y(6)
```

```
x(7)=y(7)*y(8)
x(8)=y(8)*y(8)
x(9)=y(9)*y(10)
```

```
x(10)=y(10)*y(10)
x(11)=y(11)*y(12)
x(12)=y(12)*y(12)
```

With LHS owner-computes

```
x(1)=y(1)*y(2)
x(2)=y(2)*y(2)
```

```
x(3)=y(3)*y(4)
x(4)=y(4)*y(4)
x(5)=y(5)*y(6)
x(6)=y(6)*y(6)
```

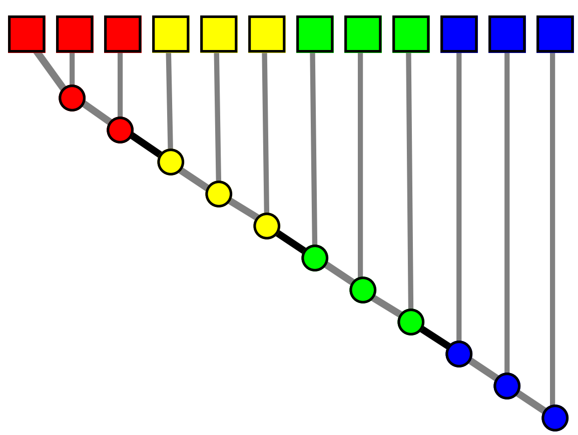
```
x(7)=y(7)*y(8)
x(8)=y(8)*y(8)
x(9)=y(9)*y(10)
x(10)=y(10)*y(10)
```

```
x(11)=y(11)*y(12)
x(12)=y(12)*y(12)
```

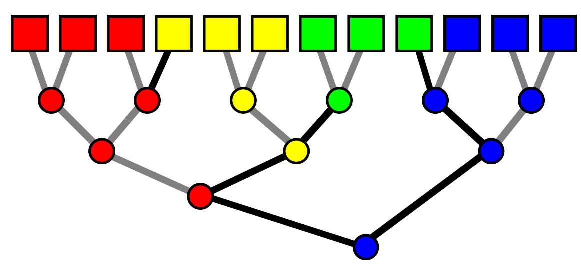
Another Example of New Parallelism

```
!HPF$ INDEPENDENT, NEW(xinc), REDUCTION(x)
DO i = 1, n
    CALL sub(i, xinc)
    x = x + xinc
END DO
```

Serial reduction

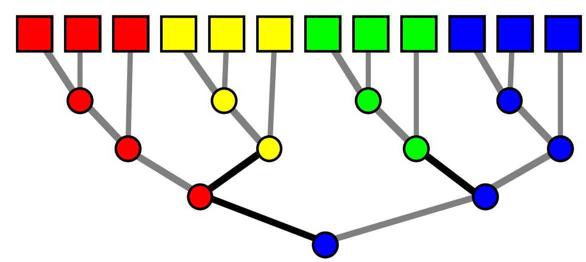


Tree reduction



Local reductions with

final tree combining



New Data Mapping Features

- **Extended Distribution Patterns**

```
!HPF$ DISTRIBUTE x( BLOCK(SHADOW=1) )
!HPF$ DISTRIBUTE y( BLOCK( (/ 12,10,10,12 /) )
!HPF$ DISTRIBUTE z( INDIRECT(map_array) )
```

- **Distribution to processor subsets**

```
!HPF$ PROCESSORS procs(1:np)
!HPF$ DISTRIBUTE b(BLOCK) ONTO procs(1:np/2-1)
```

- **Distribution of derived type components**

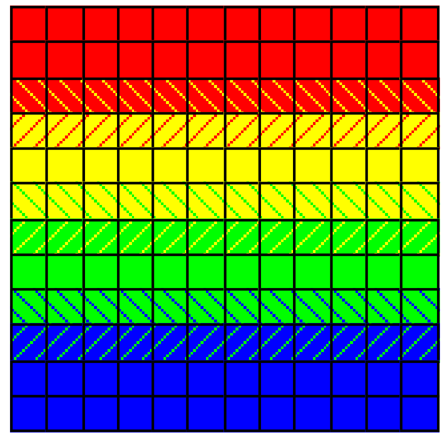
```
TYPE set_of_meshes
    REAL p(100,100), q(100,100), r(100,100)
    !HPF$ DISTRIBUTE (BLOCK,*) :: p, q, r
END TYPE
TYPE tree
    TYPE(tree), POINTER, DIMENSION(100) :: children
    !HPF$ DYNAMIC children
    REAL value
END TYPE
TYPE(tree) t
ALLOCATE( t%children )
!HPF$ REDISTRIBUTE t%children(BLOCK)
```

CRPC

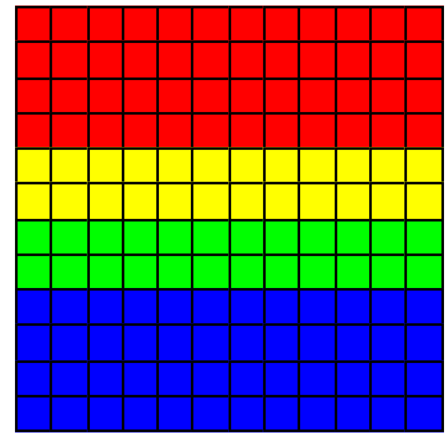
Examples of New Data Mappings

```
!HPF$ DISTRIBUTE x(BLOCK(SHADOW=1),*)

!HPF$ DISTRIBUTE y(BLOCK((/4,2,2,4/)),*)

!HPF$ DISTRIBUTE z(*,INDIRECT(map))


!HPF$ PROCESSORS p(4)

!HPF$ DISTRIBUTE b(*,BLOCK) ONTO p(3:4)
```

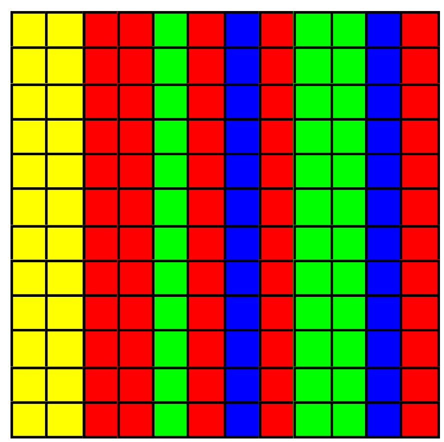
(BLOCK(SHADOW=1),*)



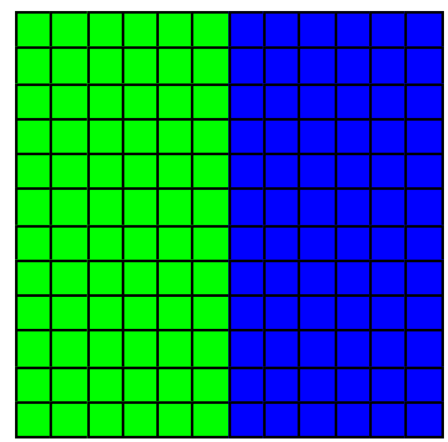
(BLOCK((/4,2,2,4/)),*)



(*,INDIRECT(map))



(*,BLOCK) ONTO p(3:4)

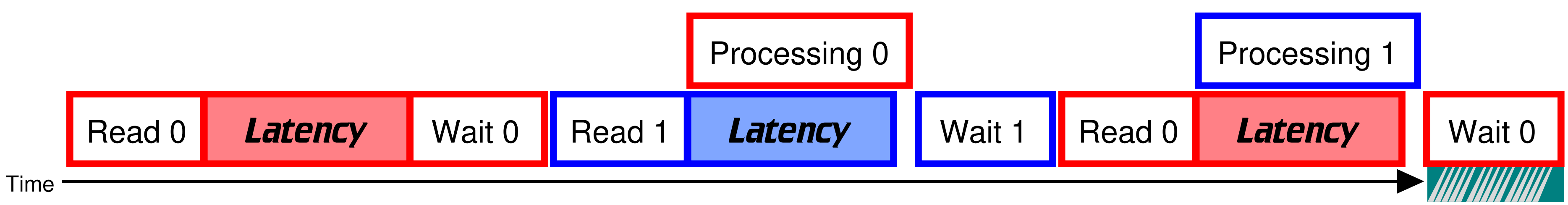


External Interfaces

- **Calling HPF from other languages**
  - How to ensure HPF is called consistently?
- **Calling other languages from HPF**
  - How to avoid unnecesary synchronization?
- **Parallel Input/Output**
  - What do *you* mean by “parallel I/O”?
  - Asynchronous I/O
- **Tools**
  - Can we define a symbol table format for debuggers, etc?

Example of Asynchronous I/O

```
i0 = 0; i1 = 1
READ (file0, ID=id0, END=100) a(i0,1:1048576)
DO
    WAIT (ID=id0, END=100)
    ! start next read into other row
    itmp = i0; i0 = i1; i1 = itmp
    READ (file0, ID=id0, END=100) a(i0,1:1048576)
    ! overlap I/O with useful work
    CALL PROCESSING( a(i1,1:1048576) )
END DO
100 CONTINUE
```



CRPC

The High Performance Fortran Model

- **Performance is determined by**
  - Parallelism
    - Vector operations
    - Independent loop iterations
  - Communication
    - Data mapping
    - Unaligned variable references
- **HPF gives a high-level description of parallelism and data distribution**
- **HPF deemphasizes low-level communications details**

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

CRPC