Combining Object-Oriented Techniques with Co-arrays in Fortran 2008

Robert W. Numrich

Minnesota Supercomputing Institute
Minneapolis, MN USA
rwn@msi.umn.edu
Fortran is a modern language

- Fortran 2003
  - Object-oriented
  - Portable C interface
  - Parametrized derived types
  - Strong typing through interfaces

- Fortran 2008
  - Co-arrays
  - First parallel addition to the language
Object-oriented
- Objects
  - User-defined derived types
  - Type-bound procedures
  - Type constructors
  - Type finalization
- Abstract types
- Inheritance
- Deferred procedure bindings
- Overloaded generic procedures
- Polymorphism
Co-arrays

- Execution model
  - SPMD programming model (MPI rank \(\leq\) CAF image)
  - Explicit synchronization

- Memory model
  - Explicit data decomposition
  - Explicit data movement

- Co-array objects
  - \texttt{real} :: x[*]
  - \texttt{type(Y)} :: a[*]
Using Co-arrays

real :: x(n)[p,*]
type(Y) :: a[*]

real :: y(n)
real :: z

y(:) = x(:) ! local copy
y(:) = x(:)[r,s] ! remote copy
a[q]%x(k) = z
real, allocatable :: a[:,:,::]

p = coresPerChip()
q = chipsPerNode()
r = nodesPerSystem()
allocate(a[p,q,*])

x = a local reference
x = a[:,q,r] on-chip reference
x = a[p,:,r] on-node reference
x = a[p,q,:] off-node reference
real :: balance

type(BankAccount) :: myAccount[*]

balance = myAccount[ftp://myBank.com]%balance
Assigning images to processors (cores?)

One-to-one
- one core to one image

Many-to-one
- many cores to one image (OpenMP)

One-to-many
- one core to many images (virtual processors)

Many-to-many
- many cores to many images (virtual processors with OpenMP)
Co-array objects

```fortran

type :: Y
    real, allocatable :: x(:)
end type

type(Y) :: a[*]
    real, allocatable :: y(:)
end type

y(:) = a[p] % x(:)

---

type :: Y
    real, allocatable :: x(:) [:]
end type

type(Y) :: a
    real, allocatable :: y(:)
end type

y(:) = a % x(:) [p]
```

Robert W. Numrich
Combining Object-Oriented Techniques with Co-arrays in Fortran
Co-array objects

Robert W. Numrich
Combining Object-Oriented Techniques with Co-arrays in Fortran
Co-array objects

\[ a\%x[q] \quad a\%x[p] \]

Robert W. Numrich
Combining Object-Oriented Techniques with Co-arrays in Fortran
A Parallel Numerical Library (Fortran 95)

- Grids
- Differential Operators
- Fields
- Block Vectors
- Vector Maps
- Block Matrices
- Vectors
- Linear Algebra
- Matrices

Robert W. Numrich
Combining Object-Oriented Techniques with Co-arrays in Fortran
Object Maps (Composite Pattern)
Example Program

program LU
    type(BlockR8Matrix) :: A[*]
    type(PivotVector)   :: Pivot[*]
    integer, parameter :: n=1000
    integer, parameter :: block=100
    integer, parameter :: p=4, q=4

    A = BlockMatrix(n, block, n, block, p, q)
    Pivot = PivotVector(A)
    call A%readBlockMatrix()
    call A%LU(Pivot)
    call A%writeBlockMatrix()
end program LU
LU Decomposition

![Graphs showing the relationship between $1/p$ and $2^p$ for different values of $p$.]
Row interchange

temp(:) = a(k,:)
a(k,:) = a(j,:) [p,myQ]
a(j,:) [p,myQ] = temp(:)

"Row Broadcast"

L0(i:n,i) = a(i:,n,i) [p,p] i=1,n

"Row/Column Broadcast"

L1 (:,:) = a(:,,:) [myP,p]
U1(:,:) = a(:,:) [p,myQ]
Distributed fields

Abstract Field
- data( :, :) [p, *]
- getData()
- setData()

Distributed Field
- Map
- Grid
- gradient()
- exchange()

Abstract Grid
- LonLat Grid
- DistrLonLat Grid

co-arrays

passed to kernels
type(DistributedField) :: a

a = DistributedField(map)

function DistributedField(map) result(a)
  type(DistributedField) :: a
  type(DistributedMap) :: map
  a%map => map
  nx = map%getX()
  ny = map%getY()
  p = map%getP()
  allocate(a%data(nx,ny)[p,*])
end function
type(PhysicalField) :: a
type(Iterator) :: iterator
real, contiguous, pointer :: ptrA(:,:)

iterator => a%getGlobalIterator()
do while(iterator%hasNext())
    ptrA => iterator%requireField()
call kernel(ptrA)
    ptrA => iterator%releaseField()
end do
Compilers that support co-arrays

- Cray has supported co-arrays for over ten years
- g95 has a preliminary implementation
- IBM under development
- gfortan in discussion phase
- Ask Intel for a multi-core implementation
The co-array model is simpler and easier to use than the MPI model.

- Co-arrays should be as good as MPI on any architecture.
- Co-arrays work best on hardware with a true global address space.
- How important are the object-oriented features of Fortran 2008?
References