More about coarrays

John Reid, ISO Fortran Convener,

JKR Associates and

Rutherford Appleton Laboratory

MPI/CAF Comparison

Parts that differ between an MPI and a CAF implementation of a Jacobi iteration code for the Laplace equation

Set up niter on all images

```
IF ( my_rank .EQ. 0) THEN
  print *, 'Number of iterations?'
  read *, niter
  niter = MIN0 ( niter, MXITER)
  print *, 'niter = ', niter
END IF
CALL MPI_Bcast ( niter, 1, &
          MPI_INTEGER, 0, &
          MPI_COMM_WORLD, ierr)
```

```
IF ( me == 1) THEN
   print *, 'Number of iterations?'
   read *, niter
   niter = MIN0 ( niter, MXITER)
   print *, 'niter = ', niter
   DO j = 2, nimages
        niter[j] = niter
   END DO
END IF
SYNC ALL
```

Get values from right neighbour window and put in local storage T

```
window_offset = 0
IF ( my_rank /= 0) THEN
  call MPI_WIN_LOCK ( MPI_LOCK_SHARED, &
     my_rank_m1, 0, &
     botm_window_handle, ierr)
  call MPI_GET ( T1d_top, nr_p2, MPI_REAL, &
     my_rank_m1, window_offset, nr_p2, &
     MPI_REAL, botm_window_handle, ierr)
  call MPI_WIN_UNLOCK ( my_rank_m1, &
     botm_window_handle, ierr)
END IF
```

```
IF ( me /= 1) THEN
  T1d_top(:) =  &
          botm_send_window(:)[me_m1]
END IF
```

Find maximum temp change

```
dt_global = 0.0
CALL MPI_Reduce ( dt, dt_global, 1, &
                  MPI_REAL, MPI_MAX, 0, &
                  MPI_COMM_WORLD, ierr)
```

```
SYNC ALL
IF ( me == 1) THEN
  dt_global = dt
  DO i = 2, nimages
    dt_global = MAX (dt_global, dt[i])
  END DO
END IF
! May be inefficient
```

Find maximum temp change - proposed

```
dt_global = 0.0
CALL MPI_Reduce ( dt, dt_global, 1, &
                  MPI_REAL, MPI_MAX, 0, &
                  MPI_COMM_WORLD, ierr)
```

```
SYNC ALL
CALL co_max(dt,result_image=1)
```

The intrinsic procedures that have been added for coarrays.

`this_image`

`this_image( )`

Index of executing image

`this_image(coarray)`

Array of cosubscripts of executing image in **`coarray`**

`this_image(coarray, dim)`

Cosubscript **`dim`** of executing image in **`coarray`**

`image_index`

`image_index(coarray, sub)`

where **`sub`** is a rank-one integer array of size the corank of **`coarray`** returns the image index of the image with these cosubscripts.

move_alloc

call move_alloc(from, to)

where **from** is allocatable and **to** is allocatable with the same type, rank and corank.

After the call, **to** has the allocation status that **from** had beforehand and **from** is deallocated.

If **from** was associated with a target, **to** becomes associated with it.

If they are coarrays, there is an implicit synchronization of all images.

Doubling the size of an array coarray

```
real, allocatable :: a(:)[:]
real, allocatable :: temp(:)[:]
   :
allocate( temp(size(a)*2)[*] )
temp(1:size(a)) = a(:)
call move_alloc(temp,a)
```

cobounds

`lcobound(coarray)`
Rank-1 integer array holding the lower cobounds

`lcobound(coarray, dim)`
Lower cobound **`dim`** of **`coarray`**

`ucobound(coarray)`
Rank-1 integer array holding the upper cobounds

`ucobound(coarray, dim)`
Upper cobound **`dim`**

The last upper cobound is the largest valid value it can have.