Further coarray features for parallel computing

John Reid, Convener WG5

In 2008, WG5 decided to move some coarray features from Fortran 2008 to a Technical Specification (TS).

In June, WG5 decided that while the overall complexity of the TS should be unchanged, the public should be invited to comment on the technical contents.

I will discuss the requirements that have been suggested and invite the audience to comment.

BCS Fortran Specialist Group
London, 29 September 2011.

Items removed in 2008

In February 2008, it was decided to move the following features into a separate Technical Report on ‘Enhanced Parallel Computing Facilities’:

1. Teams and features that require teams.
2. The collective intrinsic subroutines.
3. The notify and query statements.
4. File connected on more than one image, unless preconnected to the unit specified by output_unit or error_unit.

I will describe these in turn.

Technical Reports are now known as Technical Specifications.

Teams

The intrinsic module iso_fortran_env contains a derived type image_team. A scalar object of this type identifies a team of images.

The same call of the intrinsic subroutine form_team must be executed on all images of a team to form the team.

This code splits images into two groups and implicitly synchronizes each of them:

```
use iso_fortran_env
integer :: i, ne
e = num_images()
type(image_team) :: team
if (this_image()<=ne/2) then
  call form_team(team,[(i,i=1,ne/2)])
else
  call form_team(team,[(i,i=ne/2+1,ne)])
end if
```

Team barrier

sync team (team) provides a barrier for the team.
Notify – query

integer image_set(p)
logical ready
notify (image_set)
query (image_set)
query (image_set, ready)

Records are kept of
n(m,t), the number of times image m completes
a notify statement with image t in its image
set and
q(t,m), the number of times image t completes
a query statement with image m in its image
set
Image t waits until n(m,t) > q(t,m) for all
images m in its image set.

The query statement has an optional logical
ready argument that allows the image to do other
work instead of waiting.

Collective subroutines

Intrinsics and involve synchronization.
All have optional argument team.

On every image, given the co-arrays
real :: x[*], y(n)[*]
real :: sum, sums(n)
call co_sum(x,sum)
returns \( \sum_p x[p] \) in sum and
call co_sum(y(:),sums(:))
returns \( \sum_p y(:)[p] \) in sums.

List of collective subroutines

c_o_all True if all values are true
c_o_any True if any value is true
c_o_count Numbers of true elements
c_o_findloc Image indices of images having
a given value
c_o_maxloc Image indices of maximum values
c_o_maxval Maximum values
c_o_minloc Image indices of minimum values
c_o_minval Minimum values
c_o_product Products of elements
c_o_sum Sums of elements

Input/output

Syntax to allow teams of images to access a
single file. Allows local buffering.

To open for a team:
OPEN(unit,...,TEAM=team,...)
There is an implied
sync team (team)
and the unit must not be opened on other images.

Only cases:

sequential write While an image is writing a
record, the processor blocks other images.
Thus each record comes from a single image.

direct access Up to the programmer to
synchronize access to a single record by more
than one image.
Bill Long’s collective subroutines

Bill Long proposes a new set of collective subroutines that are not image control statements.

The new intrinsic subroutine
`co_bcast(source,source_image[,team])` would broadcast a value to all images of a team.

The new intrinsic subroutine
`co_reduce(source,operation & [,result,team,result_image])` would provide a general routine for operations not currently covered. `operation` is an external procedure that defines a binary, commutative operation.

Only `co_max`, `co_min`, and `co_sum` of the old collectives are retained. Their features are slightly altered to improve performance.

Robert Numrich suggests the removal of teams.

He says

*The ability to couple two coarray codes already exists using MPI intercommunicators or one of many frameworks out there. These frameworks just need to allow coarray codes as components.*

and

*A coarray code should be the same whether it is run alone or run as a team coupled with another coarray code. With the current definition of teams this is not true. Both codes will need to be altered to run as teams.*

WITH TEAM Construct

The Rice University group suggest adding a `with team` construct. If I understand it correctly, we have

```fortran
with team (team)
  : ! All coindexing here is with
  : ! respect to the team, including
  : ! within any procedures invoked.
end with team
```

Looks attractive to me, but I may have missed the snags.