Co-Arrays

John Reid,

JKR Associates, UK

Abstract

It was decided in Delft to include co-arrays in the next revision. They provide a simple parallel extension, based on a second set of subscripts to address different ‘images’.

They are the brain-child of Bob Numrich (Minneapolis, formerly Cray). A subset has been implemented by Cray.

BCS Fortran AGM

2 June 2005

Basics

- SPMD – Single Program, Multiple Data
- Replicated to a number of **images**
- Number of images fixed during execution
- Each image has its own set of **local** variables
- All images start by executing main program and mostly execute asynchronously
- Variables declared as co-arrays are accessible on another image through set of array subscripts, delimited by [ ]
- Intrinsics: `this_image`, `num_images`, `sync_all`, `sync_team`, `flush_memory`, collectives such as `co_add`
- Critical construct

Examples of co-array syntax

```
real :: r[*], s[0:*], x(n)[*]
type(u) :: u2(m,n)[np,*]
! Co-arrays always have assumed
! co-size (equal to number of images)

real :: t
integer p, q, index(n)
!  Local variables
     :
t = s[p]
x(:) = x(:)[p]
! Reference without [] is to local part
x(:)[p] = x(:)
u2(i,j)%b(:) = u2(i,j)[p,q]%b(:)
```

Images have indices 1, 2, ..., `num_images()` and co-subscript lists are mapped to image indices by the usual rule.

Example: redistribution

Consider redistributing the array
`a(1:kx,1:ky)[1:kz]`
from
`b(1:ky,1:kz)[1:kx]`,
where `max(kx,kz)` ≤ `num_images()`.

```
iz = this_image(a)
if (iz<=kz) then
   do ix = 1, kx
      a(ix,:) = b(:,iz)[ix]
   end do
end if
```

The `if` construct is needed so that no action is taken on the images on which we have no data.

Implementation model

The compiler may arrange that a co-array, when originally declared, occupies the same set of addresses within each image:

- A co-array must have the same set of bounds on all images
- There is an implicit synchronization of all images at an allocate or deallocate statement so that they all perform their allocations and deallocations in the same order.

On a shared-memory machine, a co-array may be implemented as a single large array.

On any machine, a co-array may be implemented so that each image can calculate the memory address of an element on any image.

Synchronization

The images normally execute asynchronously. If one image relies on another image having taken an action, explicit synchronization is needed.

For example, to read data on image 1 and get it to other images:

```
if(this_image()==1) read(*,*)p
call sync_all()
p = p[1]
```

Critical sections

Exceptionally, it may be necessary to limit execution to one image at a time:

```
critical
   p[6] = p[6] + 1
   :
end critical
```

Procedures

A subobject of a co-array without [ ], may be passed to a co-array.

- The ordinary rules of Fortran 95 apply to the local part and the co-array part is defined afresh.
- The interface must be explicit.
- No copy-in copy-out.

The rules for resolving generic procedure references remain unchanged.

No co-array syntax is permitted in pure procedures.

Dynamic co-arrays

Only dynamic form: the allocatable co-array.

Automatic arrays or array-valued functions would require automatic synchronization, which would be awkward.

Co-Arrays and SAVE

Unless allocatable or a dummy argument, a co-array must be given the SAVE attribute.

This is to avoid the need for synchronization when co-arrays go out of scope on return from a procedure.

Optimization

Most of the time, the compiler can optimize as if the image is on its own, using its temporary storage such as cache, registers, etc.

Structure components

Types may have allocatable co-array components, but structures of this type must be scalar.

However, a co-array may be of a derived type with allocatable or pointer components, which allows the size to vary from image to image.

Pointers must have targets in their own image:

```
q => z[i]%p       ! Not allowed
allocate(z[i]%p) ! Not allowed
```

Input/output

All images reference the same file system and a single set of input/output units.

To open for a team:

```
OPEN(unit,...,TEAM=connect-team,...)
```

There is an implied

```
call sync_team(connect-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.

Other items accepted in Delft

- J3-03 Execute external program
- J3-13 Internal procedure as actual argument
- J3-19 Extend intrinsics such as `ASIN` to complex arguments
- J3-39 Rank plus co-rank limited to 15.
- J3-43 Contiguous attribute
- J3-44 `INTENT(SCRATCH)`
- J3-46 `DO CONCURRENT`
- RU-03 Deleted: statement functions; obsolescent: ENTRY
- UK-05 Guarantee support of `selected_int_kind(18)`