Co-Array Fortran

Phil Russell
Alex Libor
Micah Wendell
What It Is

Formally called F--, 
Small set of semantic extensions to Fortran 95
Simple syntactic extension to Fortran 95
Single Program Multiple Data, SPMD, parallel processing
What It Is

Robust, efficient parallel language.
Requires learning only a few new rules.
Rules handle two fundamental issues:
  Work distribution
  Data distribution.
Work Distribution

A single program is replicated a fixed number of times.
Each replication has its own set of data objects.
Each replication of the program is called an image.
Each image executes asynchronously.
Work Distribution

The normal rules of Fortran apply
The execution path may differ from image to image.
The programmer determines the actual path for the image with

A unique image index
Normal Fortran control constructs
Explicit synchronizations.
Work Distribution

Code between synchronizations

The compiler is free to use all its normal optimization techniques, as if only one image is present.
Data Distribution

Specify the data relationships
One new object, the co-array, is added to the language
An example…
Data Distribution

REAL, DIMENSION(N)[*] :: X,Y

\[ X(:, \) = Y(:,[Q]\)

The above statement declares that each image has two real arrays of size N. If Q has the same value on each image, the effect of the assignment statement is that each image copies the array Y from image Q and makes a local copy in array X.
Data Distribution

(index) follow the normal Fortran rules within one memory image.

[index] provide access to objects across images and follow similar rules.

[bounds] in co-array declarations follow the rules of assumed-size arrays since co-arrays are always spread over all the images.
Data Distribution

The programmer uses co-array syntax only where it is needed.

A co-array reference with no square brackets is a reference to the object in the local memory.

Co-array syntax should appear only in isolated parts of the code.

If not, too much communication among images?

Flags compiler to avoid latency

Flags programmer to rethink
Extended Fortran 90 Array Syntax

A way of expressing remote memory operations. Here are some simple examples:

\[
\begin{align*}
X &= Y[\text{PE}] & \text{get from } Y[\text{PE}] \\
Y[\text{PE}] &= X & \text{put into } Y[\text{PE}] \\
Y[:] &= X & \text{broadcast } X \\
Y[\text{LIST}] &= X & \text{broadcast } X \text{ over subset of } \text{PE's} \\
in & & \\
Z(\cdot) &= Y[:] & \text{collect all } Y \\
S &= \text{MINVAL}(Y[:]) & \text{min (reduce) all } Y \\
B(1:1)[1:1:N] &= S & \text{S scalar, promoted to shape} (1:1:M,1:1:N)
\end{align*}
\]
Input/Output

Input/output problem with SPMD programming models
Fortran I/O assumes dedicated single-process access to an open file

Often violated when it is assumed that I/O from each image is completely independent.
Input/Output

Co-Array Fortran includes only minor extensions to Fortran 95 I/O,
All the inconsistencies of earlier programming models have been avoided
There is explicit support for parallel I/O.
I/O is compatible with both process-based and thread-based implementations.
Other Fortran 95 additions:
Several Intrinsics

NUM_IMAGES() returns the number of images,
THIS_IMAGE() returns this image's index between 1 and NUM_IMAGES()
SYNC_ALL() is a global barrier
To only wait for the relevant images to arrive.
SYNC_ALL(WAIT=LIST)
More Intrinsics

SYNC_TEAM(TEAM=TEAM)
SYNC_TEAM(TEAM=TEAM,WAIT=LIST)
START_CRITICAL and END_CRITICAL
Adding Synch Functionality

SYNC_MEMORY().

This routine forces the local image to both complete any outstanding co-array writes into "global" memory and refresh from global memory any local copies of co-array data it might be holding (in registers for example).

Image synchronization implies co-array synchronization.

A call to SYNC_MEMORY() is rarely required

Implicitly called before and after virtually all procedure calls including Co-Array's built in image synchronization intrinsics.
Image and co-array synchronization

- Example: exchanging an array with your north and south neighbors:

  ```
  COMMON/XCTILB4/ B(N,4)[*]
  SAVE /XCTILB4/

  CALL SYNC_ALL(
    WAIT=(/IMG_S,IMG_N/) )
  B(:,3) = B(:,1)[IMG_S]
  B(:,4) = B(:,2)[IMG_N]
  CALL SYNC_ALL(
    WAIT=(/IMG_S,IMG_N/) )
  ```
Array Exchange
Synchronization Explained

The first SYNC_ALL waits until the remote B(:,1:2) is ready to be copied
The second waits until it is safe to overwrite the local B(:,1:2).
Only nearest neighbors are involved in the sync.
It is always safe to replace SYNC_ALL(WAIT=LIST) calls with global SYNC_ALL() calls
    Often is significantly slower.
    Either the preceding or succeeding synchronization may be avoidable.
Synch Optimization

The majority of remote co-array access optimization is minimizing the synchronization
Frequency of synchronization
Cover the minimum number of images

On machines without global memory hardware, array syntax (rather than DO loops) should always be used for remote memory operations

Copying co-array's into local temporary buffers before they are required might be appropriate
Data Parallel Cumulative Sum

- In data parallel programs, each image is either performing the same operation or is idle.
- For example here is a data parallel fixed order cumulative sum:

```plaintext
REAL SUM[*]
CALL SYNC_ALL( WAIT=1 )

DO IMG= 2,NUM_IMAGES()
   IF (IMG==THIS_IMAGE()) THEN
      SUM = SUM + SUM[IMG-1]
   ENDIF
ENDIF
CALL SYNC_ALL( WAIT=IMG )
ENDDO
```
Data Parallel Performance Critique

SYNC_ALL waiting on just the active image improves performance
still NUM.Images() global sync
An Alternative to Data Parallel

- A better alternative may be to minimize synchronization by avoiding the data parallel overhead entirely:

```fortran
REAL SUM[*]
ME = THIS_IMAGE()
IF (ME.GT.1) THEN
   CALL SYNC_TEAM( TEAM=(/ME-1,ME/) )
   SUM = SUM + SUM[ME-1]
ENDIF
IF (ME.LT.NUM_IMAGES()) THEN
   CALL SYNC_TEAM( TEAM=(/ME,ME+1/) )
ENDIF
```
Alternative Performance Analysis

Now each image is involved in at most two sync's: the images just before and just after it in image order. The first SYNCTEAM call on one image is matched by the second SYNCTEAM call on the previous image.
Benefits (or: In Summary)

The Co-Array Fortran synchronization intrinsics can:

- Improve the performance of data parallel algorithms
- Provide implicit program execution control as an alternative to the data parallel approach.
```
Amusement

```
```