Coarray Fortran (CAF) 2.0
Department of Computer Science, Rice University, Houston, TX
Project URL: http://caf.rice.edu

Objectives

| Expressiveness | Support irregular and adaptive applications; support construction of sophisticated parallel applications and parallel libraries |
| Scalability     | Scale to petascale architectures and beyond |
| Orthogonality   | Support complex concepts with minimal language extensions |
| Multithreading  | Exploit multicore and multi-threaded processors |
| Performance     | Deliver top performance: enable users to overlap communication latency with computation |
| Portability     | Support development of portable high performance programs |
| Interoperability| Interoperate with legacy parallel computing models such as MPI, OpenMP, and CUDA |

Key Features

Events for point-to-point synchronization and asynchrony

- **event**: synchronization object for anonymous pairwise coordination
- Safe synchronization space: can allocate as many events as desired
- **event_init**: event initialization
- **event_notify**: nonblocking signal to an event; a pairwise fence between sender and target image
- **event_wait**: blocking wait for notification of an event
- **event_trywait**: nonblocking check to see if an event has been signaled

Asynchrony support

- Completion of asynchronous operations managed two ways:
  - Explicit model: notify an event upon completion
  - Implicit model: both `cofence` and `finish` block "round up" outstanding operations
- Asynchronous collective operations support either explicitly or implicitly

- Predicated asynchronous copy overlaps computation and communication
  - `copy_async(dest, src, cr, cr, dr)`
- `cr` (copy ready: optional): an event indicating that the data may now be copied from src to dest
- `dr` (source ready: optional): an event indicating that the source data may be safely overwritten

CAF 2.0 offers greater expressiveness than the coarray features in Fortran 2008 yet it still yields performance comparable to that of MPI

Partitioned Global Address Space (PGAS) memory view

Like Unified Parallel C (UPC) and Chapel, CAF 2.0 features a two-level partitioned view of memory in which data is either local or remote. Unlike them, however, accesses that may touch remote memory are always explicitly flagged with square brackets.

![Partitioned Global Address Space (PGAS) memory view](image)

Array allocation in CAF 2.0:

- `integer, allocatable :: A(:)[*]`: declares coarray A accessible to all image processes
- `integer :: B(1:40)`: declares local array B, not accessible to other image process

Process subsets: Teams

- `team:: A(1:50)[*]`: ordered sequence of process images
- `team:: B(1:40)`: block-structured

Accessing a coarray from a specific team:

- `b(1:100) [1]`: accesses elements in coarray on image 1 of the current default team.
- `b(1:100) [1]@some_team`: accesses elements in coarray on image 1 of `some_team` within `myteam`.

TEAM INTRINSICS AND STATEMENTS

- `team_world` predefined team that consists of all images (equivalent to MPI COMM_WORLD).
- `team_default` the default team for the current scope (initially `team_world`).
- `team_rank(myteam)` returns the team-relative rank of a given image process.
- `team_size(myteam)` returns the number of images in a given team.
- `team_split(parent_team, color, key, new_team)` forms new teams as subsets of an existing one; equivalent to MPI COMM_SPLIT.
- `with team myteam - end with team myteam` sets the default team to `myteam` within its scope.

A rich set of collective operations

CAF 2.0 offers greater expressiveness than the coarray features in Fortran 2008 yet it still yields performance comparable to that of MPI

Other CAF 2.0 features

- **Synchronization**: block-structured finish... and finish construct as in X10
- **Mutual exclusion**: locks, critical sections, and locksets
- **Function shipping**: synchronous invocation of remote functions using a `spawn` statement

Development of CAF 2.0 is supported by the Department of Energy’s Office of Science under cooperative agreements DE-FG02-07ER25800 and DE-FG02-06ER25754.

The Rice CAF 2.0 compiler uses the Rose compiler (Lawrence Livermore National Lab), the Open Fortran Parser (Los Alamos National Lab) and the GASNet communications library (University of California Berkeley).

Contributors

- John Mellor-Crummey (PI)
- Mark Krentel
- Karthik Murphy
- Dung Nguyen
- William N. Scherer III
- Scott Warren
- Chaoran Yang

John Mellor-Crummey (PI)