Parallel Runtime Interface for Fortran

A compiler and implementation independent interface for supporting the parallel features of the Fortran language

https://go.lbl.gov/prif

December, 2023
Motivation

What’s this for?

- Enable a compiler to target multiple implementations of PRIF
  - I.e. enable a vendor to supply their own parallel runtime
- Enable a PRIF implementation to be used by multiple compilers
- Isolate a compiler’s support of the parallel features of the language from any particular details of the communication infrastructure
- Our group’s experience with UPC and OpenCoarrays has shown this to be valuable
Parallel Features

- **Statements**
  - Synchronization
    - Explicit: sync all, sync images, sync memory, sync team
    - Implicit: allocate, deallocate, stop, end, move_alloc
  - Events: event post, event wait
  - Notify: notify wait
  - Error termination: error stop
  - Locks: lock, unlock
  - Failed images: fail image
  - Teams: form team, change team
  - Critical sections: critical, end critical

- **Coarray Accesses** ([...])
- **Intrinsic functions**: num_images, this_image, lcobound, ucobound, team_number, get_team, failed_images, stopped_images, image_status, coshape, image_index

- **Intrinsic subroutines**
  - Collective subroutines: co_sum, co_max, co_min, co_reduce, co_broadcast
  - Atomic subroutines: atomic_add, atomic_and, atomic_cas, atomic_define, atomic_fetch_add, atomic_fetch_and, atomic_fetch_or, atomic_fetch_xor, atomic_or, atomic_ref, atomic_xor
  - Other subroutines: event_query

- **Types, kind type parameters, and values**
  - Intrinsic derived types: event_type, team_type, lock_type, notify_type
  - Atomic kind type parameters: atomic_int_kind and atomic_logical_kind
  - Values: stat_failed_image, stat_locked, stat_locked_other_image, stat_stopped_image, stat_unlocked, stat_unlocked_failed_image
PRIF Design Overview

Parallel Features Directly Translatable to Use of Fortran Library

```fortran
me = this_image()
call prif_this_image(image_index=me)

call co_sum(a, result_image=1)
call prif_co_sum( &
a, result_image=1_c_int)

arr[1] = some_calc()
call prif_put( &
arr_coarray_handle, &
int([1], c_intmax_t), &
some_calc(), &
int(storage_size(arr)/8, c_size_t), &
c_loc(arr))
```
PRIF Design Overview: Responsibilities

**Compiler**
- Establish and initialize static coarrays prior to main
- Track corank of coarrays
- Track local coarrays for implicit deallocation when exiting a scope
- Initialize a coarray with SOURCE= as part of allocate-stmt
- Provide prif_critical_type coarrays for critical-constructs
- Provide final subroutine for all derived types that are finalizable or that have allocatable components that appear in a coarray
- Variable allocation status tracking, including use of MOVE_ALLOC

**PRIF Implementation**
- Track coarrays for implicit deallocation at end-team-stmt
- Allocate and deallocate a coarray
- Reference a coindexed-object
- Team stack abstraction
- form-team-stmt, change-team-stmt, end-team-stmt
- Intrinsic functions related to parallel Fortran, like num_images, etc
- Atomic subroutines
- Collective subroutines
- Synchronization statements
- Events, notify
- Locks
- critical-construct
Next Steps

- Submit PRIF Design Doc to LLVM-Project Repository
- Finish tests for proper behaviour of parallel features
- Finish implementation in Caffeine
- (Find help with) Integration into flang
- Track progress: [https://github.com/BerkeleyLab/flang-testing-project/projects/7](https://github.com/BerkeleyLab/flang-testing-project/projects/7)
- Solicit Feedback:
  - Discourse Post
  - Email: lbl-flang@lbl.gov
  - Specification Working Draft: [https://go.lbl.gov/prif](https://go.lbl.gov/prif)
  - We welcome issues and PRs at the above GitHub Repository
Questions?

- Email: lbl-flang@lbl.gov
- Specification Working Draft: https://go.lbl.gov/prif
Acknowledgements

- This research is supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration
- This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231
Who We are

We have experience developing parallel runtimes, parallel applications, Flang frontend parallel features, and parallel unit tests:


Why not OpenCoarrays?

- Is hardwired to gfortran, e.g., many procedures manipulate gfortran-specific descriptors
- The interface implicitly assumes a MPI backend
- Only the MPI layer is maintained (GASNet & OpenSHMEM layers are legacy codes)
- Lacks full support for some parallel features (e.g., teams).
- Has a bus factor of ~1.
What is GASNet?