A Full-Stack Exploration of Language-Based Parallelism in Fortran 2023

Katherine Rasmussen, Damian Rouson, Dan Bonachea, Brad Richardson
Computer Languages and Systems Software Group and NERSC
Lawrence Berkeley National Laboratory, USA

fortran.lbl.gov

fortran@lbl.gov

Abstract

This poster explores native parallel features in Fortran 2023 through the lens of supporting applications with libraries, compilers, and parallel runtimes. The language revision informally named Fortran 2008 introduced parallelism in the form of Single Program Multiple Data (SPMD) execution with two broad feature sets: (1) loop-level parallelism via `do concurrent` and (2) a Partitioned Global Address Space (PGAS) comprised of distributed "coarray" data structures. Fortran's native parallelism has demonstrated high performance [1] and reduced the burden of inserting what sometimes amounts to more directives than code. Several compilers support both feature sets, typically by translating `do concurrent` into serial `do` loops annotated by parallel directives and by translating SPMD/PGAS features into direct calls to a communication library. Our research focuses primarily on two questions: (1) can the compiler's parallel runtime library be developed in the language being compiled (Fortran) and (2) can we define an interface to the runtime that liberates compilers from being hardwired to one runtime and vice versa. We are answering these questions by developing the Parallel Runtime Interface for Fortran (PRIF) [2] and the Co-Array Fortran Framework of Efficient Interfaces to Network Environments (Caffeine) [3]. Caffeine is initially targeting adoption by LLVM Flang, a new open-source Fortran compiler developed by a broad community in industry, academia, and government labs. We are also exploring the use of these features in Inference-Engine, a deep learning library designed to facilitate neural network training and inference for high-performance computing applications written in modern Fortran.

References

[1] S. Garain, D. S. Balsara, and J. Reid, "Comparing Coarray Fortran (CAF) with MPI for several structured mesh PDE applications," *Journal of Computational Physics*, vol. 297, pp. 237–253, 2015, doi:10.1016/j.jcp.2015.05.020.

[2] D. Bonachea, K. Rasmussen, B. Richardson, and D. Rouson, "Parallel Runtime Interface for Fortran (PRIF) Specification, Revision 0.4," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-2001604, July 2024, doi:10.25344/S4WG64.

[3] D. Rouson and D. Bonachea, "Caffeine: CoArray Fortran Framework of Efficient Interfaces to Network Environments," in *Proceedings of the Eighth Annual Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC2022)*, November 2022, doi:10.25344/S4459B.

A Full-Stack Exploration of Language-Based Parallelism in Fortran 2023

Katherine Rasmussen, Dan Bonachea, Brad Richardson, Damian Rouson
Lawrence Berkeley National Laboratory, USA
fortran.lbl.gov

Background

Parallel Fortran

The first widely used programming language, Fortran evolved to embrace modern programming paradigms with the Fortran 2003 advent of object-oriented programming and the Fortran 2008 introduction of parallel programming. Expanded further in 2018 and 2023, the native parallel features support the execution of multiple images (program instances, e.g., SPMD processes); coarrays (distributed data structures supporting one-sided communication); synchronization mechanisms; collective procedures; atomic variables; events (counting semaphores); locks and critical blocks; image teams (subsets); and loop-level parallelism via **`do concurrent`**.

LLVM Flang

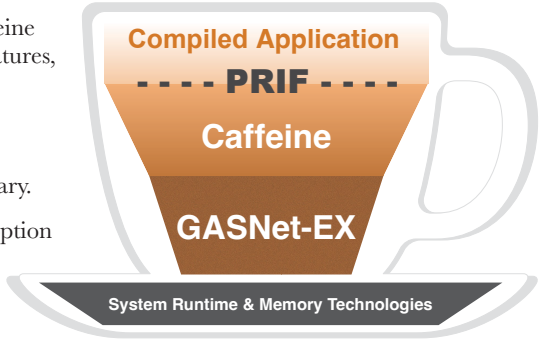
The open-source LLVM Flang Fortran compiler parses parallel Fortran, but cannot yet compile programs for multi-image execution. We develop an interface and a supporting runtime library that will add parallel support to Flang and other compilers.

Parallel Runtime Interface for Fortran (PRIF)

- PRIF is the first compiler- & runtime-library-agnostic interface to multi-image parallel Fortran features: any compiler targeting PRIF can use any library that supports PRIF.
- PRIF specifies a "**`prif`**" Fortran module containing derived types, named constants, and procedure interfaces that support parallel Fortran features.
- PRIF procedures correspond to features defined by the Fortran specification, e.g.,
  - **`prif_num_images`** supports the Fortran intrinsic function **`num_images`**
  - **`prif_allocate_coarray`** supports coarray declarations
- A compiler targeting PRIF is responsible for transforming parallel features in the user's Fortran source code into corresponding PRIF library procedure calls.
- For more information, see the PRIF specification:
  D. Bonachea, K. Rasmussen, B. Richardson, and D. Rouson, "Parallel Runtime Interface for Fortran (PRIF) Specification, Revision 0.4," Lawrence Berkeley National Laboratory (LBNL), Tech. Rep. LBNL-2001604, July 2024, doi:10.25344/S4WG64.

Caffeine: A PRIF Implementation

- Caffeine is a PRIF implementation written primarily in Fortran.
- Caffeine currently supports most commonly used parallel Fortran features including program launch and termination; coarray allocation and communication; collective subroutines; synchronization; image enumeration; and more.
- We are continuing work on Caffeine to support all parallel Fortran features, with atomics and events feature support up next.
- Caffeine runs atop GASNet-EX, LBNL's exascale networking library.
- We are initially targeting the adoption of PRIF and Caffeine by the LLVM Flang compiler.



- For more on Caffeine, see:
  Rouson & Bonachea (2022) "*Caffeine: CoArray Fortran Framework of Efficient Interfaces to Network Environments*" LLVM-HPC'22 doi:10.25344/S4459B

Research Questions

- Can the compiler's parallel runtime library be developed in the language being compiled (Fortran)?
- Can we define an interface that liberates parallel Fortran compilers from being hardwired to one parallel runtime library and vice versa?

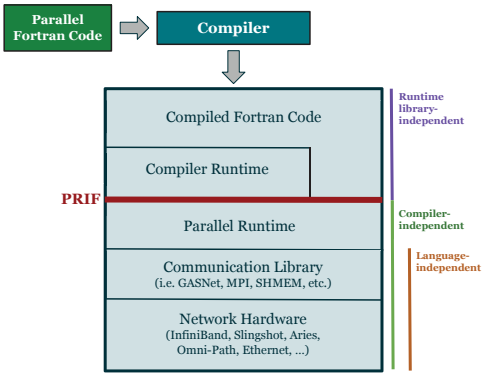


Figure 1: Parallel Fortran Software Stack including PRIF

Fortran source code with parallel features

```
me = this_image()
call co_sum(a, result_image=1)
```

Corresponding PRIF calls

```
call prif_this_image(image_index=me)
call prif_co_sum(a, result_image=1_c_int)
```

Figure 2: Fortran intrinsic procedure calls and their PRIF equivalents

Outcomes

- Contributed static semantics tests, compile-time error checking, and frontend bug fixes to Flang.
- Published the PRIF specification and submitted a design document to LLVM.
- Developing the Caffeine parallel runtime with many widely used features already supported.
- Investigating the use of parallel Fortran features in deep learning algorithms in LBNL's Inference-Engine library: go.lbl.gov/inference-engine.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research.
This research was supported in part 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.