Lowering and Runtime Support for Fortran's Multi-Image Parallel Features using LLVM Flang, PRIF, and Caffeine

LLVM-HPC2025: The 11th Workshop on the LLVM Compiler Infrastructure in HPC

Dan Bonachea	Lawrence Berkeley National Laboratory (LBNL)
Katherine Rasmussen	Lawrence Berkeley National Laboratory (LBNL)
Damian Rouson	Lawrence Berkeley National Laboratory (LBNL)
Jean-Didier Pailleux	SiPearl, R&D
Etienne Renault	SiPearl, R&D
Brad Richardson	Amentum

Read the paper

Fortran's Multi-Image Parallel Features

Multi-image

Coarray features
(allocation, deallocation, accesses, ...)
+
Multi-image parallel features
(synchronization, collective subroutines, atomic subroutines, teams of images, …)

Status

Introduced in Fortran 2008 standard, expanded in Fortran 2018, 2023

LLVM only supports parsing since 2020
Lack of maturity with regards to other compilers:
CCE (2008), OpenUH (2010), ifort/ifx (2011), GCC (2015), nAG (2022), etc.

The absence of robust multi-image support in LLVM could discourage Fortran users: closing the gap with other compilers is desirable

This Paper !

Demonstrate adding support for multi-image features into LLVM

Assess the viability of the lowering into LLVM IR through an extensive and comparative evaluation

Minimal example of a multi-image Fortran program

`this_image`: intrinsic that returns the 1-based index of the calling image

`num_images`: intrinsic that returns returns the number of images in the SPMD parallel execution.

```
program main
    integer :: i
    if (this_image() .eq. 1) then
        i = num_images()
    endif
end program main
```

Compilers supporting multi-image Fortran rely on a parallel runtime library (e.g., MPI, GASNet, proprietary layers) to provide the communication support for multi-image features

Parallel Runtime Interface for Fortran (PRIF)

PRIF

Runtime Agnostic Library

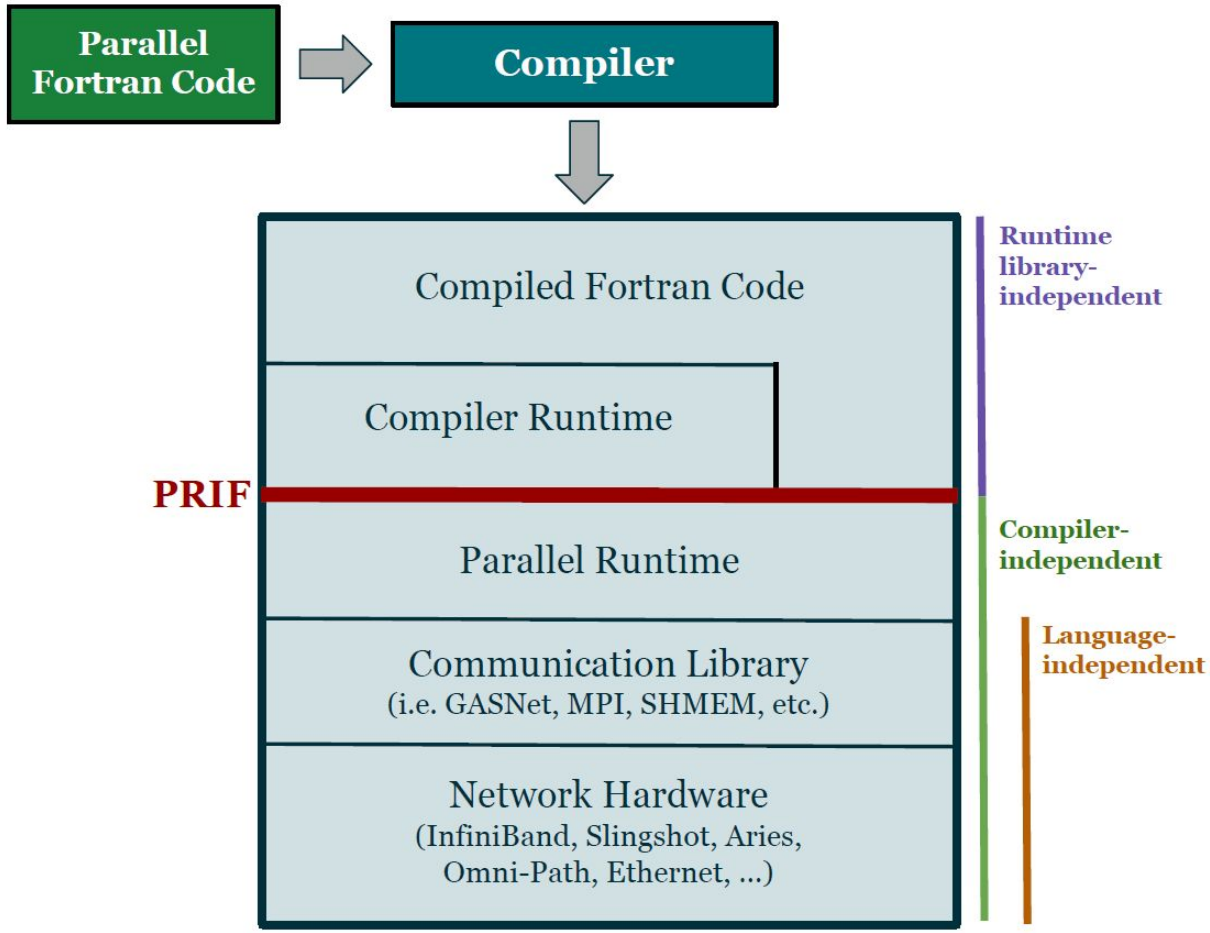
Generic interface to simplify adding support for multi-image features to a compiler

Open standard

Governed by a multi-institutional committee

Portable

Across shared- and distributed-memory systems



CoArray Fortran Framework of Efficient Interfaces to Network Environments (Caffeine)

Caffeine

First implementation of PRIF

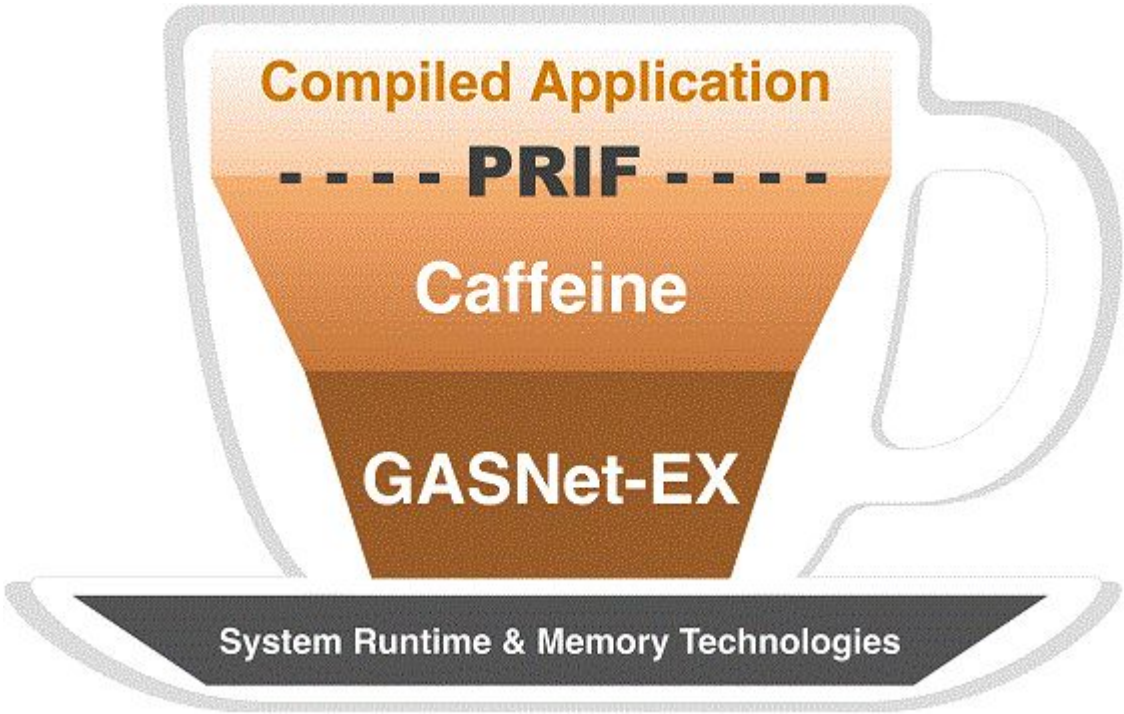
Development started late 2021

Written mostly in Fortran

Demonstrates that Fortran's non-parallel features can provide support for the multi-image features

GASNet-EX

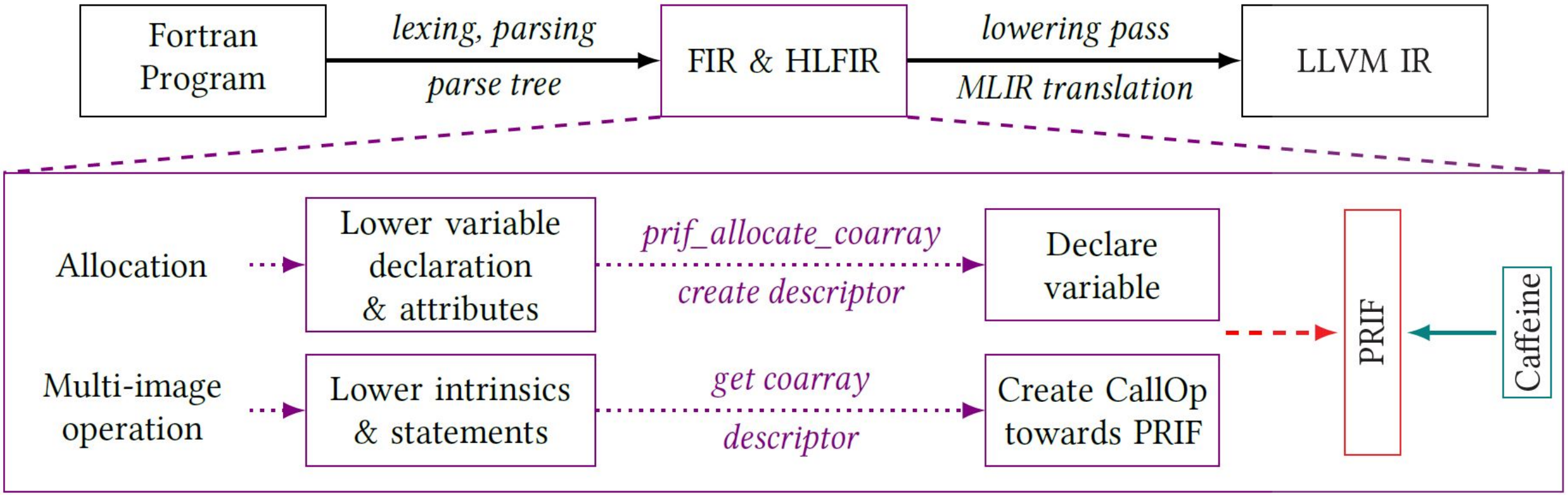
Implemented atop the GASNet-EX communication library



Caffeine's support for the multi-image parallel features of Fortran (as of Caffeine v0.5.2)

Multi-image Fortran Feature	Status
Program startup and shutdown (incl. normal and error termination): `STOP`, `ERROR STOP`, `END PROGRAM` statements	yes
Collective subroutines: `CO_{BROADCAST,SUM,MIN,MAX,REDUCE}`	yes
Image queries: `THIS_IMAGE`, `NUM_IMAGES`, etc, intrinsic functions	yes
Synchronization: `SYNC {ALL,IMAGES,MEMORY,TEAM}` statements	yes
Storage management: Coarray allocation, deallocation and coarray aliases	yes
Coarray Queries: `LCOBOUND`, `UCOBOUND`, `COSHAPE`, etc.	yes
Contiguous and strided coarray access: Coarray puts and gets	yes
Teams: `TEAM_TYPE` intrinsic type and `{FORM,CHANGE,END} TEAM` statements	yes
Events: `EVENT_TYPE` intrinsic type, `EVENT_QUERY` subroutine and `EVENT {POST,WAIT}` statements	yes
Notifications: `NOTIFY_TYPE` intrinsic type and `NOTIFY WAIT` statement	yes
Atomics: `ATOMIC_{INT,LOGICAL}_KIND` kind parameters and `ATOMIC_{DEFINE,REF,...}` subroutines	yes
Critical construct: `CRITICAL` and `END CRITICAL`	no
Locks: `LOCK` and `UNLOCK` statements	no
`FAIL IMAGE` statement	no

Lowering in LLVM



Lowering in LLVM: details

Allocations

Step #1

Tag coarray variables with a specific semantic symbol

Step #2

Intercept coarray allocations and substitute them with prif_allocate_coarray

Coindexed expressions and MIF-related intrinsics

Step #1

Lookup to retrieve coarray descriptor handle

Step #2

Translate HLFIR / FIR calls directly into PRIF calls

Lowering in LLVM: Example

```
program main
    integer :: i
    if (this_image() .eq. 1) then
        i = num_images()
    endif
end program main
```

```
...
fir.call @_QMprifPprif_this_image_no_coarray(%3,
     %4) fastmath<contract> : (!fir.box<none>, !
    fir.ref<i32>) -> ()
%5 = fir.load %4 : !fir.ref<i32>
%c1_i32 = arith.constant 1 : i32
%6 = arith.cmpi eq, %5, %c1_i32 : i32
fir.if %6 {
  %7 = fir.alloca i32
  fir.call @_QMprifPprif_num_images(%7) fastmath
      <contract> : (!fir.ref<i32>) -> ()
  %8 = fir.load %7 : !fir.ref<i32>
  hlfir.assign %8 to %2#0 : i32, !fir.ref<i32>
}
...
```

Evaluation: Benchmark Description

Caf-testsuite

Suite of correctness tests for coarrays

Split into micro-benchmarks and benchmarks
Check semantics and behavior

Metrics

Latency and Bandwidth

Cafbench

Performance benchmarking suite

Evaluates scalability performance in multi-image execution environment

Metrics

Latency and Bandwidth

Evaluation: Software / Hardware Experimental Setup

LLVM Flang prototype	prif-llvm-hpc25 (commit c17dd15)
GNU Gfortran	15.0.0
HPE CCE Fortran	18.0.0
GASNet-EX	2024.5.0
Open-MPI	5.0.8

Shared Memory System

AArch64 Neoverse-V1

Graviton3, 64 cores@2.5 Ghz, RHEL9, kernel v5.14

x86_64 Intel Xeon

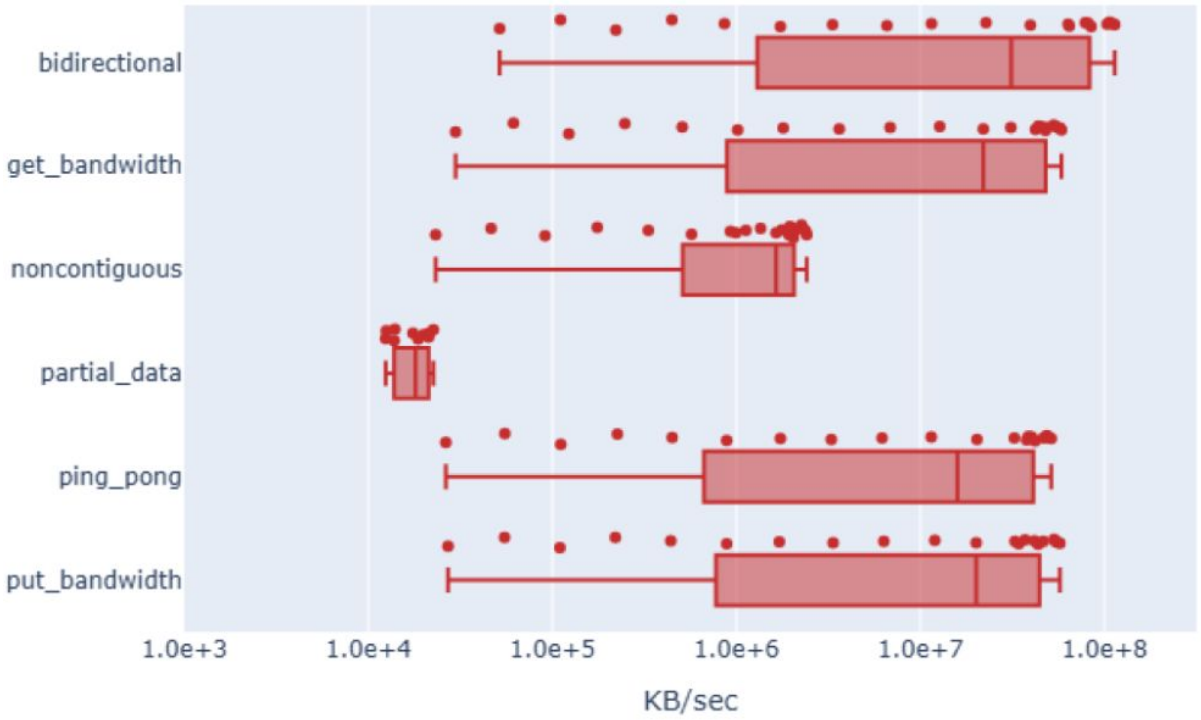
SapphireRapids, 80 cores@2.1 Ghz, RHEL9, kernel v5.14

Distributed Memory System

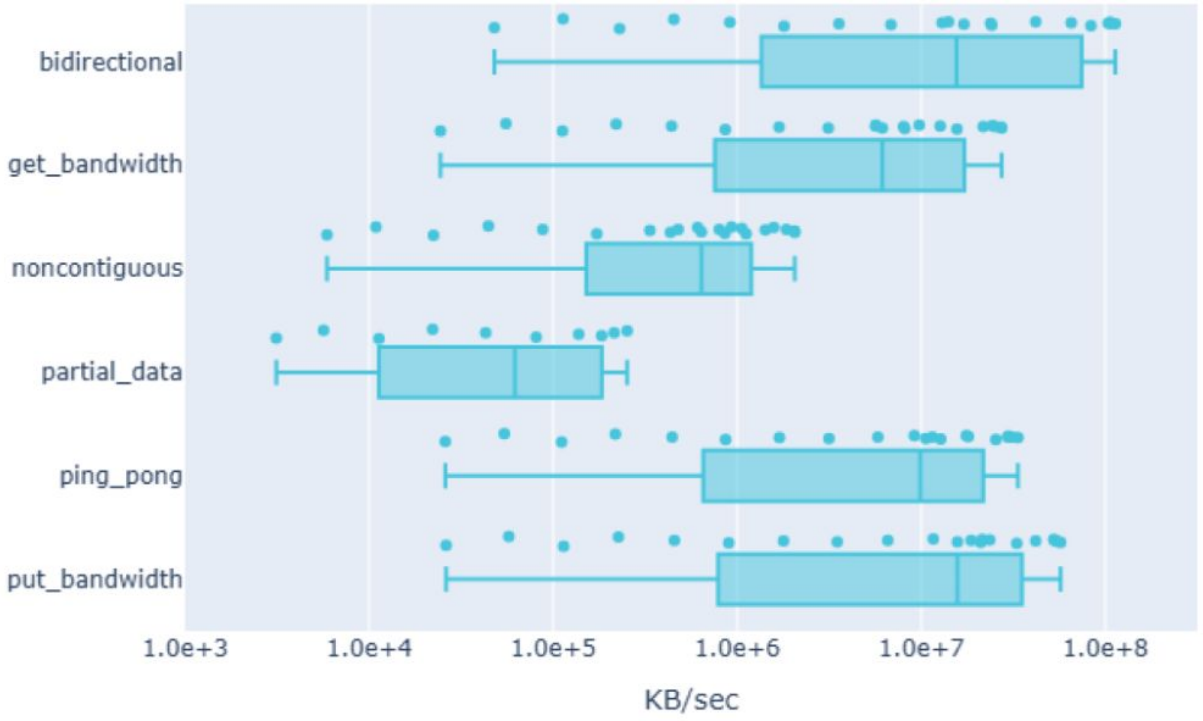
HPE Cray EX supercomputer

NERSC Perlmutter, CPU partition
Each node is 2x64 cores AMD EPYC 7663 Milan CPU @2.45 Ghz

Caf-testsuite on shared-memory systems



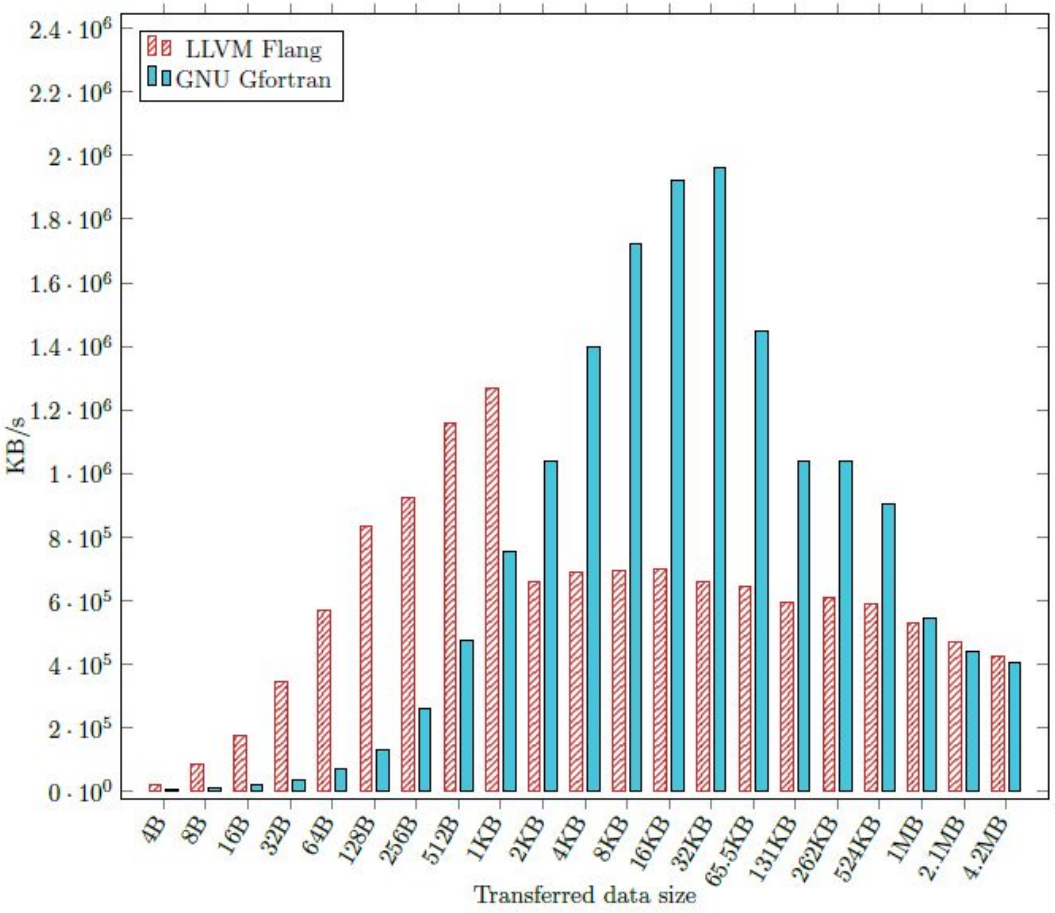
(c) `flang` and Caffeine on AArch64



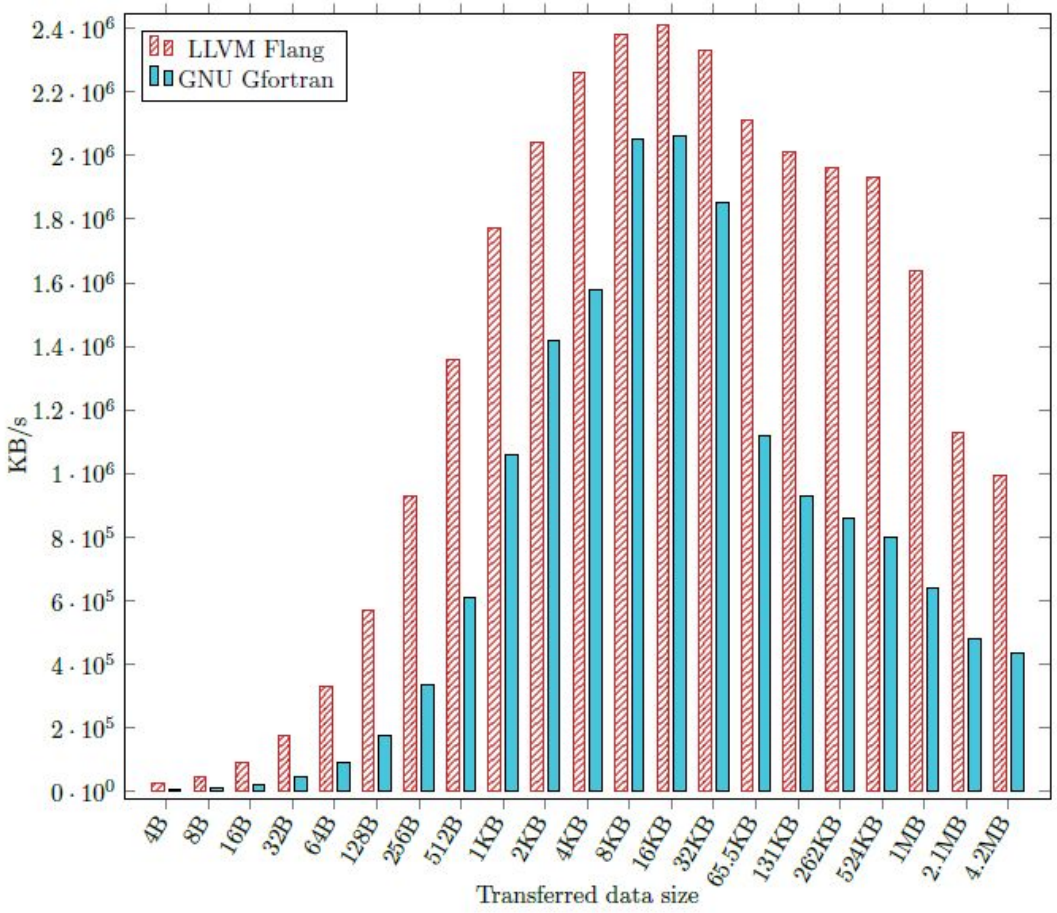
(d) GNU Gfortran and OpenCoarrays on AArch64

Similar results observed on x86_64

Caf-testsuite: zooms into noncontiguous test

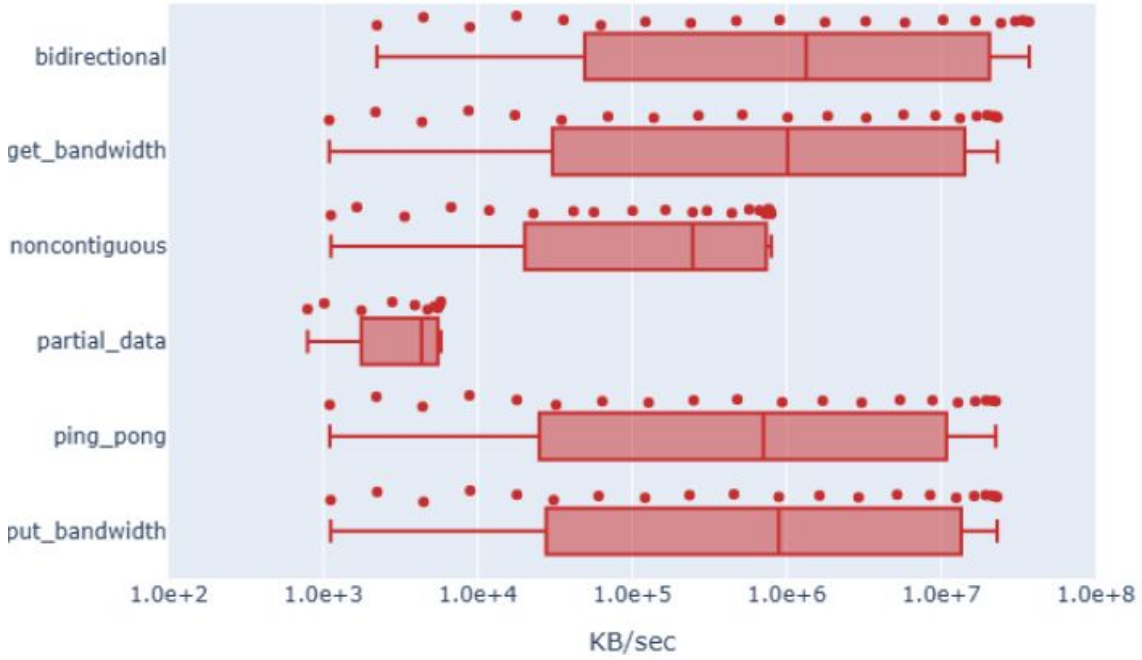


(a) x86-64

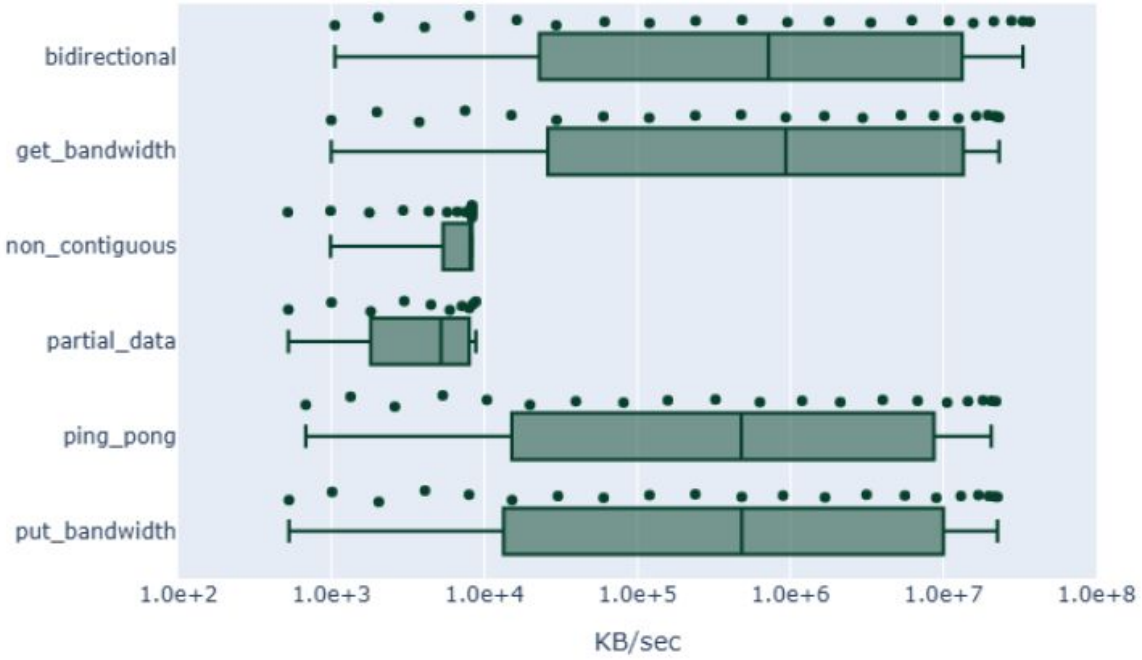


(b) AArch64

Caf-testsuite: Distributed Memory across two nodes



(a) `flang` and Caffeine



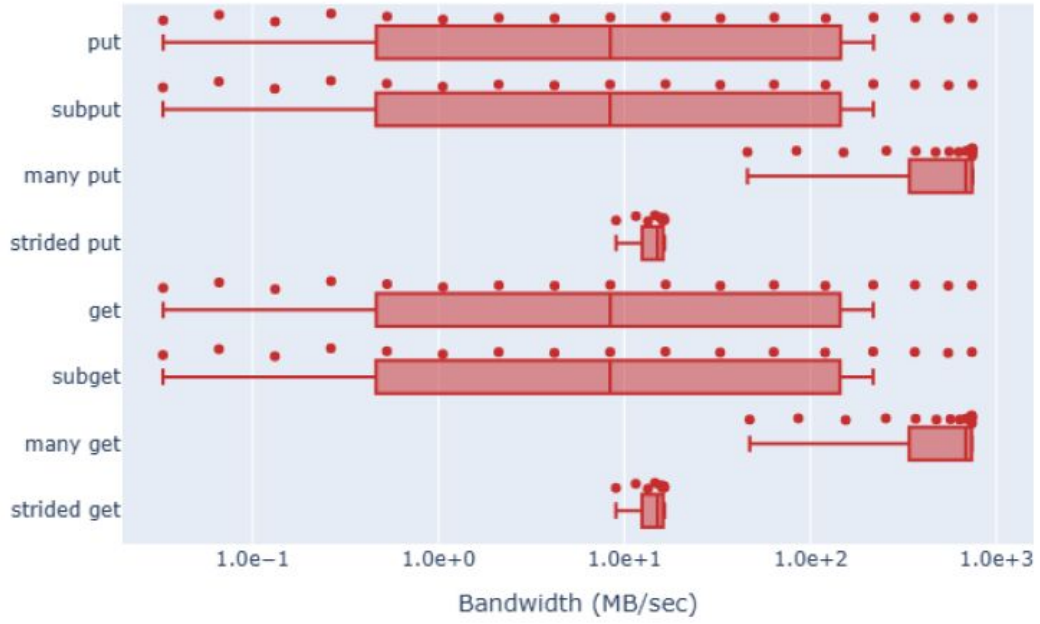
(b) HPE Cray Fortran

Caf-testsuite: latency across all systems

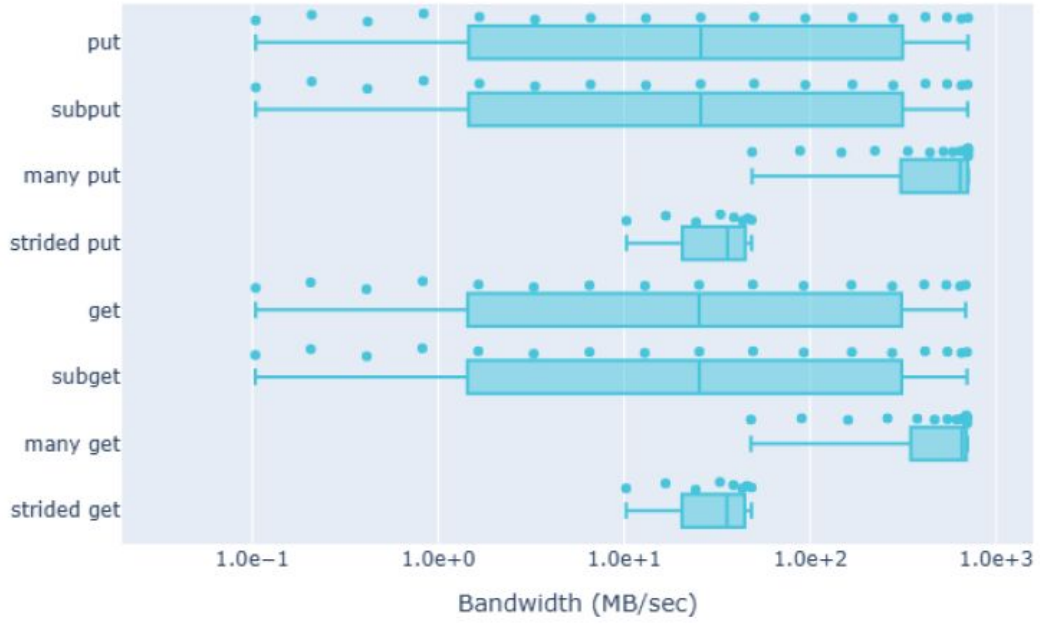
Shared Memory	AArch64 Flang + Caffeine	AArch64 Gfortran + OpenCoarrays	x86_64 Flang + Caffeine	x86_64 Gfortran + OpenCoarrays
PUT	119 ns	141 ns	63 ns	178 ns
GET	129 ns	136 ns	63 ns	180 ns

Distributed Memory	Flang + Caffeine	HPE CCE
PUT	3.64 ms	7.17 ms
GET	3.53 ms	4.30 ms

Cafbench



(a) flang and Caffeine on AArch64



(b) GNU Gfortran and OpenCoarrays on AArch64

Results are consistent with Caf-testsuite evaluation

HPE CCE currently unable to compile Cafbench due to compiler defects, which have been reported to the vendor

Conclusion

Evaluation

Caf-testsuite
for correctness and benchmarking

Cafbench
for benchmarks and micro-benchmarks

Shared and Distributed Memory
to stress various configurations

Upstream LLVM

SPMD program initialization
and -fcoarray option (PR #151675)

Basic image identification intrinsics
`this_image`, `num_images` (PR #154081)

Image synchronization statements
`sync all`, `sync images`, `sync memory` (PR #154166)

Collective subroutine instrinsics
`co_broadcast`, `co_min`, `co_max`, `co_sum` (PR #154770)

Current and Future Work

New MLIR Dialect: Multi-Image Fortran (MIF)

Centralize all lowering in one place

Better organization and readability

Similar to the CUF dialect

Extension of Fortran,
Coexists with FIR/HLFIR,
Define operations, types and transformation pass

Upstream LLVM

Moving all operations into MIFDialect: PR #161179
Team features (without coarray) : PR #165573

Under review

Program Terminaison (stop and error stop): PR #166787

Listing 3: MIF Dialect output example

```
func.func @_QQmain() attributes {fir.bindc_name =
    "MAIN"} {
  ...
  %4 = mif.this_image -> i32
  %c1_i32 = arith.constant 1 : i32
  %5 = arith.cmpi eq, %4, %c1_i32 : i32
  fir.if %5 {
    %6 = mif.num_images -> i32
    hlfir.assign %6 to %2#0 : i32, !fir.ref<i32>
  }
  ...
}
```

Current and Future Work

Improvement

Better support for coarray dummy arguments

manage the offset with the PRIF 0.6 updated revision

Support for allocatable members of coarrays

use dedicated PRIF procedures for this type of allocation

Optimizing access operations for coarray

Upstream Roadmap

Coarray allocation and deallocation

Basic features

`coshape`, `lcobound`, `ucobound`, `image_index`, `image_status`, `fail_image`, `failed_images`, `stopped_images`, …

Coarray access operations

PUT and GET

…