UNIVERSITY OF CAMBRIDGE

Bringing a scientific application to the distributed world using PGAS

Performance, Portability and Usability of Fortran Coarrays

Jeffrey Salmond

August 15, 2017

Research Software Engineering
University of Cambridge

Summary

We aim to investigate the

- performance,
- portability and
- usability

of Fortran 2008 coarrays for porting large, complex scientific applications to distributed memory.

Outline

- Coarrays & implementations of coarrays
- Synthetic benchmarks
- Porting a scientific code: TROVE

What are Coarrays

Coarrays are:

- a PGAS extension of Fortran,
- Fortran 2008 adds remote access to variables,
- Fortran 2015 adds collectives, atomics and teams.

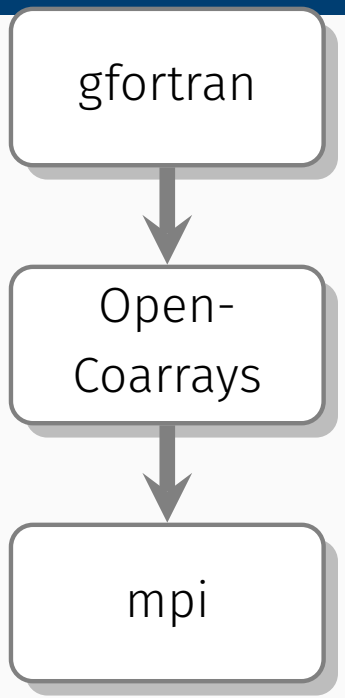
```
real :: x(10)[*]

x(:) = x(:)[1]
 !call mpi_get(x, 10, MPI_REAL, 1, disp, 10, MPI_REAL, mywin, ierr)

sync all
 !mpi_barrier(MPI_COMM_WORLD, ierr)

call co_sum(x, result_image=1)
 !mpi_reduce(x, x, 10, MPI_REAL, MPI_SUM, 0, MPI_COMM_WORLD, ierr)
```

Coarray implementations

	F2008	F2015	OpenMP	MPI
gfortran + OpenCoarrays	✓	✓	✓	✓
Intel Parallel Studio	✓	×	✓	✓
OpenUH + GASNet	✓	?	✓	
Cray	✓	✓	?	✓

Coarray implementations: gfortran + OpenCoarrays

- gfortran frontend generates calls to libcaf
  `_gfortran_caf_get (...)`
- OpenCoarrays supplies libcaf
- OpenCoarrays libcaf calls into a standard MPI library

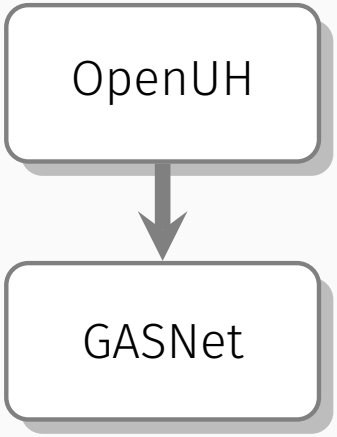


✓ gfortran + mpi are very widely supported
  ⇒ can run (almost) anywhere
✓ gfortran can compile most things
× MPI not the ideal target for implementing coarray support
× gasnet backend exists but is ‘unsupported’ (and doesn’t compile)

- Intel implementation uses a similar structure

Coarray implementations: OpenUH + GASNet

- OpenUH compiler frontend
- communication backends provided by GASNet



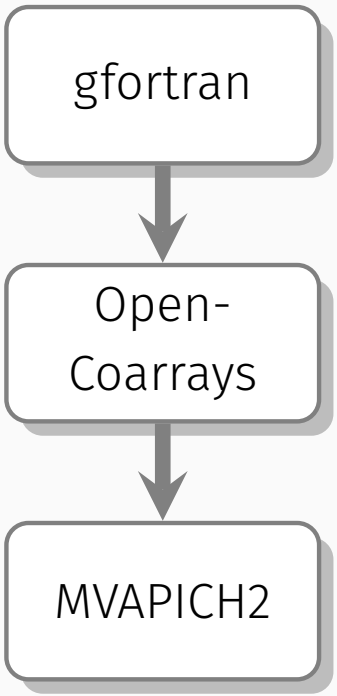
✓ potentially higher performance

× OpenUH as compliant with complex (nasty) science code

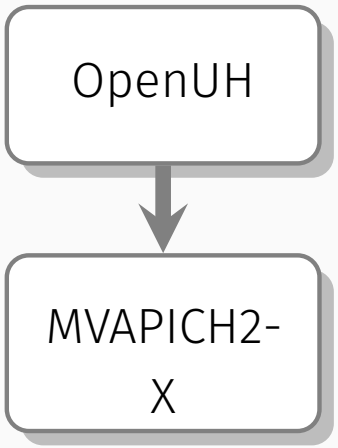
× OpenUH not simple to deploy

Coarray implementations: using MVAPICH2-X

Use MVPAICH2 MPI



Use MVPAICH2-X GASNet conduit



Synthetic Benchmarks

- EPCC Fortran Coarray micro-benchmark suite
- OSU microbenchmarks.

All measurements performed on with

- Intel Broadwell (E5-2650)
- Mellanox EDR

OpenCoarrays
: gfortran 7.1.0 + OpenCoarrays 1.8.10 + MVAPICH2 2.2

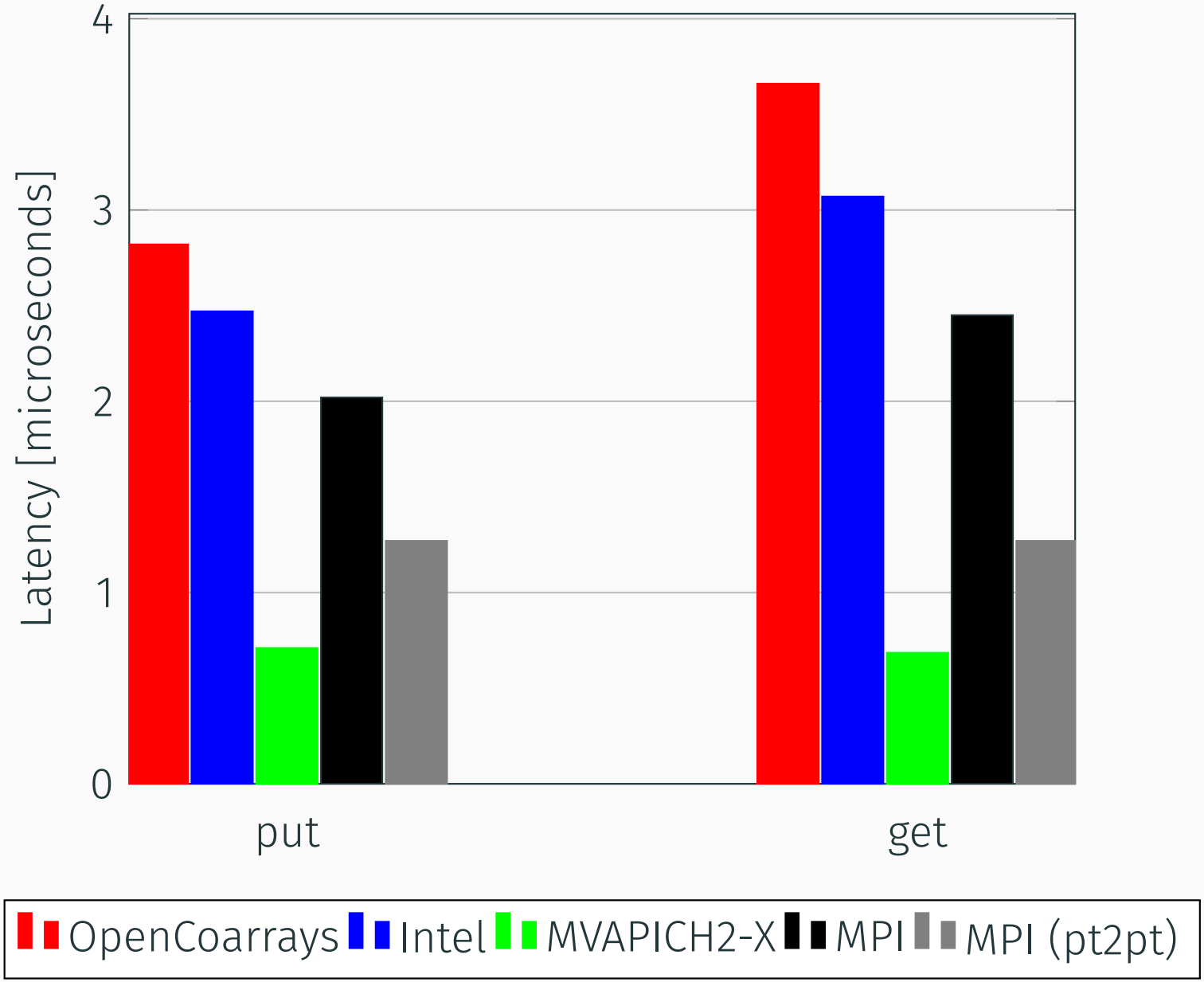
Intel
: Intel Parallel Studio (Intel compiler + Intel MPI) 17.4

MVPAICH2-X
: OpenUH 3.1.0 + MVPIACH2-X 2.2

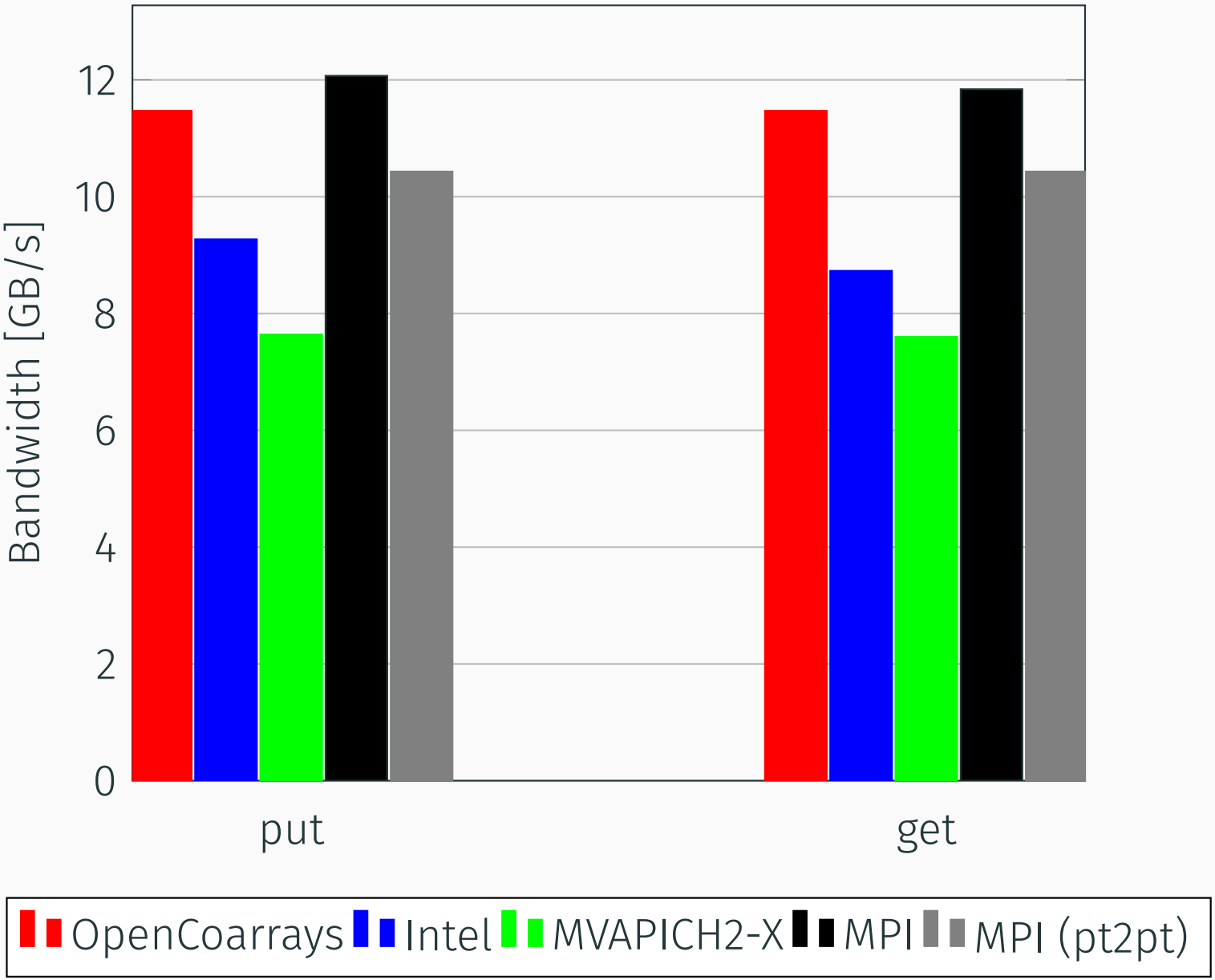
MPI put/get (using MPI-3 put and get)
: gfortran 7.1.0 + MVAPICH2-X 2.2

MPI (p2p) (using MPI send to fake puts and gets)
: gfortran 7.1.0 + MVAPICH2-X 2.2

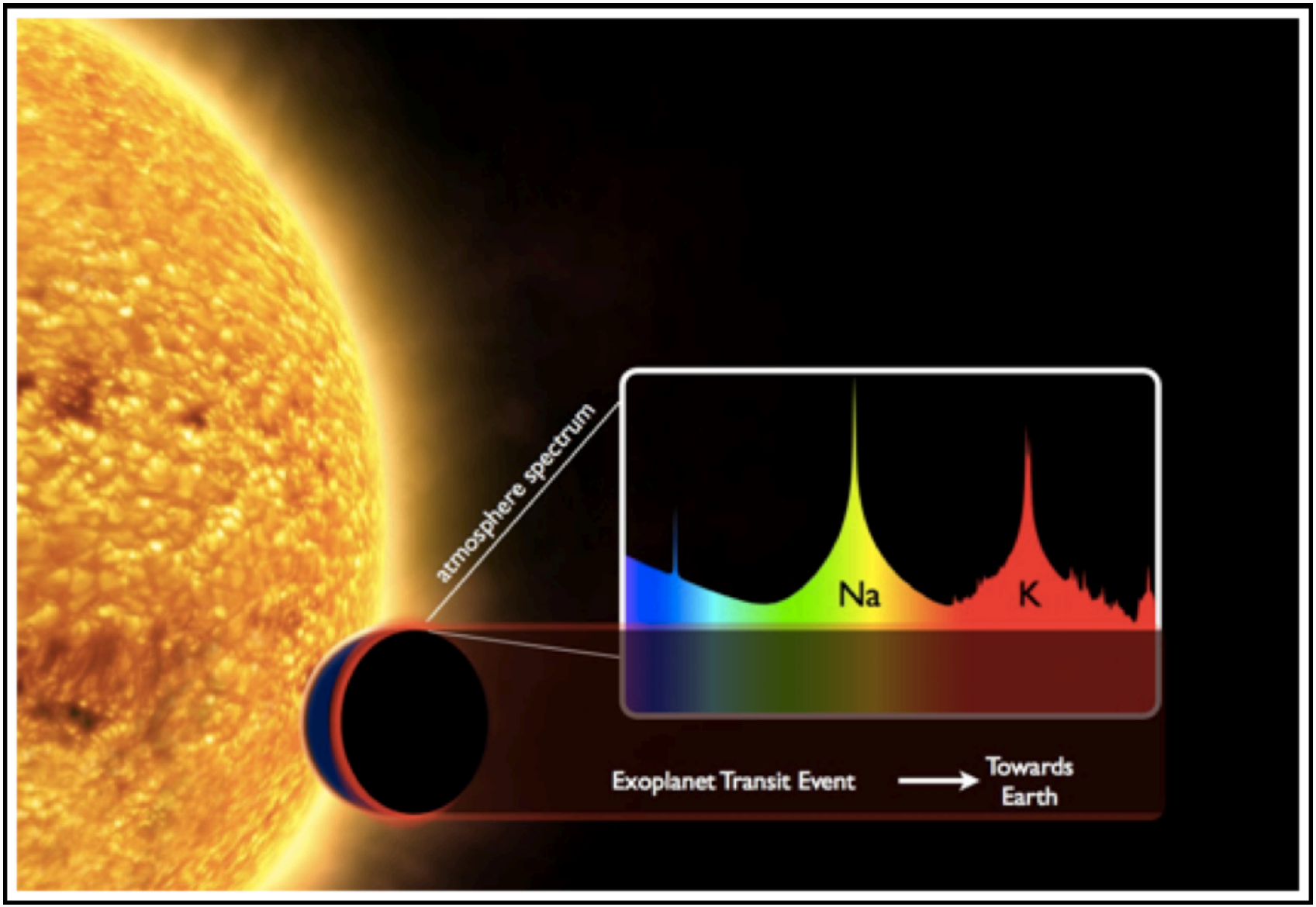
Synthetic Benchmarks: Latency



Synthetic Benchmarks: Bandwidth



TROVE


atmosphere spectrum
Na
K
Exoplanet Transit Event
Towards
Earth

TROVE

- part of the ExoMol (exoplanet molecular line search) project
- looking at the composition of atmospheres on exoplanets
- ultimately searching for aliens!

TROVE has been developed by Sergey Yurchenko (currently at UCL)

- under active development with many contributors
- written in modern Fortran
- developed targeting shared memory :-(
- $\sim$ 160k lines of code

TROVE: Why distributed memory

- TROVE has long run-times (>1M CPU Hours)
- Construction & diagonalisation of 1000s of matrices
- Matrices of size 1M x 1M
- Scientists want to compute bigger problems

Why Coarrays?

- Translating a large code base to distributed memory is daunting
- Maintainers prefer not to use MPI
- PGAS approach allows an ‘incremental’ approach

TROVE: code before

```
g = 0

$!omp parallel do private(h,phi,D) reduce(+:g)
do iterm = 1, N
  phi = !construct Hamiltonian
  D = !construct basis set
  h = matmul(transpose(D), matmul(phi, D))

  g = g + h
enddo
$!omp end parallel do

call diagonalize(g)
```

- Each loop iteration takes >10 seconds
  $\Rightarrow$ almost embarrassingly parallel

TROVE: code after

```
g = 0
do iterm = 1, N
  if (mod(iterm, num_images()) /= this_image()) cycle

  phi = !construct Hamiltonian
  D = !construct basis set
  h = matmul(transpose(D), matmul(phi, D))

  g = g + h
enddo

call co_sum(g)
call diagonalize(g)
```

- very few changes required from OpenMP
- scheduling is basic

TROVE: Results

Coarrays implementation works!

Before

Ideal scaling on 4 socket node

After

Ideal scaling on 4 nodes with 1 socket each

Problems

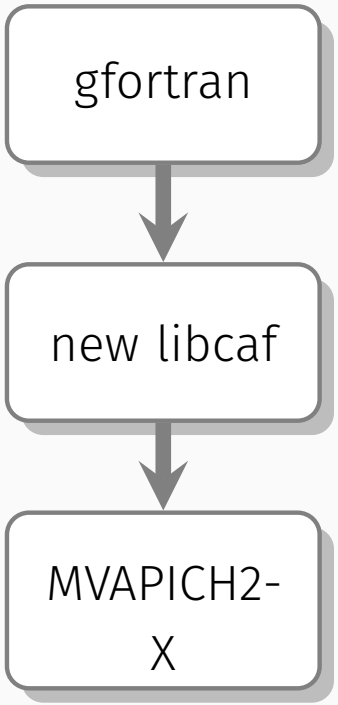
- TROVE not compatible with OpenUH
  $\Rightarrow$ can't use MVAPICH2-X :-(

Future Work

- Extracting more parallelism
- MPI interoperability

Wishlist

Building a coarray implementation with gfortran and an MVAPICH2-X based libcaf.



- gfortran frontend generates calls to libcaf
- a new implementation of libcaf
- this new libcaf calls MVAPICH2-X

  - combines the portable and friendly gfortran
  - with the high-performance of MVAPICH2-X

Thank you!