Performance Portability of a Coarray Fortran atmospheric Model

Damian Rouson

Ethan Gutman
Alessandro Fanfarillo

Brian Friesen

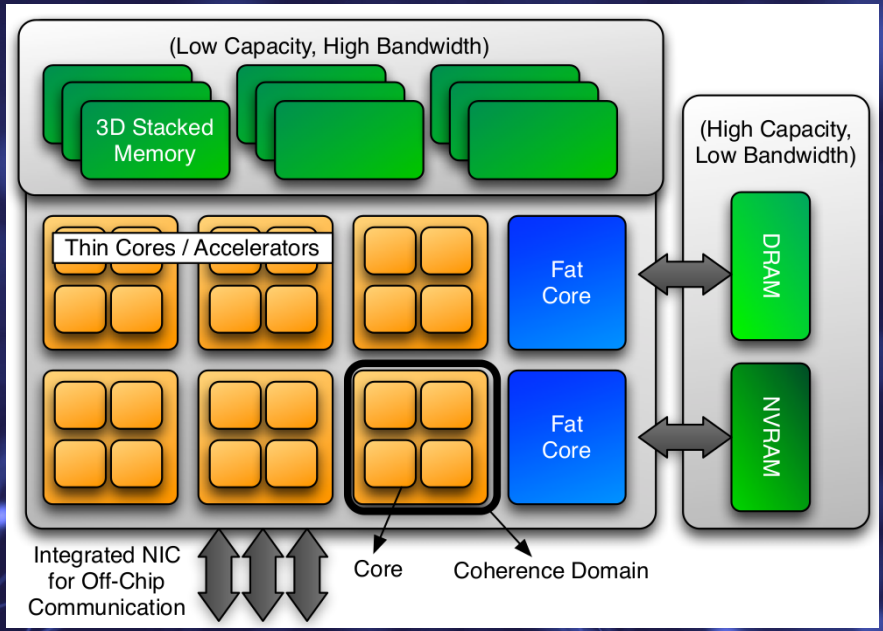
Overview

- Fortran 2018 in a Nutshell
- ICAR & Coarray ICAR
- Results
- Conclusions

Overview

- **Fortran 2018 in a Nutshell**
  - Motivation: exascale challenges
  - Background: SPMD & PGAS in Fortran 2018
  - Beyond CAF
- ICAR & Coarray ICAR
- Results
- Conclusions

EXASCALE CHALLENGES & Fortran 2018 Response



Source: Ang, J.A., Barrett, R.F., Benner, R.E., Burke, D., Chan, C., Cook, J., Donofrio, D., Hammond, S.D., Hemmert, K.S., Kelly, S.M. and Le, H., 2014, November. Abstract machine models and proxy architectures for exascale computing. In *Hardware-Software Co-Design for High Performance Computing (Co-HPC)*, 2014 (pp. 25-32). IEEE.

Billion-way concurrency with high levels of on-chip parallelism

- events
- collective subroutines
- richer set of atomic subroutines
- teams

Higher failure rates

- failed-image detection

Expensive data movement

- one-sided communication
- teams (locality control)

Heterogeneous hardware on processor

- events

Single Program Multiple Data (SPMD)

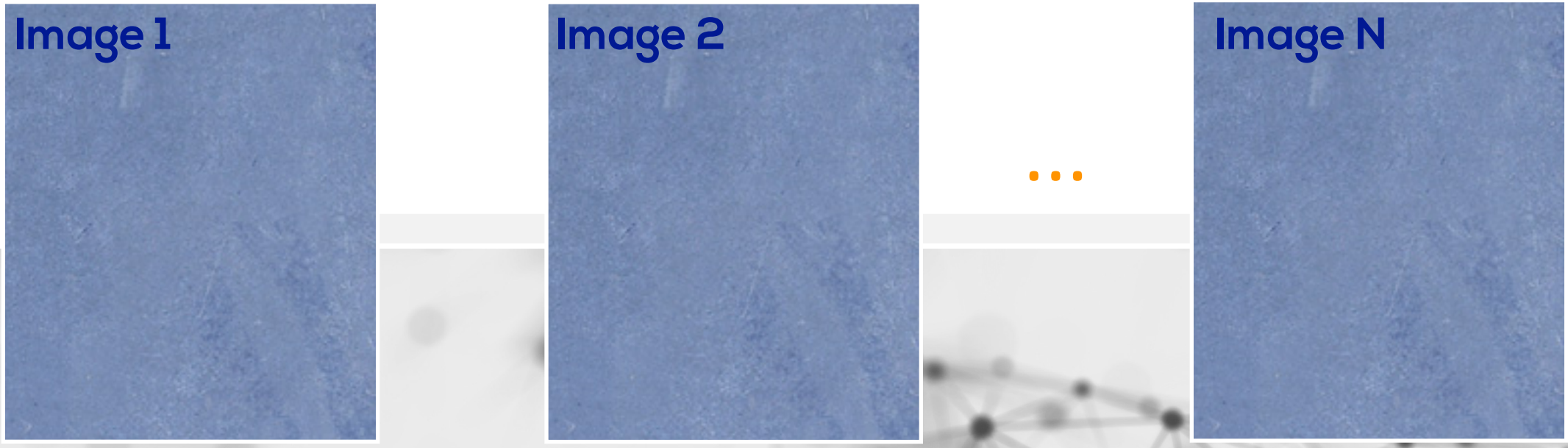
Images

Single Program Multiple Data (SPMD)

Images ⟶

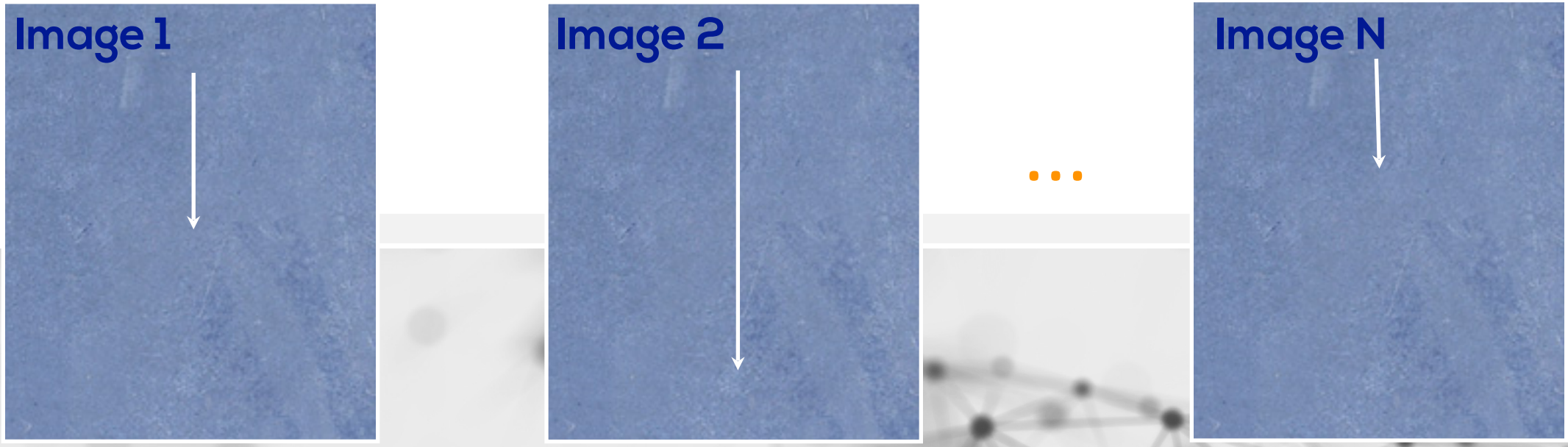
Single Program Multiple Data (SPMD)

Images ⟶ {processes | threads}



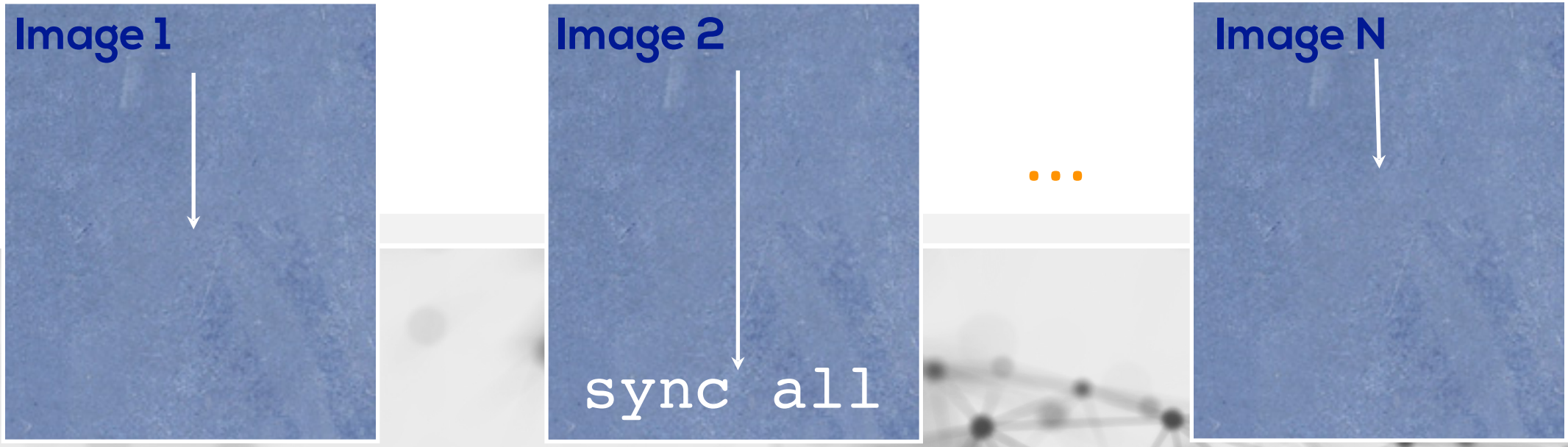
Single Program Multiple Data (SPMD)

Images ⟶ {processes | threads}



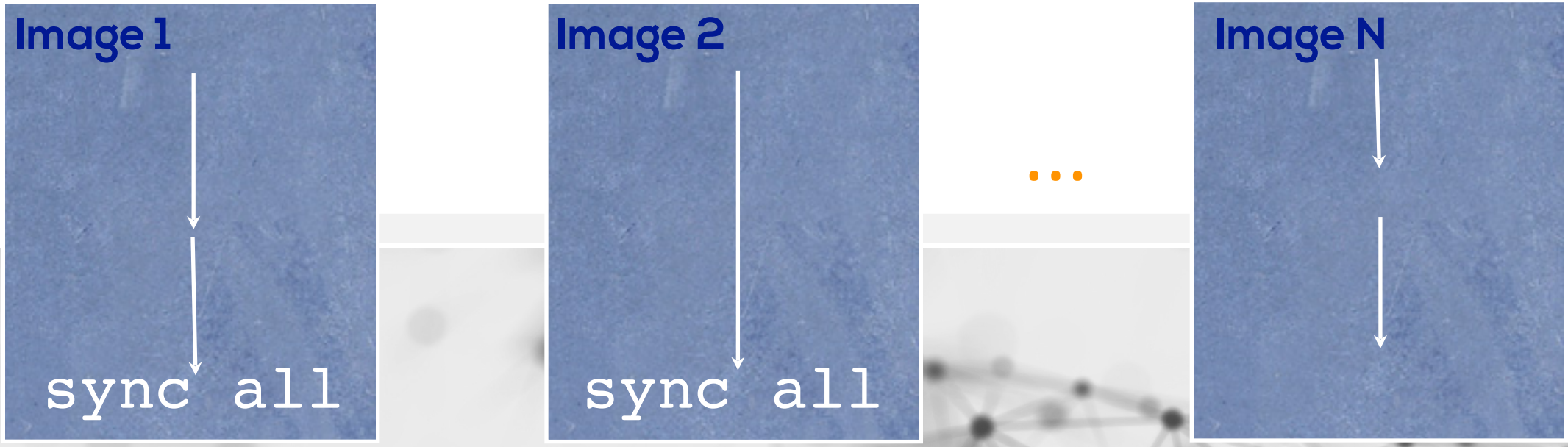
Single Program Multiple Data (SPMD)

Images ⟶ {processes | threads}



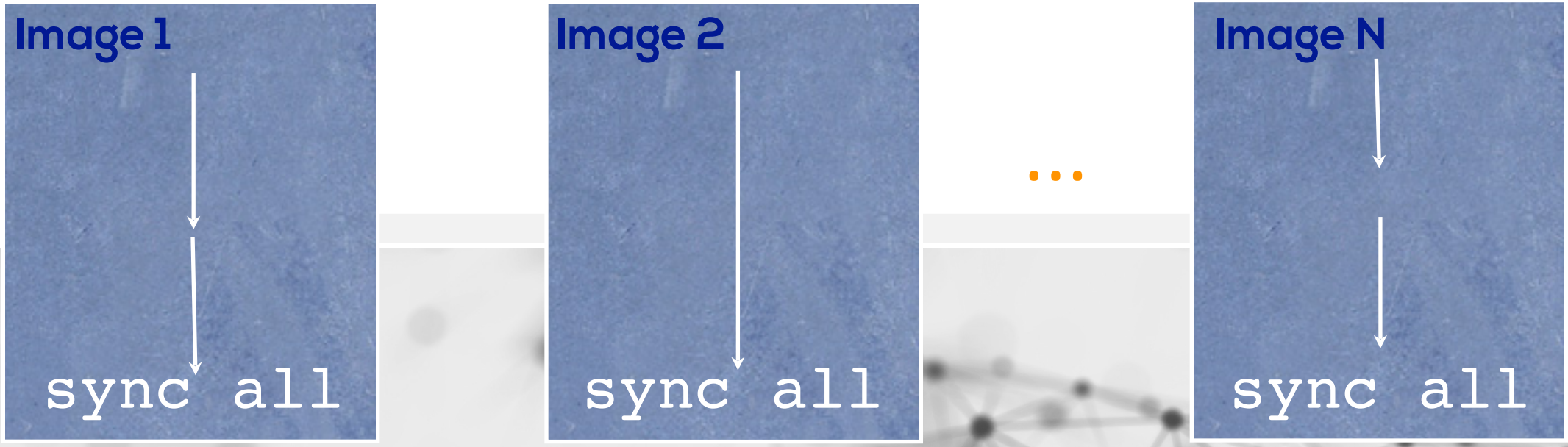
Single Program Multiple Data (SPMD)

Images ⟶ {processes | threads}



Single Program Multiple Data (SPMD)

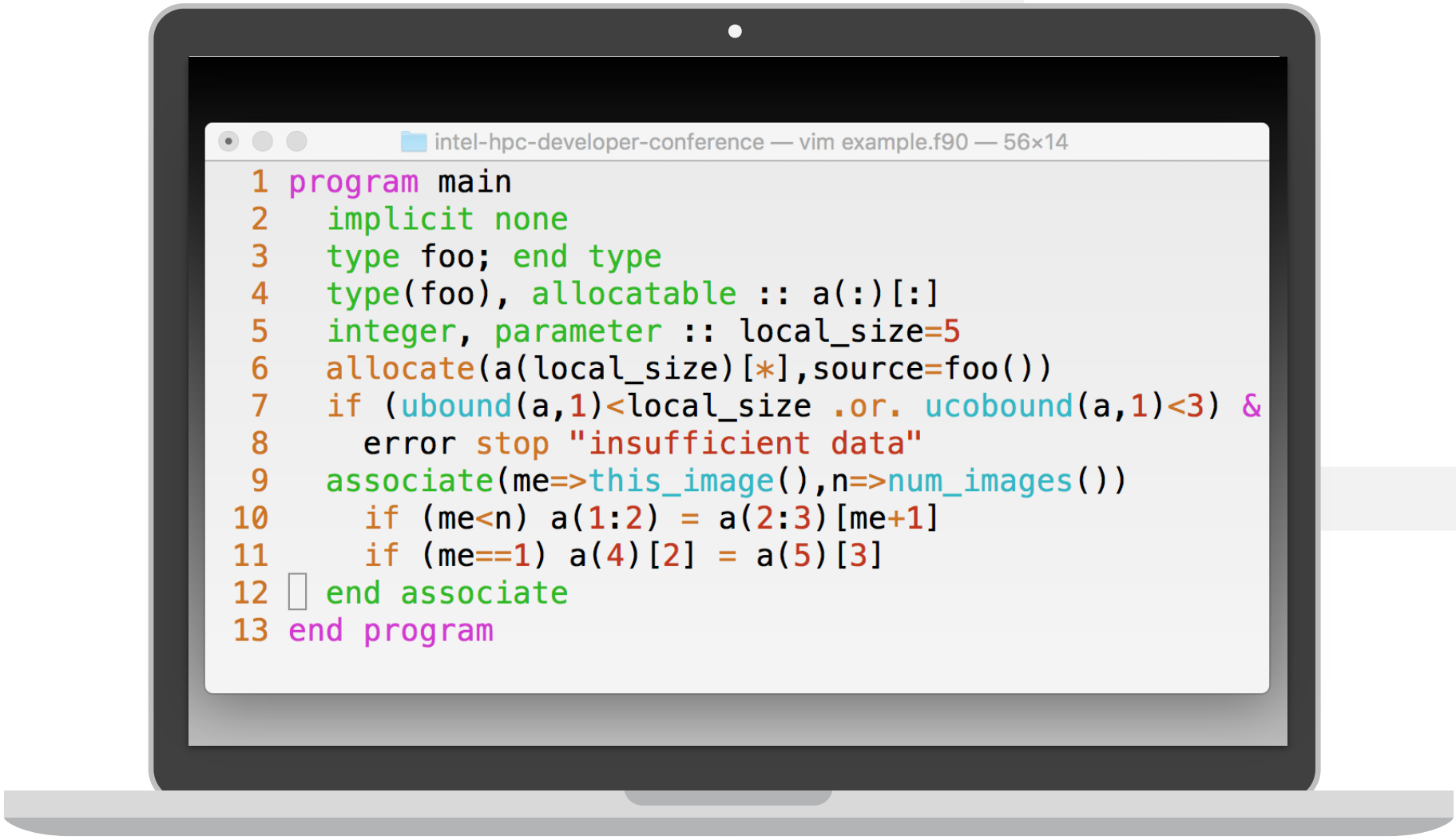
Images ⟶ {processes | threads}



Images execute asynchronously up to a programmer-specified synchronization:

```
sync all
sync images
allocate/deallocate
```

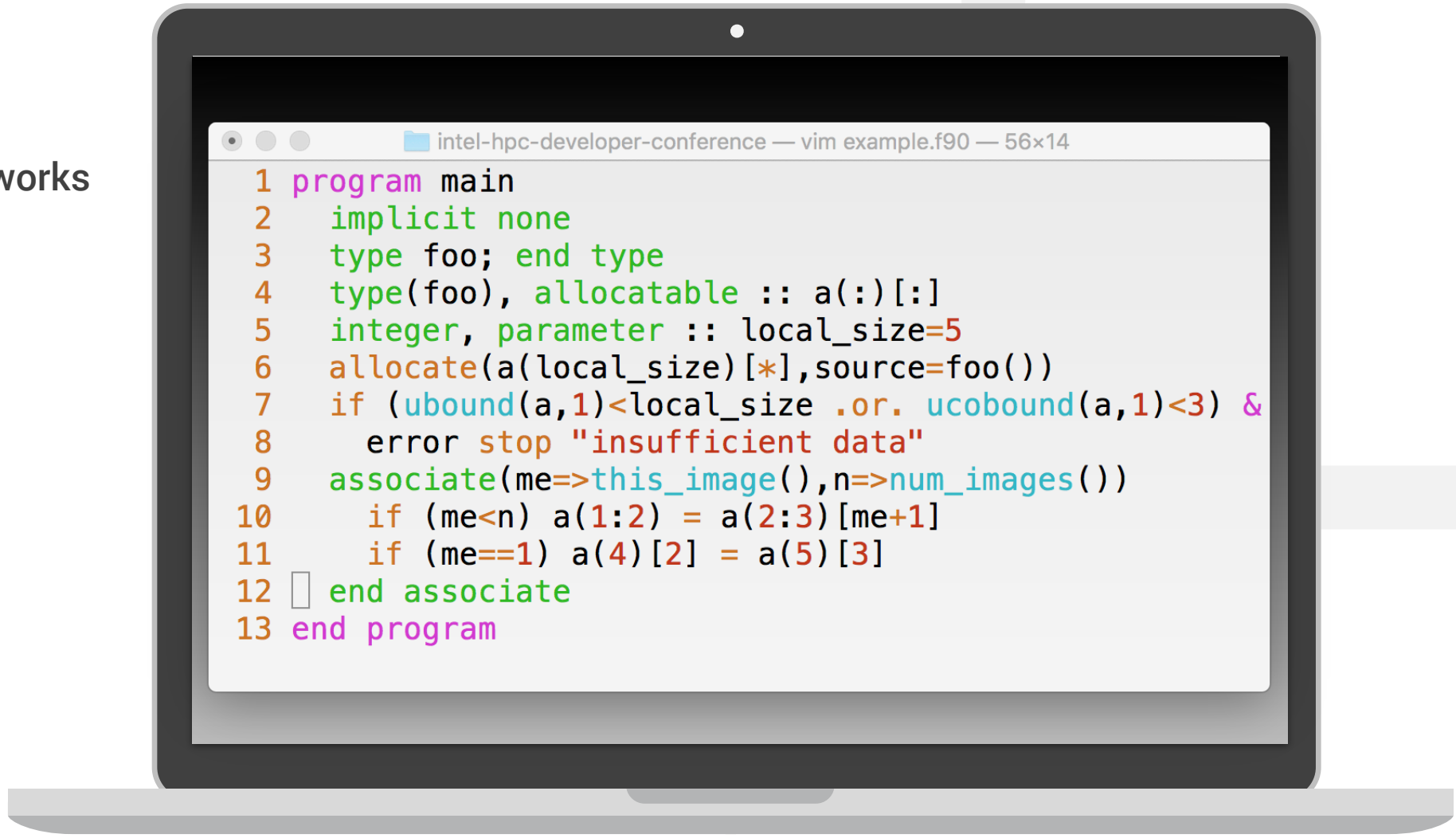
Partitioned Global Address Space (PGAS)



Partitioned Global Address Space (PGAS)

Coarrays integrate with other languages features:

- Fortran 90 array syntax works on local data.
- Communicate objects



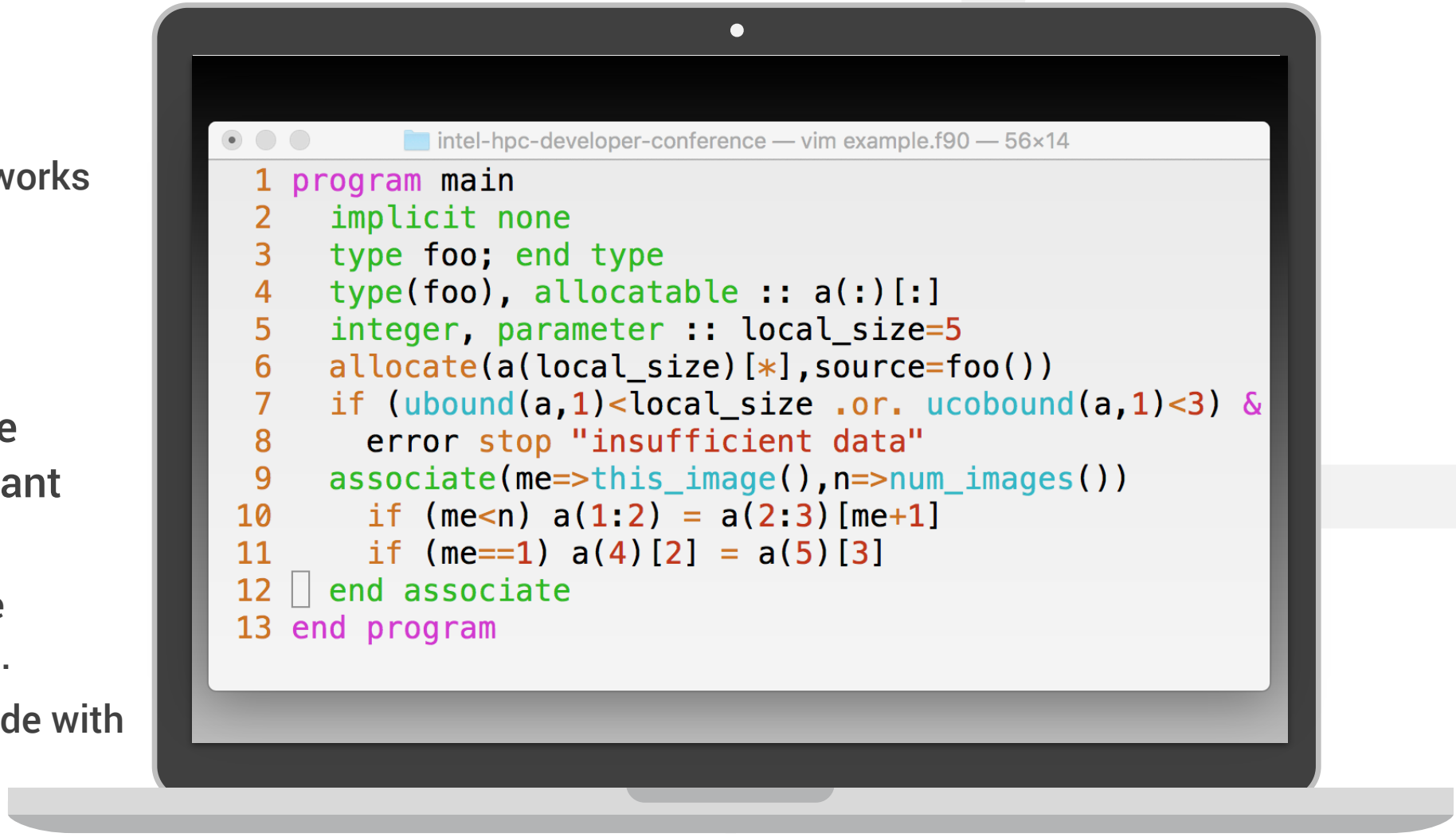
Partitioned Global Address Space (PGAS)

Coarrays integrate with other languages features:

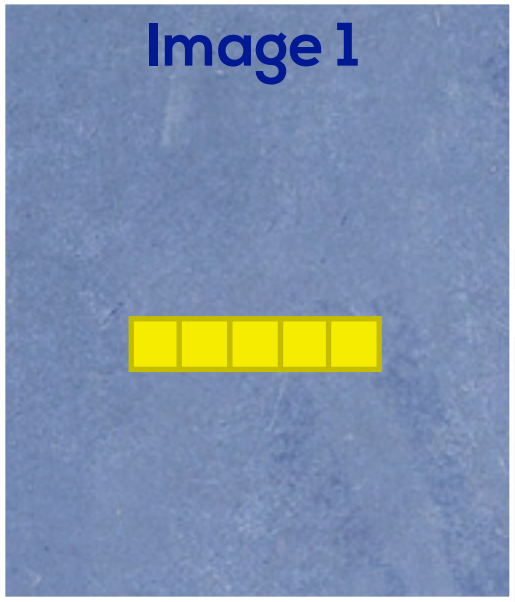
- Fortran 90 array syntax works on local data.
- Communicate objects

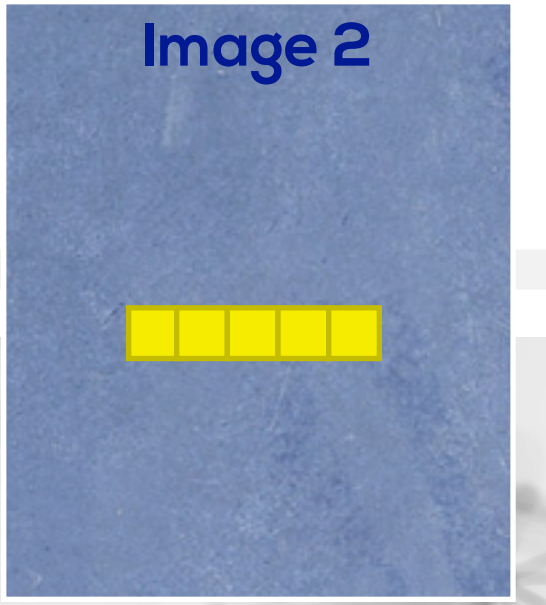
The ability to drop the square brackets harbors two important implications:

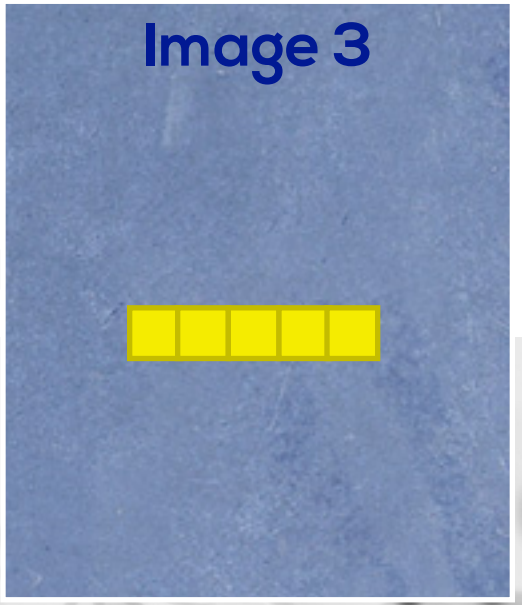
- Easily determine where communication occurs.
- Parallelize legacy code with minimal revisions.



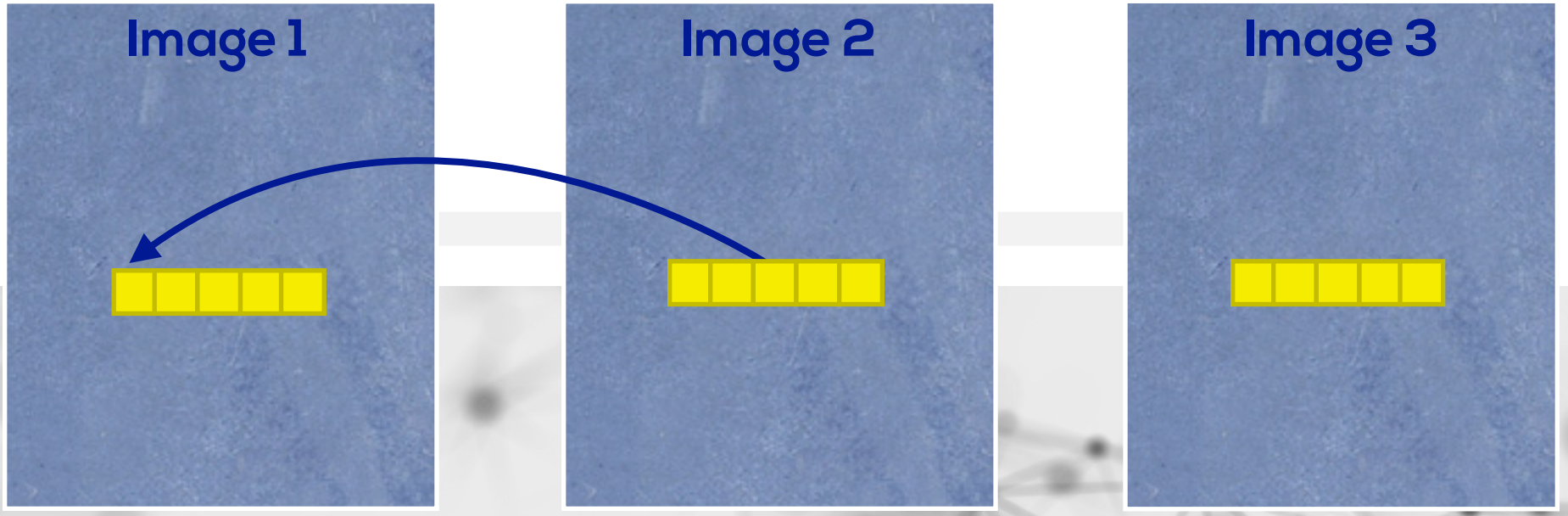
```
if (me<n) a(1:2) = a(2:3)[me+1]
```



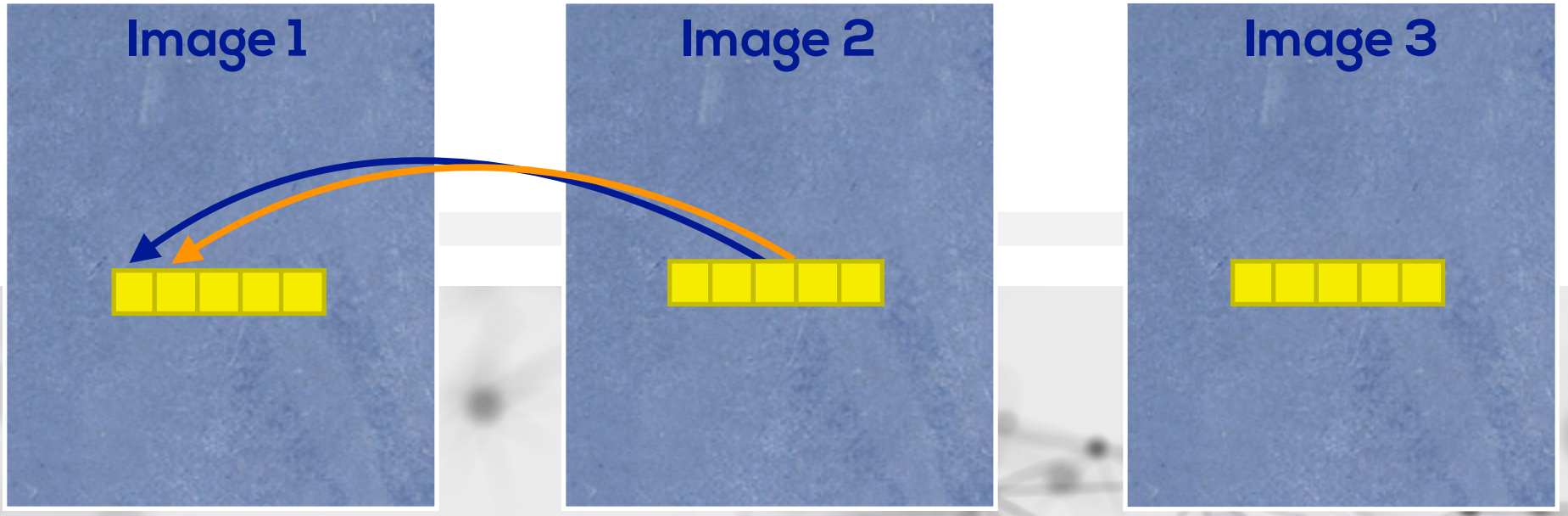




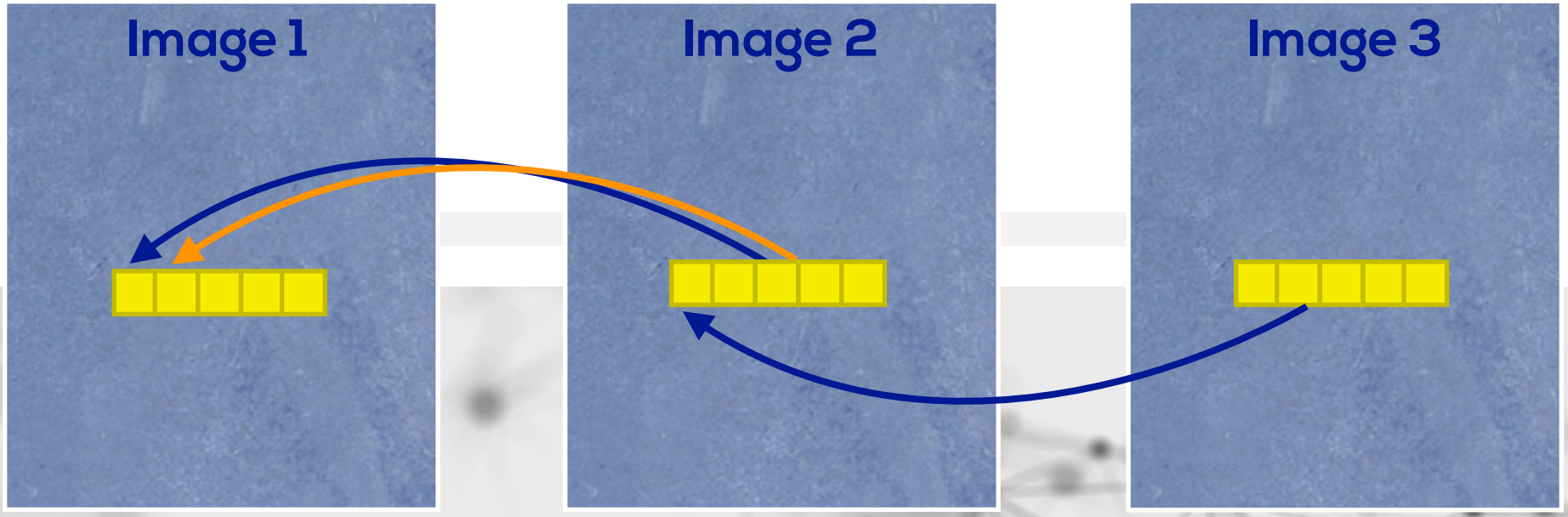
```
if (me<n) a(1:2) = a(2:3)[me+1]
```



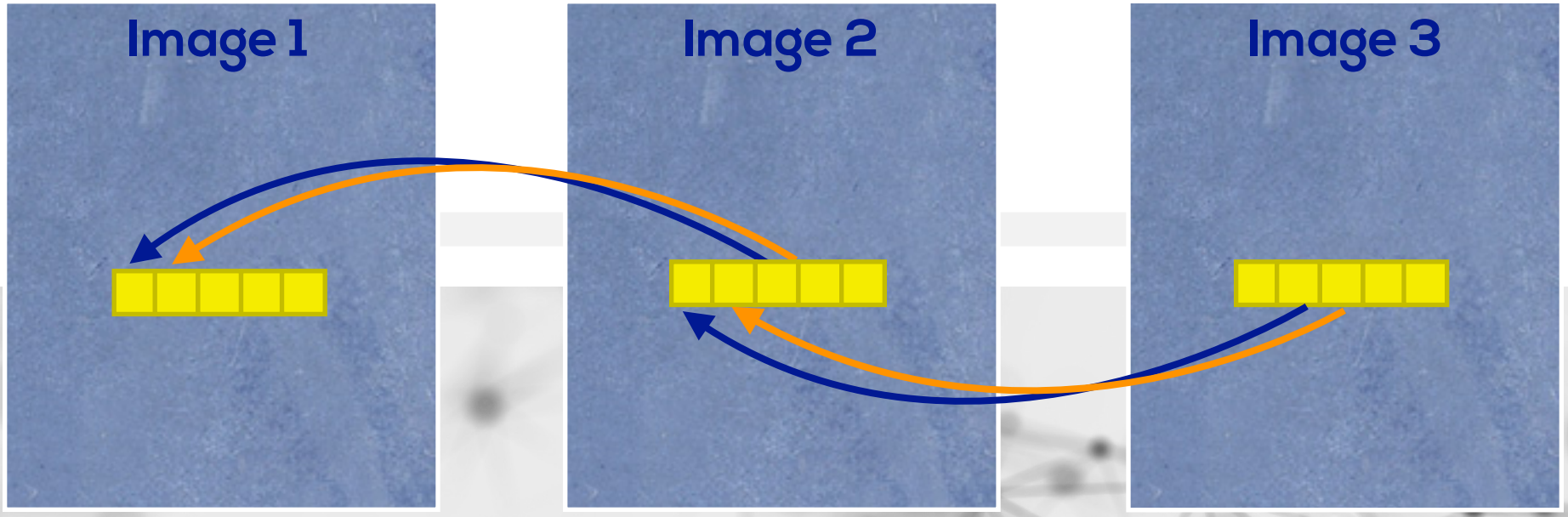
```
if (me<n) a(1:2) = a(2:3)[me+1]
```



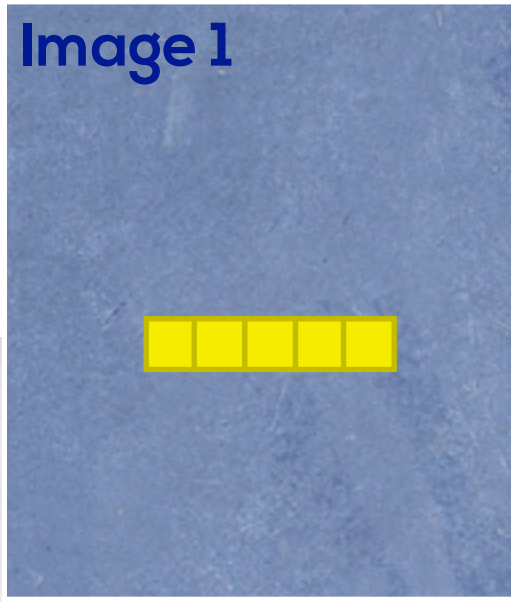
```
if (me<n) a(1:2) = a(2:3)[me+1]
```

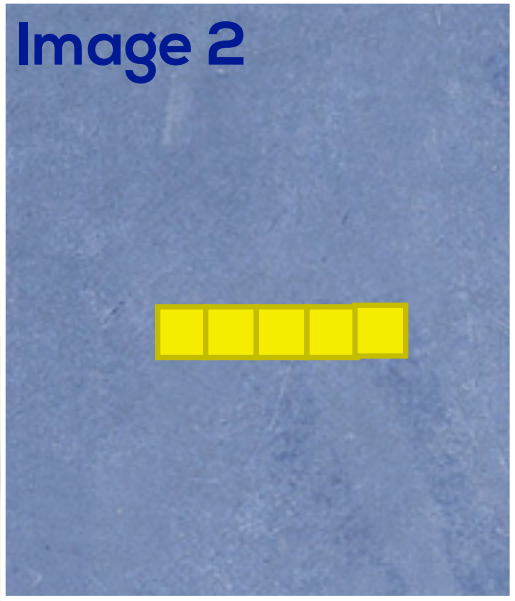


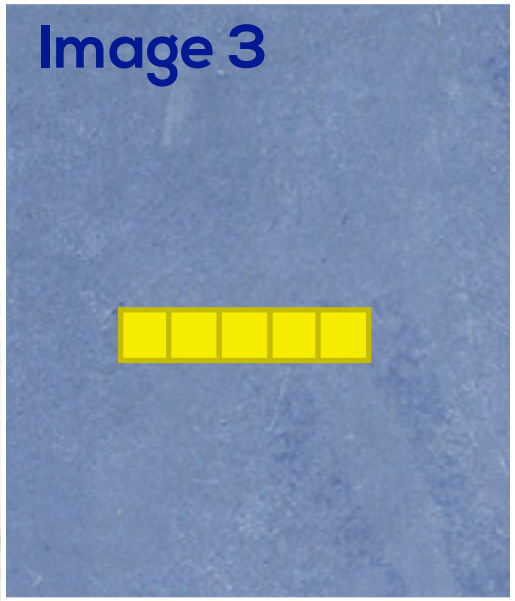
```
if (me<n) a(1:2) = a(2:3)[me+1]
```



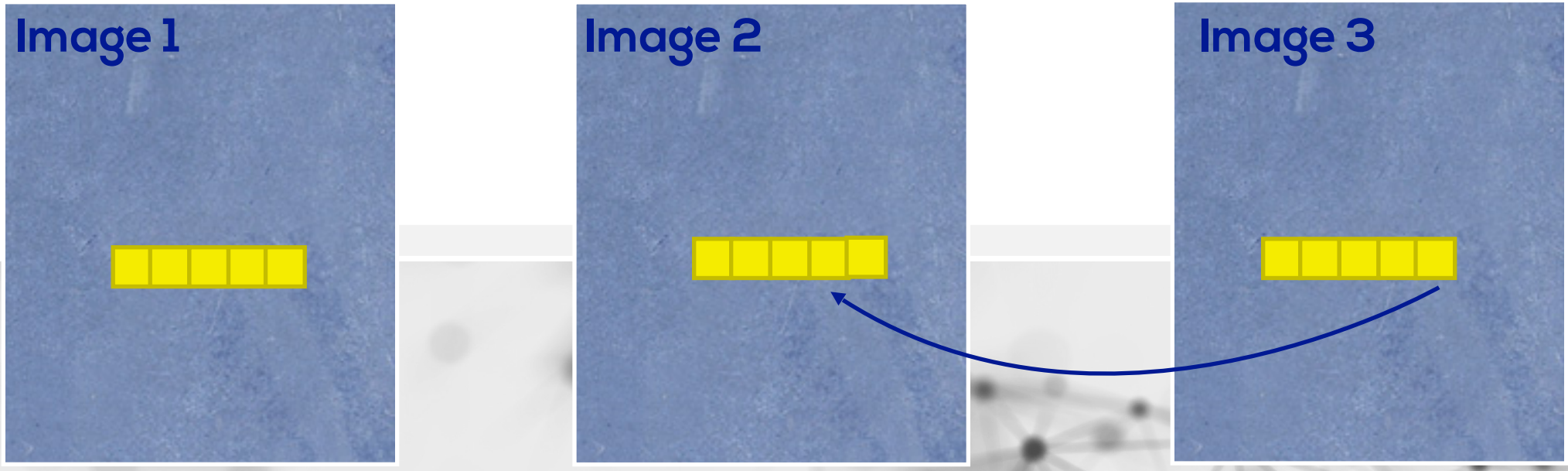
```
if (me==1) a(4)[2] = a(5)[3]
```







```
if (me==1) a(4)[2] = a(5)[3]
```



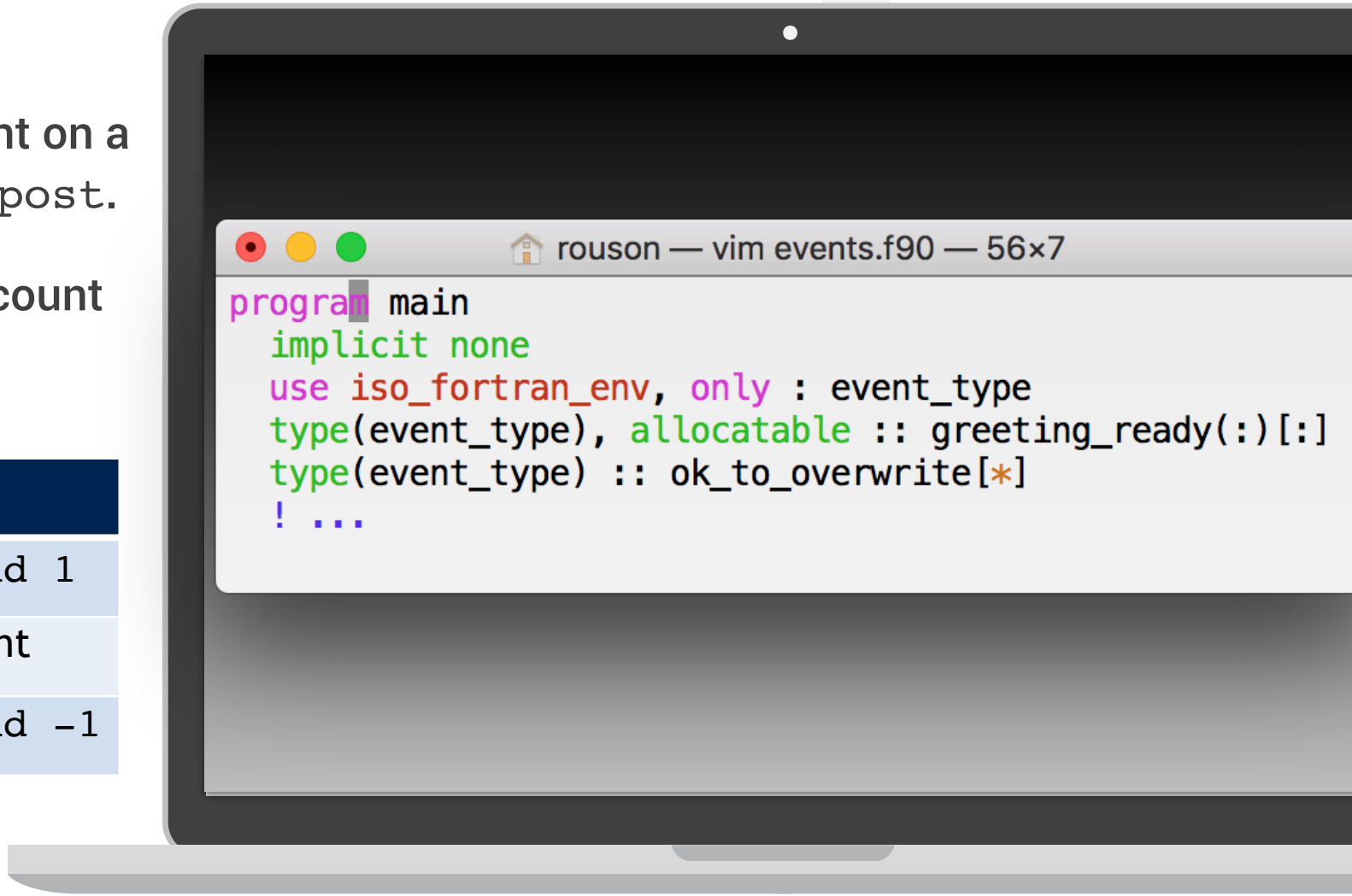
Segment Ordering: Events

An intrinsic module provides the derived type `event_type`, which encapsulates an `atomic_int_kind` integer component default-initialized to zero.

An image increments the event count on a remote image by executing `event post`.

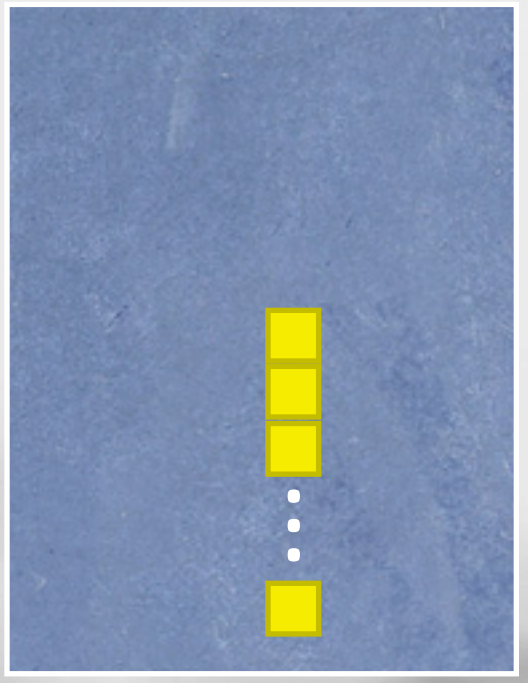
The remote image obtains the post count by executing `event_query`.

	Image Control	Side Effect
`event post`	☒	`atomic_add 1`
`event_query`		defines count
`event wait`	☒	`atomic_add -1`

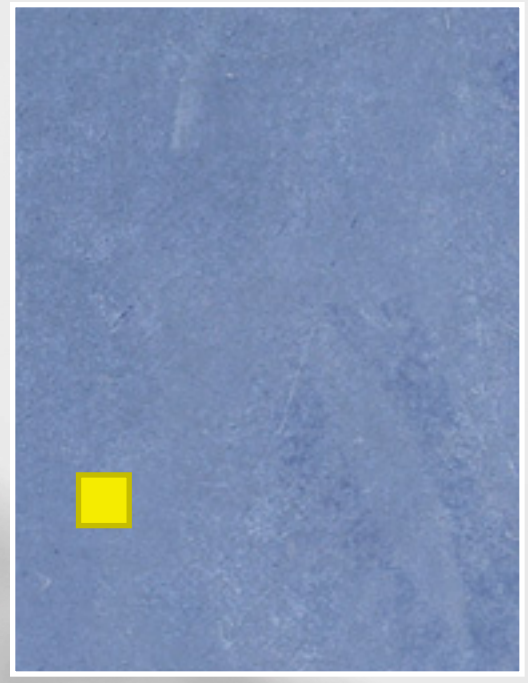


Events

Hello, world!



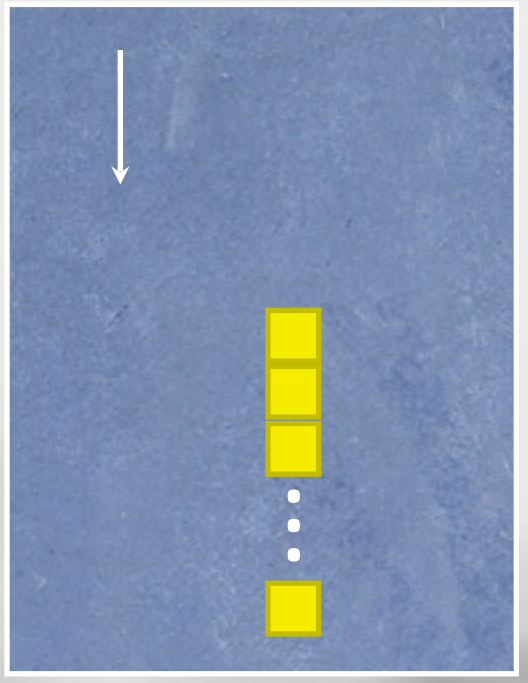
greeting_ready(2:n)[1]



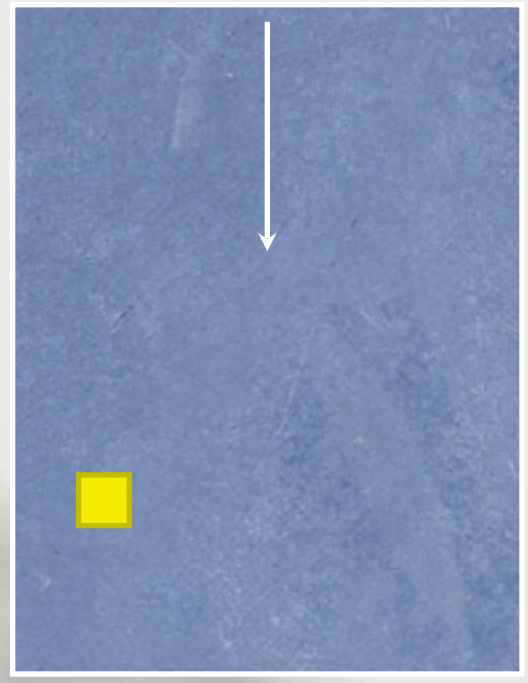
ok_to_overwrite[3]

Events

Hello, world!



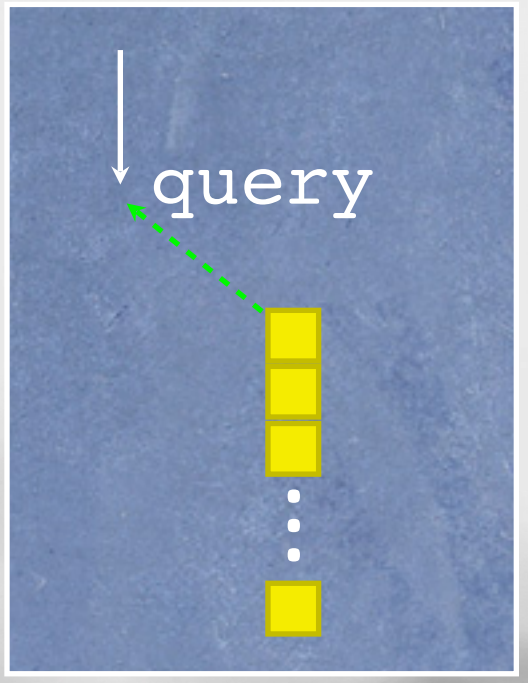
greeting_ready(2:n)[1]



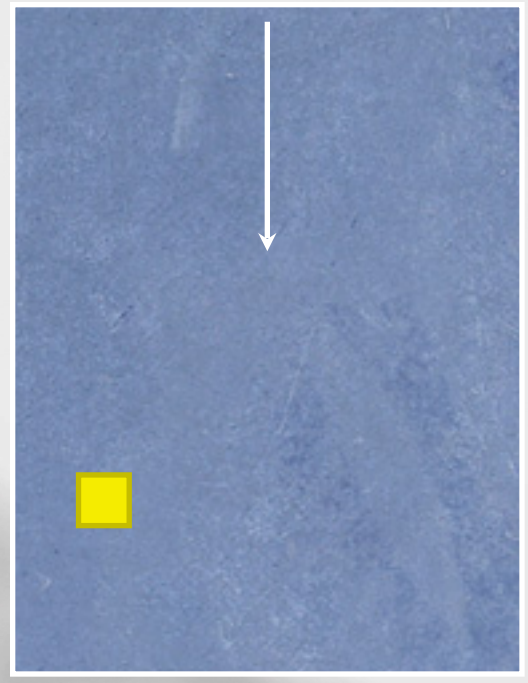
ok_to_overwrite[3]

Events

Hello, world!



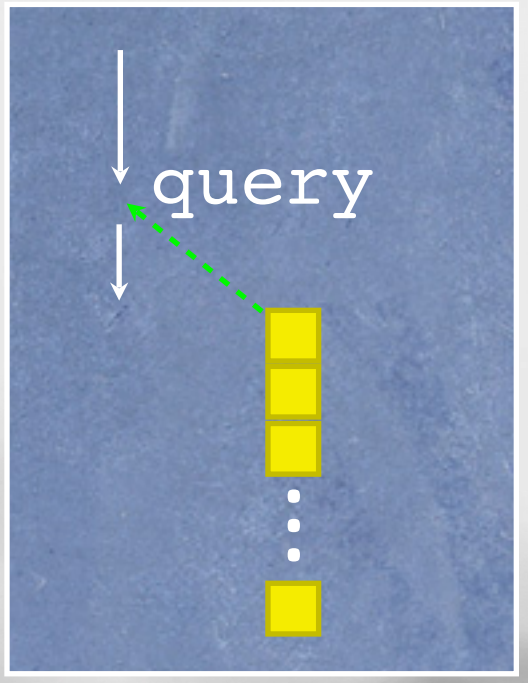
greeting_ready(2:n)[1]



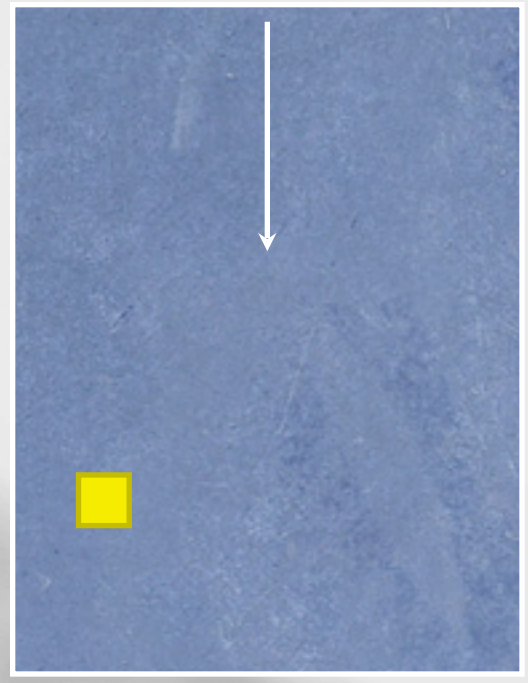
ok_to_overwrite[3]

Events

Hello, world!



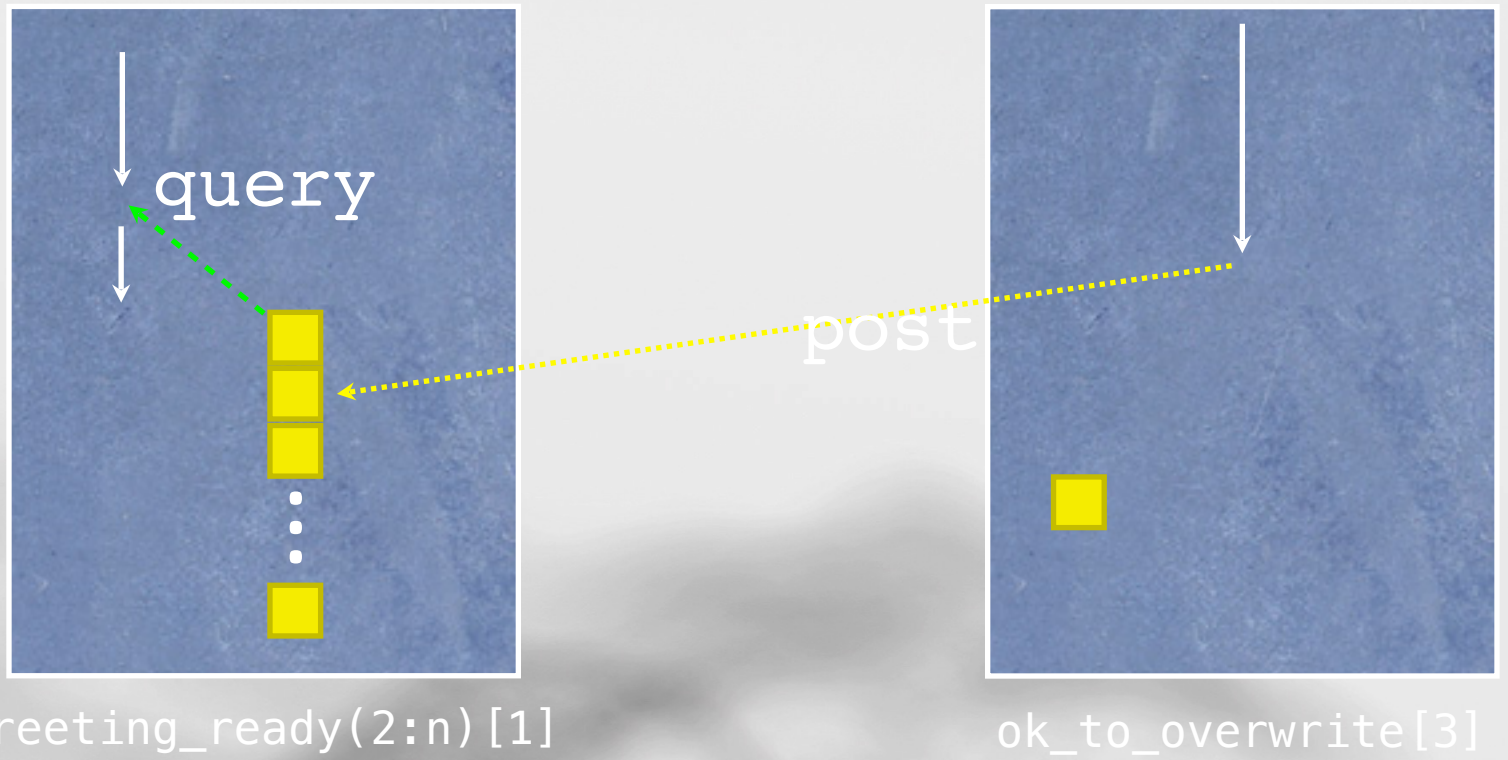
greeting_ready(2:n)[1]



ok_to_overwrite[3]

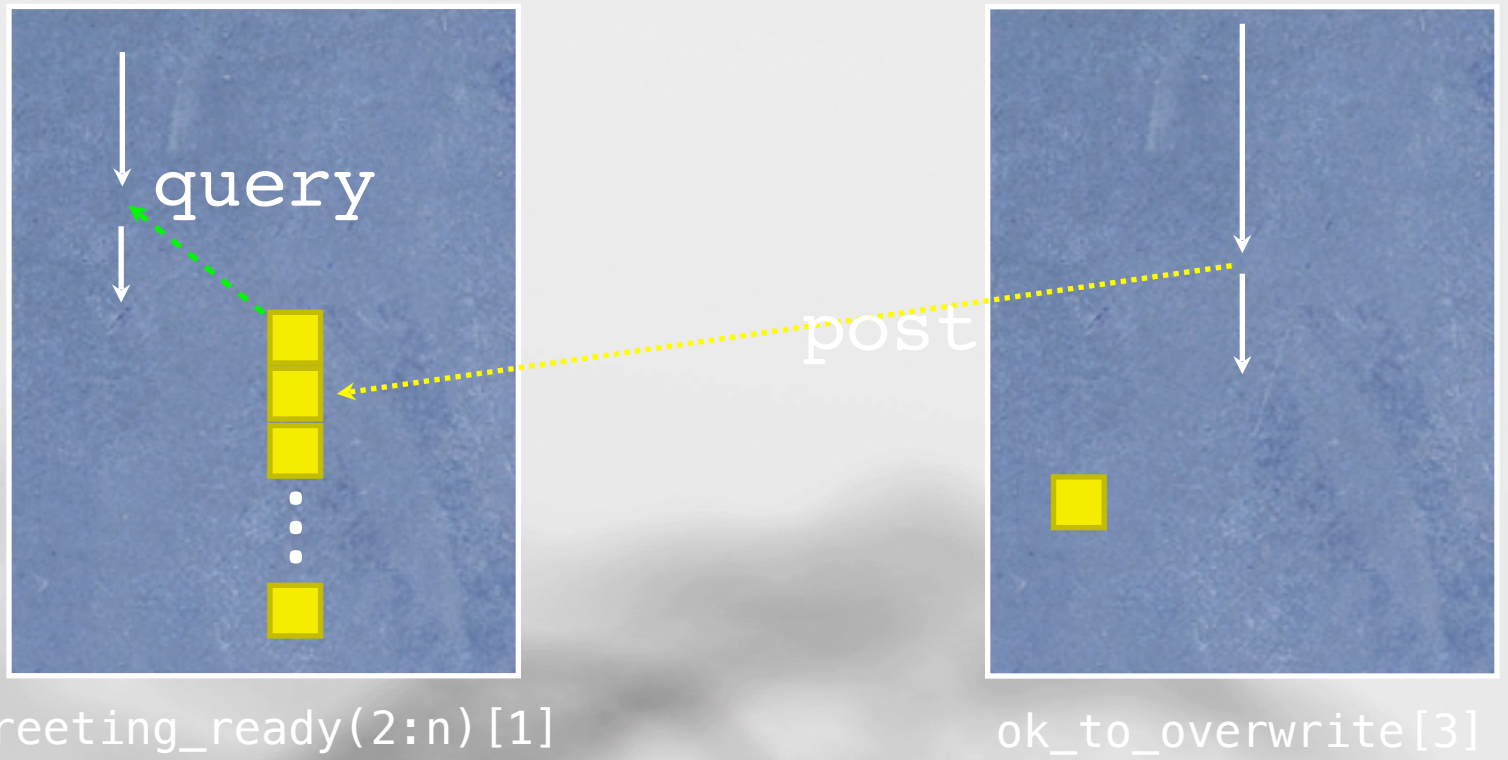
Events

Hello, world!



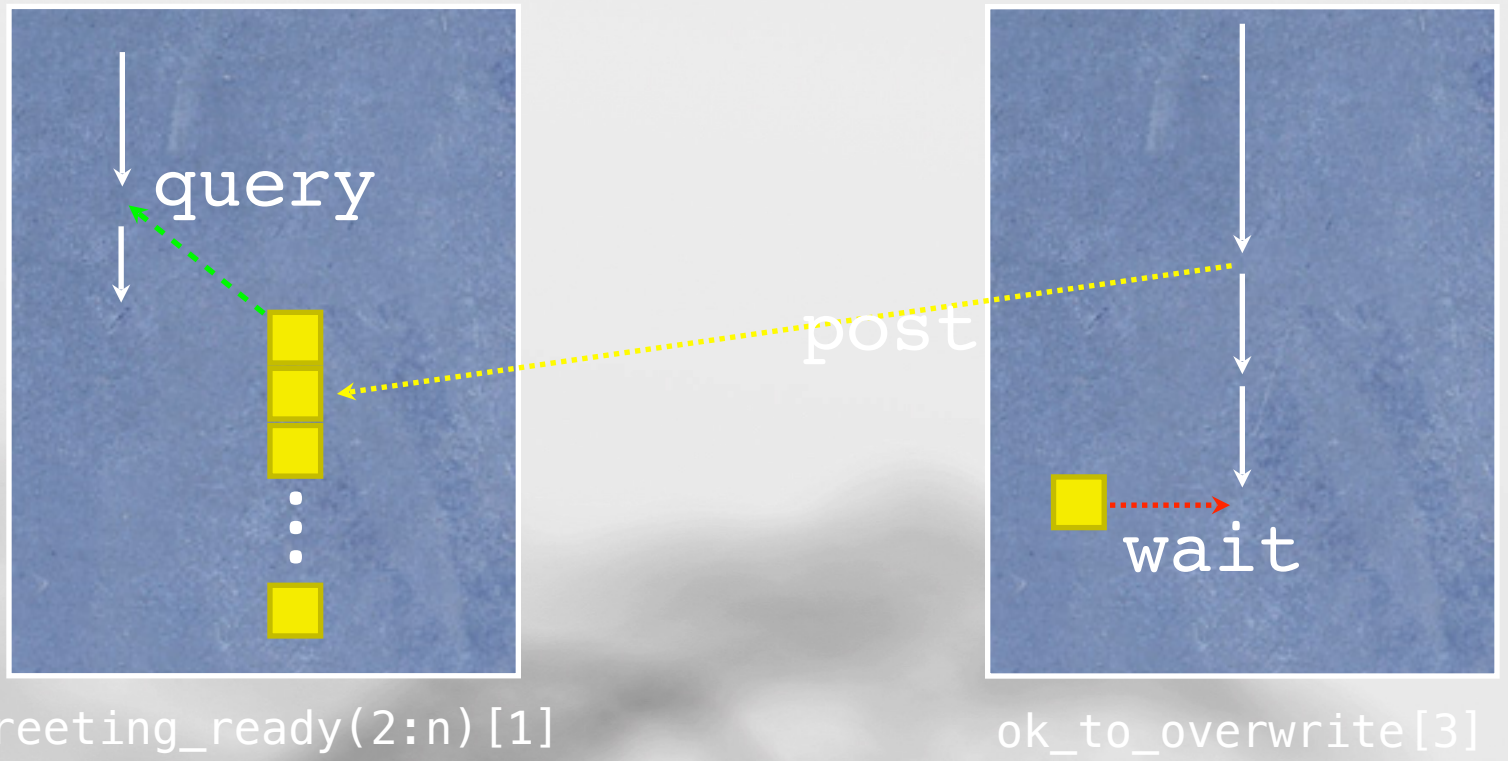
Events

Hello, world!



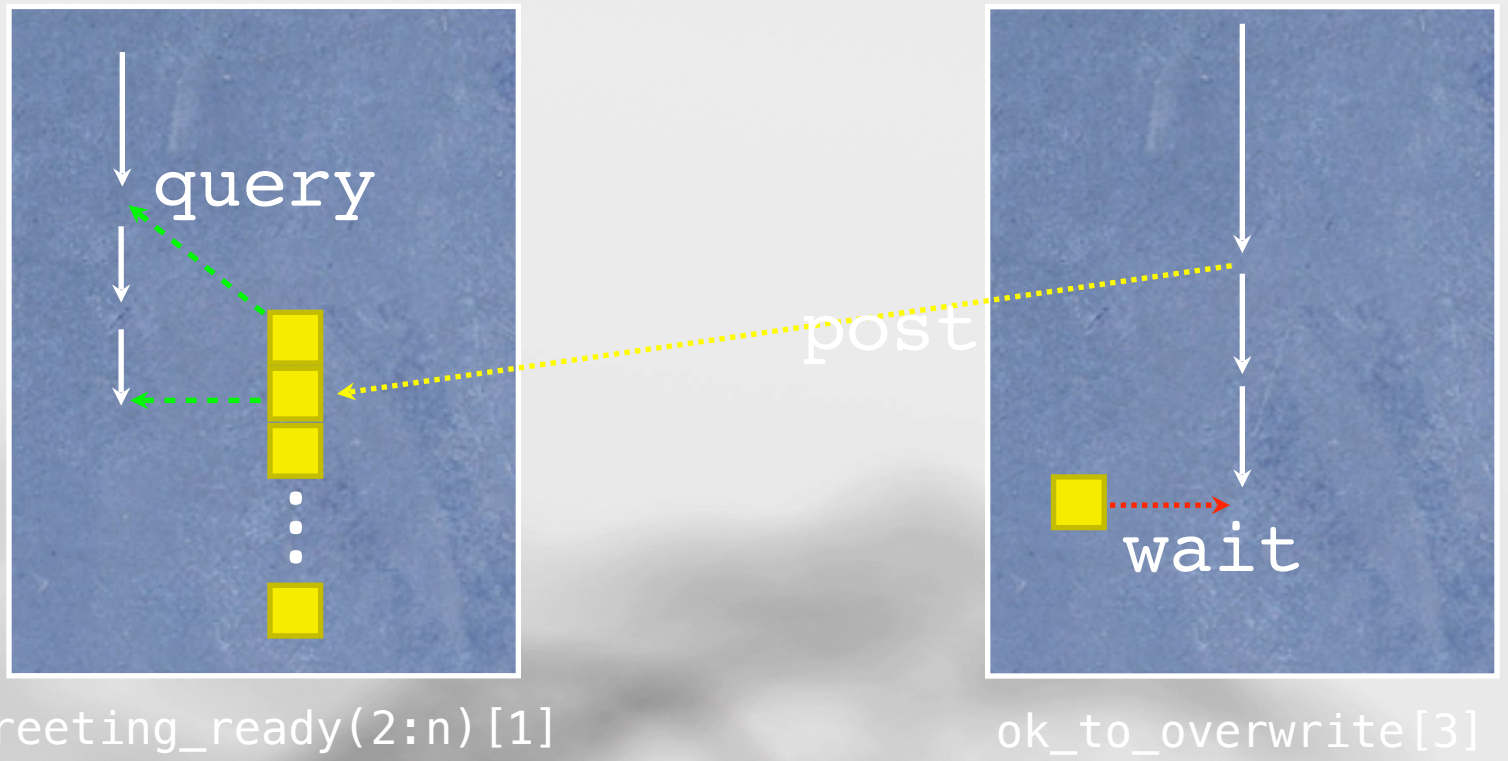
Events

Hello, world!



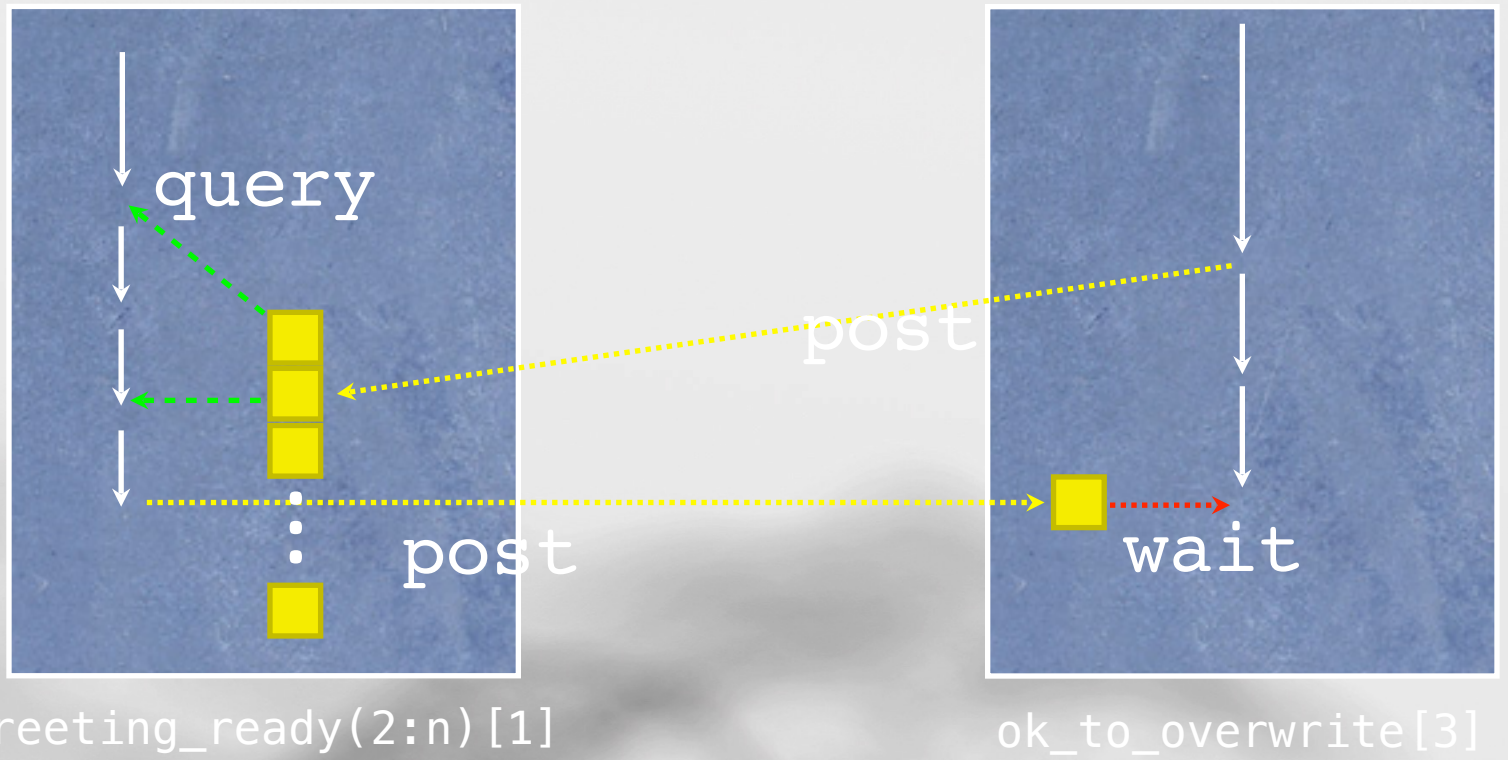
Events

Hello, world!



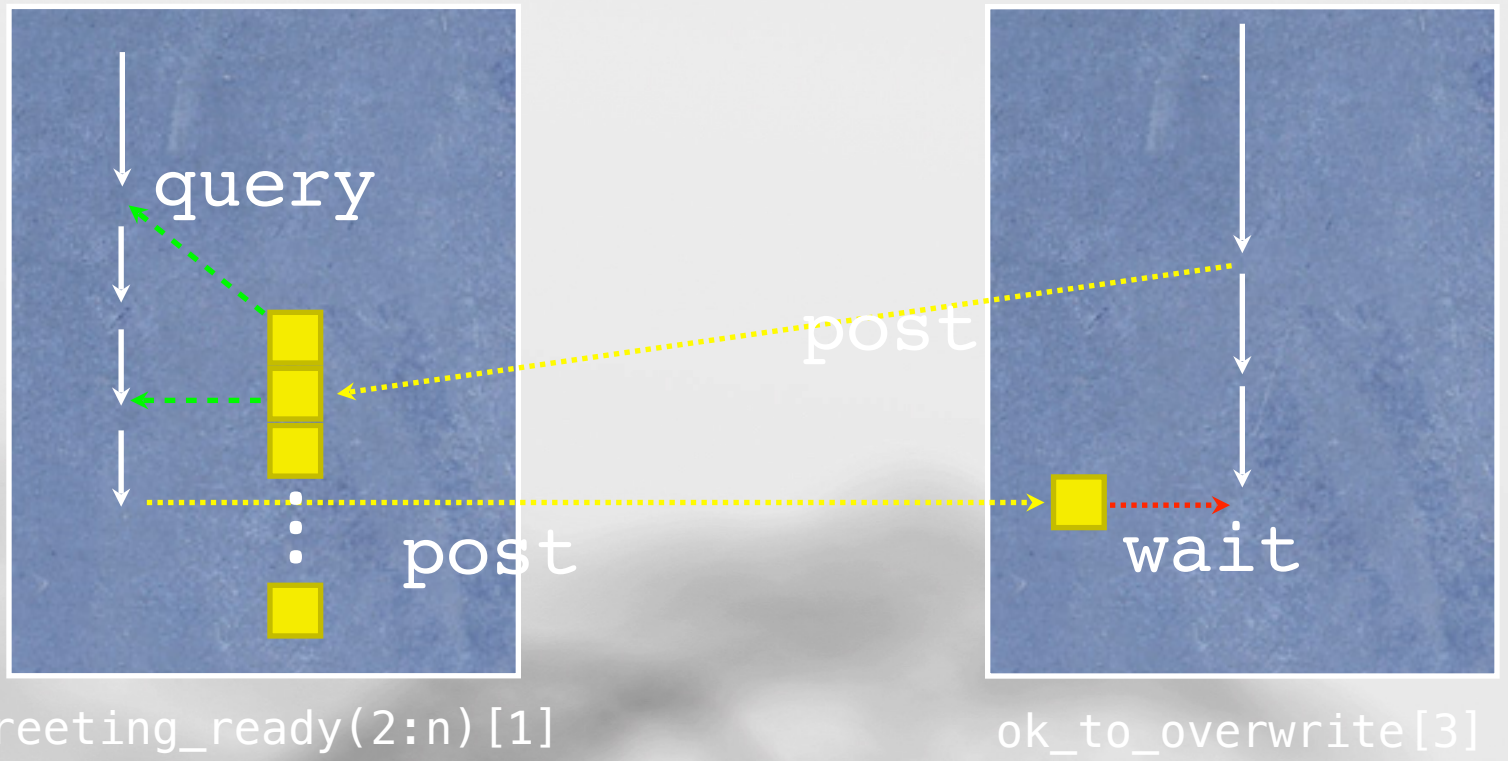
Events

Hello, world!



Events

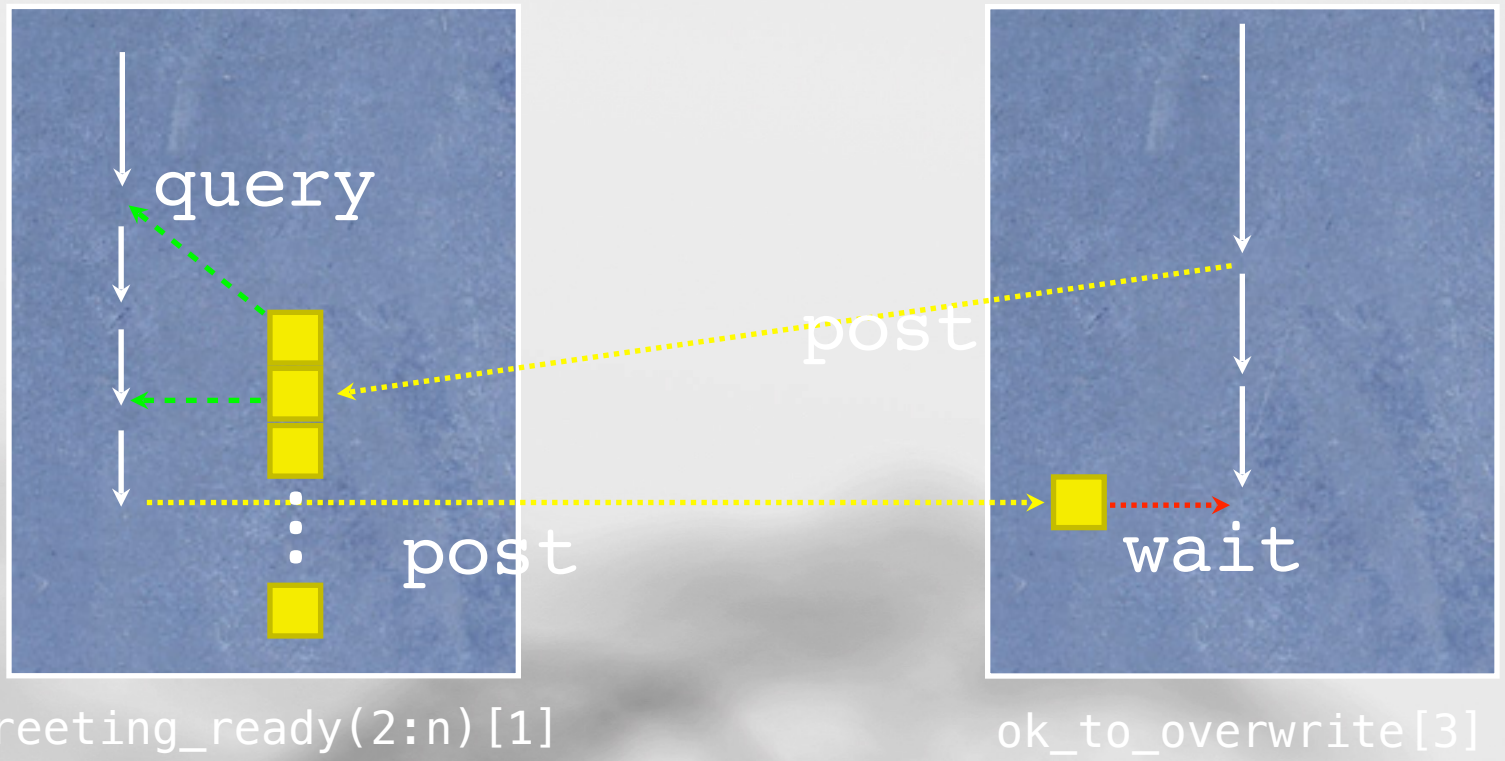
Hello, world!



Performance-oriented constraints:
- Query and wait must be local.
- Post and wait are disallowed in do concurrent constructs.

Events

Hello, world!

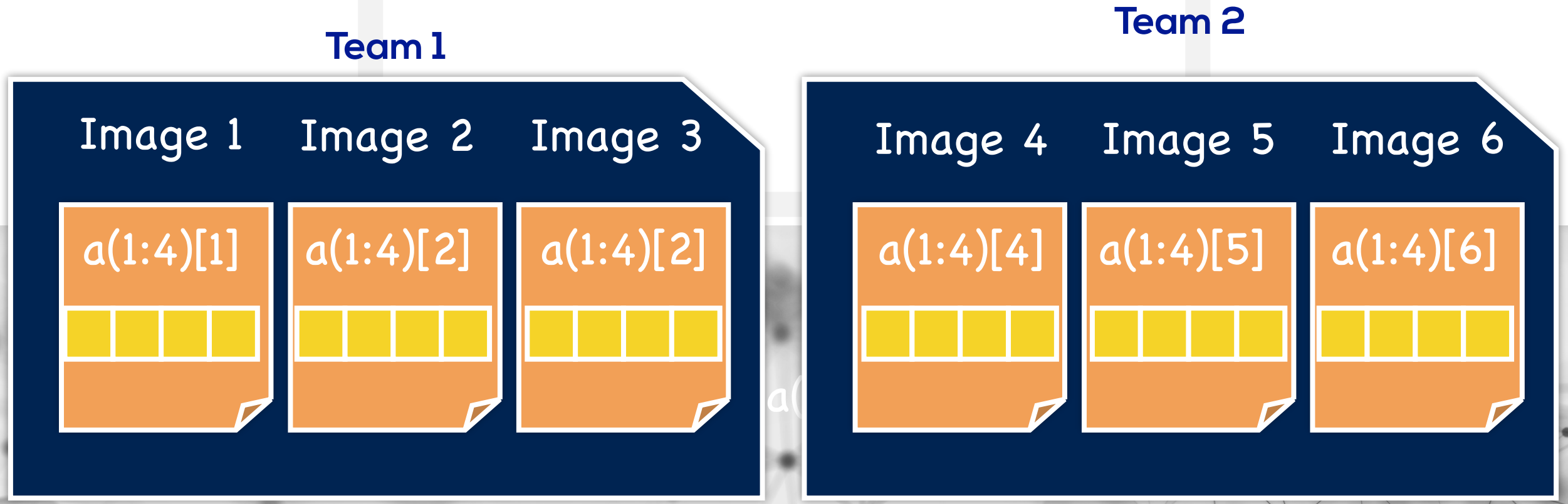


Performance-oriented constraints:
- Query and wait must be local.
- Post and wait are disallowed in do concurrent constructs.

Pro tips:
- Overlap communication and computation.
- Wherever safety permits, query without waiting.
- Write a spin-query-work loop & build a logical mask describing the remaining work.

TEAM

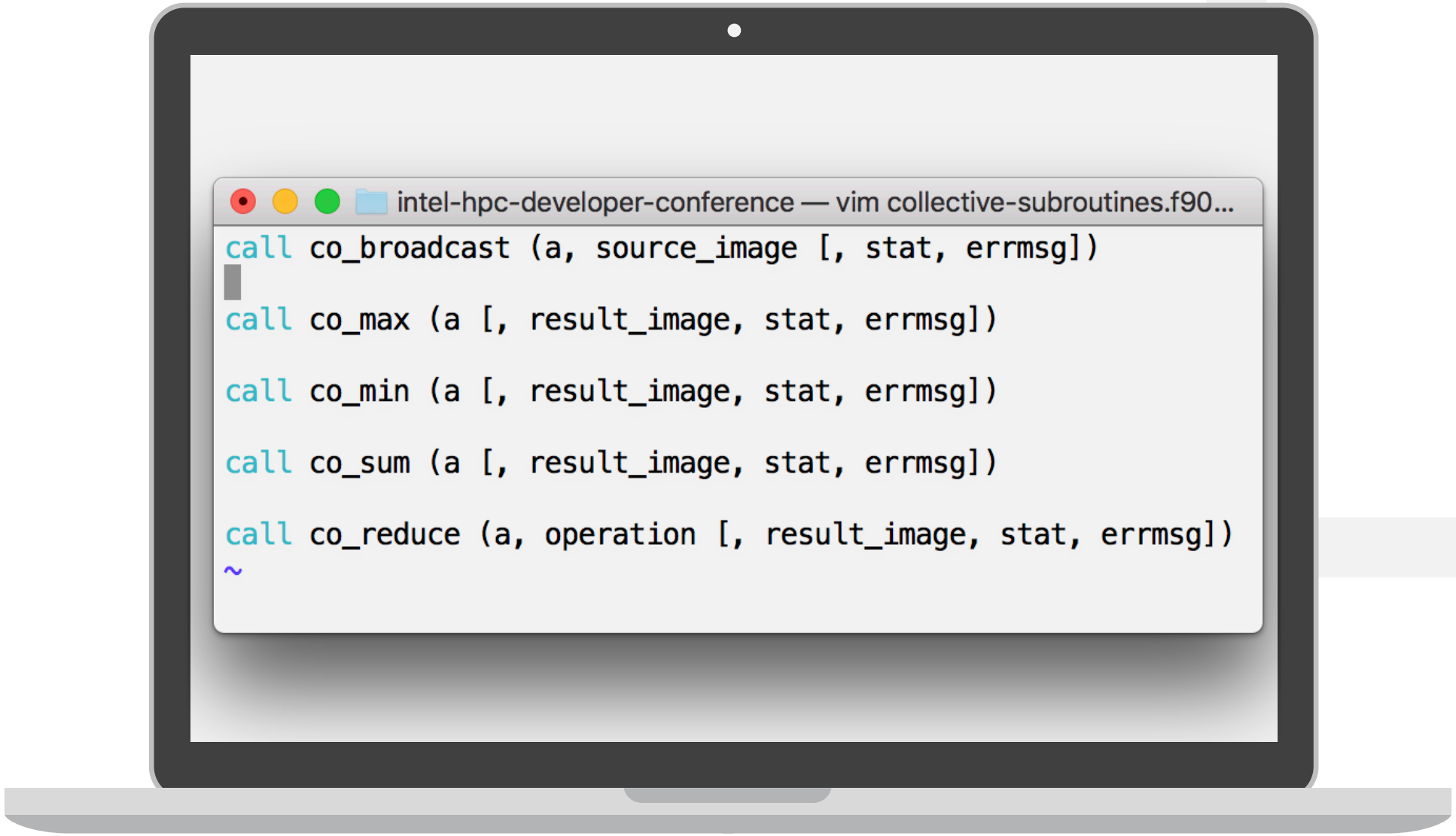
A set of images that can readily execute independently of other images



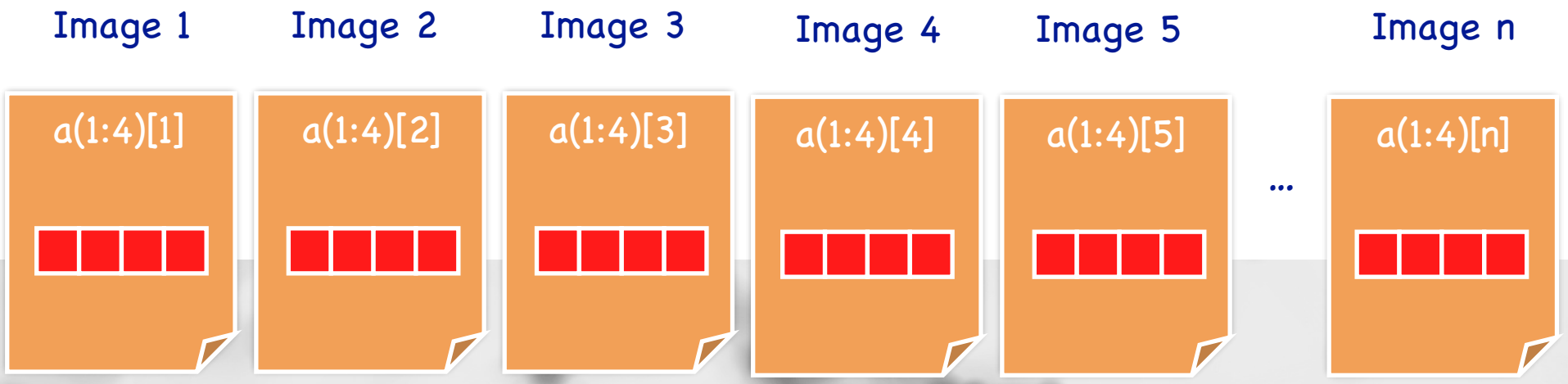
Collective Subroutines

- Each non-failed image of the current team must invoke the collective.
- After parallel calculation/communication, the result is placed on one or all images.
- Optional arguments: `stat, errmsg, result_image, source_image`
- All collectives have `intent(inout)` argument `A` holding the input data and may hold the result on return, depending on `result_image`
- No implicit synchronization at beginning/end, which allows for overlap with other actions.
- No image's data is accessed before that image invokes the collective subroutine.

Extensible Set of Collective Subroutines

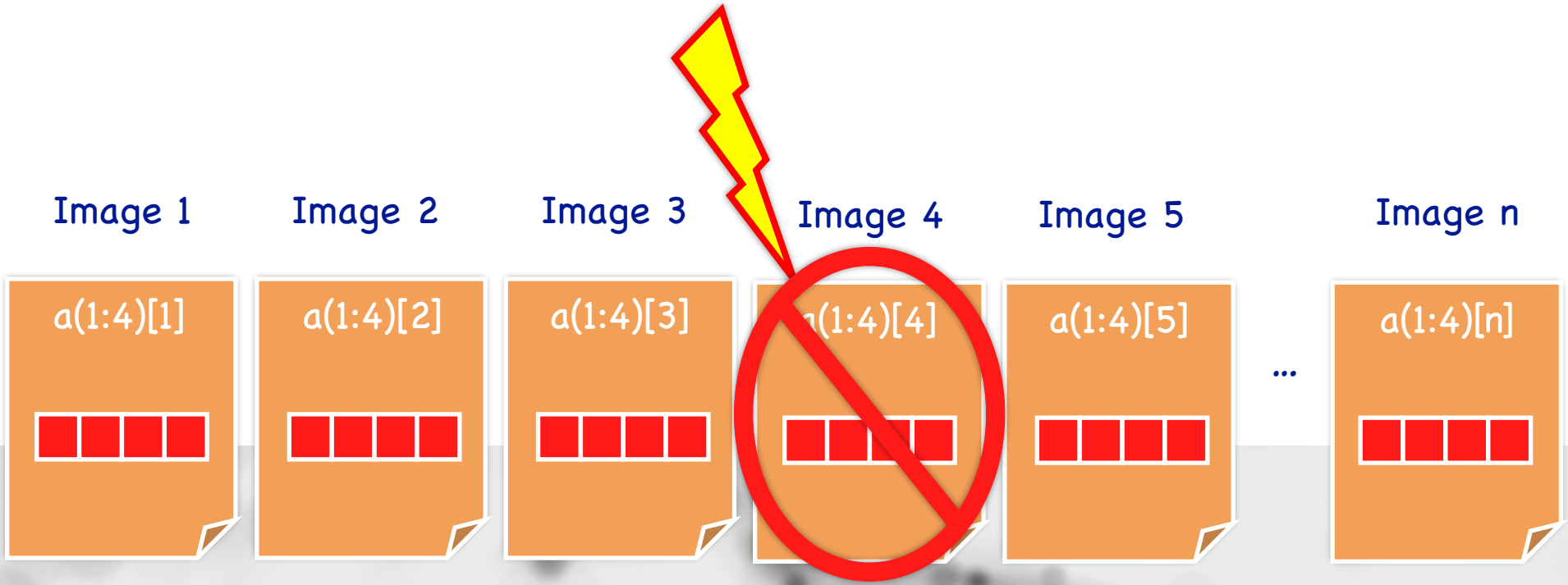


Failure Detection



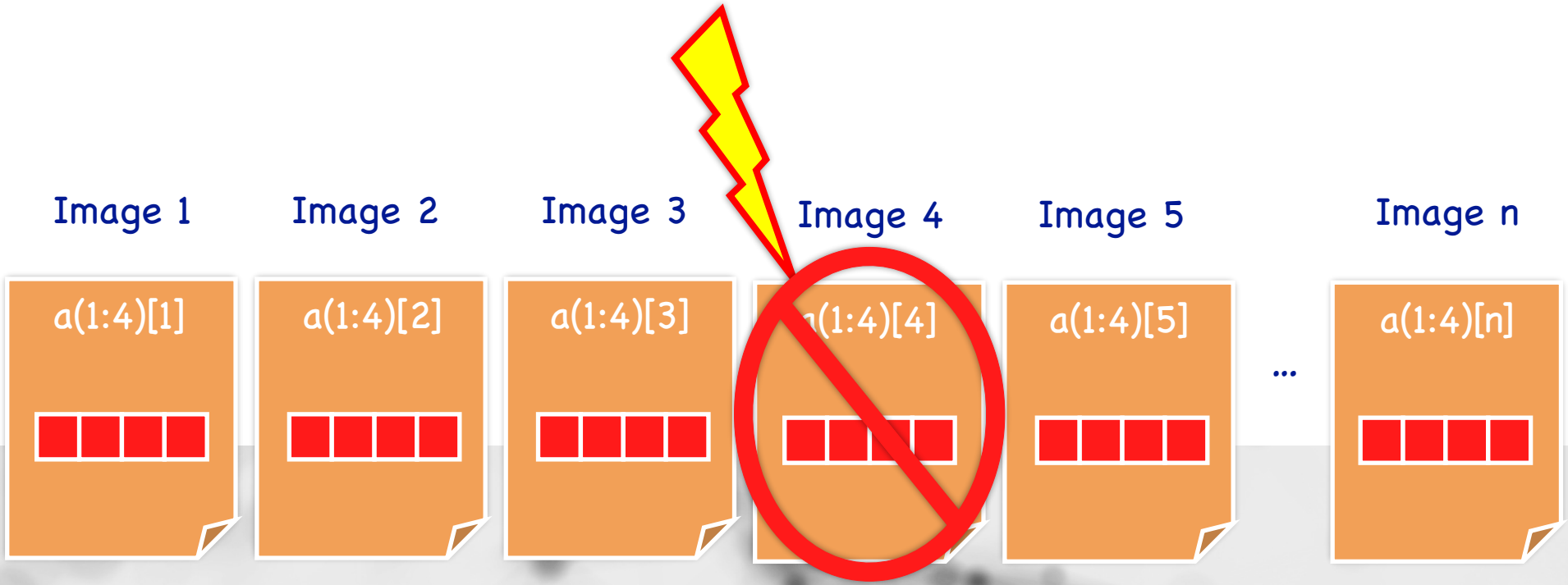

```
use iso_fortran_env, only : STAT_FAILED_IMAGE
integer :: status
sync all(stat=status)
if (status==STAT_FAILED_IMAGE) call fault_tolerant_algorithm()
```

Failure Detection




```
use iso_fortran_env, only : STAT_FAILED_IMAGE
integer :: status
sync all(stat=status)
if (status==STAT_FAILED_IMAGE) call fault_tolerant_algorithm()
```

Failure Detection



FORTRAN 2018
Failed-Image Detection

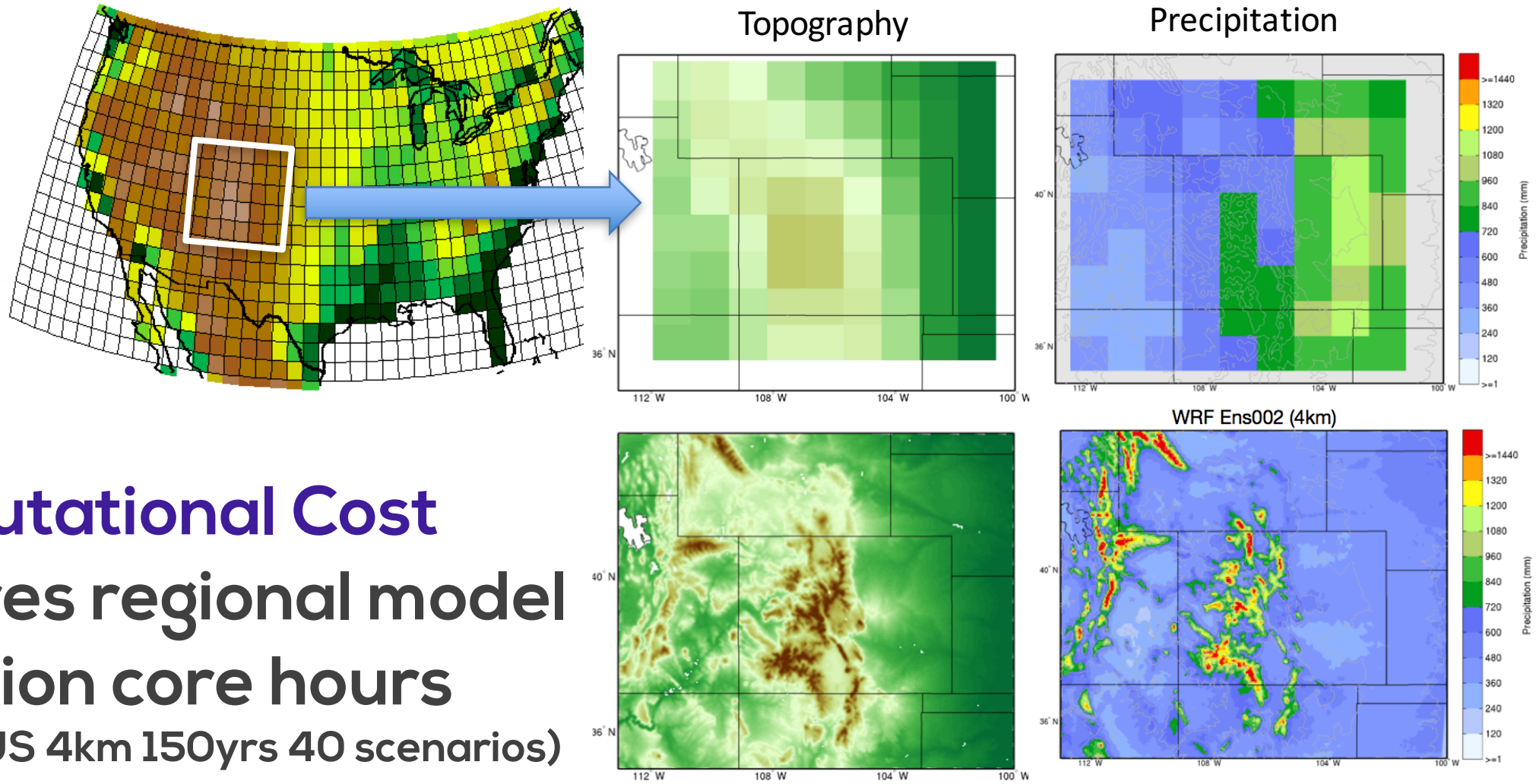
- FAIL IMAGE (simulates a failures)
- IMAGE_STATUS (checks the status of a specific image)
- FAILED_IMAGES (provides the list of failed images)
- Coarray operations may fail. STAT= attribute used to check correct behavior.

Overview

- Fortran 2018 in a Nutshell
- **ICAR & Coarray ICAR**
- Results
- Conclusions

The Climate Downscaling Problem

Climate model (top row) and application needs (bottom row)



Computational Cost

High-res regional model

>10 billion core hours

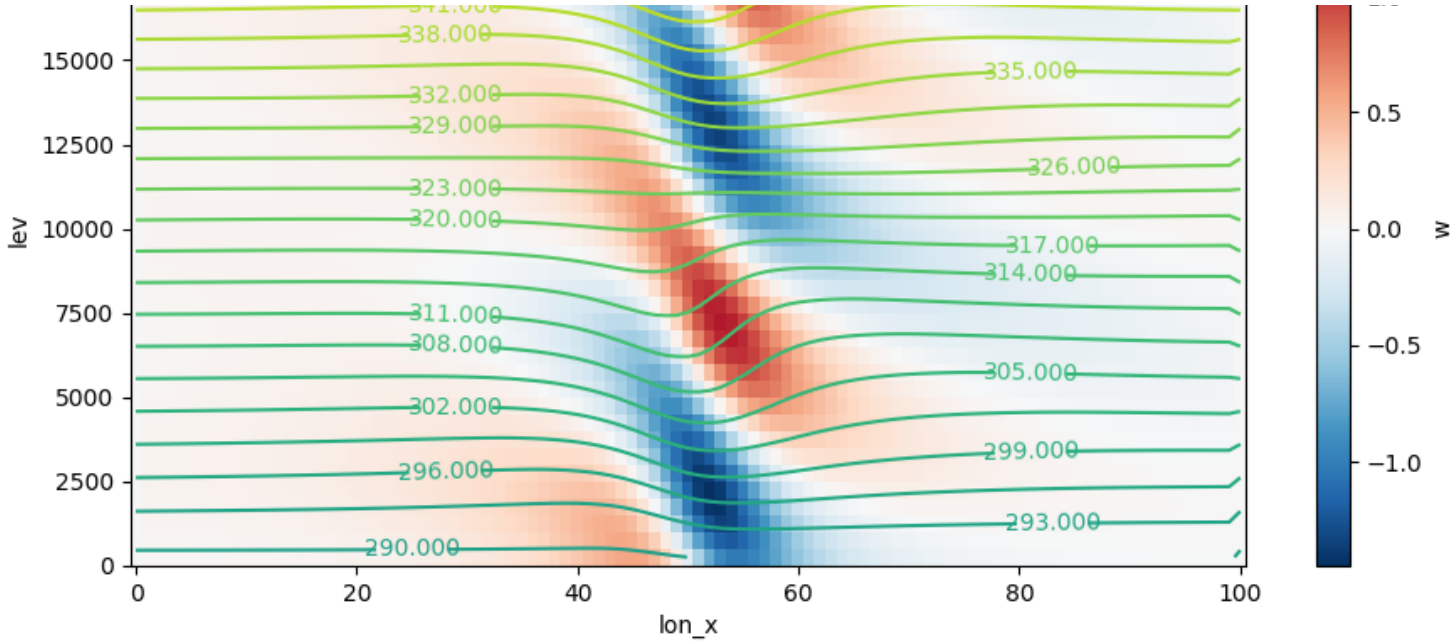
(CONUS 4km 150yrs 40 scenarios)

The Intermediate Complexity Atmospheric Research Model (ICAR)

- Analytical solution for flow over topography (right)
- 90% of the information for 1% of the computational cost

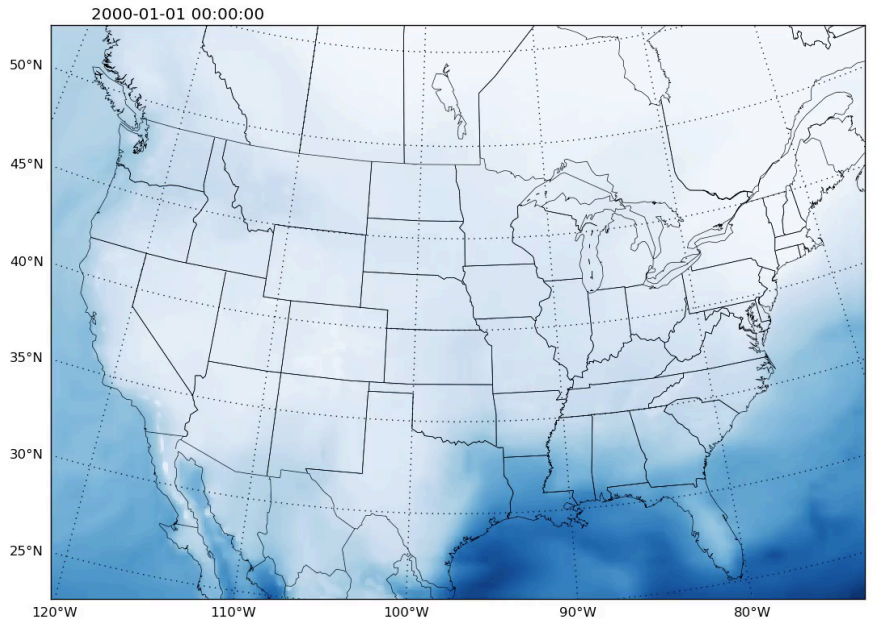
- Core Numerics:
  - 90% of cost = Cloud physics
    - grid-cells independent
  - 5% of cost = Advection
    - fully explicit, requires local neighbor communication

ICAR Wind Field over Topography



Gutmann et al (2016) J. of Hydrometeorology, 17 p957. doi:10.1175/JHM-D-15-0155.1

ICAR
Intermediate Complexity Atmospheric Research



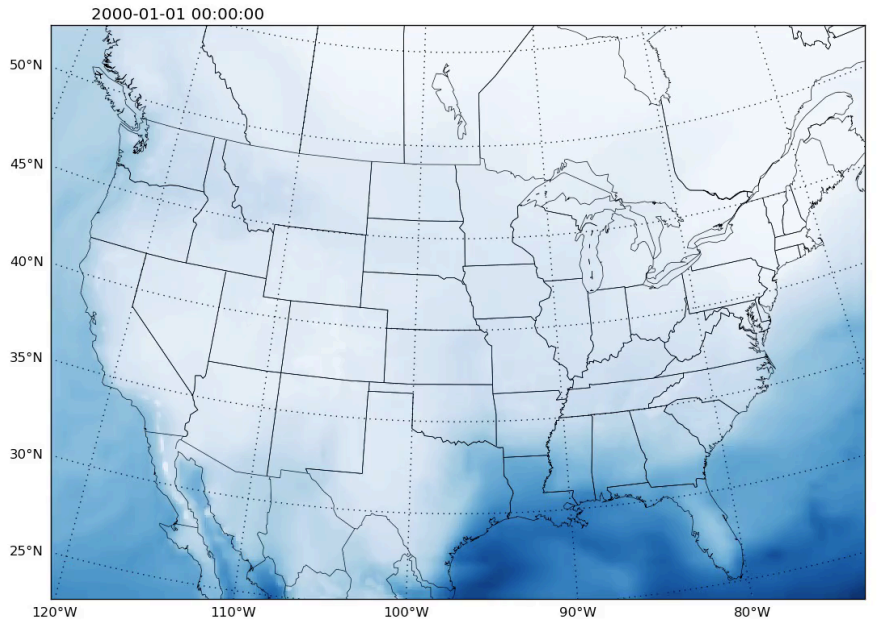
Animation of
Water vapor (blue) and
Precipitation (Green to Red)
over the contiguous United States

Output timestep : 1hr
Integration timestep : Variable (~30s)
Boundary Conditions: ERA-interim (historical)

Run on 16 cores with OpenMP

Limited scalability

ICAR
Intermediate Complexity Atmospheric Research



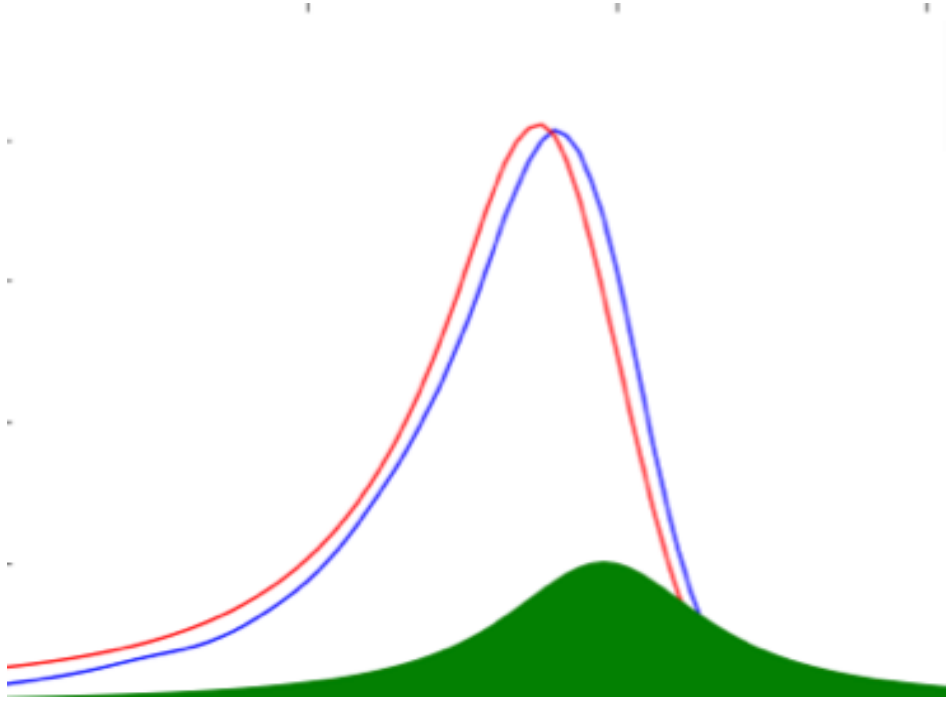
Animation of
Water vapor (blue) and
Precipitation (Green to Red)
over the contiguous United States

Output timestep : 1hr
Integration timestep : Variable (~30s)
Boundary Conditions: ERA-interim (historical)

Run on 16 cores with OpenMP

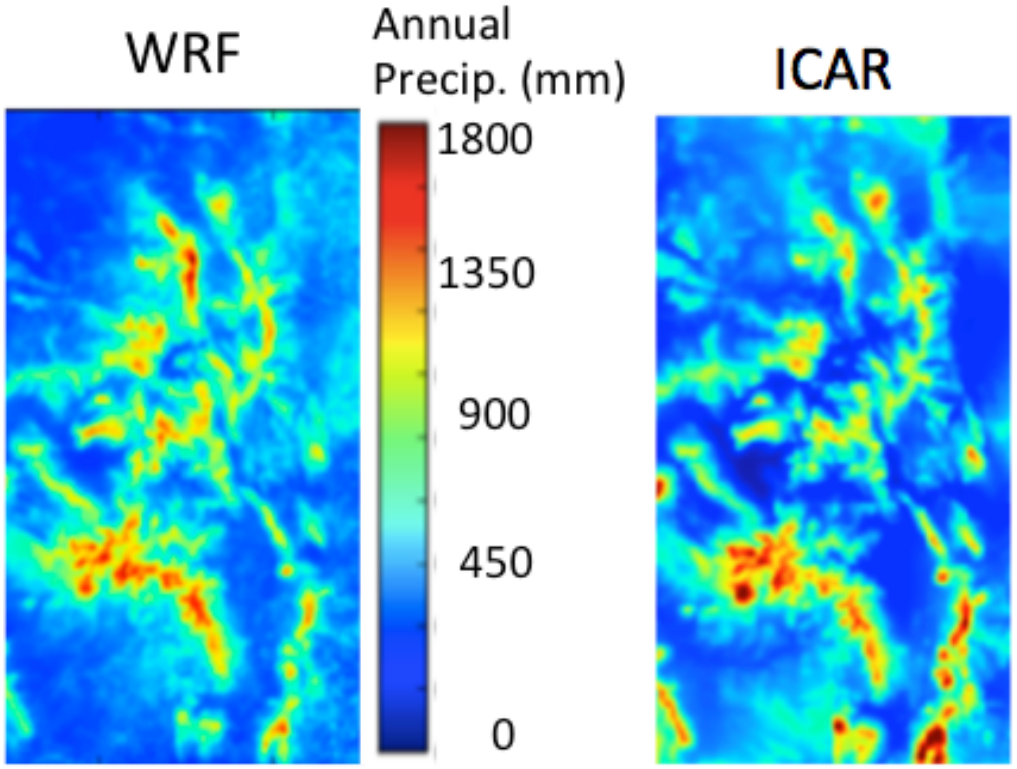
Limited scalability

ICAR comparison to "Full-physics" atmospheric model (WRF)



Ideal Hill case

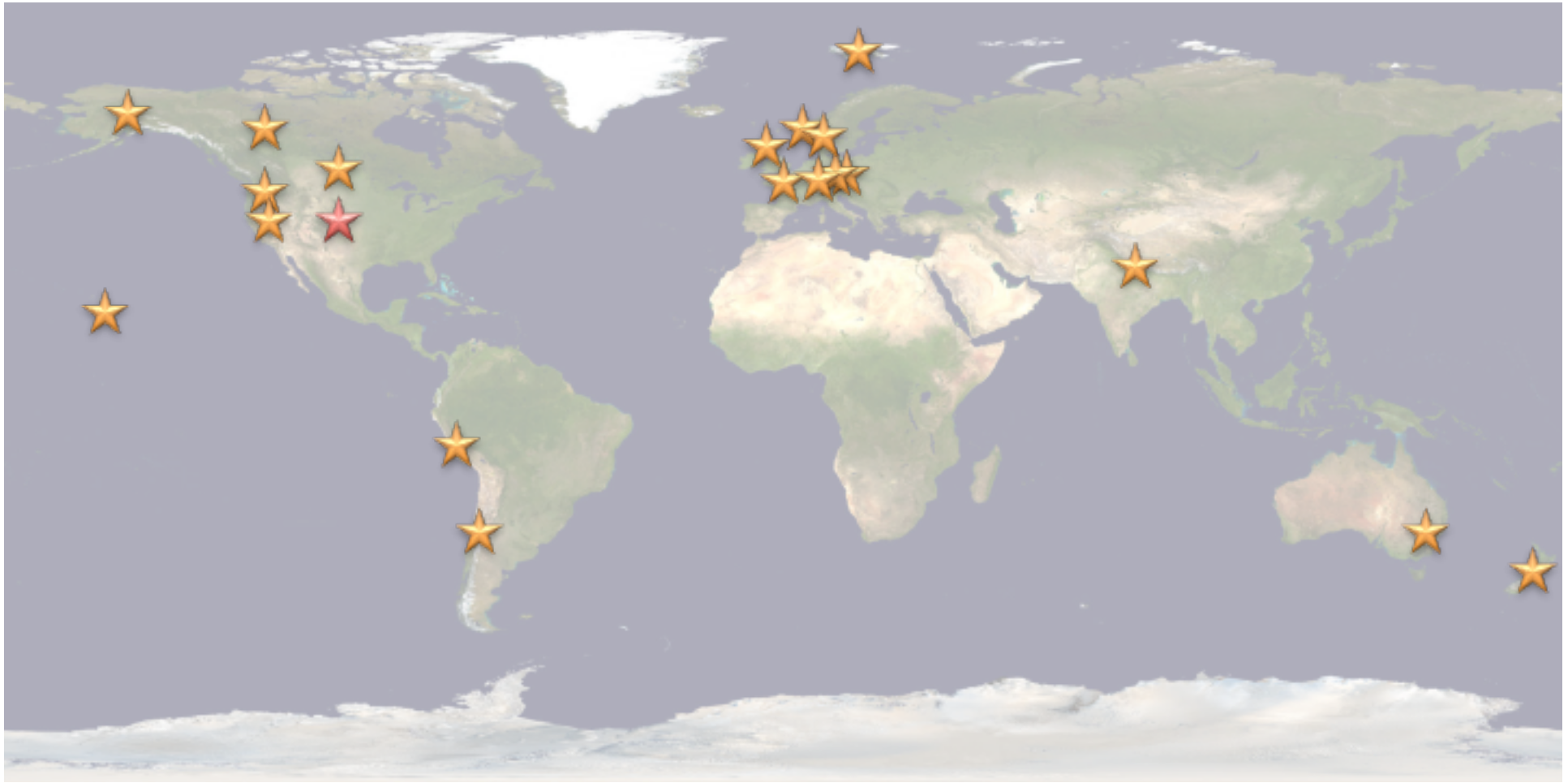
ICAR (red) and WRF (blue) precipitation over an idealized hill (green)



Real Simulation

WRF (left) and ICAR (right) Season total precipitation over Colorado Rockies

ICAR Users and Applications



http://github.com/NCAR/icar.git

Coarray ICAR Mini-App

- Object-oriented design
- Collective broadcast of initial data
- Overlaps communication & computation via one-sided “puts.”
- Coarray halo exchanges with 3D, 1st-order upwind advection (~2000 lines of new code)
- Cloud microphysics (~5000 lines of pre-existing code)

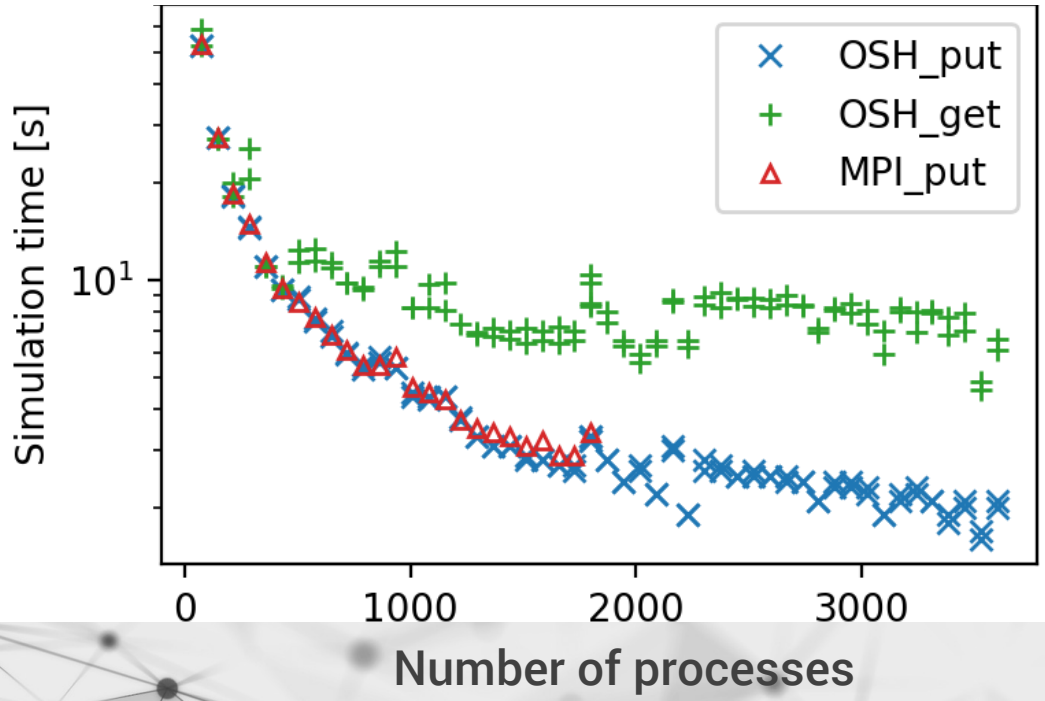
Overview

- Fortran 2018 in a Nutshell
- ICAR & Coarray ICAR
- **Results**
- Conclusions

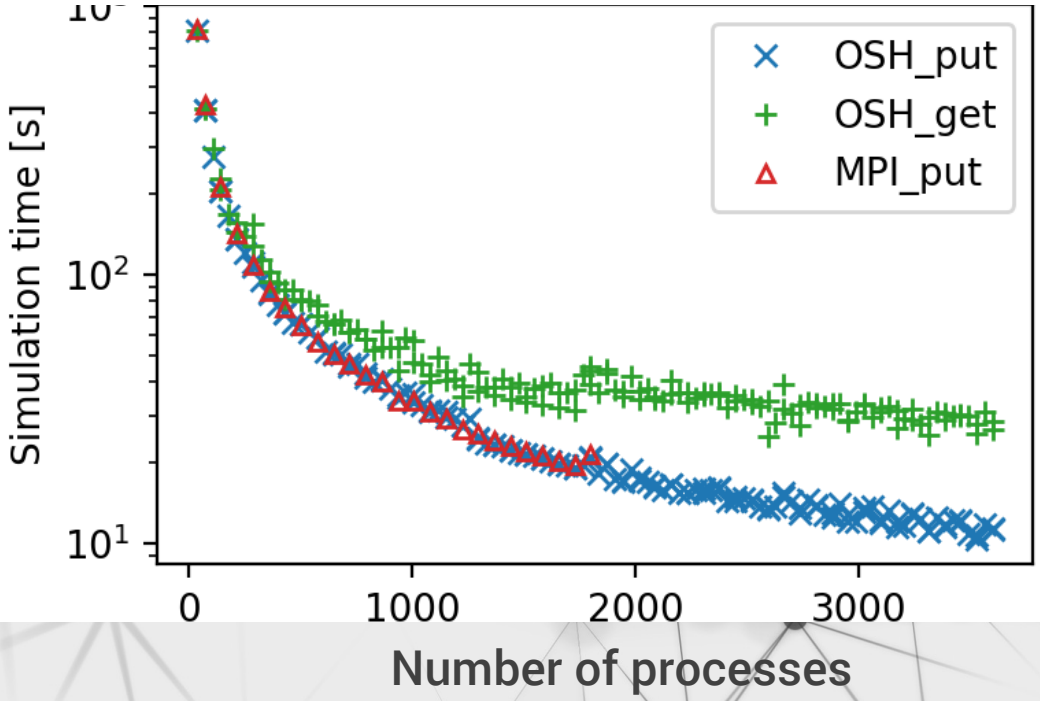
Coarray ICAR
Simulation Time

- OpenSHMEM vs MPI
- Puts vs gets

500 x 500 x 20 grid



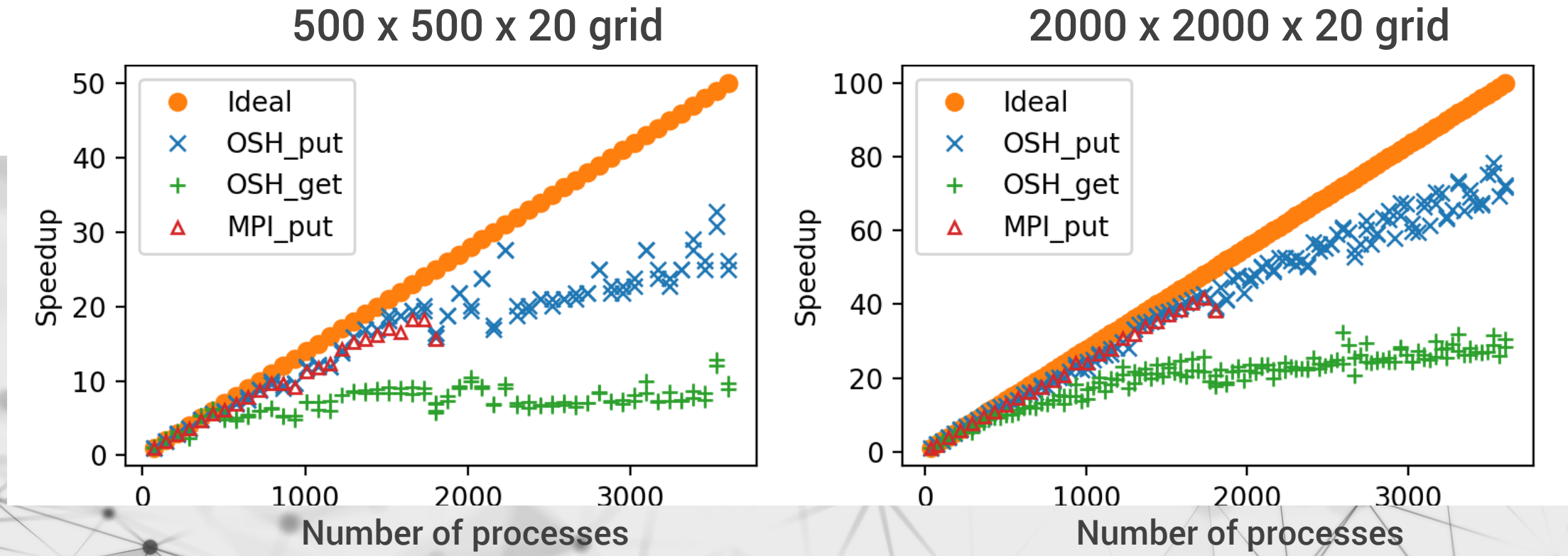
2000 x 2000 x 20 grid



GCC 6.3 on NCAR “Cheyenne” SGI ICE XA Cluster: 4032 nodes, 2 x 18-core Xeon Broadwell @ 2.3 GHz

Coarray ICAR

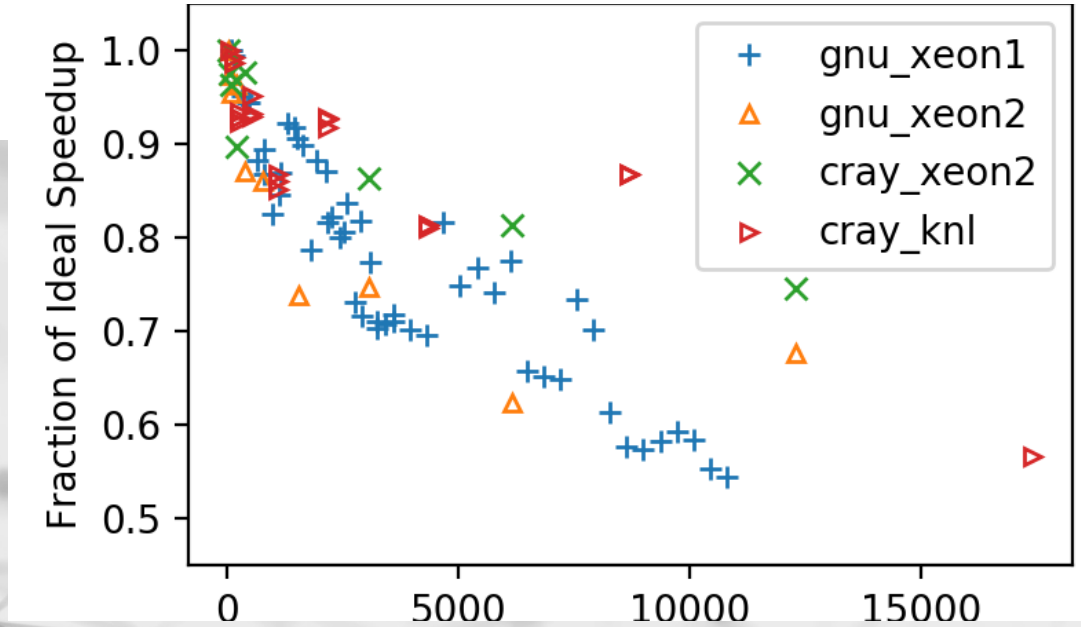
Speedup

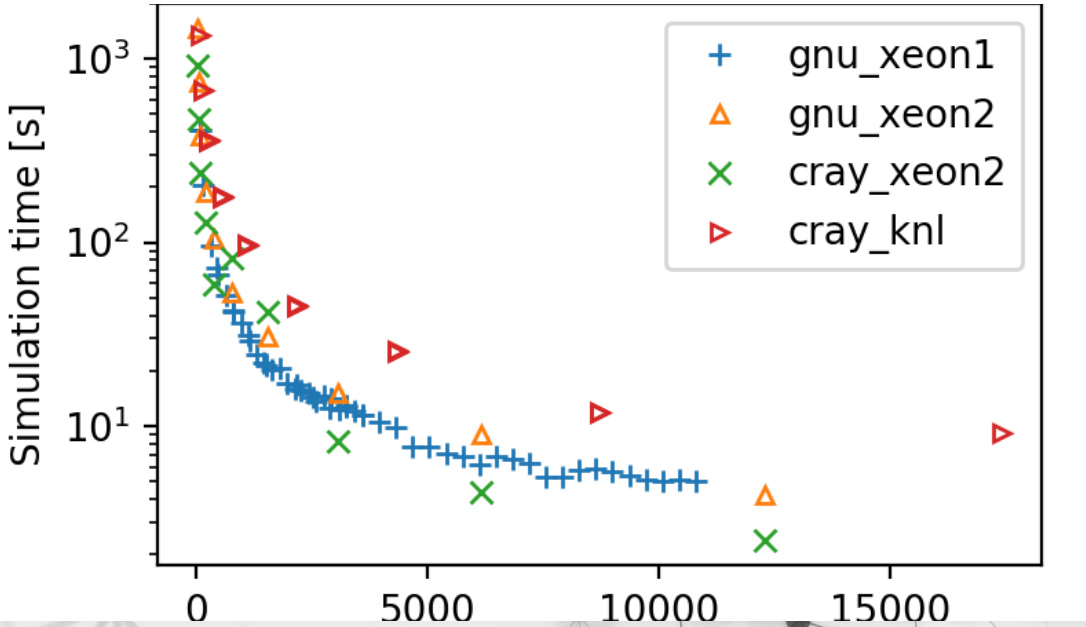


GCC 6.3 on NCAR “Cheyenne” SGI ICE XA Cluster: 4032 nodes, 2 x 18-core Xeon Broadwell @ 2.3 GHz

Coarray ICAR

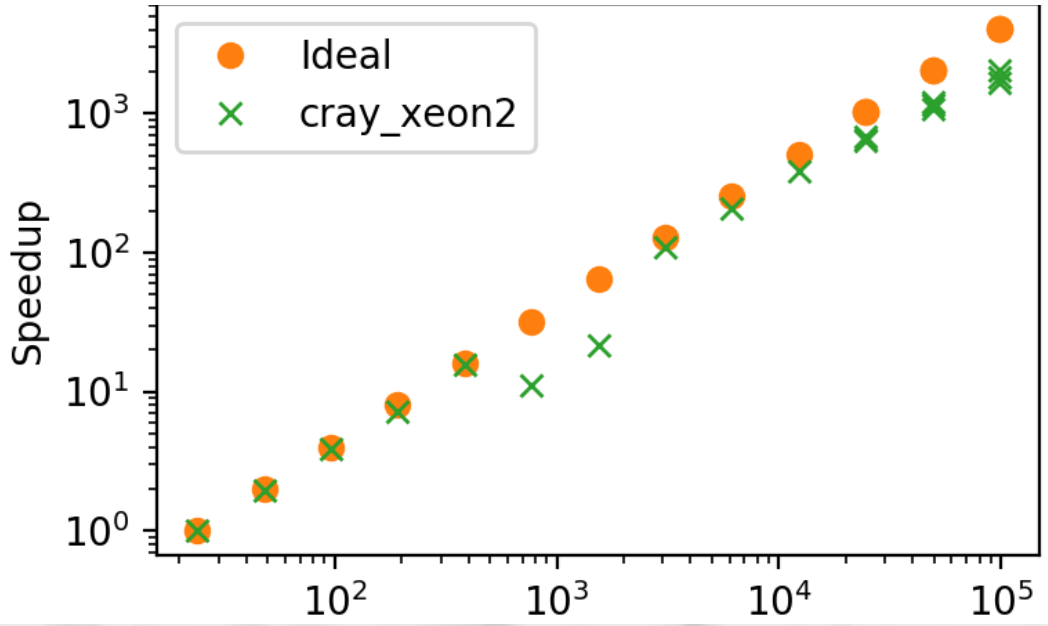
Xeon vs KNL



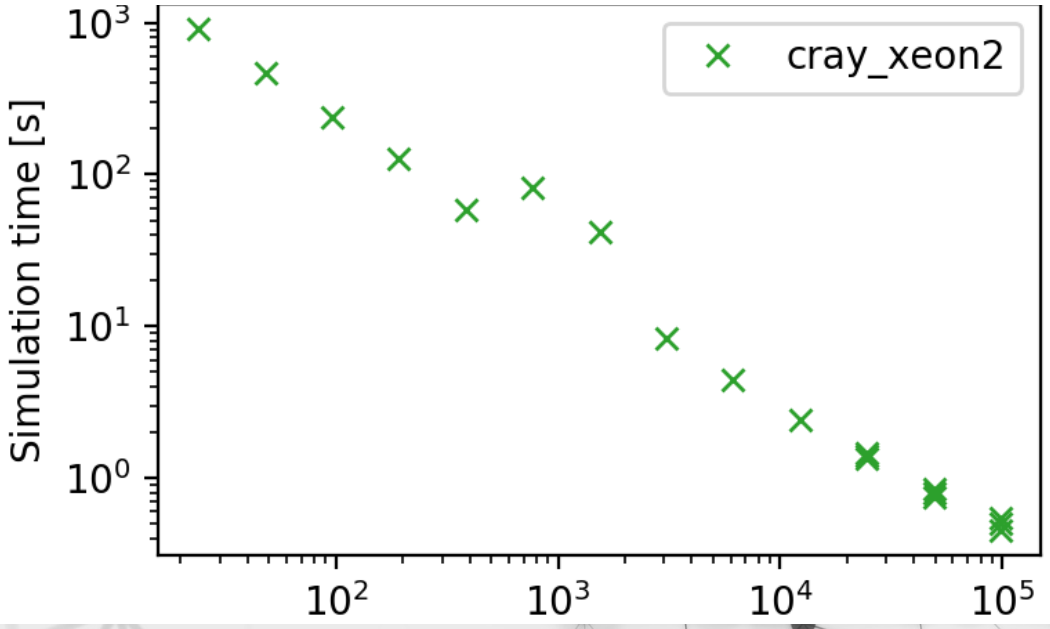


Compilers & platforms: GNU on Cheyenne; Cray compiler on Edison (Broadwell), Cori (KNL).

Coarray ICAR
At Scale



Number of processes



Number of processes

Cray compiler on Edison.

Conclusions

- Fortran 2018 is a PGAS language supporting SPMD programming at scale.
- Programming-model agnosticism is a life-saver.
- High productivity pays off: from shared-memory parallelism to 100,000 cores in ~100 person-hours.