Towards Exascale Computing with Fortran 2015

Alessandro Fanfarillo
National Center for Atmospheric Research

Damian Rouson
Sourcery Institute

Outline

- Parallelism in Fortran 2008
  - SPMD
  - PGAS
- Exascale challenges
- Fortran 2015 targeting exascale challenges
  - Events
  - Teams
  - Collective subroutines
  - Failed-image detection
  - Richer set of atomic subroutines
- Scientific kernels

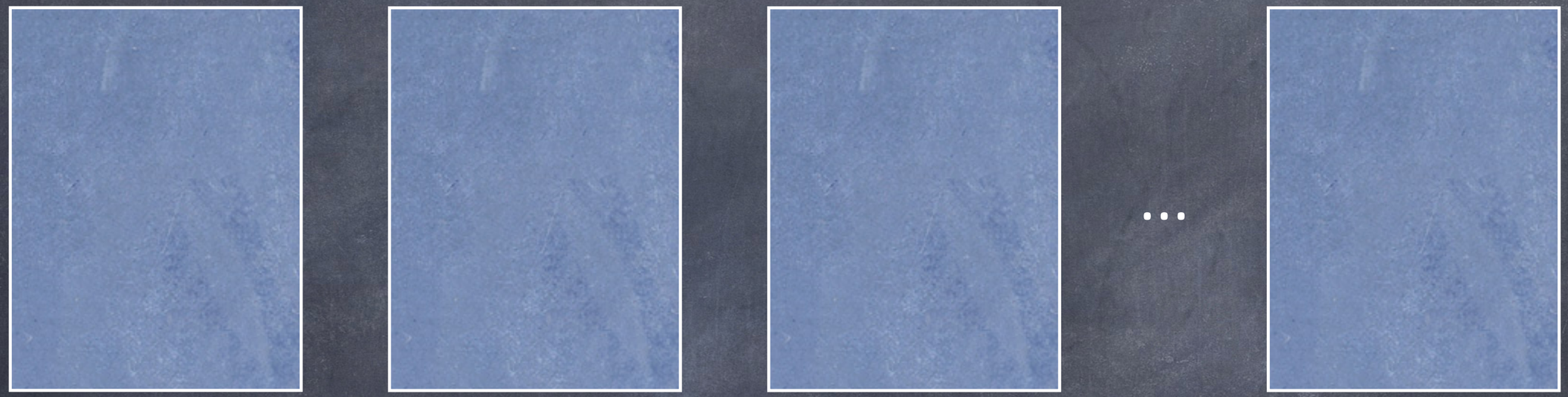
SPMD

Images -> {processes | threads }

SPMD

Images -> {processes | threads }

Image 1 Image 2 Image 3 ... Image n

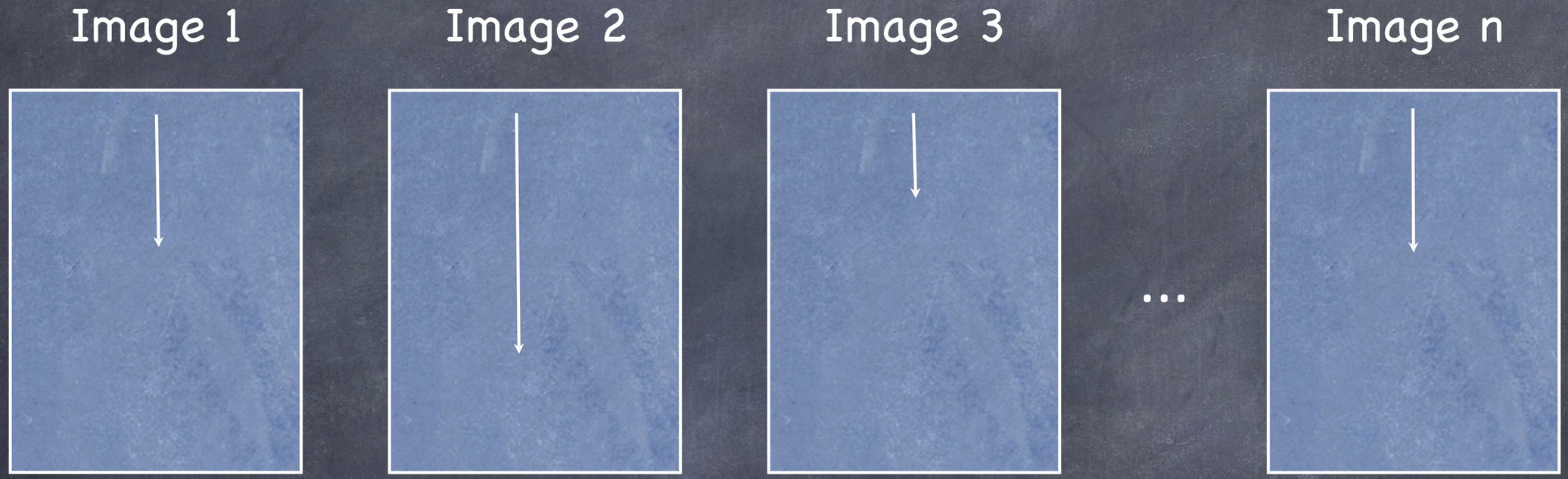


Images execute asynchronously until they reach a programmer-specified synchronization:

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

SPMD

Images -> {processes | threads }

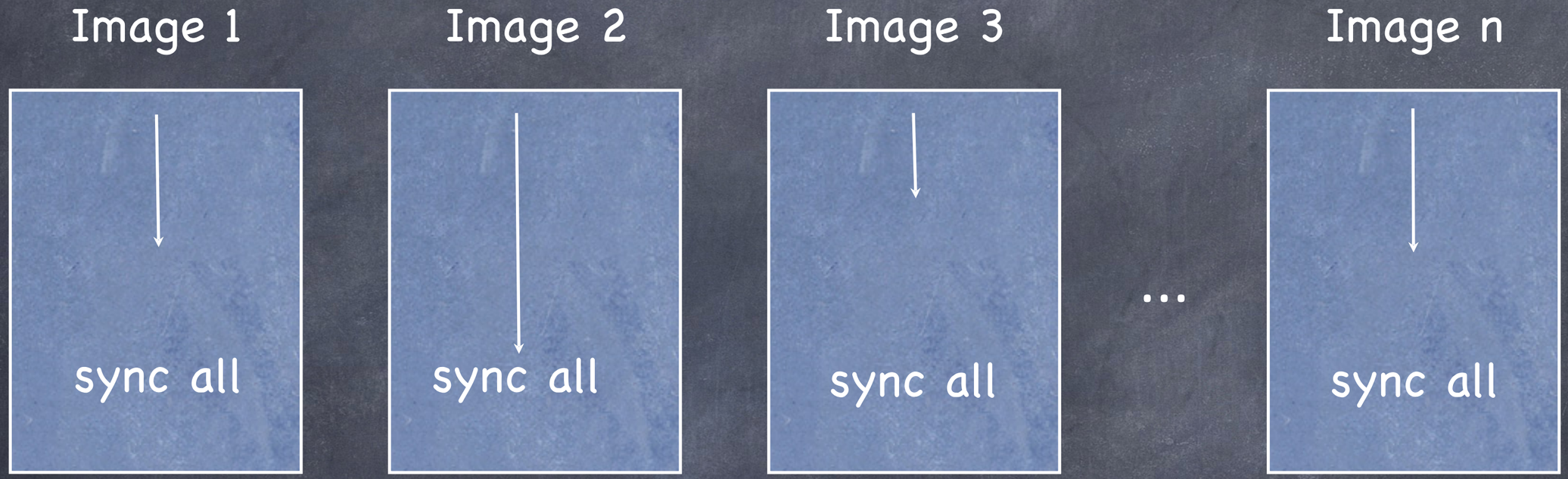


Images execute asynchronously until they reach a programmer-specified synchronization:

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

SPMD

Images -> {processes | threads }

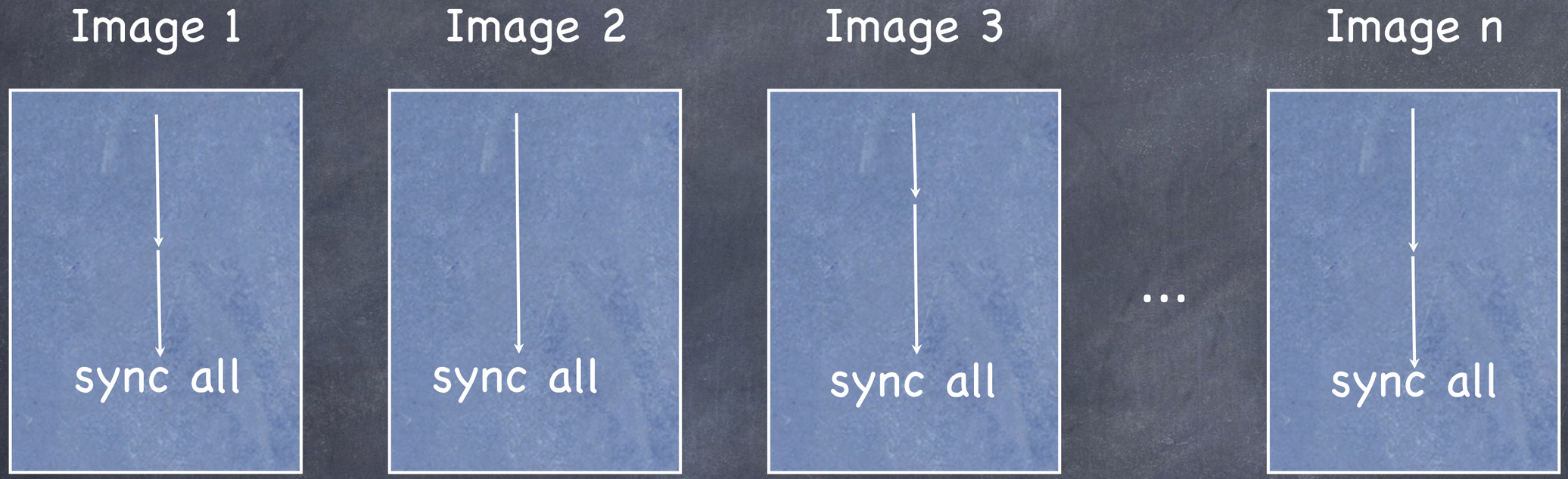


Images execute asynchronously until they reach a programmer-specified synchronization:

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

SPMD

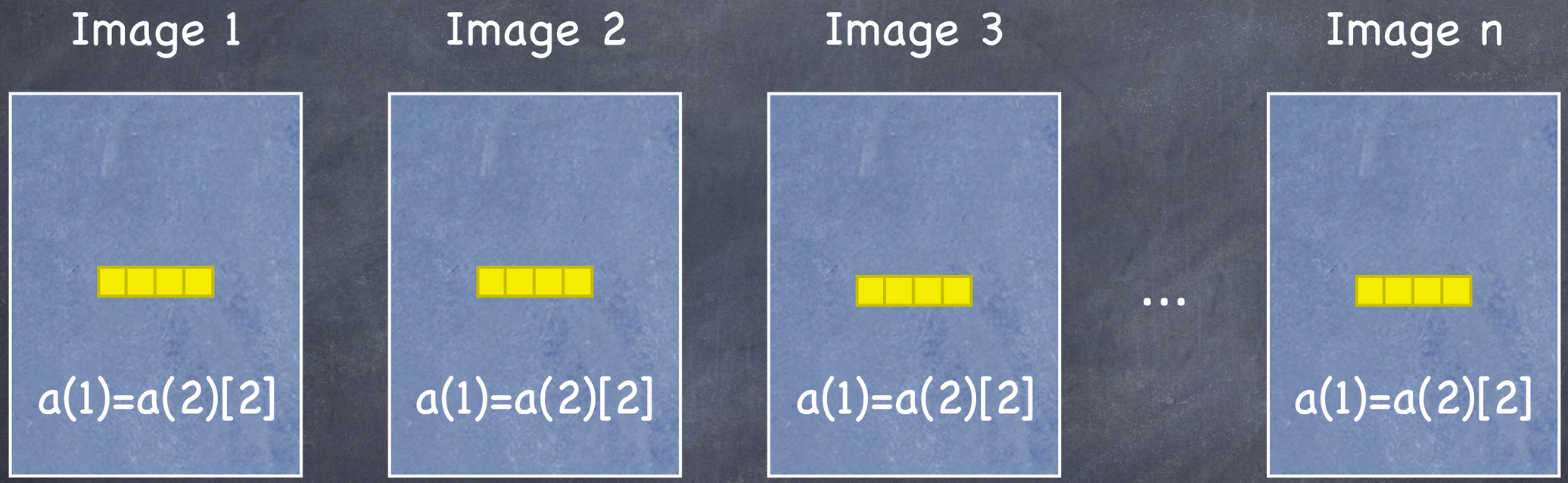
Images -> {processes | threads }



Images execute asynchronously until they reach a programmer-specified synchronization:

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

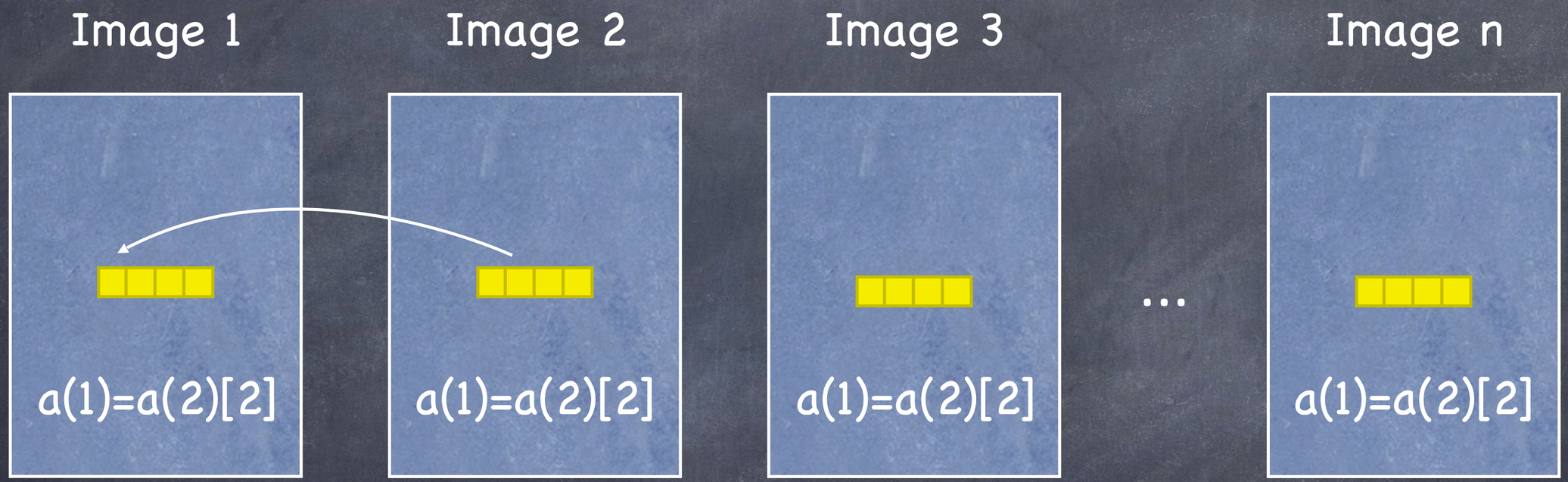
PGAS



Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

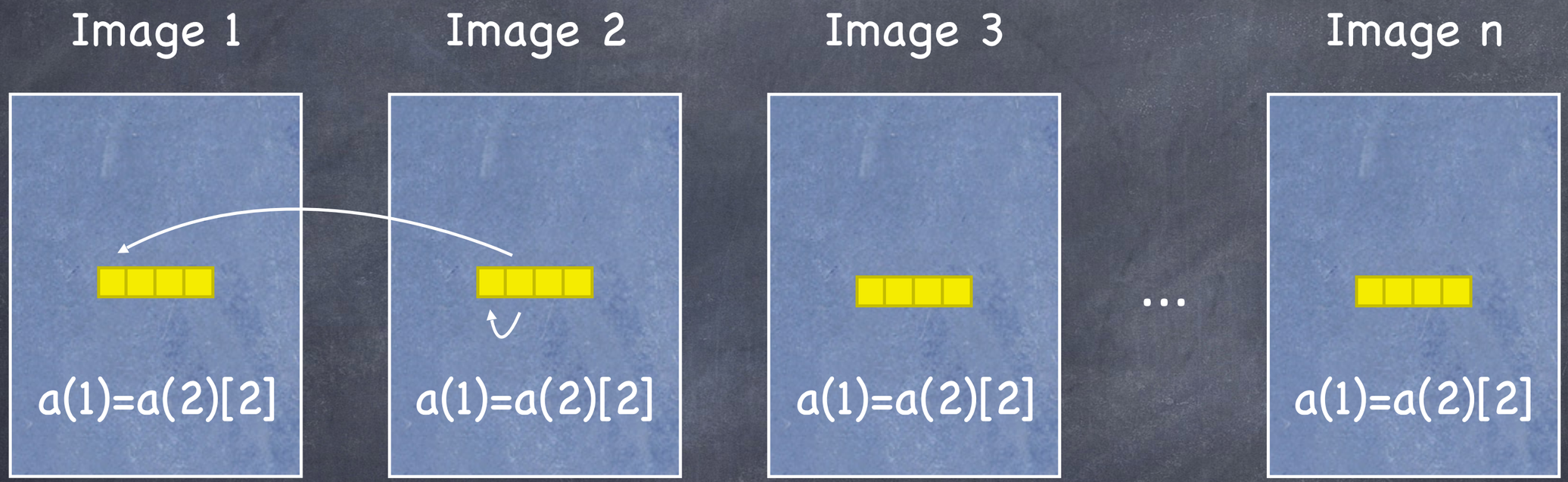
PGAS



Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

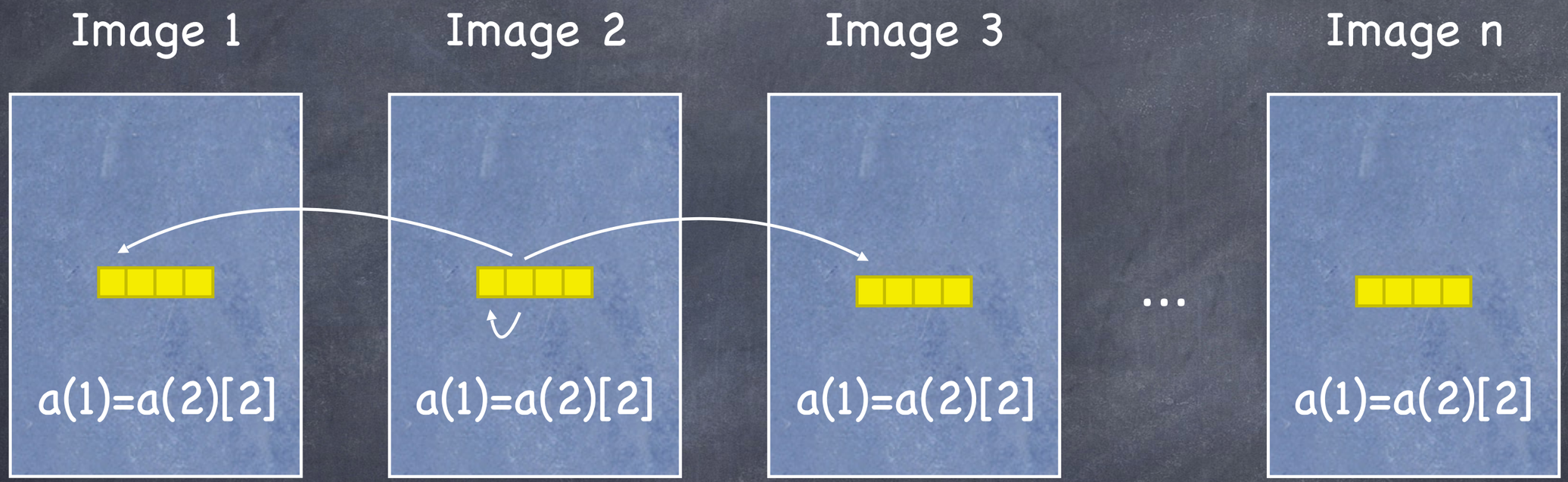
PGAS



Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

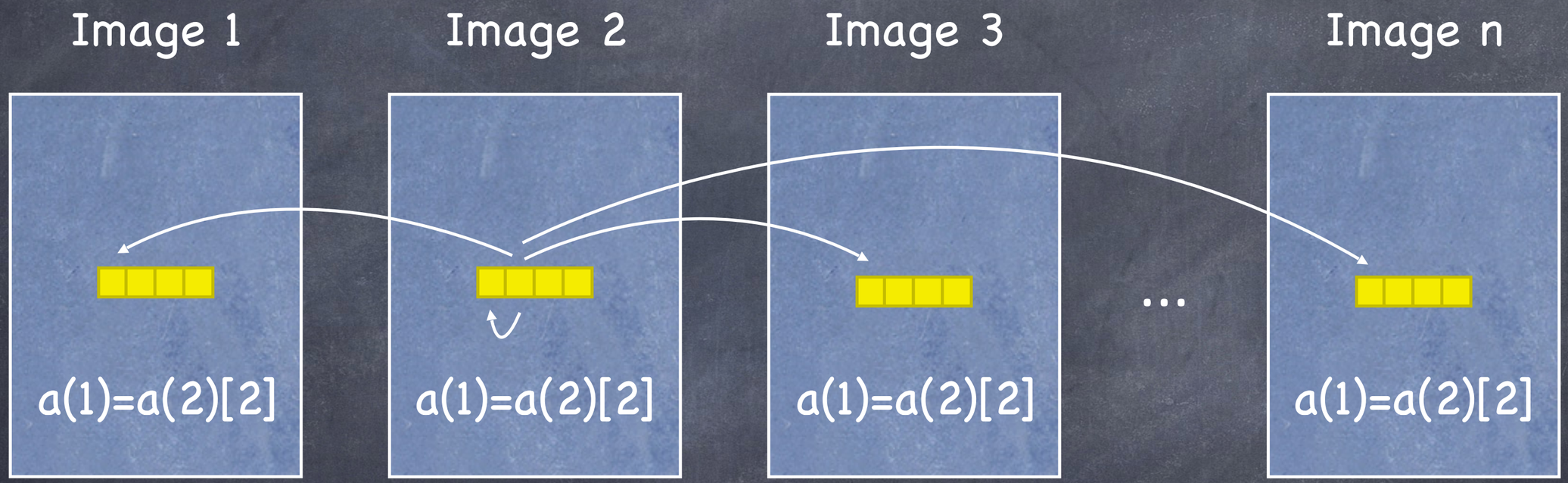
PGAS



Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

PGAS

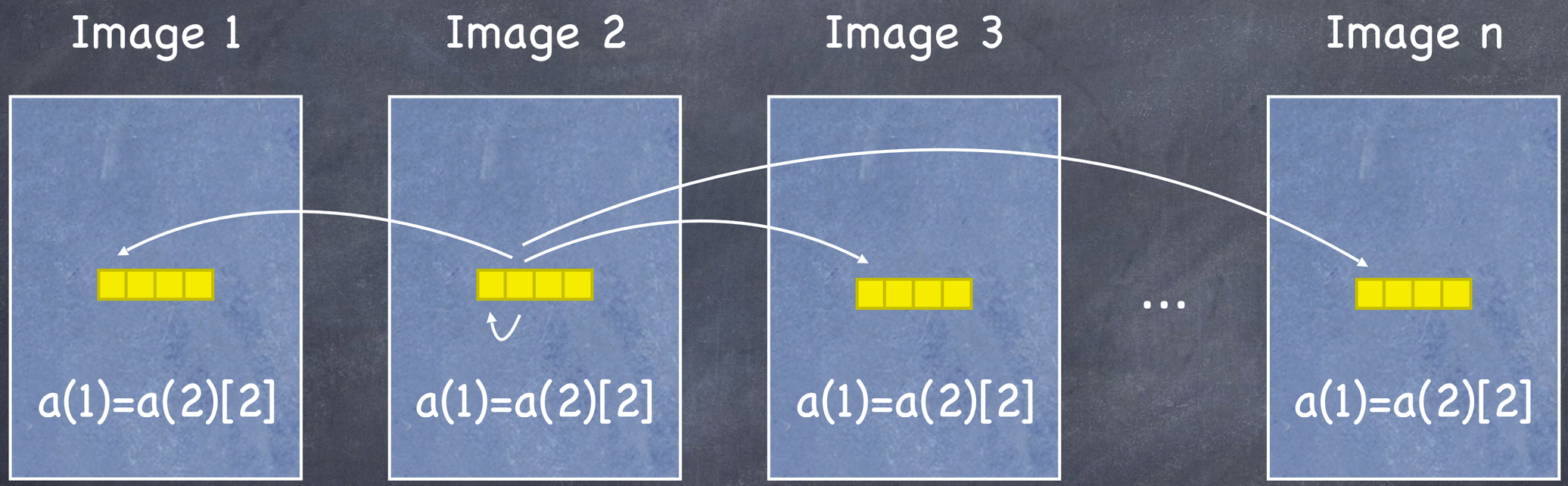


Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

PGAS

```
type(foo), allocatable :: a(:)[:]
integer, parameter :: local_size=100
allocate(a(local_size)[*])
```



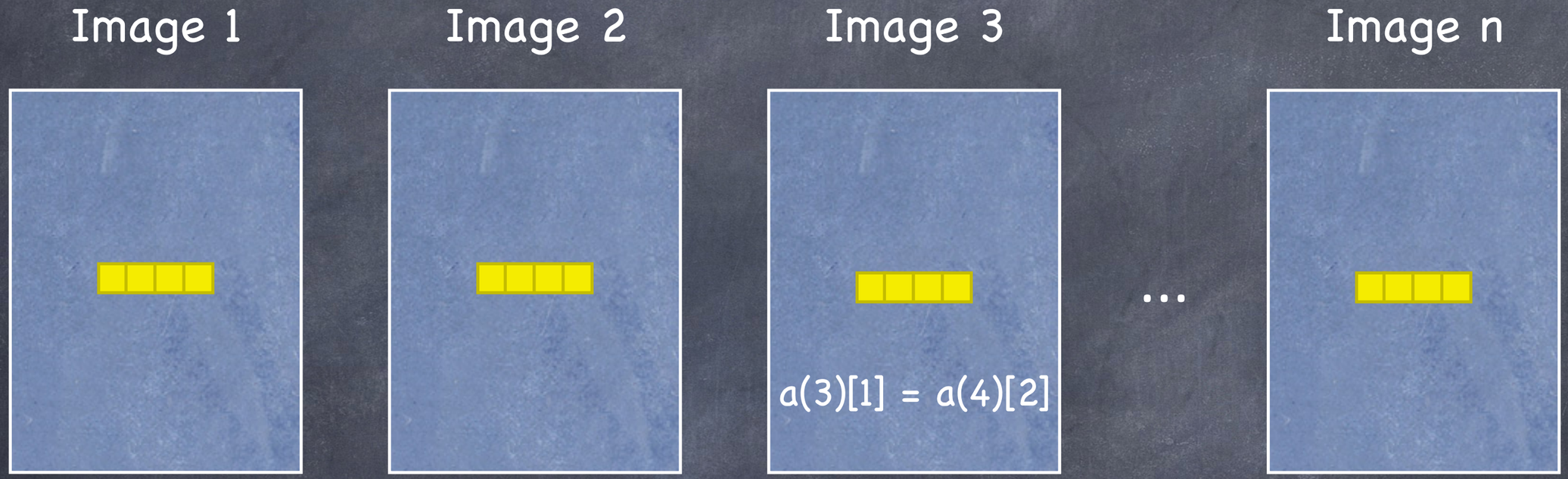
The ability to drop the square brackets harbors two important implications:

1. One can easily determine where communication might happen.
2. Parallelized legacy codes need not change where communication is not required.

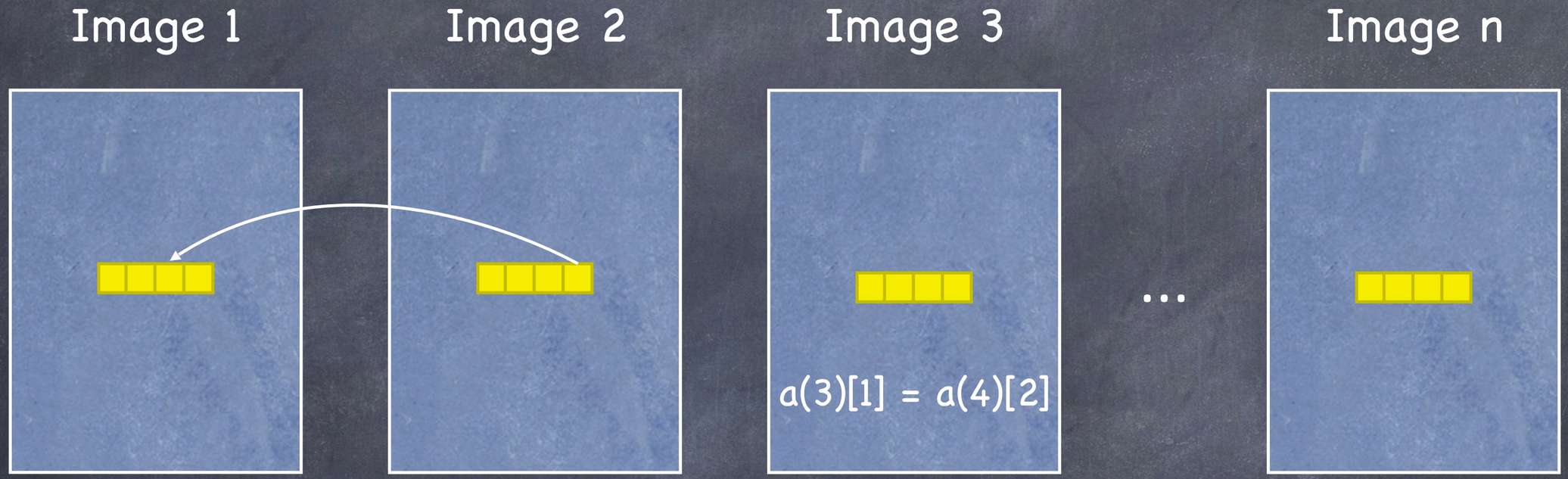
Coarrays integrate with other language features:

1. All the usual Fortran 90 array syntax applies to Fortran 2008 coarrays.
2. "a" can be an object of derived type.

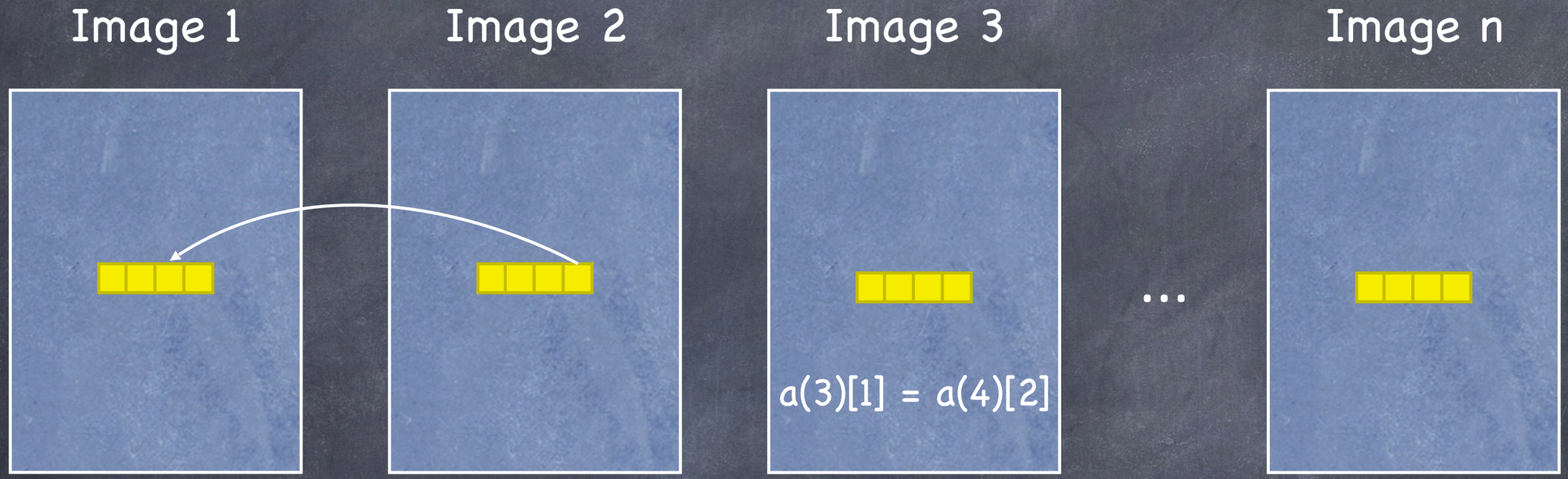
User-Defined Communication Patterns



User-Defined Communication Patterns



User-Defined Communication Patterns



Communication between two images that is invoked remotely:

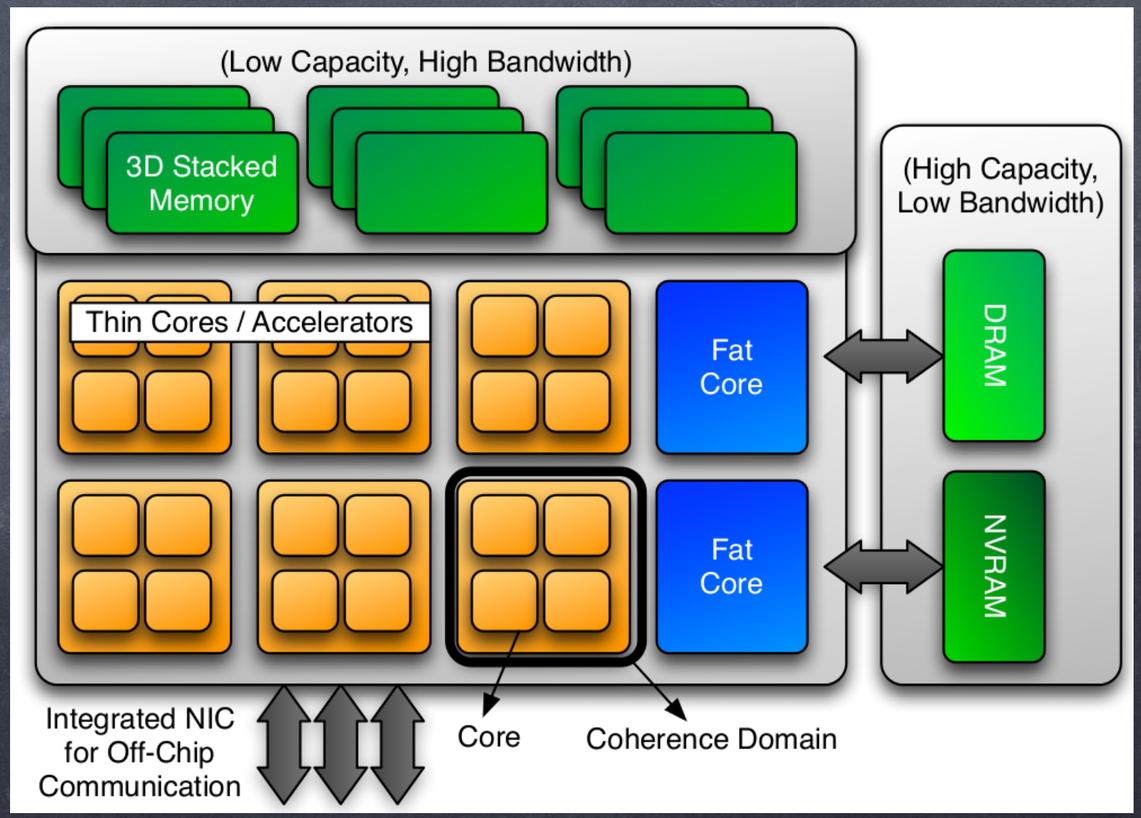
```
if (this_image()==3) a(3)[1] = a(4)[2]
```

Outline

- Parallelism in Fortran 2008
  - SPMD
  - PGAS
- Exascale challenges
- Fortran 2015 targeting exascale challenges
  - Events
  - Teams
  - Collective subroutines
  - Failed-image detection
  - Richer set of atomic subroutines
- Scientific kernels

Exascale Challenges -> Fortran 2015 Responses

- High parallelism within a single chip -> events, collectives, atomics
- Heterogeneous hardware within a single chip -> events
- Higher failure rates -> failed-image detection
- Locality Control -> Teams



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.

Outline

- Parallelism in Fortran 2008
  - SPMD
  - PGAS
- Exascale challenges
- Fortran 2015 targeting exascale challenges
  - Events
  - Teams
  - Collective subroutines
  - Failed-image detection
  - Richer set of atomic subroutines
- Scientific kernels

Events

An event is an intrinsic derived type that encapsulates a private integer (atomic_int_kind) component default-initialized to zero:

```
use iso_fortran_env, only : event_type

type(event_type), allocatable :: greeting_ready(:)[:]
type(event_type) :: ok_to_overwrite[*]
```

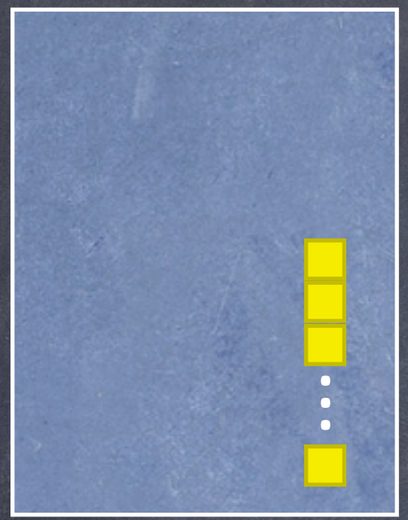
User-Controlled Segment Ordering: One image posts an event to a remote image, e.g., to signal that data is ready for pickup. The remote image might query the event to obtain its atomic integer count, which holds the number of posts.

New statements and subroutines:

Statement /Procedure	Image Control	Additional Side Effect
event post	✓	atomic_add 1
event_query		defines count
event wait	✓	atomic_add -1

Events

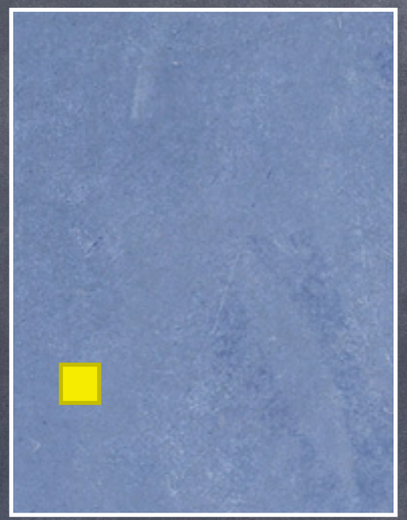
Image 1



greeting_ready(:)[1]

Image 2

Image 3



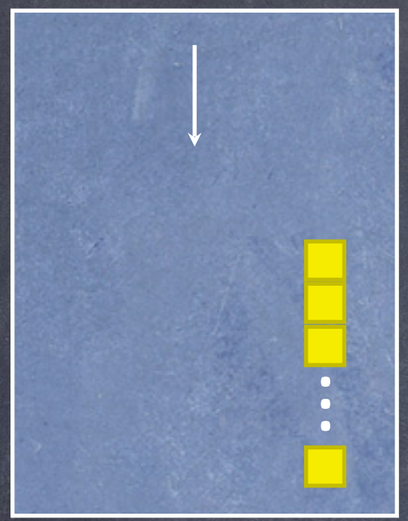
ok_to_overwrite[3]

...

Image n

Events

Image 1



greeting_ready(:)[1]

Image 2

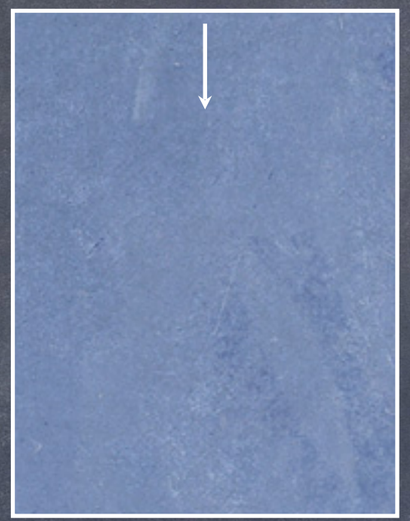
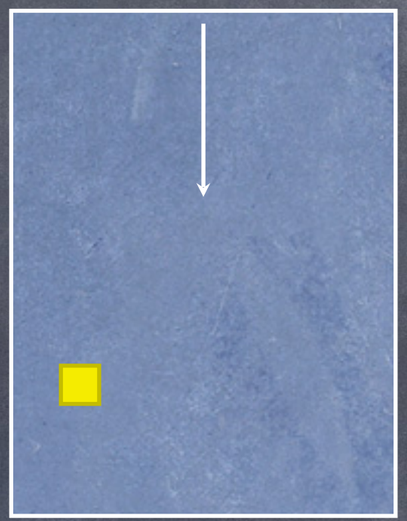


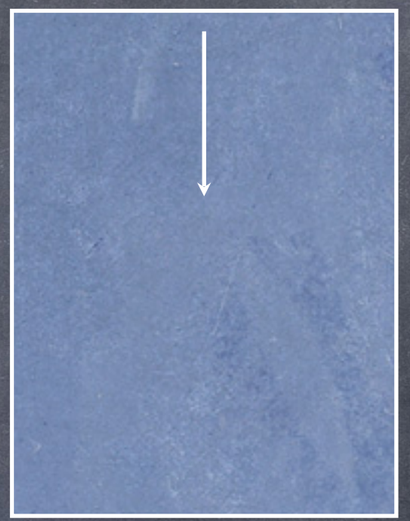
Image 3



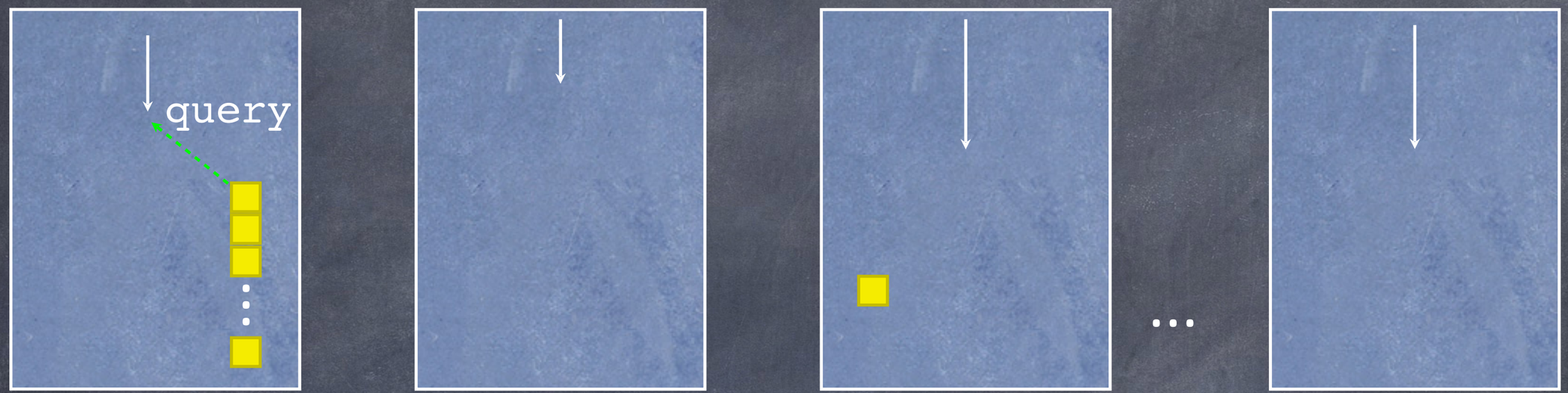
ok_to_overwrite[3]

...

Image n



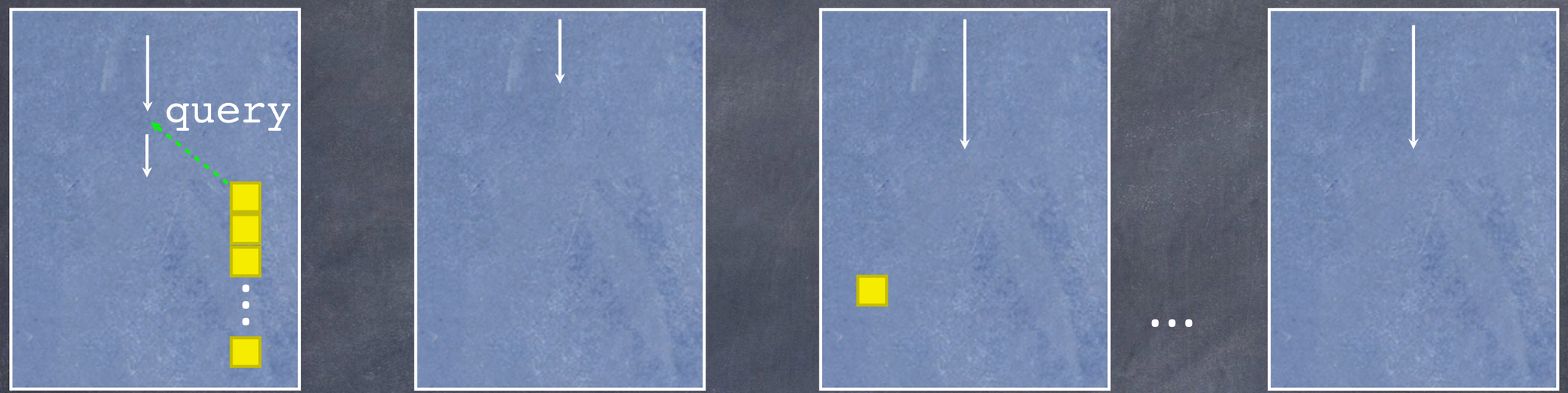
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

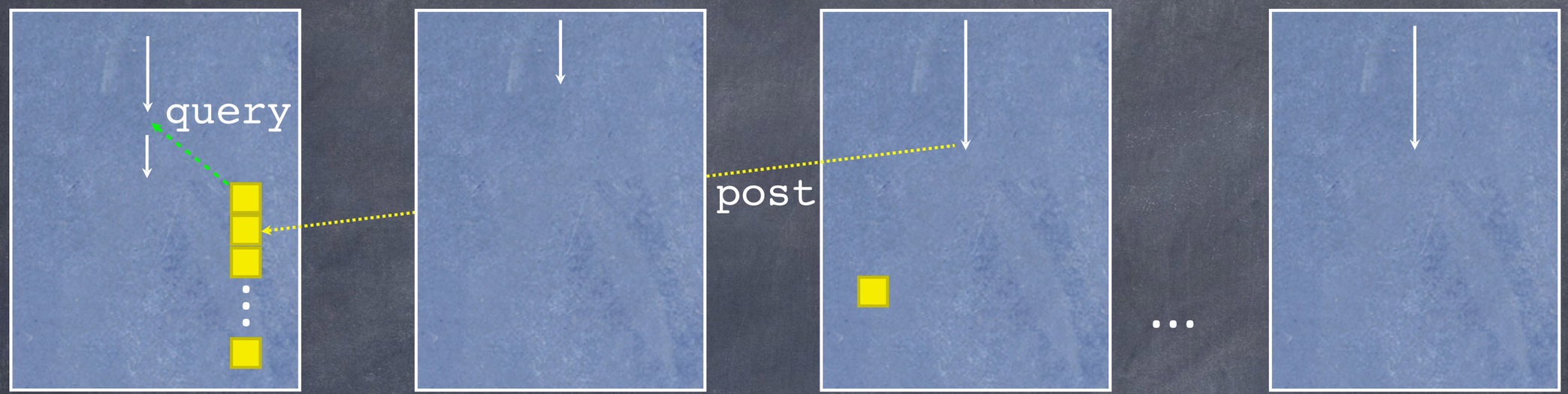
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

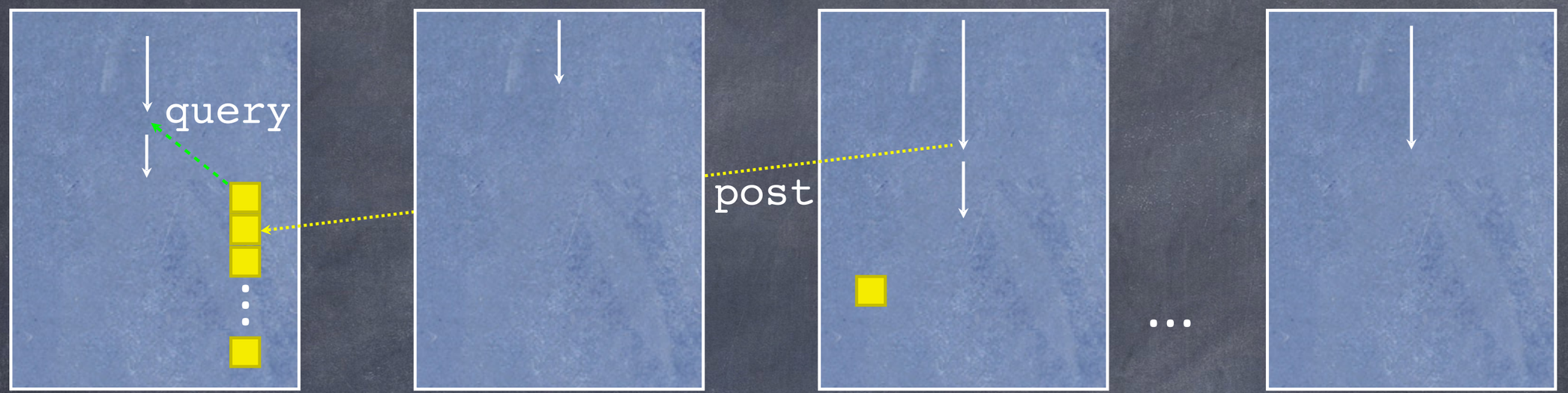
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

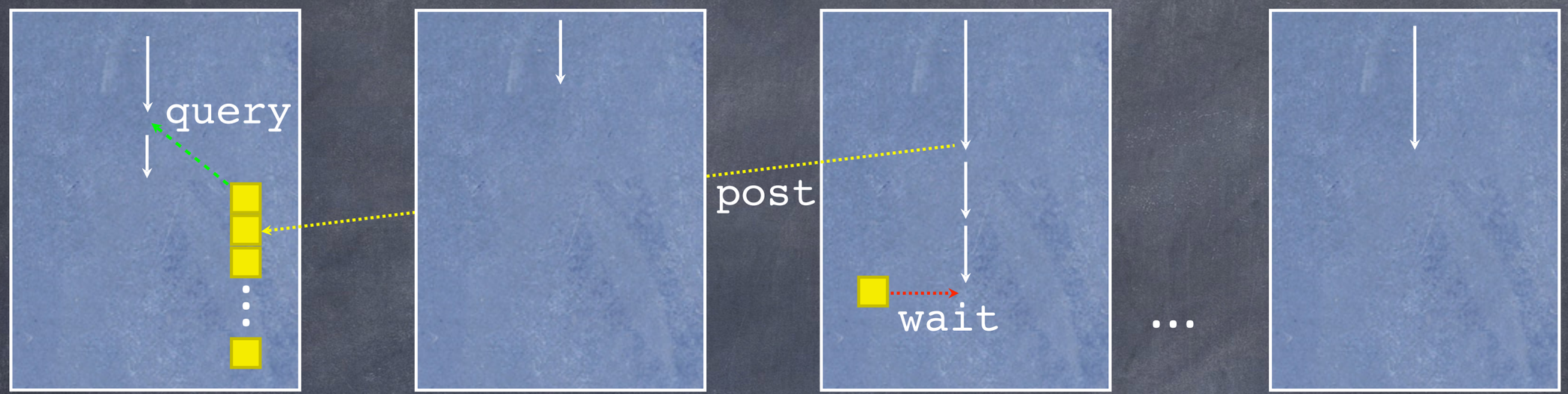
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

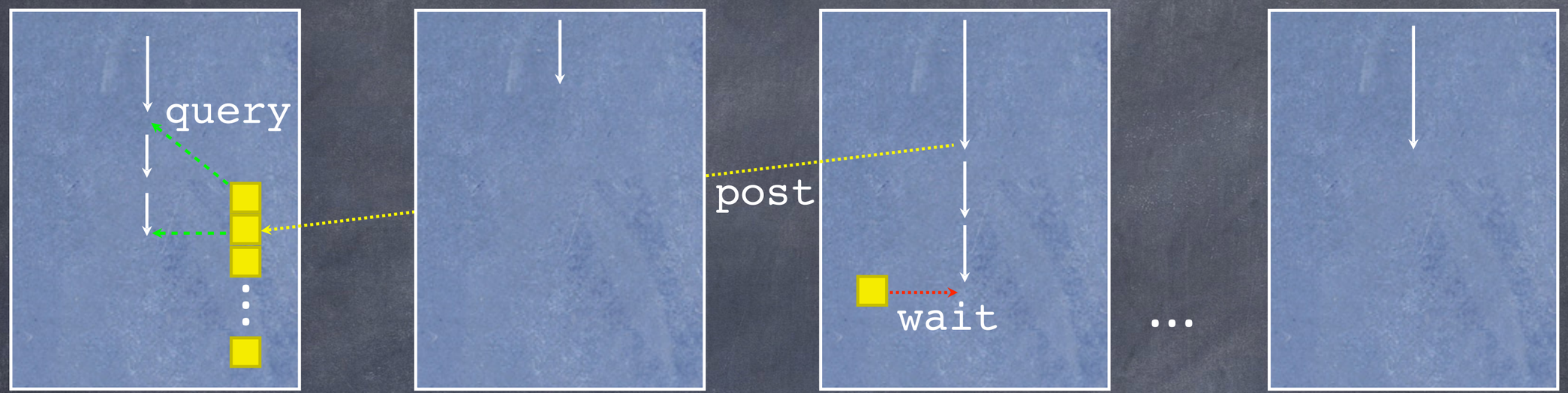
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

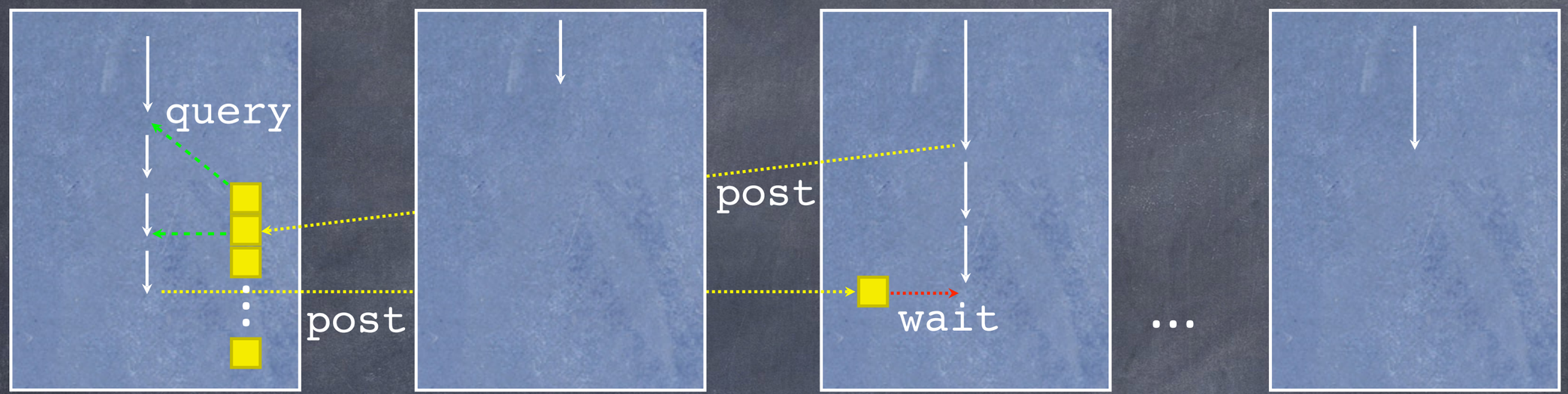
Events



greeting_ready(:)[1]

ok_to_overwrite[3]

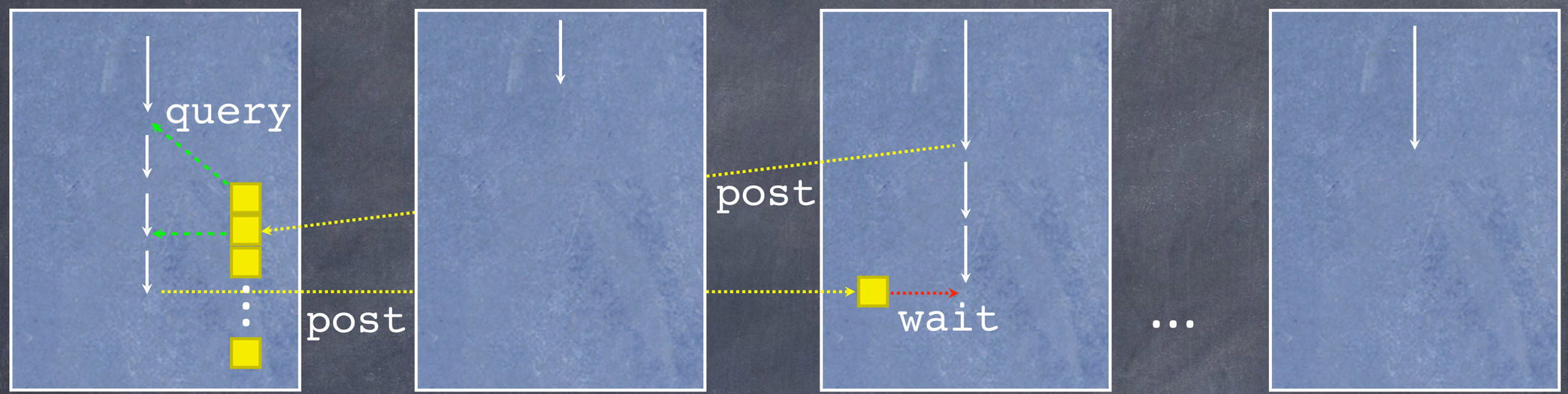
Events



greeting_ready(:)[1]

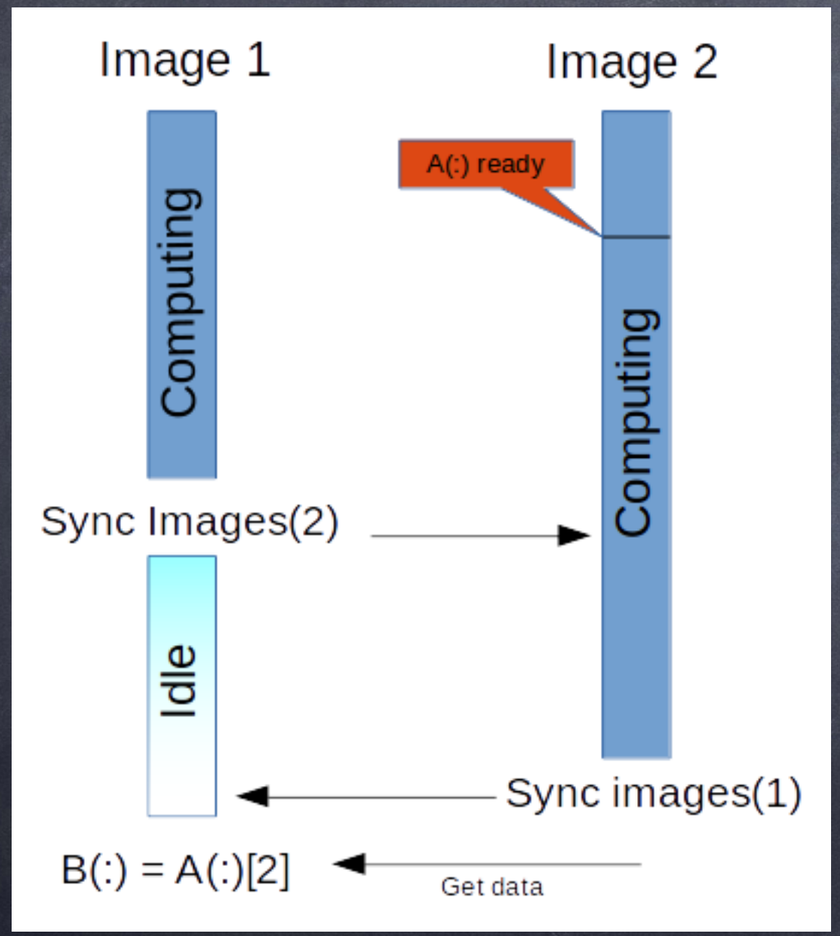
ok_to_overwrite[3]

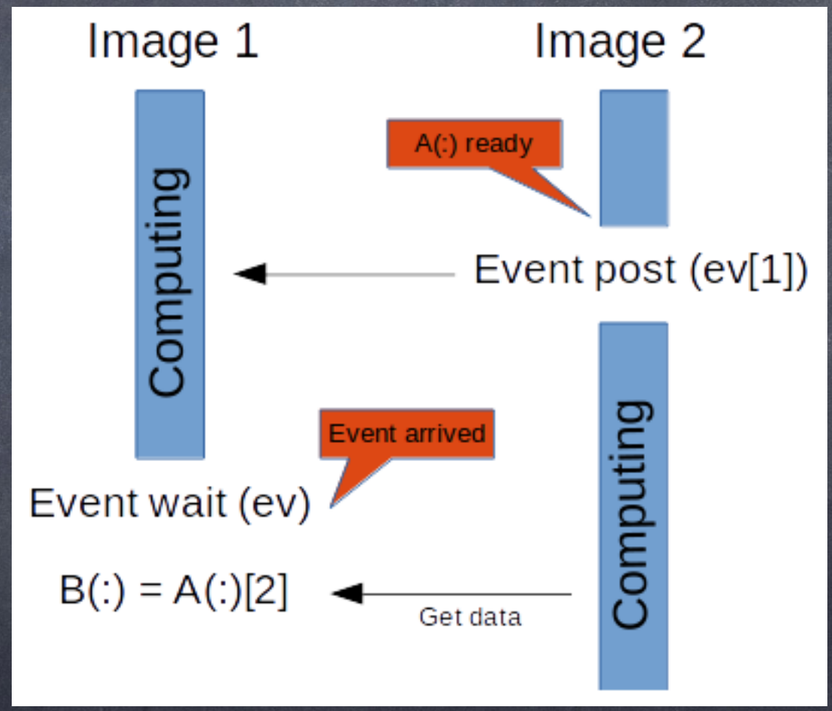
Events



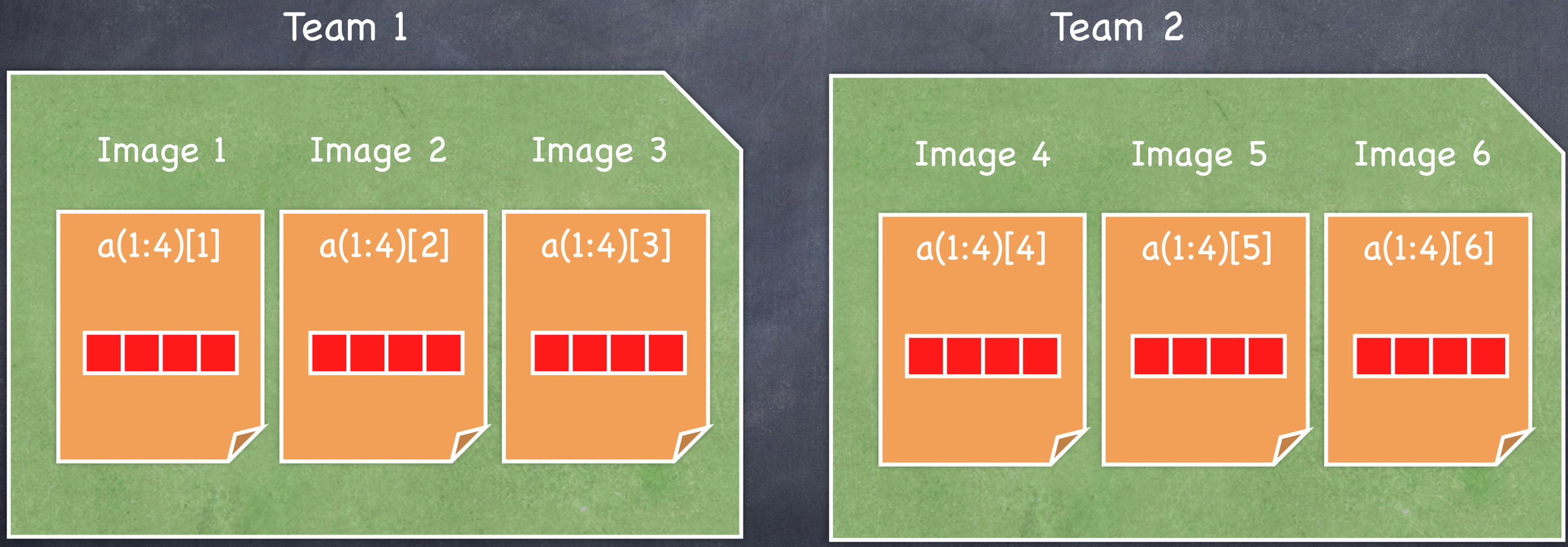
- Performance-Oriented Constrains
- The standard obligates the compiler to enforce the following constrains:
  - Query and wait must be local.
  - Post and wait are disallowed in `do concurrent` loops.
- Additional Performance-Oriented Tips
  - Wherever safety permits, query without waiting
  - Write a spin-query-work loop & construct logical masks describing the remaining work.

Overlap Computation/Communication





Teams



Team: set of images that can readily execute independently of other images

TS 18508 Additional Parallel Features in Fortran WG5/N2027 , August 22, 2014

Collective Subroutines

- Each non-failed image of the current team must invoke the collective.
- Parallel calculation/communication and result is placed on one or all images.
- Optional arguments: STAT, ERRMSG, RESULT_IMAGE, SOURCE_IMAGE.
- All collectives have an INTENT(INOUT) argument A holding the input data on entry and may hold the result on return (depending on RESULT_IMAGE).
- No implicit synchronization at beginning/end: allows overlap with other actions.
- No image's data is accessed before that image invokes the collective subroutine.

TS 18508 Additional Parallel Features in Fortran WG5/N2027 , August 22, 2014

Extensible Set of Collectives

- CO_BROADCAST (A, SOURCE_IMAGE [, STAT, ERRMSG])
- CO_MAX (A [, RESULT_IMAGE, STAT, ERRMSG])
- CO_MIN (A [, RESULT_IMAGE, STAT, ERRMSG])
- CO_SUM (A [, RESULT_IMAGE, STAT, ERRMSG])
- CO_REDUCE (A, OPERATOR [, RESULT_IMAGE, STAT, ERRMSG])

TS 18508 Additional Parallel Features in Fortran WG5/N2027 , August 22, 2014

Collective Sum: Definition

CO_SUM (A [, RESULT IMAGE, **STAT**, **ERRMSG**])

Description Sum elements on the current team of images.

Arguments

- **A** shall be of numeric type. It shall have the same type and type parameters on all images of the current team. It is an INTENT(INOUT) argument. If it is a scalar, the computed value is equal to a processor-dependent and image-dependent approximation to the sum of the values of A on all images of the current team. If it is an array it shall have the same shape on all images of the current team and each element of the computed value is equal to a processor-dependent and image-dependent approximation to the sum of all the corresponding elements of A on the images of the current team.

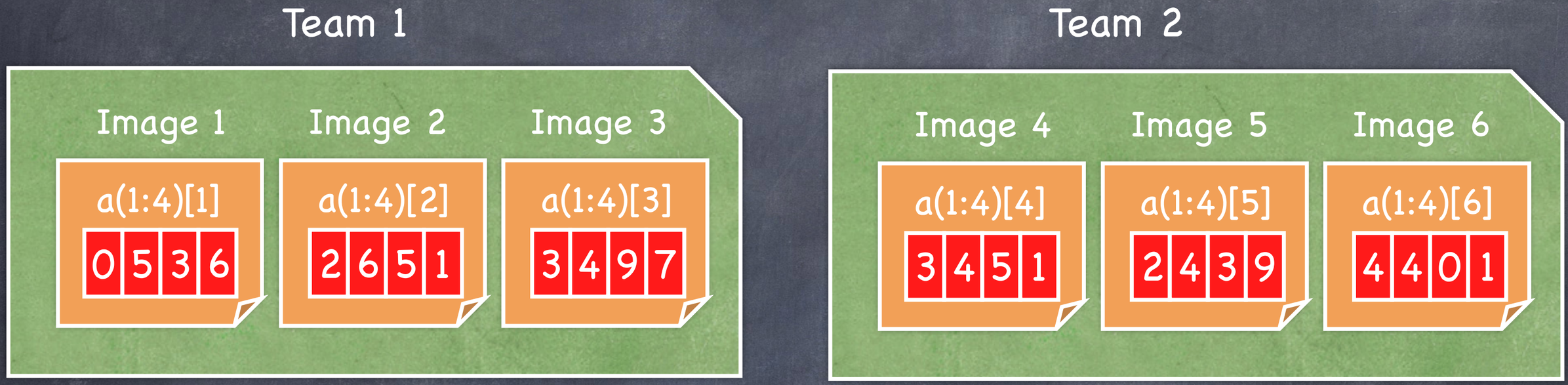
- **RESULT IMAGE** (optional) shall be a scalar of type integer. It is an INTENT(IN) argument. If it is present, it shall be present on all images of the current team, have the same value on all images of the current team, and that value shall be the image index of an image of the current team.

- **STAT** (optional) shall be a scalar of type default integer. It is an INTENT(OUT) argument.

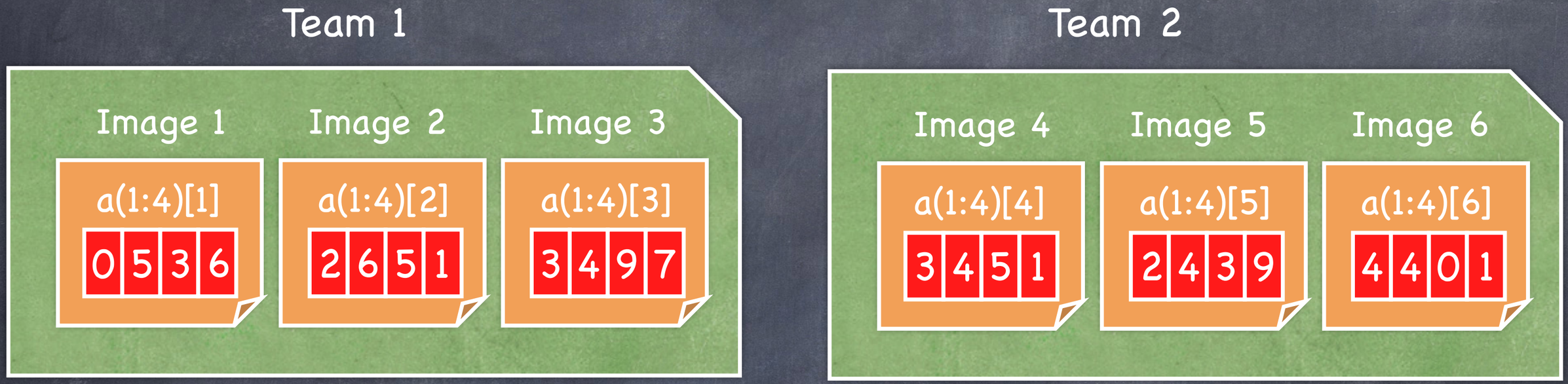
- **ERRMSG** (optional) shall be a scalar of type default character. It is an INTENT(INOUT) argument.

TS 18508 Additional Parallel Features in Fortran WG5/N2027 , August 22, 2014

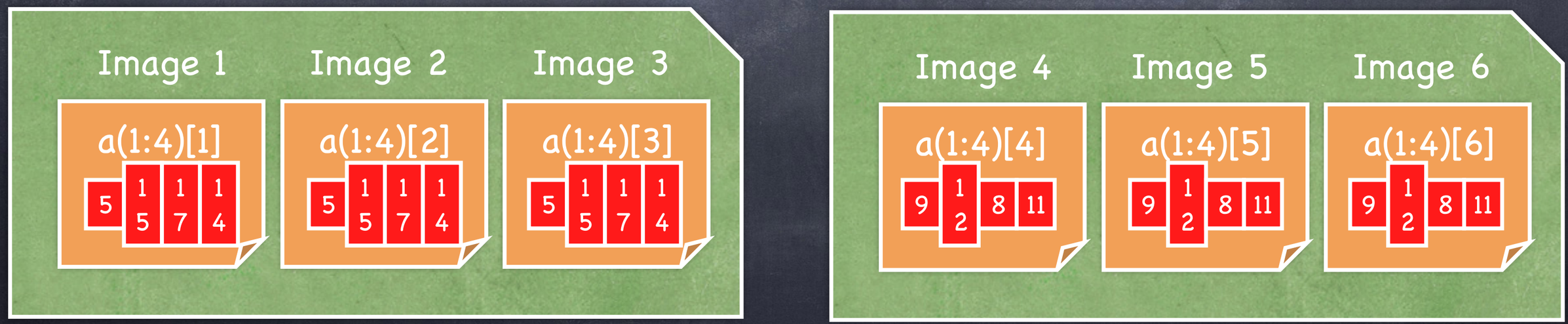
Collective Sum: Example



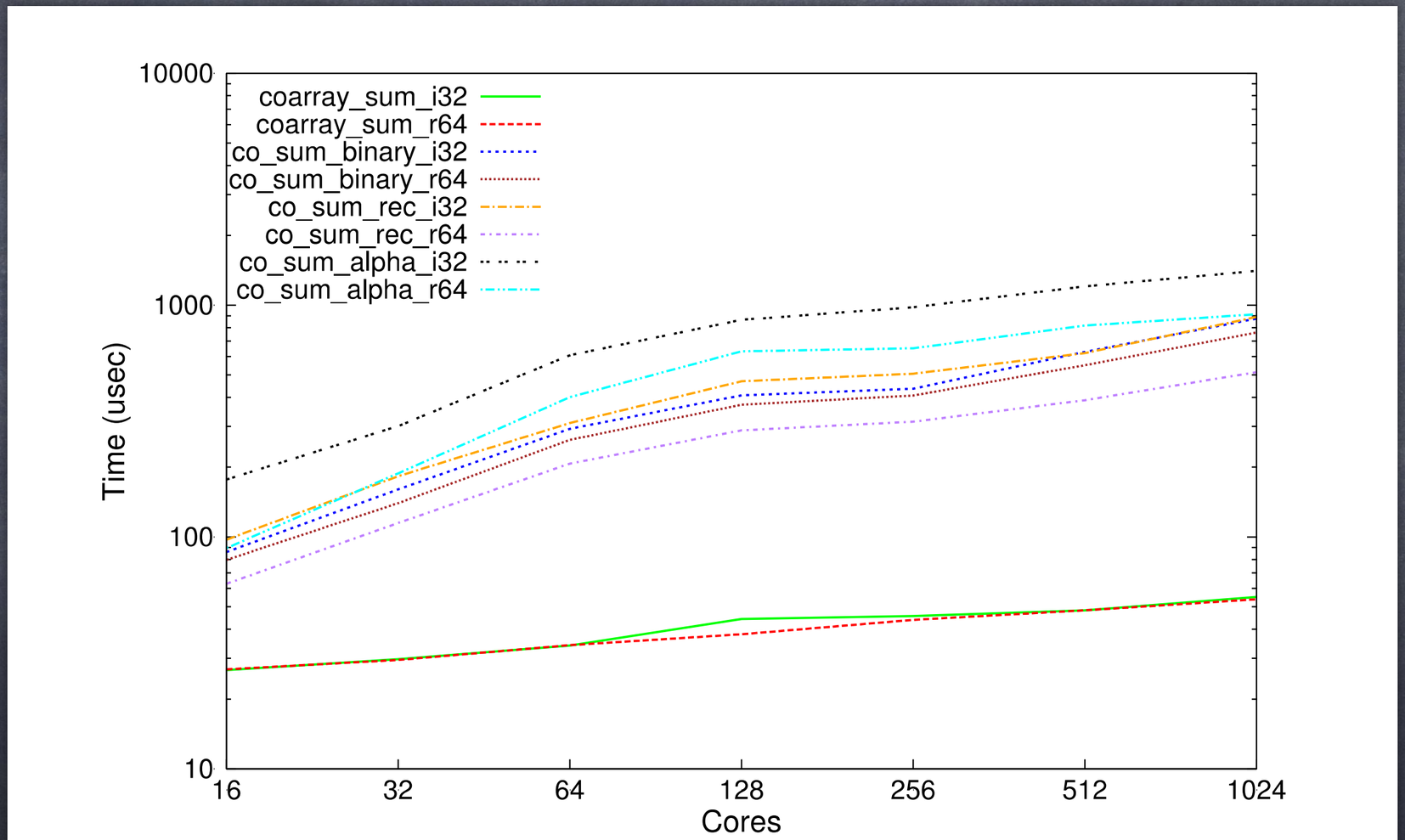
Collective Sum: Example




```
call co_sum(a)
```

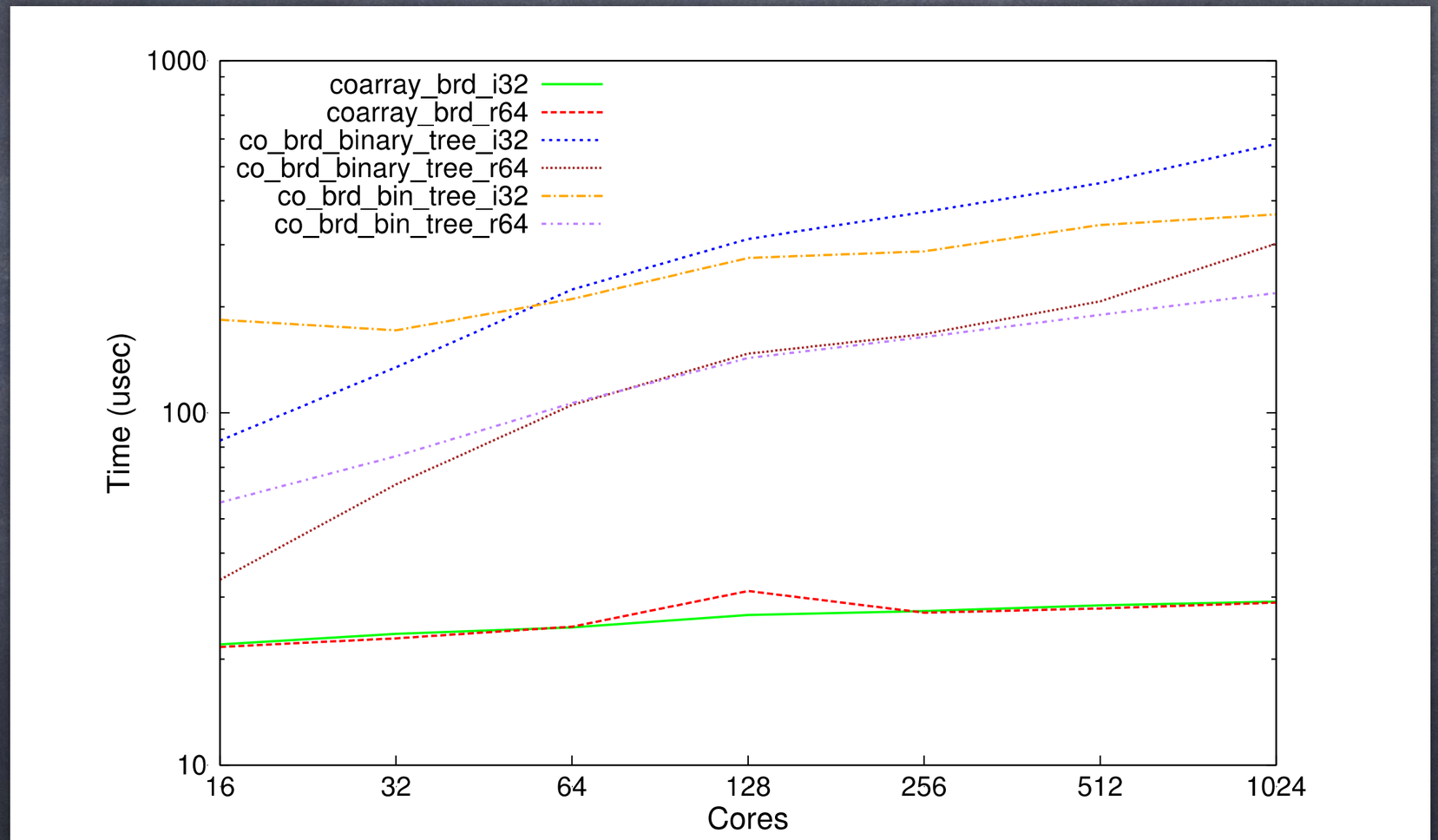


Performance of Fortran 2008 User-Defined Collective vs Fortran 2015 co_sum Intrinsic



Execution time for user-defined collective sum reductions and Fortran 2015 intrinsic co_sum
Hopper at NERSC: Cray XE6, peak performance of 1.28 Petaflops/sec, 153,216 compute cores,
212 Terabytes of memory, and 2 Petabytes of disk.

Performance of Fortran 2008 User-Defined Collective vs Fortran 2015 co_broadcast Intrinsic



Execution time for user-defined collective broadcast reductions and Fortran 2015 intrinsic co_sum
Hopper at NERSC: Cray XE6, peak performance of 1.28 Petaflops/sec, 153,216 compute cores,
212 Terabytes of memory, and 2 Petabytes of disk.

Fault Tolerance

Why do we need fault tolerance?

Exascale machines composed by millions of nodes

Each node composed by thousand of cores

MTBF - one node	1 year	10 years	120 years
MTBF - one million nodes	30 seconds	5 minutes	1 hour

Checkpointing/Restart cannot be effective with high failure rates

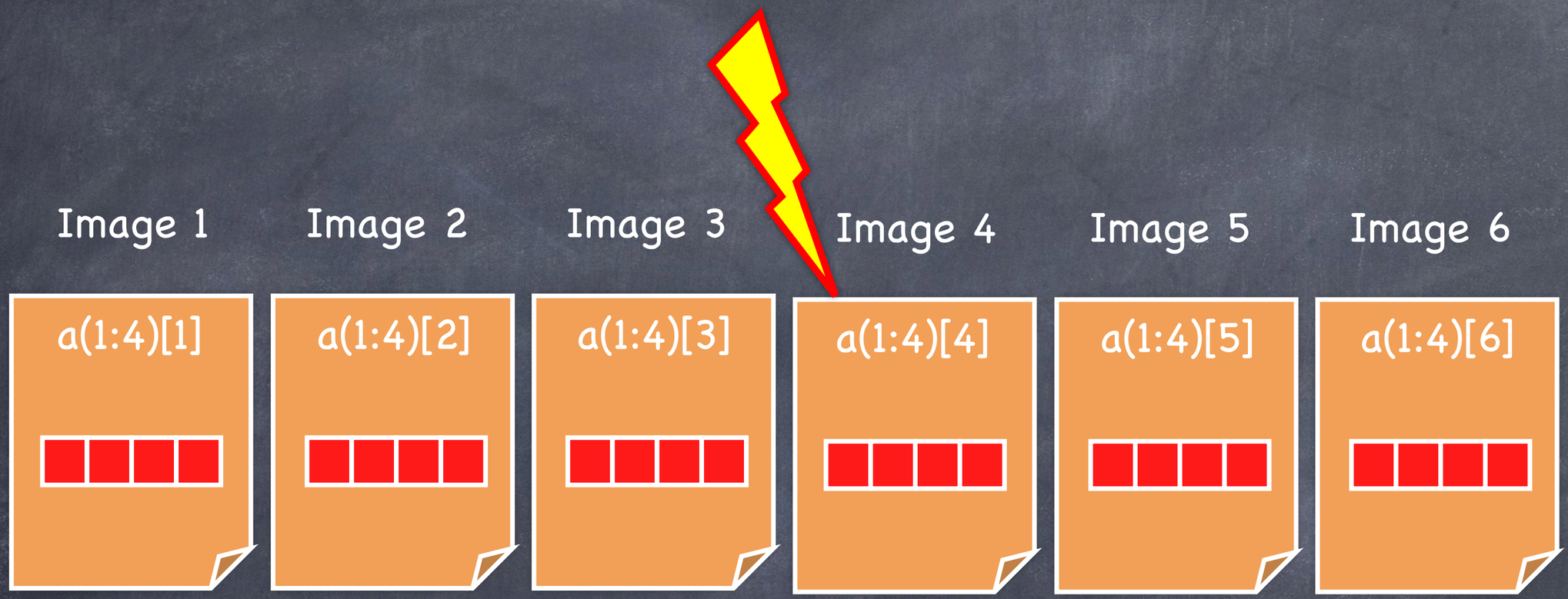
Failed Images

Failed Images provides support for failures detection

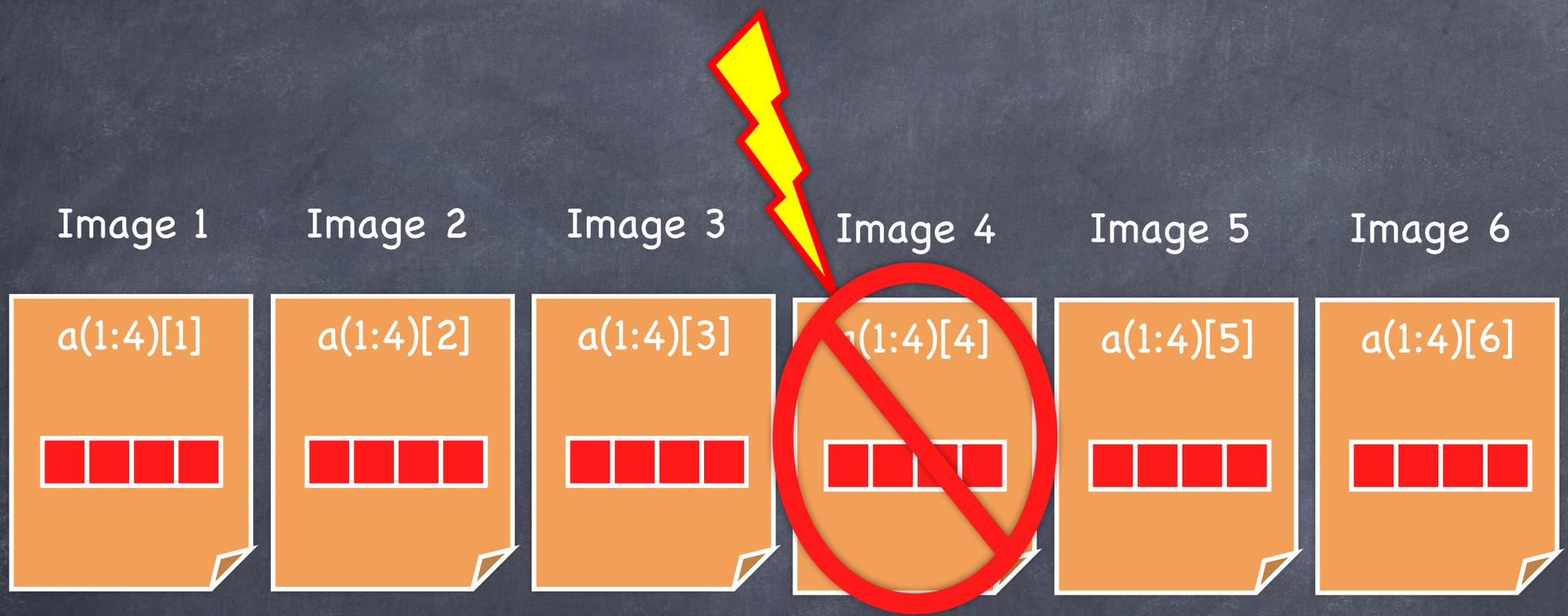
- FAIL IMAGE (simulates a failures)
- IMAGE_STATUS (checks the status of a specific image)
- FAILED_IMAGES (provides the list of failed images)
- Coarray operations may fail. STAT= attribute used to check correct behavior.

Failure Detection

Failure Detection



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()
```

Outline

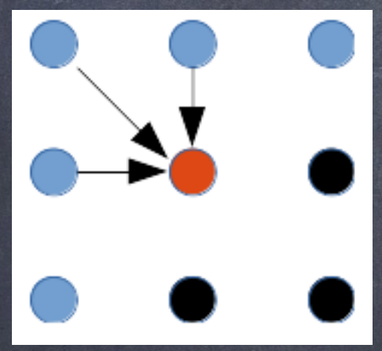
- Parallelism in Fortran 2008
  - SPMD
  - PGAS
- Exascale challenges
- Fortran 2015 targeting exascale challenges
  - Events
  - Teams
  - Collective subroutines
  - Failed-image detection
  - Richer set of atomic subroutines
- Scientific kernels

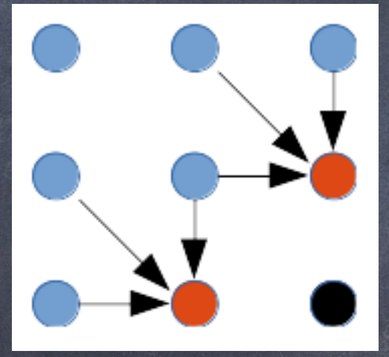
Scientific Kernels (PRK)

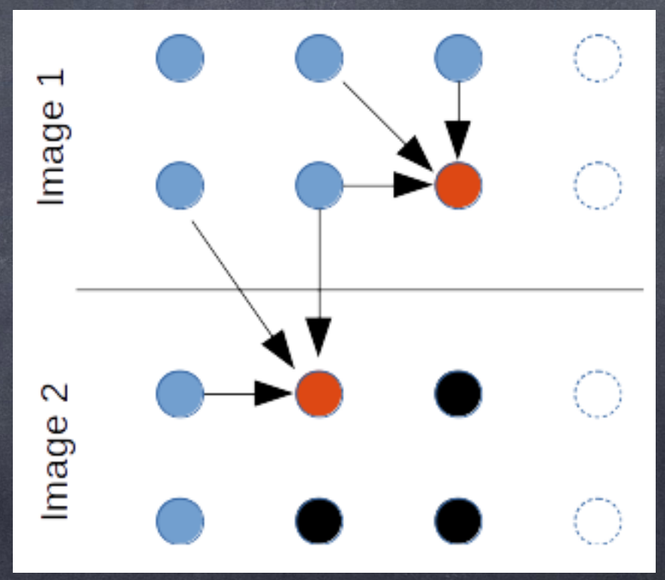
- Parallel Research Kernels: suite of representative scientific parallel codes.
  - Distributed Transpose (not examined)
  - Peer-to-peer synchronization
  - 2D Stencil (halo exchange)

Sync_p2p Kernel

Stencil with demanding data dependence resolved using sw pipeline techniques.

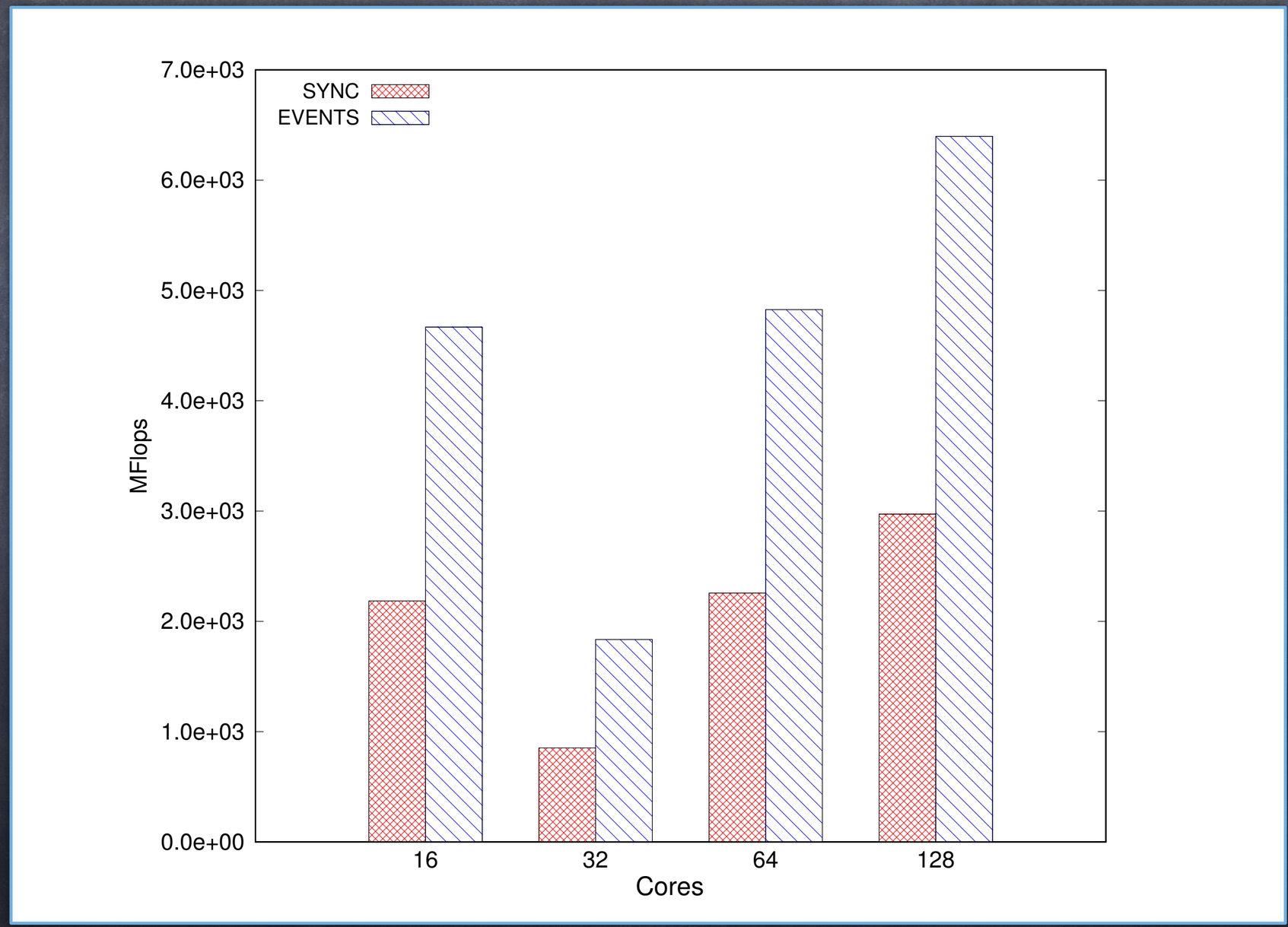






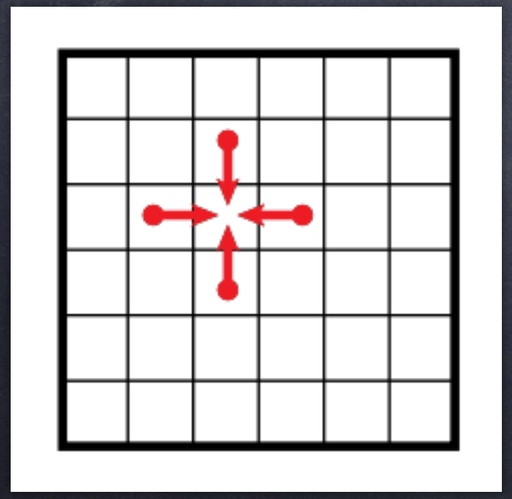
Events provide the right level of granularity for this case

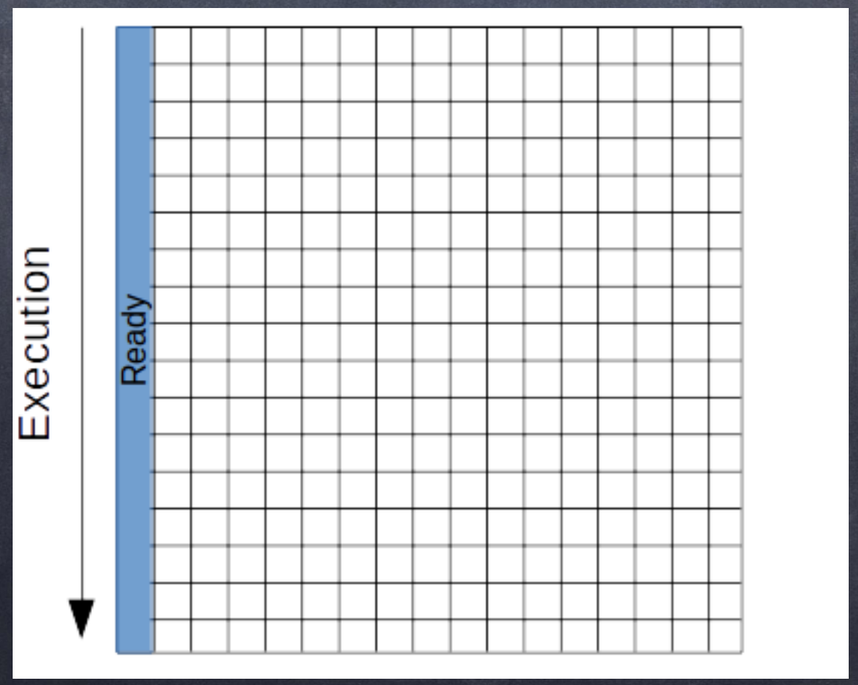
Sync_p2p on Stampede

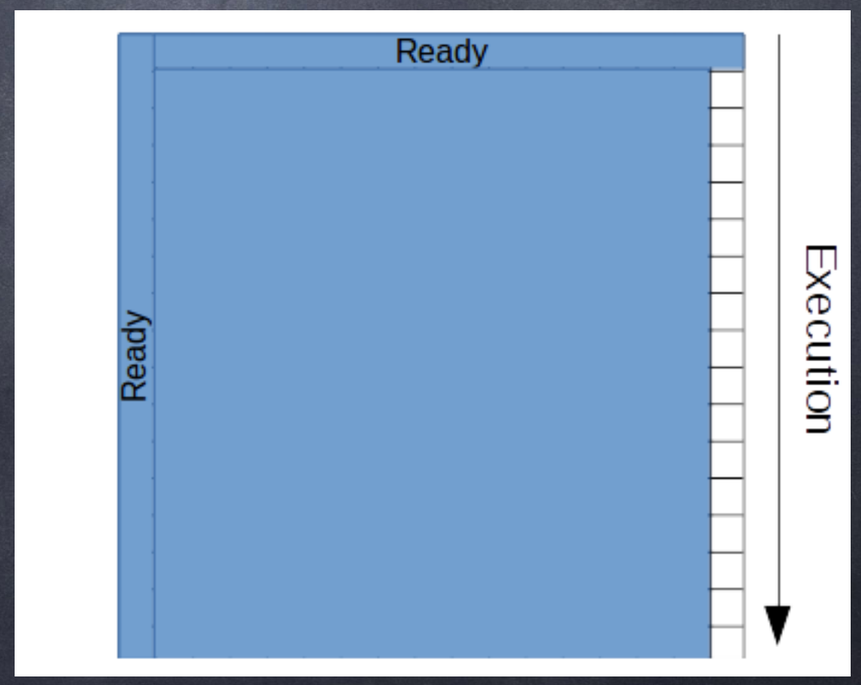


Stencil Kernel

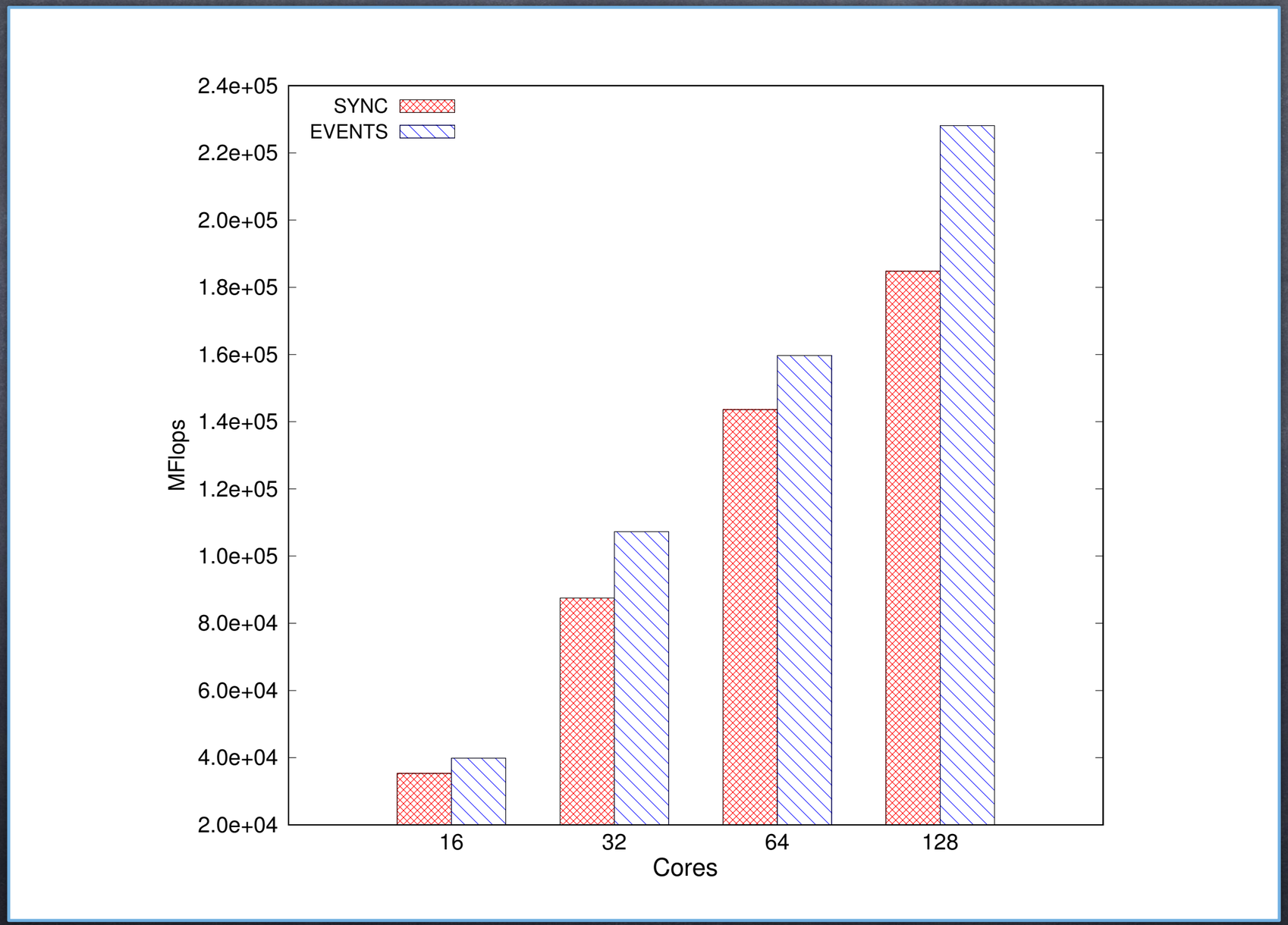
- 2D Stencil operator
- One of the most common communication patterns: halo exchange
- Events can be used on every side
- Sync images/Sync all imply computation and communication to be separate





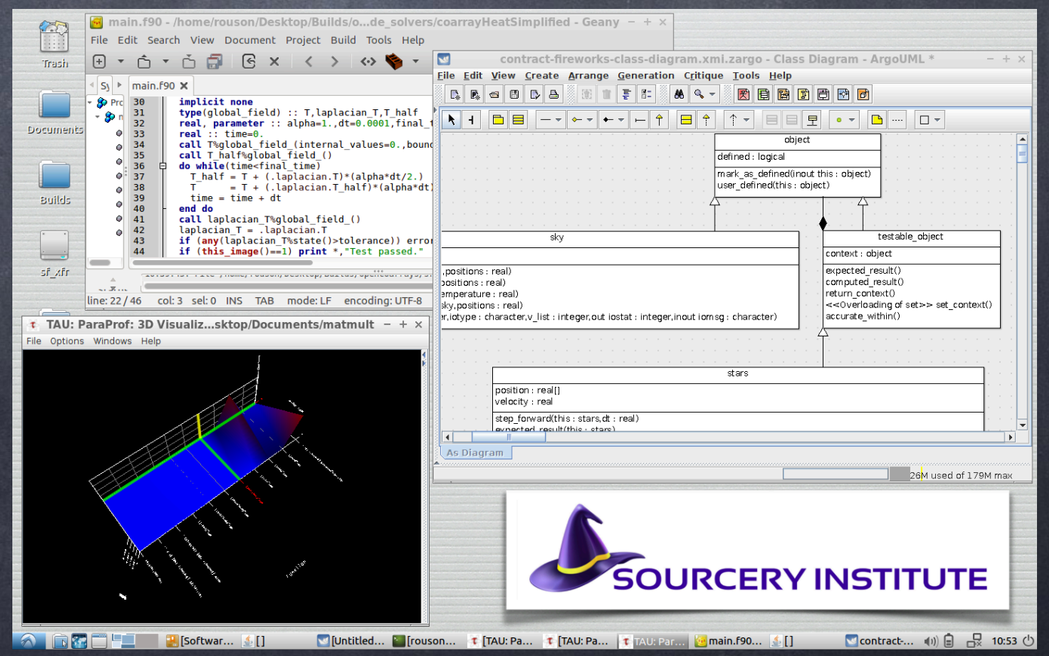


Stencil Kernel on Stampede



Compiler Support & Video Tutorials

- GCC 7 + OpenCoarrays support
  - Fortran 2008 parallel features.
  - Fortran 2015 events, collectives, atomics.
  - Unsupported: Teams, Failure detection (pending)
- The Sourcery Institute (SI) Linux virtual machine (VM) contains GCC 7, OpenCoarrays, and numerous useful tools for modern Fortran development:
  - Download free VM at sourceryinstitutute.org/store
  - SI VM is used in Stanford EARTH/CME 214 course
  - View video tutorials at sourceryinstitute.org/videos
  - Other compilers:
    - Cray supports Fortran 2008 + 2015 collectives
    - Intel supports Fortran 2008 parallel features



Related Research Literature

- Coarray Fortran (CAF)
  - See http://www.opencoarrays.org/publications
  - See http://www.sourceryinstitute.org/publications
- HPC Applications of CAF
  - Speedup of 33% relative to MPI-2 on 80K cores for European weather model: Mozdzynski, G., Hamrud, M., & Wedi, N. (2015). A Partitioned Global Address Space implementation of the European Centre for Medium Range Weather Forecasts Integrated Forecasting System. International Journal of High Performance Computing Applications, 1094342015576773.
  - Performance competitive with MPI-3 for several applications: Garain, S., Balsara, D. S., & Reid, J. (2015). Comparing Coarray Fortran (CAF) with MPI for several structured mesh PDE applications. Journal of Computational Physics.
  - Speedup of 50% relative to MPI-2 for plasma fusion code on 130,000 cores: Preissl, R., Wichmann, N., Long, B., Shalf, J., Ethier, S., & Koniges, A. (2011, November). Multithreaded global address space communication techniques for gyrokinetic fusion applications on ultra-scale platforms. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (p. 78). ACM.

Acknowledgements

- Tobias Burnus and Andre Vehreschild, GCC developers
- Salvatore Filippone, Cranfield University
- Zaak Beekman, Princeton University
- Valeria Cardellini, University of Rome "Tor Vergata"
- Dan Nagle, NCAR
- Kenneth Craft and Jeff Hammond, Intel