Portable, MPI-Interoperable Coarray Fortran

Chaoran Yang,[1] Wesley Bland,[2]
John Mellor-Crummey,[1] Pavan Balaji[2]

[1]Department of Computer Science
Rice University
Houston, TX

[2]Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL

Partitioned Global Address Space Languages

- Why PGAS Languages today?
    - need for shared data outstrips node-level memory
    - access data with shared-memory abstractions within and across nodes
- Example PGAS Languages
    - Unified Parallel C (C)
        - http://upc.wikinet.org
    - Titanium (Java)
        - http://titanium.cs.berkeley.edu
    - Coarray Fortran (Fortran)
        - http://caf.rice.edu
- Related efforts:
    - X10 (IBM)
        - http://x10-lang.org
    - Chapel (Cray)
        - http://chapel.cray.com
    - Fortress (Oracle)
        - http://projectfortress.java.net

Partitioned Global Address Space Languages

vs. MPI

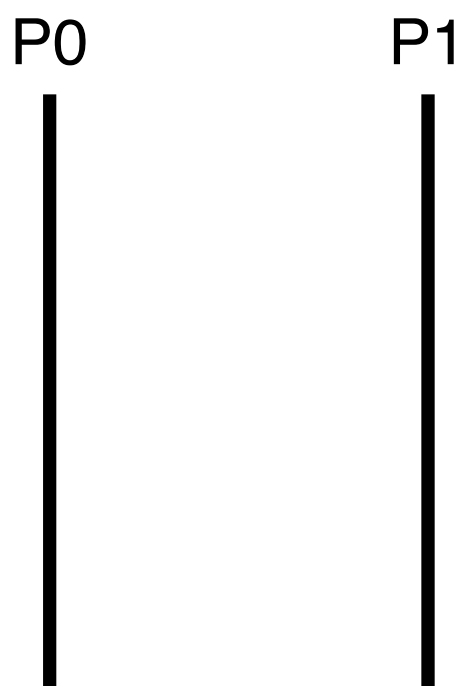
- Why PGAS Languages today?
  - need for shared data outstrips node-level memory
  - access data with shared-memory abstractions within and across nodes
- Example PGAS Languages
  - Unified Parallel C (C)
    - http://upc.wikinet.org
  - Titanium (Java)
    - http://titanium.cs.berkeley.edu
  - Coarray Fortran (Fortran)
    - http://caf.rice.edu
- Related efforts:
  - X10 (IBM)
    - http://x10-lang.org
  - Chapel (Cray)
    - http://chapel.cray.com
  - Fortress (Oracle)
    - http://projectfortress.java.net

MPI-interoperability

- Hard to adopt new programming models in existing applications **incrementally**
- Interoperable problems in new programming models (examples later)
  - Error-prone
  - Duplicate runtime resources
- Benefits of interoperable programming models
  - Leverage high-level libraries that are built with MPI
  - Hybrid programming models combine the strength of different models

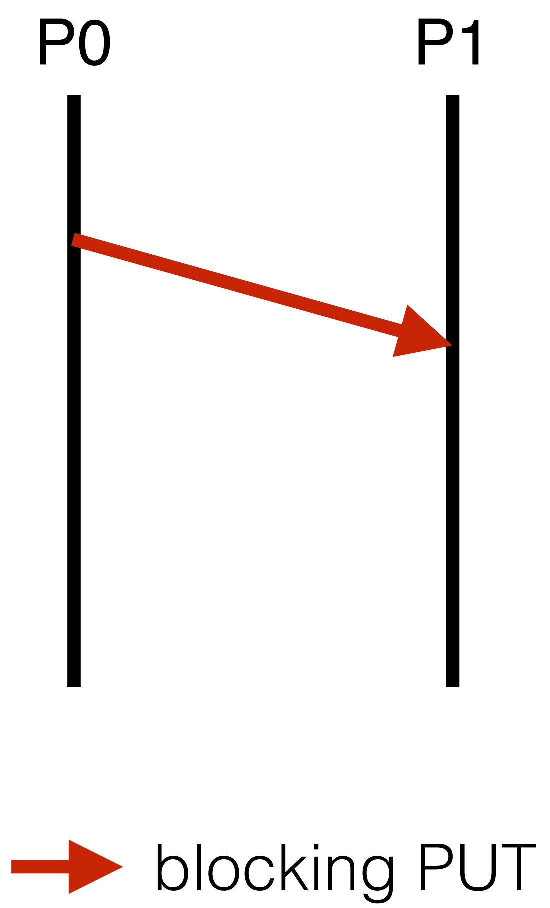
Using multiple runtimes is error-prone

```
PROGRAM MAY_DEADLOCK
   USE MPI
   CALL MPI_INIT(IERR)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, MY_RANK, IERR)
   IF (MYRANK .EQ. 0) A(:)[1] = A(:)
   CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
   CALL MPI_FINALIZE(IERR)
END PROGRAM
```



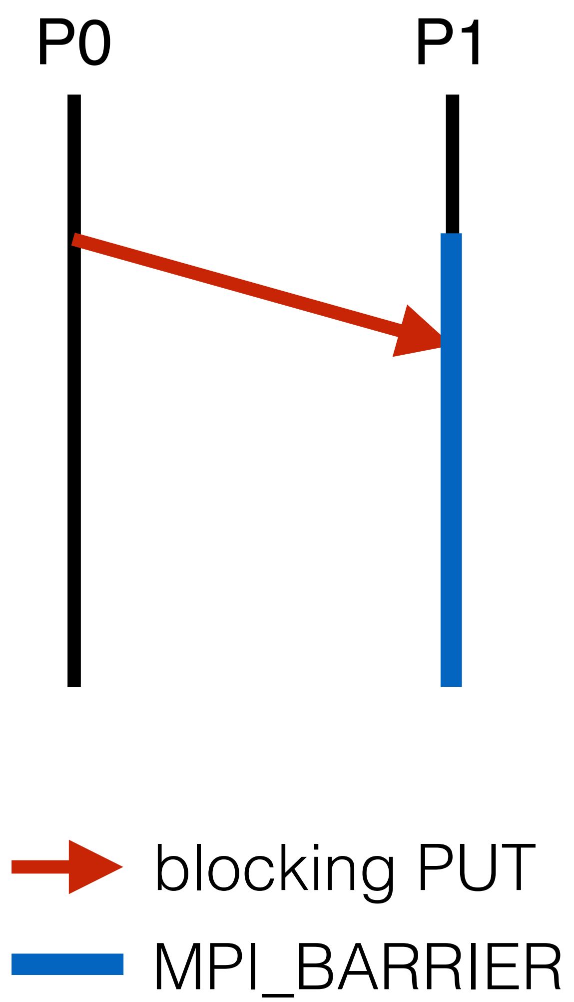
Using multiple runtimes is error-prone

```
PROGRAM MAY_DEADLOCK
   USE MPI
   CALL MPI_INIT(IERR)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, MY_RANK, IERR)
   IF (MYRANK .EQ. 0) A(:)[1] = A(:)
   CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
   CALL MPI_FINALIZE(IERR)
END PROGRAM
```



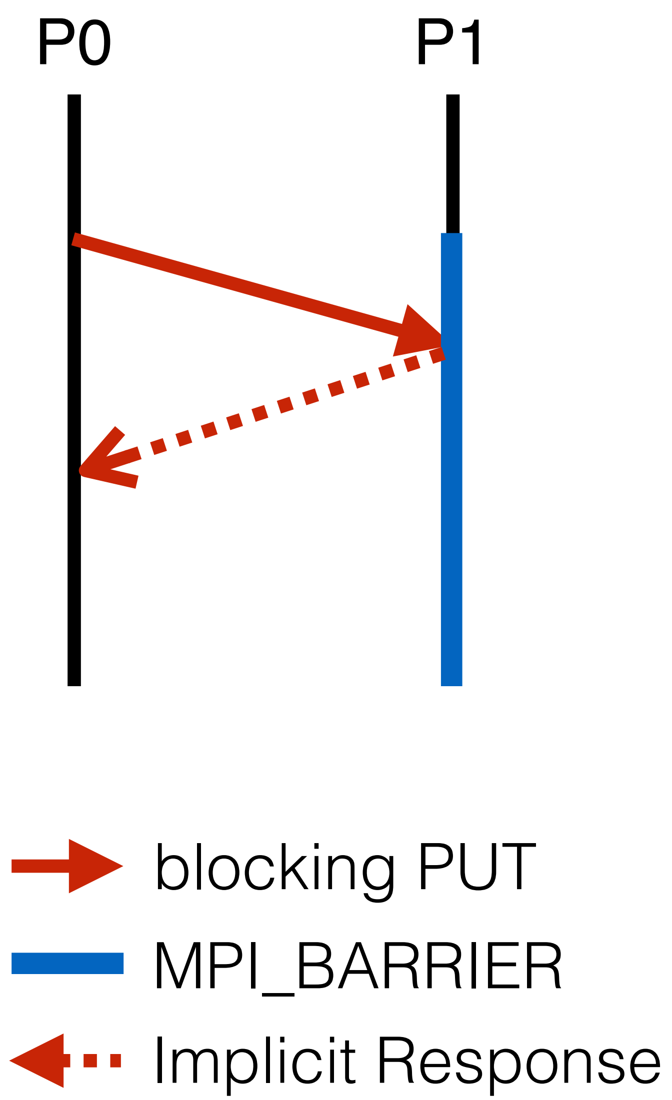
Using multiple runtimes is error-prone

```
PROGRAM MAY_DEADLOCK
   USE MPI
   CALL MPI_INIT(IERR)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, MY_RANK, IERR)
   IF (MYRANK .EQ. 0) A(:)[1] = A(:)
   CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
   CALL MPI_FINALIZE(IERR)
END PROGRAM
```

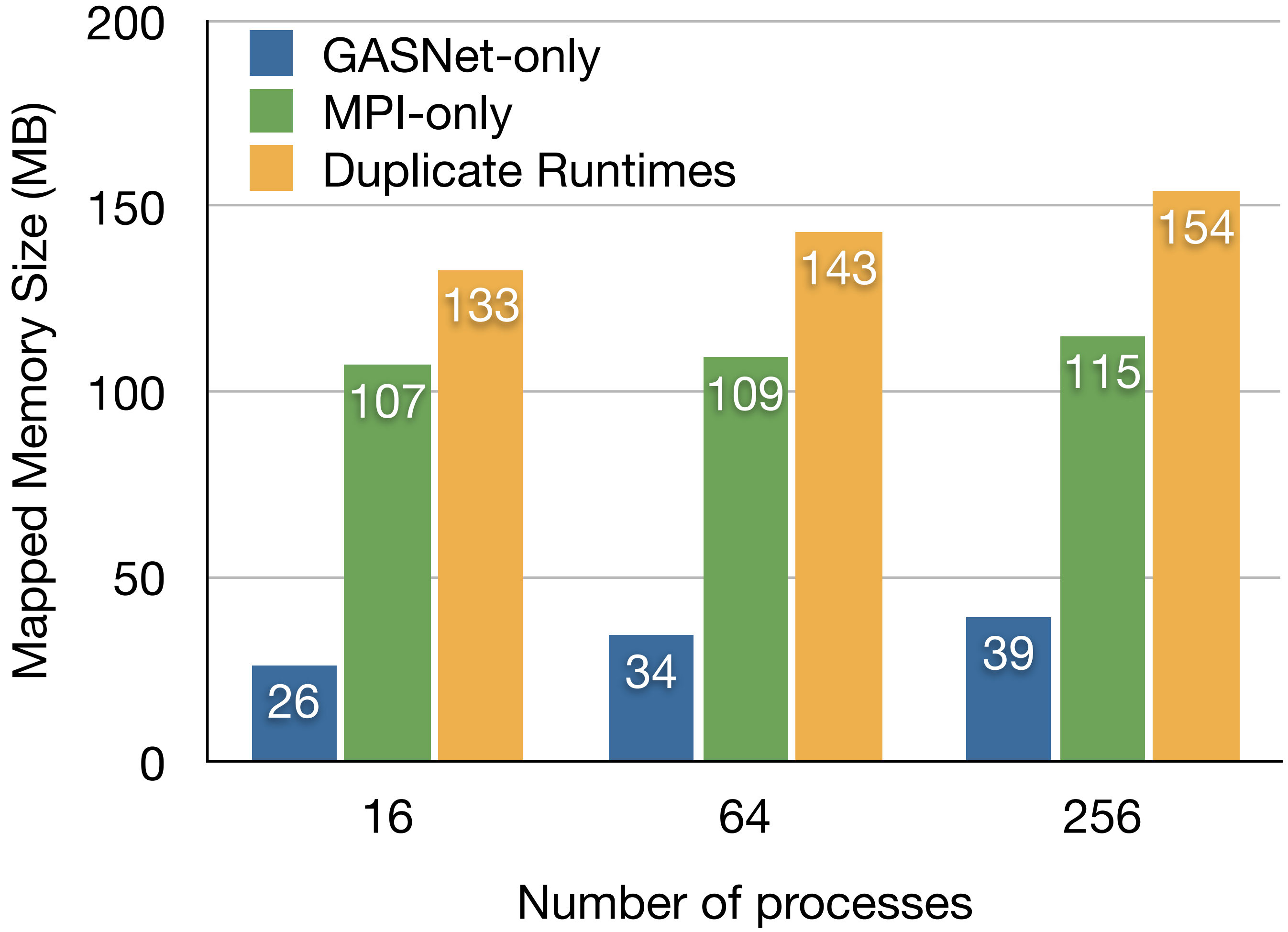


Using multiple runtimes is error-prone

```
PROGRAM MAY_DEADLOCK
   USE MPI
   CALL MPI_INIT(IERR)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, MY_RANK, IERR)
   IF (MYRANK .EQ. 0) A(:)[1] = A(:)
   CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
   CALL MPI_FINALIZE(IERR)
END PROGRAM
```



Using multiple runtimes duplicates resources

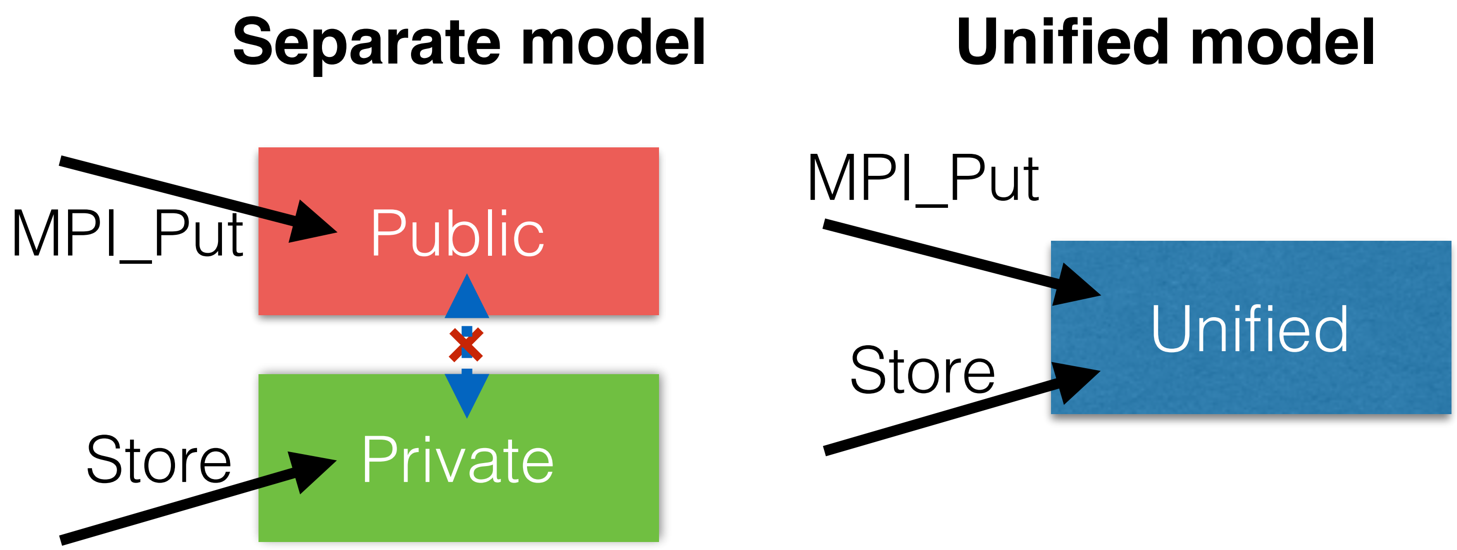


- Memory usage is measured right after initialization
- **Memory usage per process increases as the number of processes increases**
- At larger scale, excessive memory use of duplicate runtimes will hurt scalability

How do we solve the problem?

Build PGAS runtime systems with MPI

- Previously MPI was considered insufficient for this goal
  - MPI-2 RMA is portable but too strict
- MPI-3 Remote Memory Access (RMA)



Build PGAS runtimes with MPI

- Does it provide full interoperability?
- Does it degrade performance?

Coarray Fortran (CAF)

- What is Coarray Fortran?
  - added to the Fortran 2008 Standard
  - a PGAS Language, SPMD Model
- What is a coarray?
  - extends array syntax with **codimensions,** e.g. **REAL :: X(10,10)[*]**
- How to access a coarray?
  - Reference with [] mean data on specified image, e.g. **X(1,:) = X(1,:)[p]**
  - May be allocatable, structure components, dummy or actual arguments

Coarray Fortran 2.0 (CAF 2.0)

“A rich extension to Coarray Fortran developed at Rice University”

- Teams (like MPI communicator) and collectives
- Asynchronous operations
  - asynchronous copy, asynchronous collectives, and function shipping
- Synchronization constructs
  - events, cofence, and finish

More details on CAF 2.0: http://caf.rice.edu and http://chaoran.me

Coarray and MPI-3 RMA

"standard CAF features"

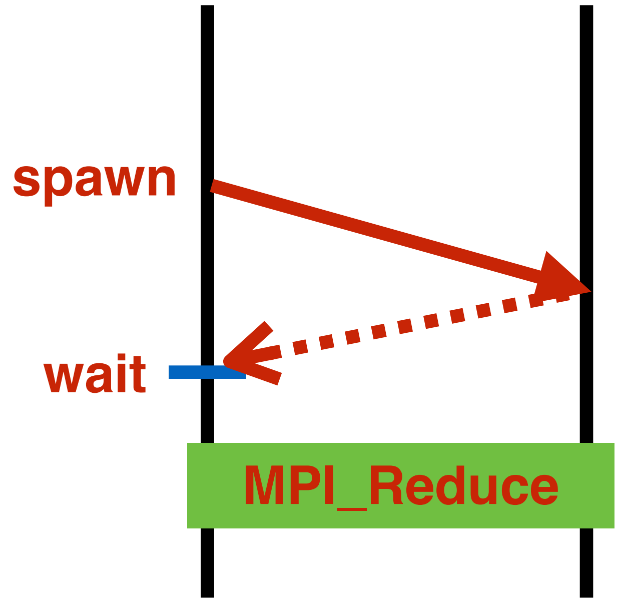
- Initialization
  - **MPI_WIN_ALLOCATE**, then **MPI_WIN_LOCK_ALL**
- Remote Read & Write
  - **MPI_RPUT** & **MPI_RGET**
- Synchronization
  - **MPI_WIN_SYNC** & **MPI_WIN_FLUSH (_ALL)**

Blue routine names are MPI-3 additions

Active Messages

"High performance low-level asynchronous remote procedure calls"

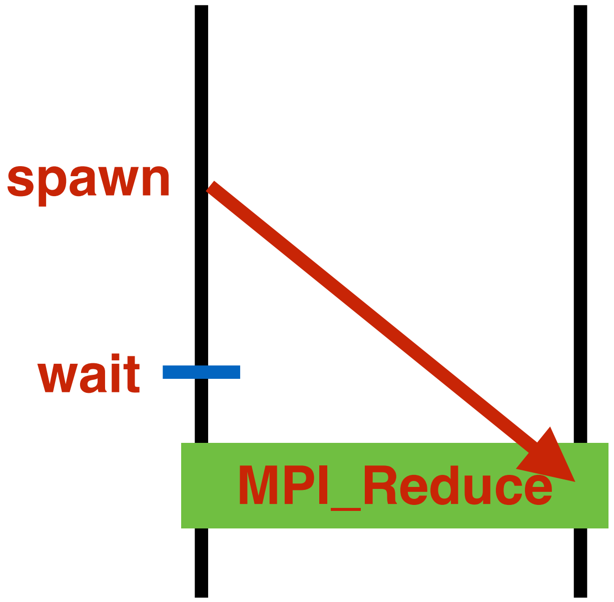
- Many CAF 2.0 features are built on top of AM
- Build AM on top of MPI's send and receive routines
    - hurt performance - cannot overlap communication with AM handlers
    - hurt interoperability - could cause deadlock



Active Messages

"High performance low-level asynchronous remote procedure calls"

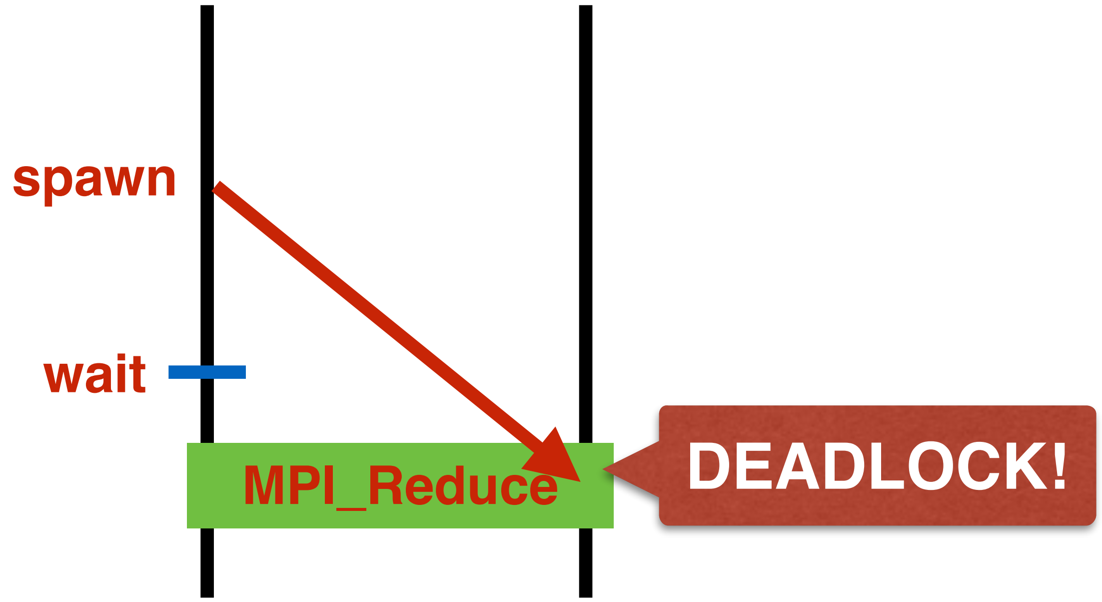
- Many CAF 2.0 features are built on top of AM
- Build AM on top of MPI's send and receive routines
    - hurt performance - cannot overlap communication with AM handlers
    - hurt interoperability - could cause deadlock



Active Messages

"High performance low-level asynchronous remote procedure calls"

- Many CAF 2.0 features are built on top of AM
- Build AM on top of MPI's send and receive routines
  - hurt performance - cannot overlap communication with AM handlers
  - hurt interoperability - could cause deadlock

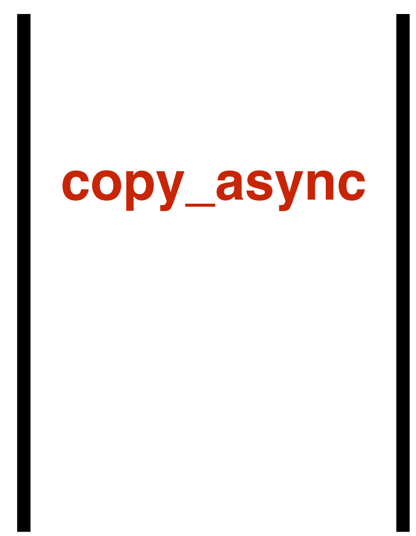


CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**

CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**

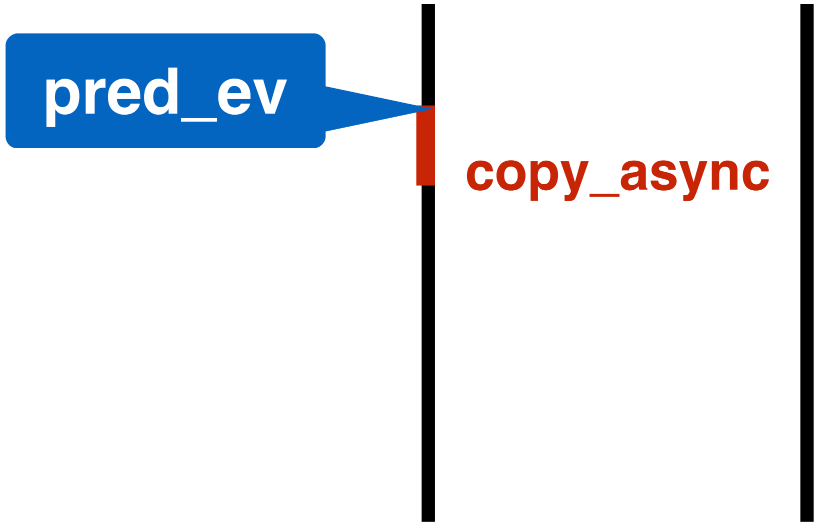


CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**

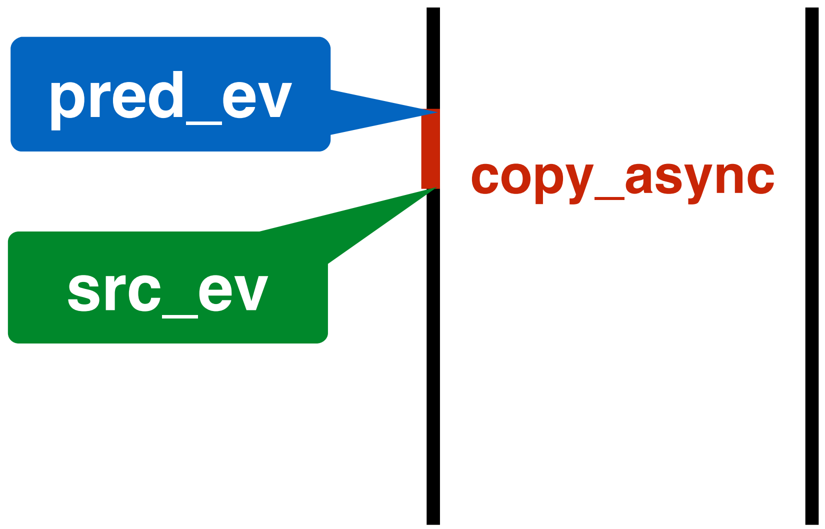
CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



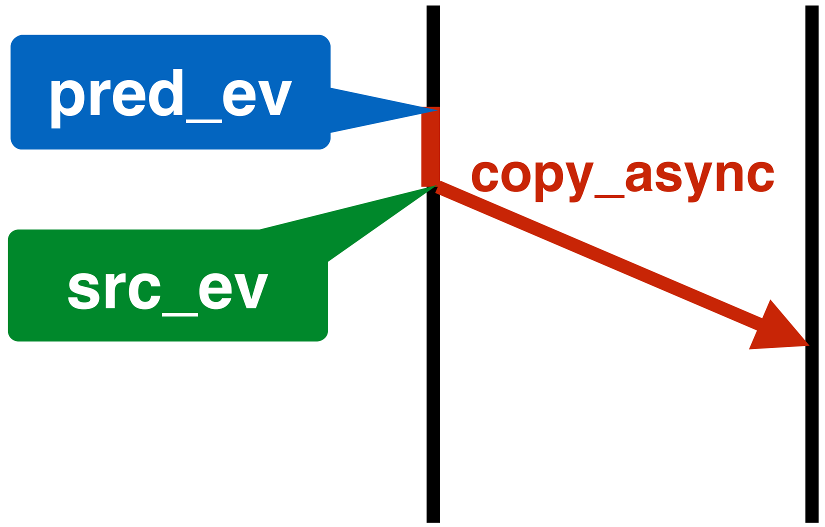
CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



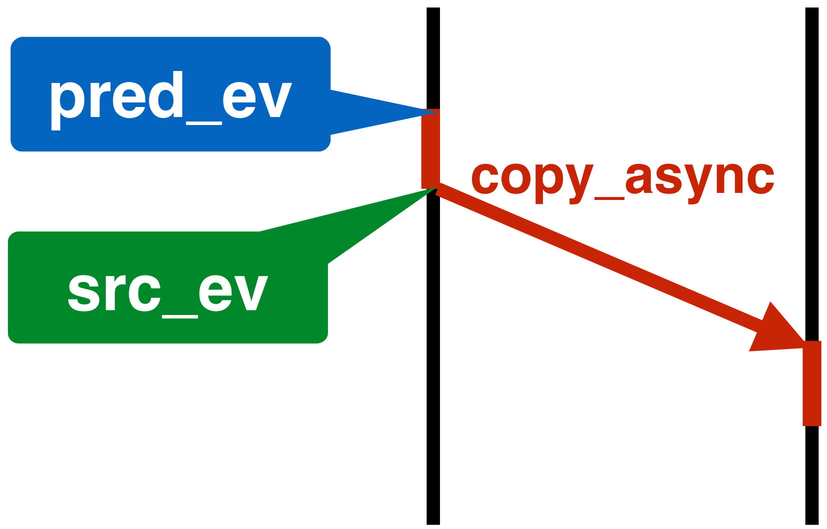
CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



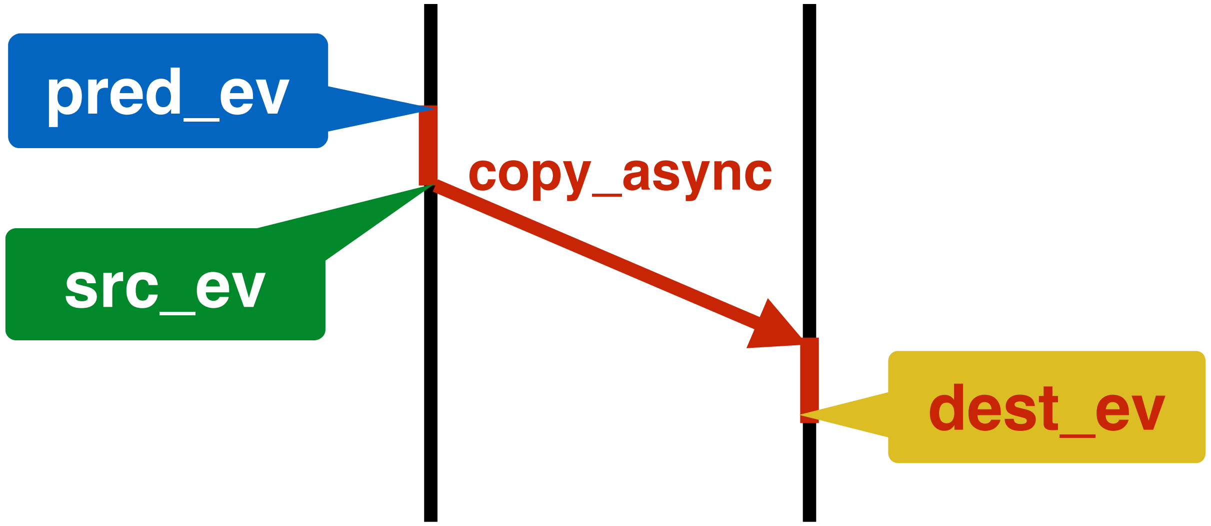
CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



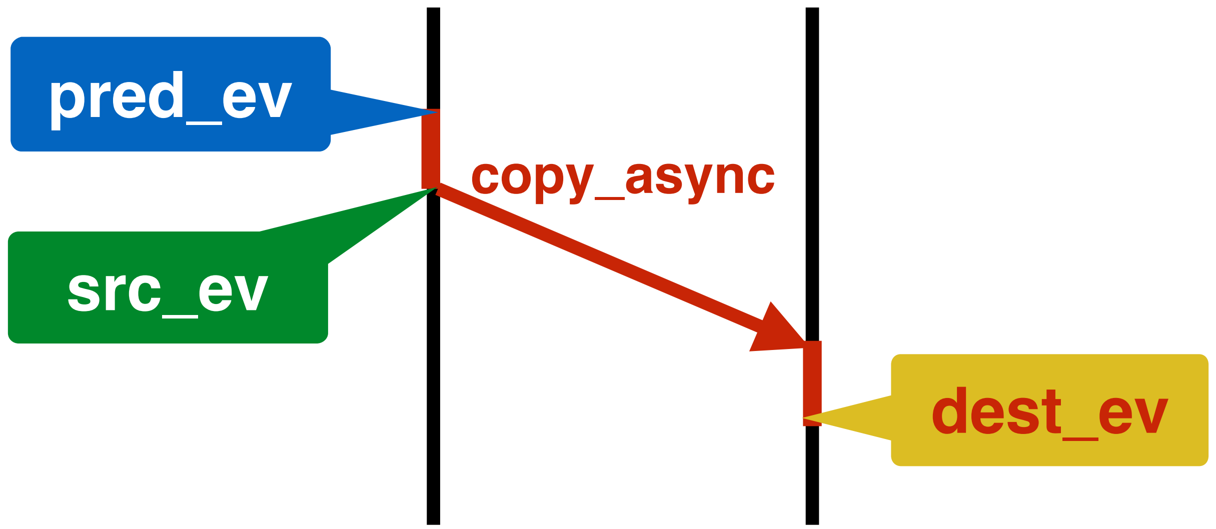
CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



CAF 2.0 Asynchronous Operations

- **copy_async(dest, src, dest_ev, src_ev, pred_ev)**



- Map **copy_async** to **MPI_RPUT** (or **MPI_RGET**)
  - when **dest_ev** should be notified? **MPI_WIN_FLUSH** is not useful
- Map **copy_async** to **Active Message**
  - MPI does not have AM support

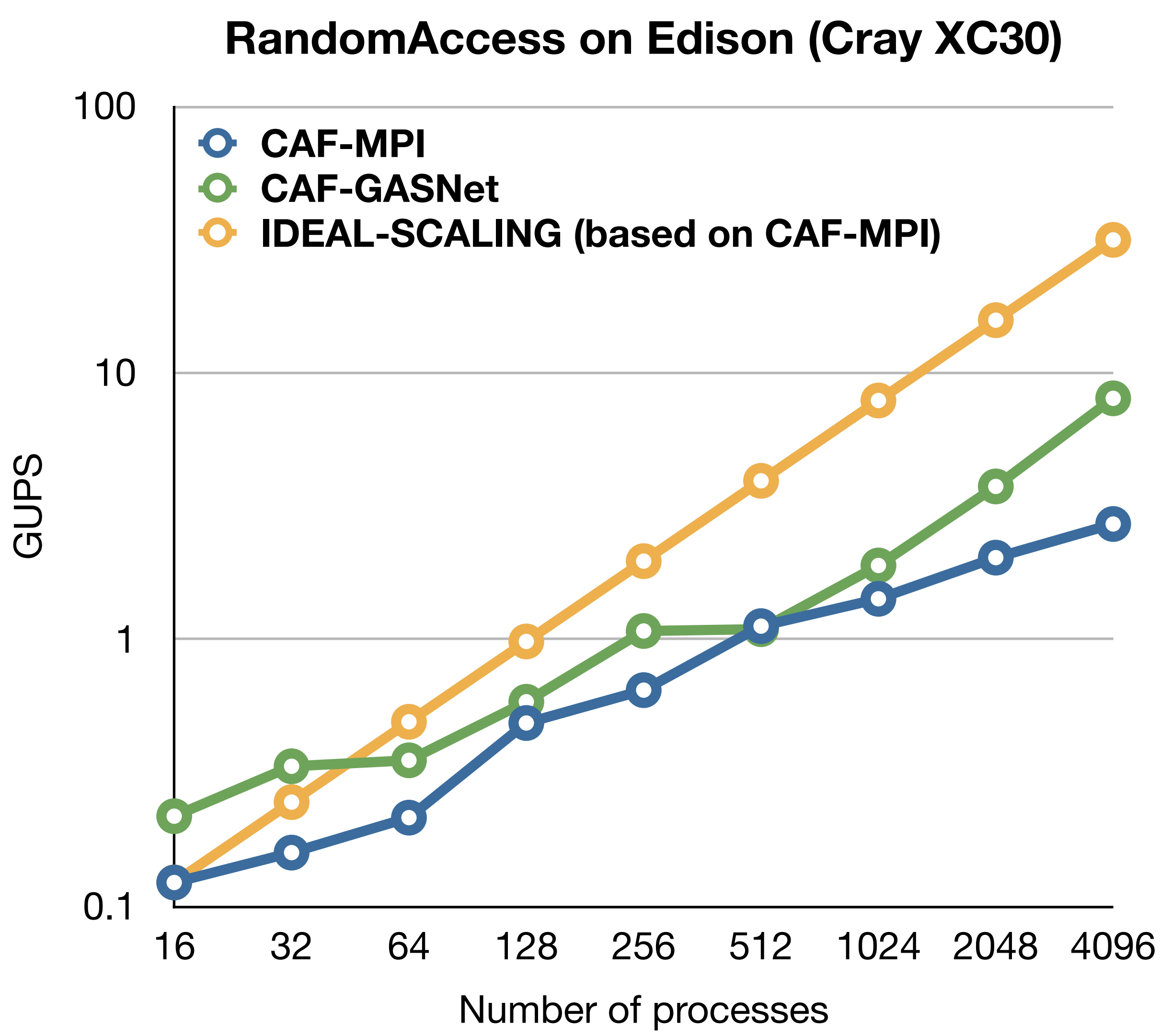
Evaluation

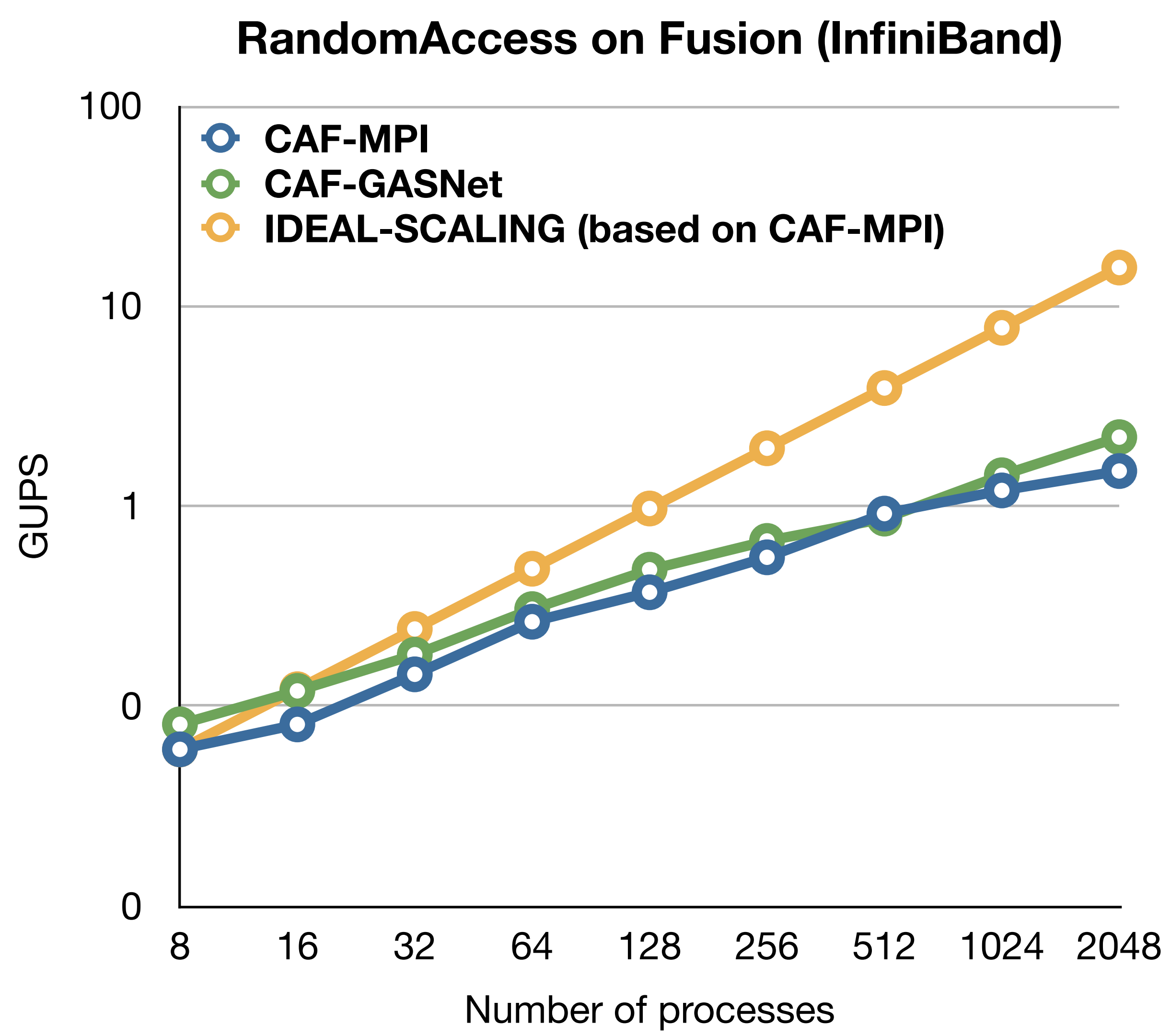
- 2 machines
  - Cluster (InfiniBand) and Cray XC30
- 3 benchmarks and 1 mini-app
  - **RandomAccess**, **FFT**, **HPL**, and **CGPOP**
- 2 implementations
  - **CAF-MPI** and **CAF-GASNet**

System	Nodes	Cores / Node	Memory / Node	Interconnect	MPI Version
Cluster (Fusion)	320	2x4	32GB	InfiniBand QDR	MVAPICH2-1.9
Cray XC30 (Edison)	5,200	2x12	64GB	Cray Aries	CRAY MPI-6.0.2

RandomAccess

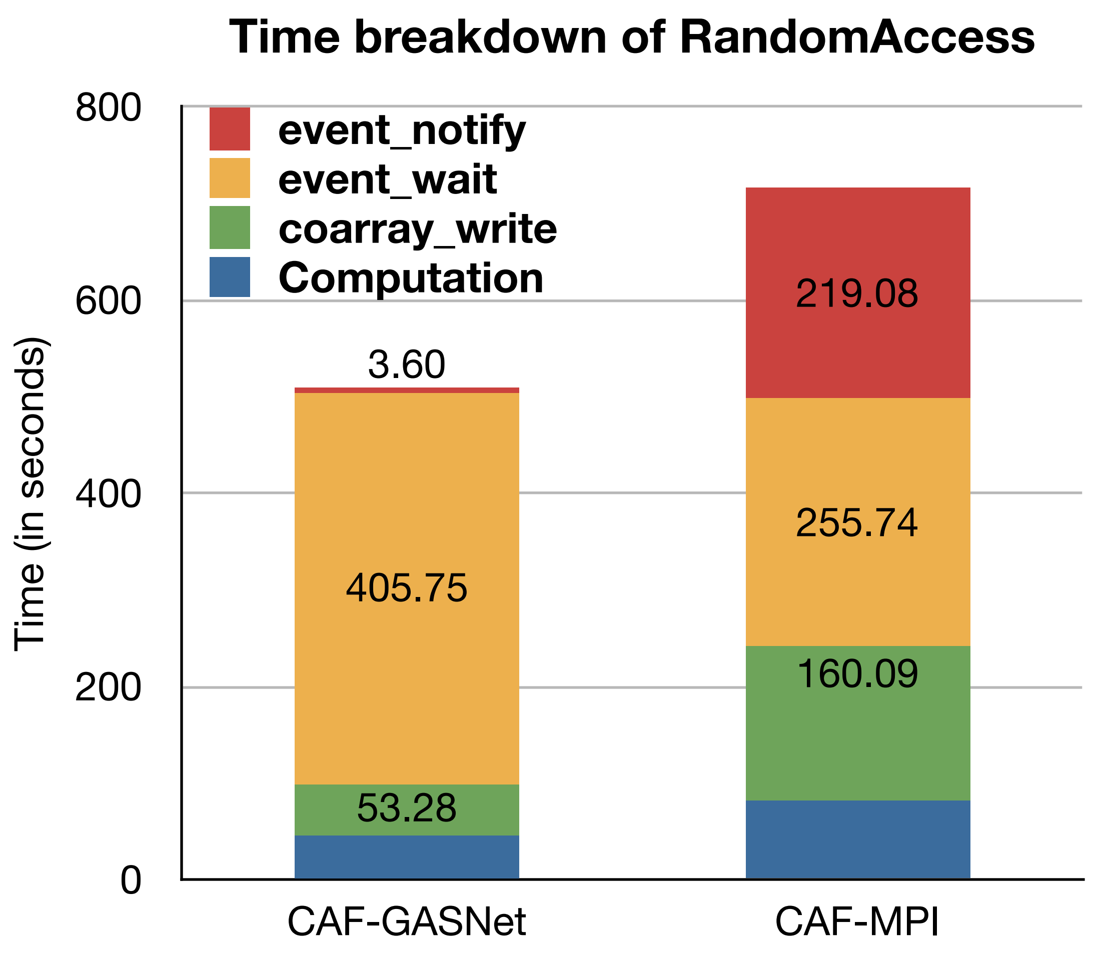
"Measures worst case system throughput"



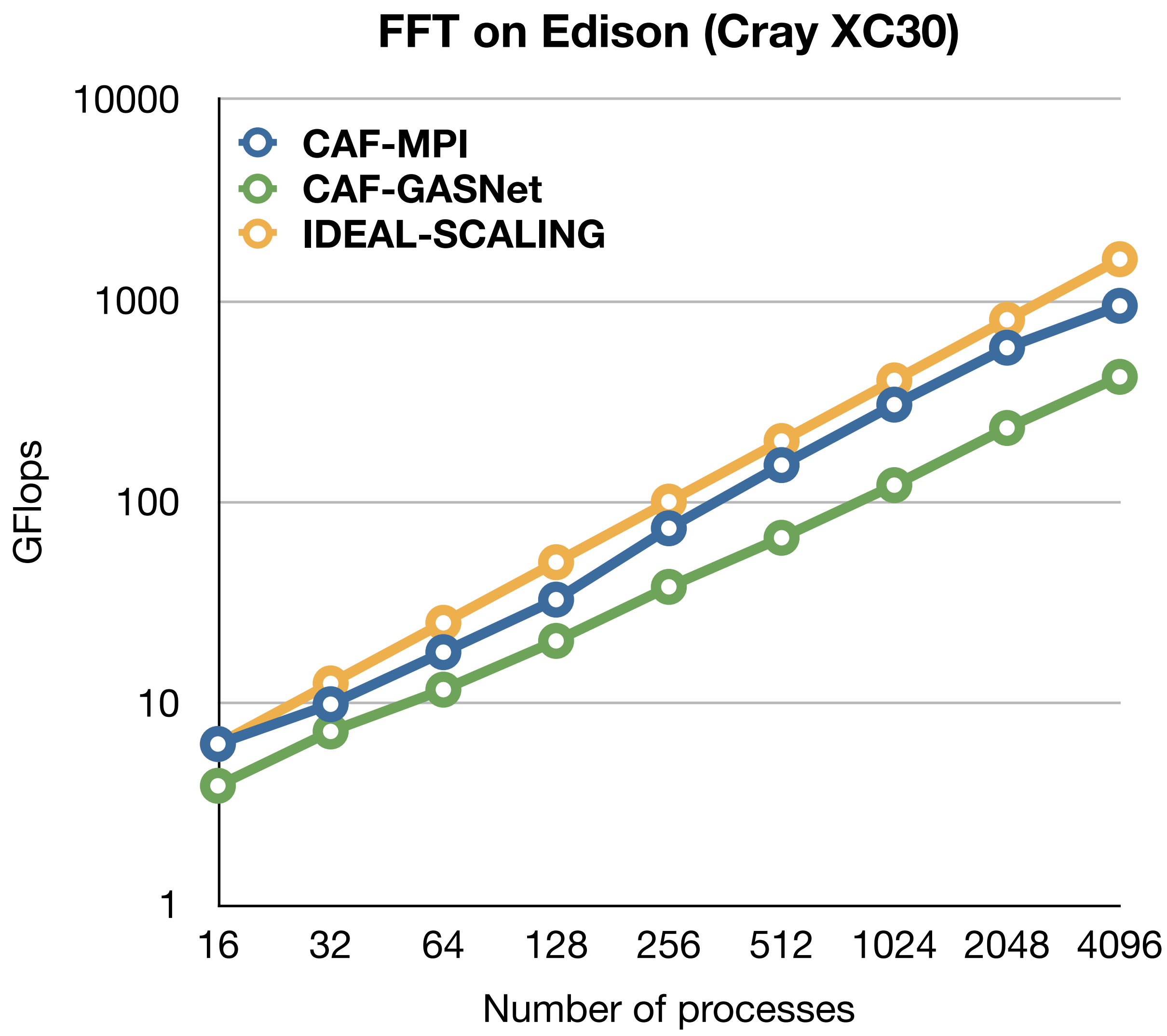


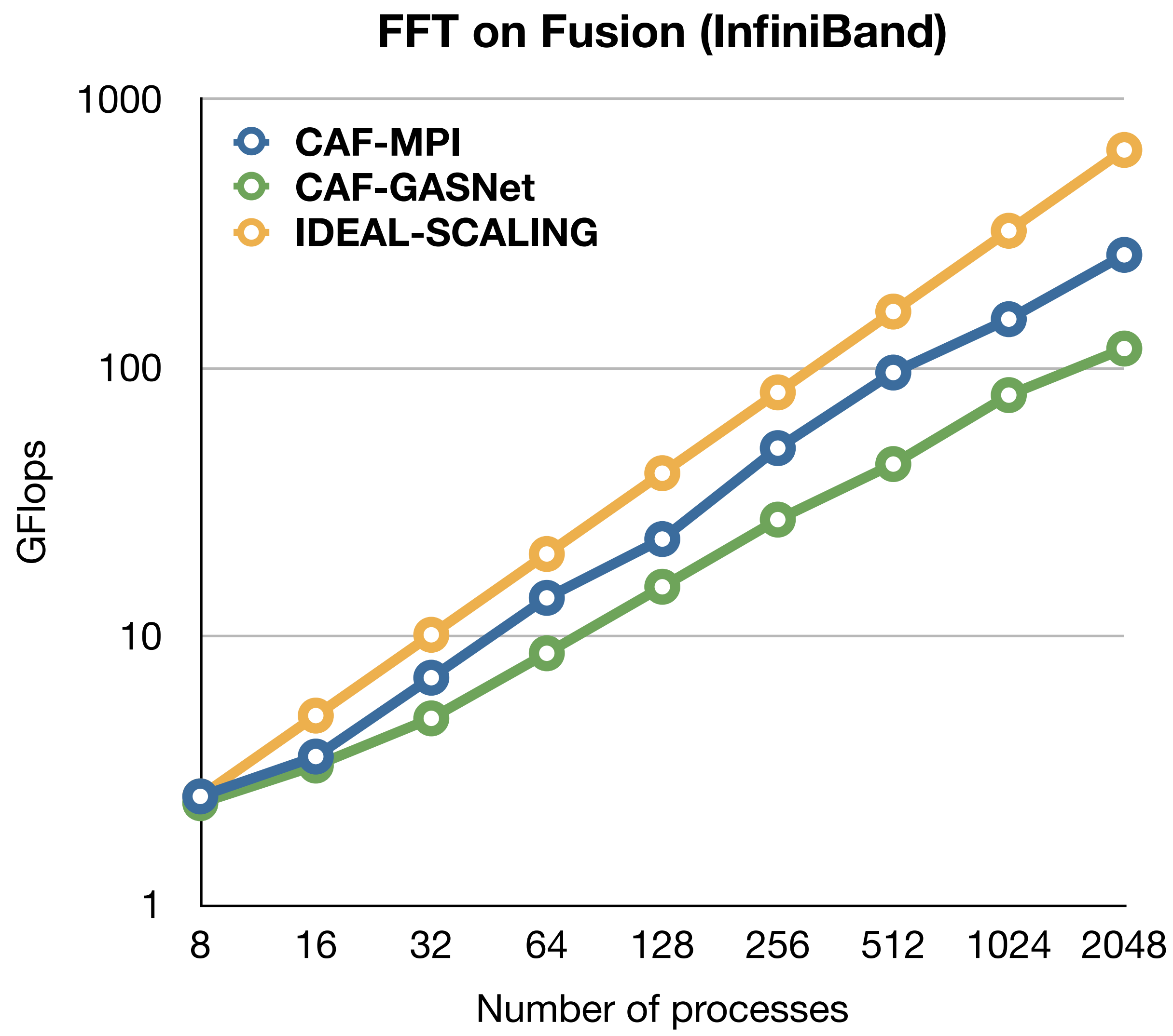
Performance Analysis of RandomAccess

- The time spent in communication are about the same
- **event_notify** is slower in CAF-MPI because of **MPI_WIN_FLUSH_ALL**



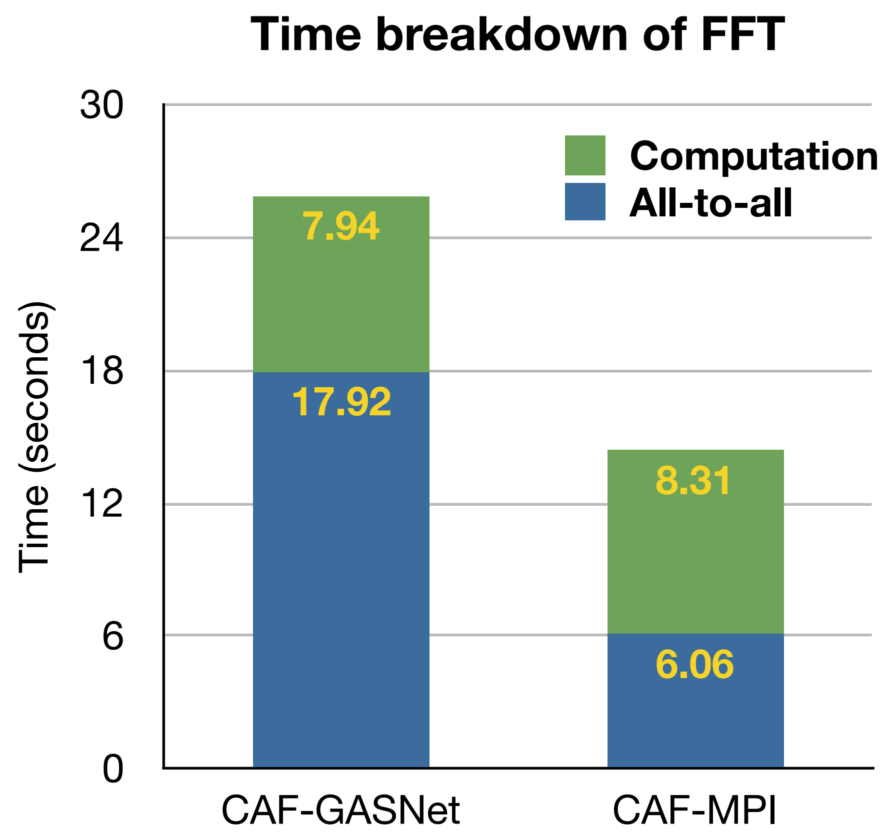
FFT


FFT on Edison (Cray XC30)
CAF-MPI
CAF-GASNet
IDEAL-SCALING
GFlops
10000
1000
100
10
1
16
32
64
128
256
512
1024
2048
4096
Number of processes


FFT on Fusion (InfiniBand)
CAF-MPI
CAF-GASNet
IDEAL-SCALING
GFlops
1000
100
10
1
8
16
32
64
128
256
512
1024
2048
Number of processes

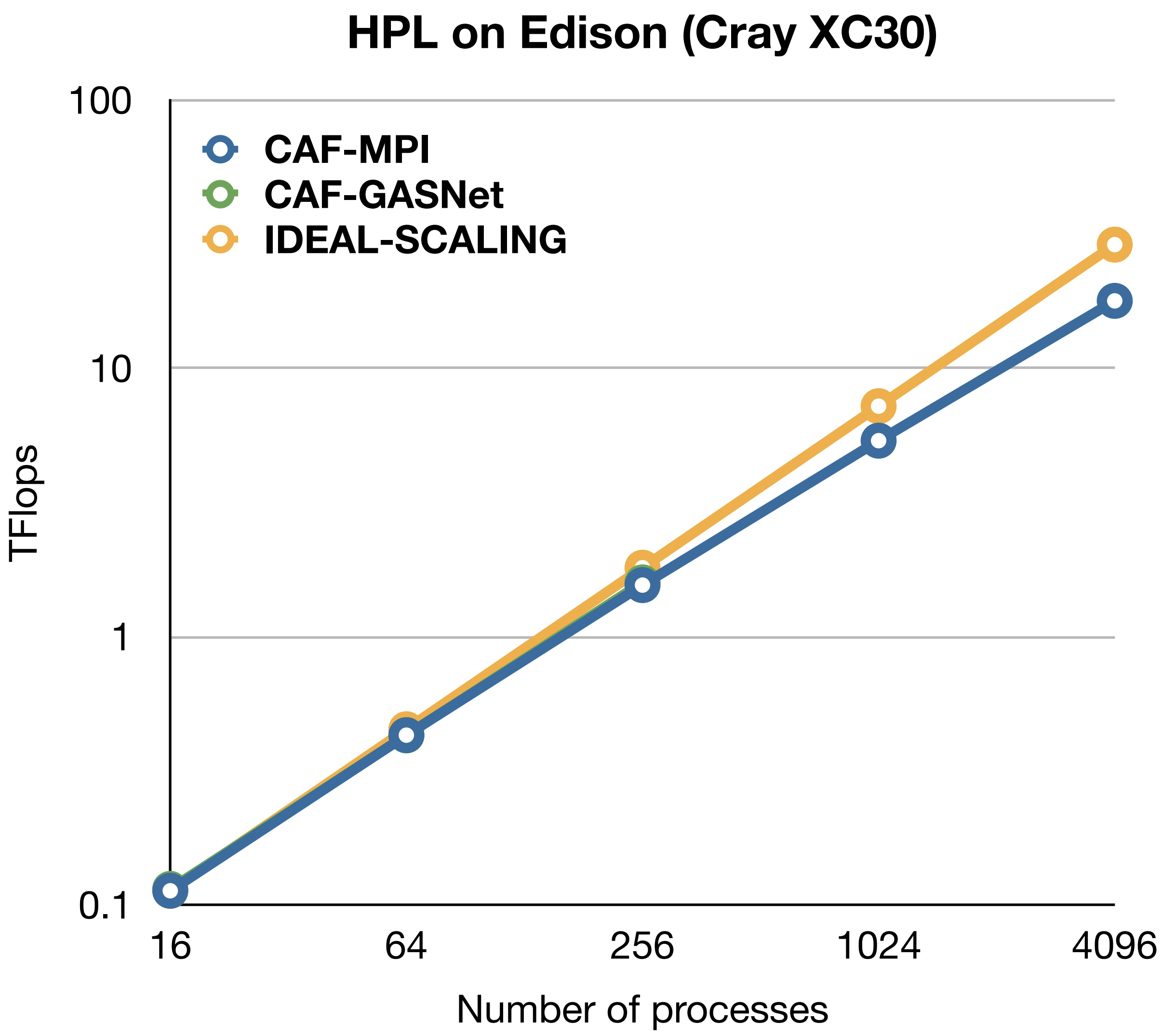
Performance Analysis of FFT

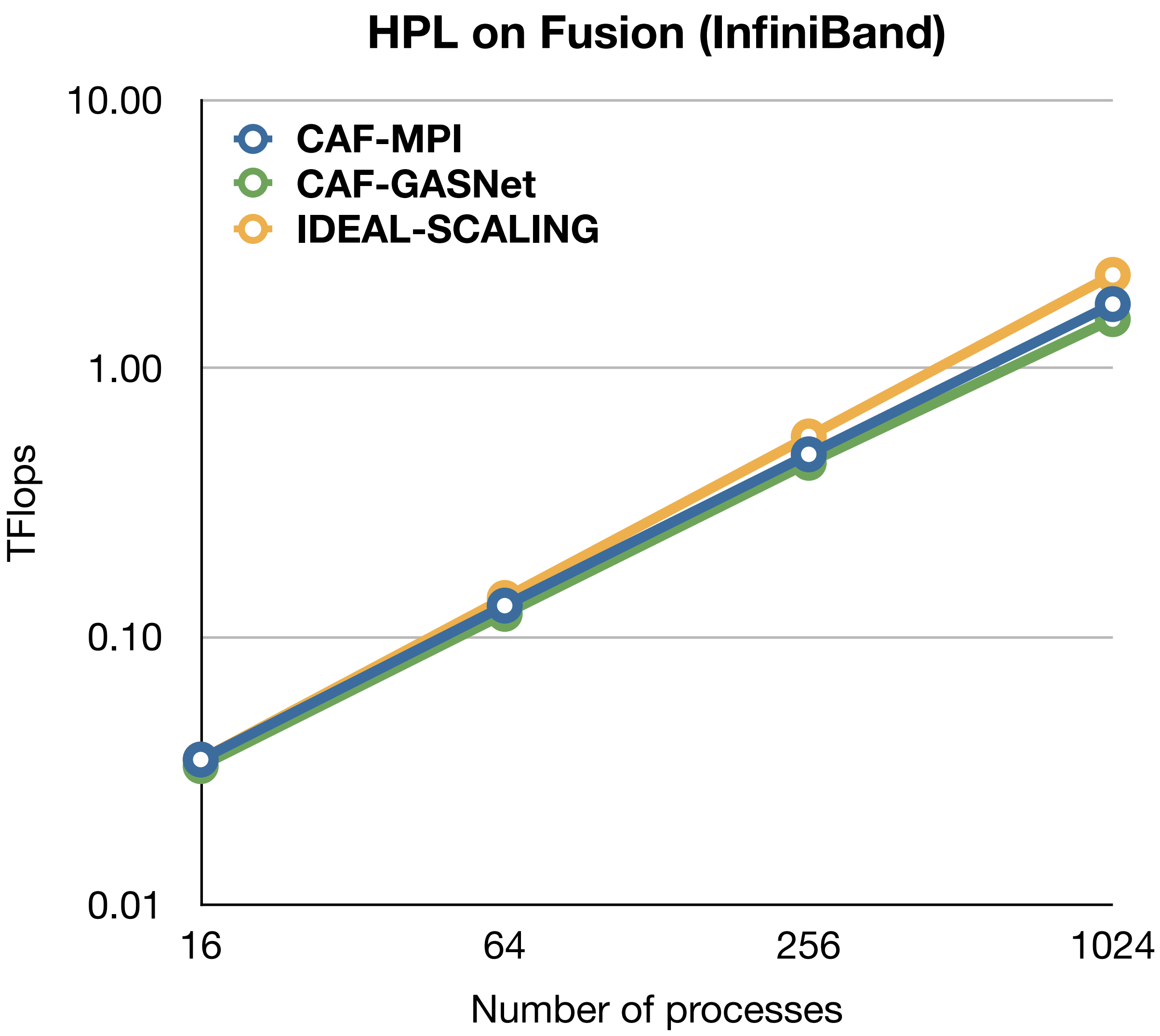
- The CAF 2.0 version of FFT solely uses ALLtoALL for communication
- CAF-MPI performs better because of fast all-to-all implementation



High Performance Linpack

"computation intensive"





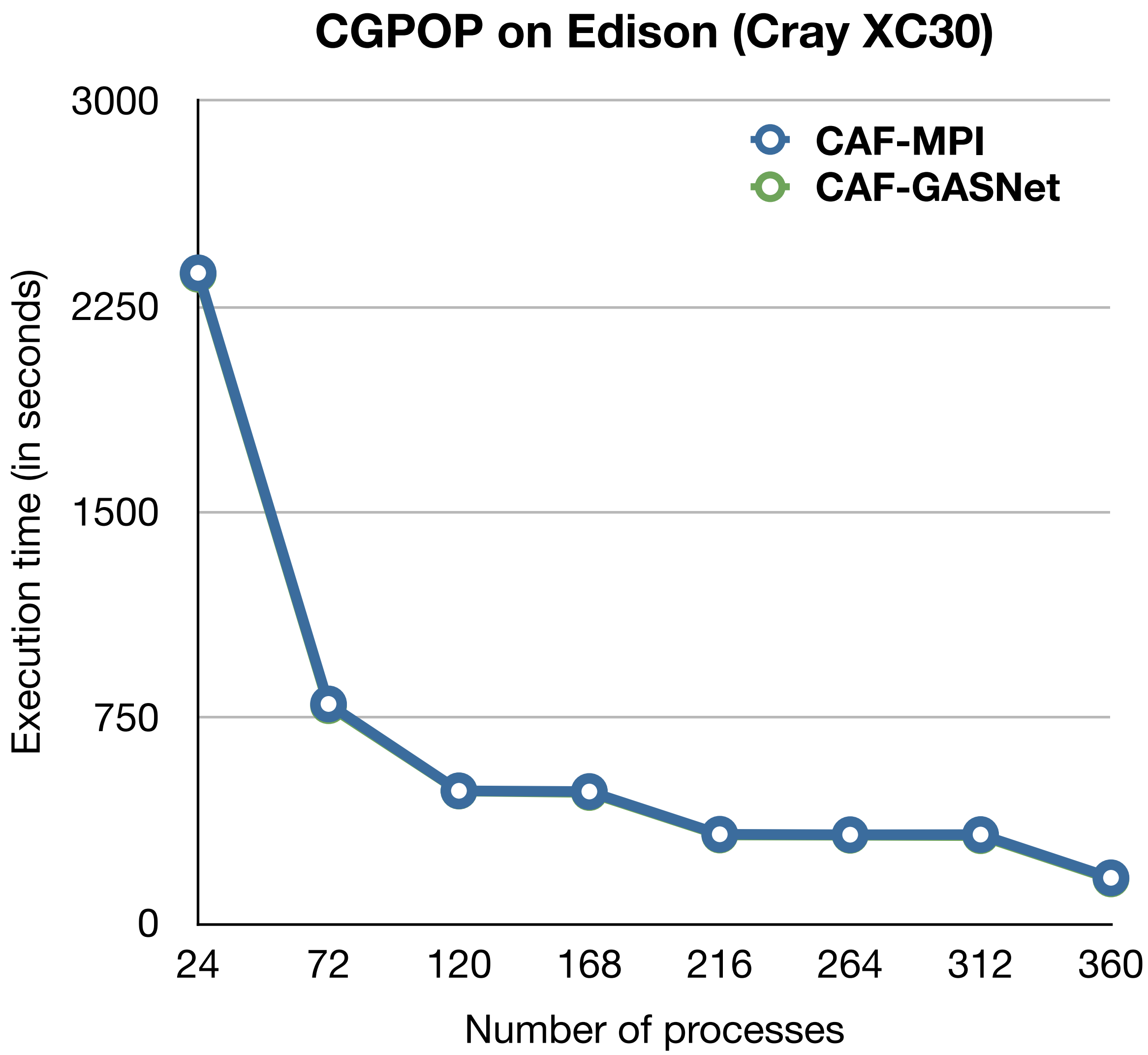
CGPOP

"A CAF+MPI hybrid application"

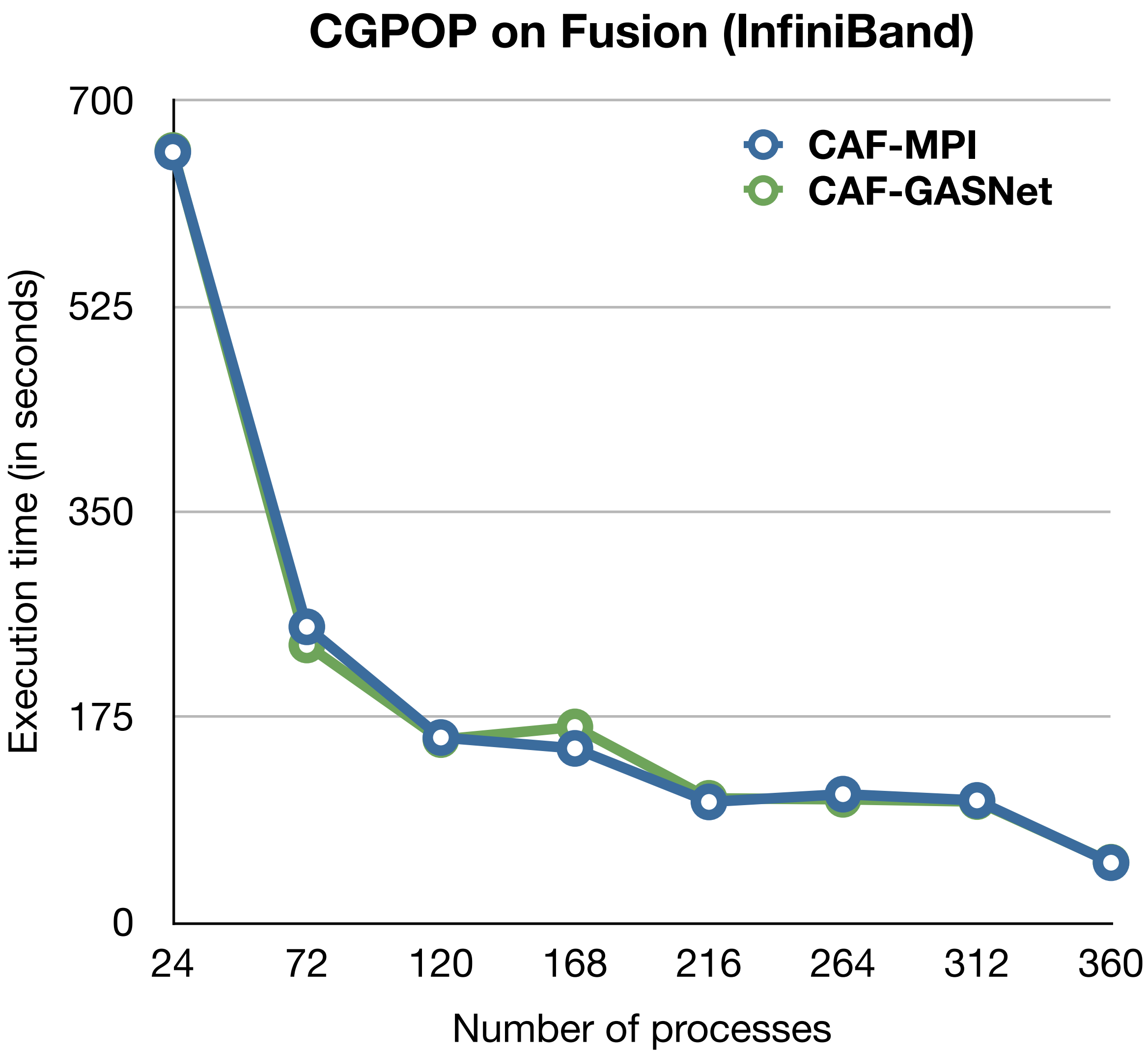
- The conjugate gradient solver from LANL Parallel Ocean Program 2.0
  - performance bottleneck of the full POP 2.0 application
- Performs linear algebra computation interspersed with two comm. steps:
  - **GlobalSum**: a 3-word vector sum (MPI_Reduce)
  - **UpdateHalo**: boundary exchange between neighboring subdomains (CAF)

Andrew I Stone, John M. Dennis, Michelle Mills Strout, "Evaluating Coarray Fortran with the CGPOP Miniapp"

CGPOP



CGPOP on Edison (Cray XC30)
CAF-MPI
CAF-GASNet
Execution time (in seconds)
3000
2250
1500
750
0
24
72
120
168
216
264
312
360
Number of processes



CGPOP on Fusion (InfiniBand)
CAF-MPI
CAF-GASNet
Execution time (in seconds)
700
525
350
175
0
24
72
120
168
216
264
312
360
Number of processes

Conclusions

use MPI to build PGAS runtimes, good or bad?

- The benefits of building runtime systems on top of MPI
  - Interoperability with numerous MPI based libraries (Standard CAF)
  - Deliver performance comparable to runtimes built with GASNet
  - MPI's rich interface is time-saving
- What current MPI RMA lacks
  - **MPI_WIN_RFLUSH** - overlap synchronization with computation
  - **Active Messages** - full interoperability