EXPERIENCES WITH MPI RMA AS A FOUNDATIONAL COMMUNICATION ABSTRACTION FOR ONE-SIDED PROGRAMMING MODELS

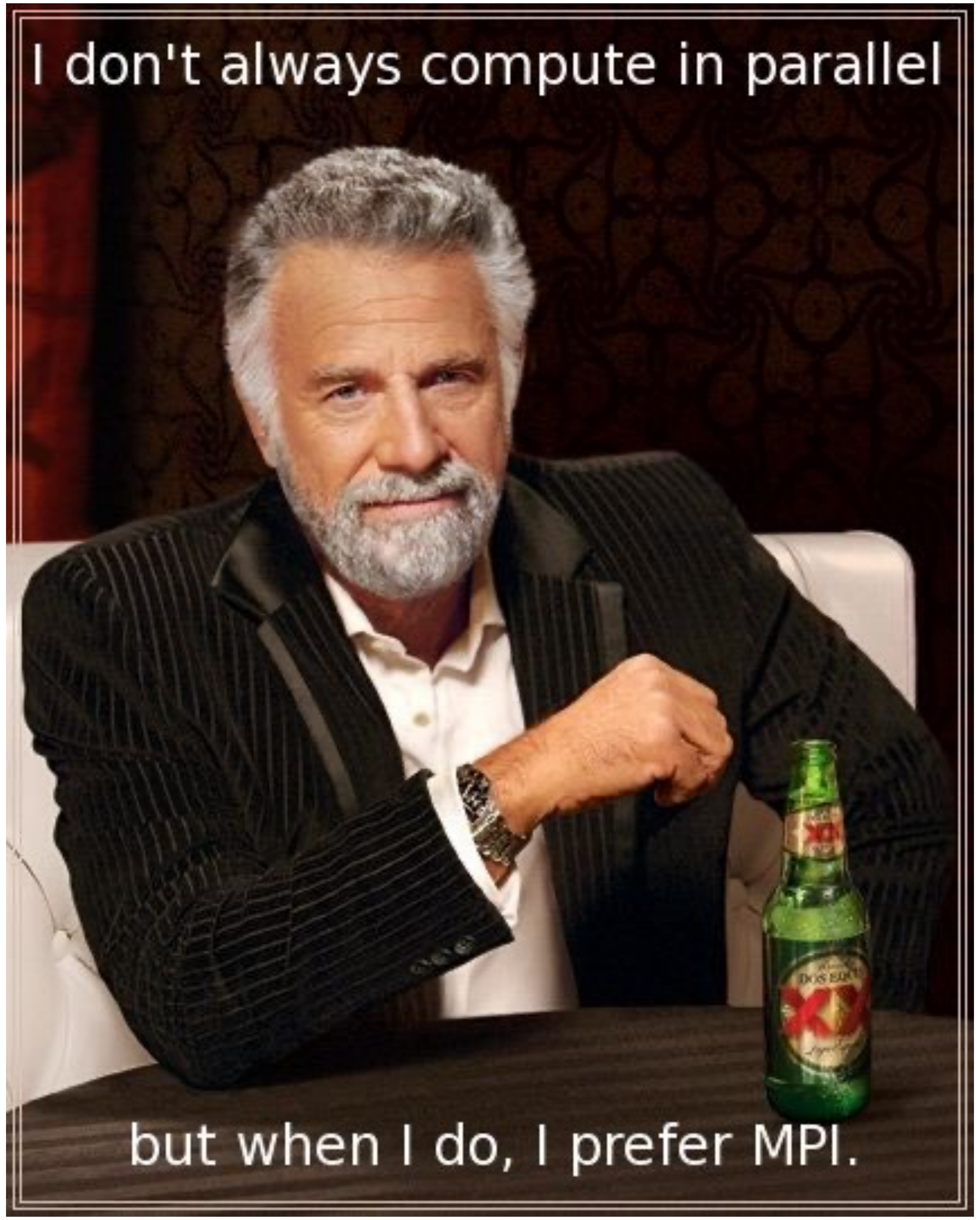
JEFF HAMMOND, PRINCIPAL ENGINEER

OUTLINE

- Background on MPI-3 RMA
- Summary of efforts to use MPI-3 RMA
- Overview of ARMCI-MPI
- Overview of OSHMPI v1
- My thoughts on the past, present and future of RMA

MOTIVATION FOR USING MPI

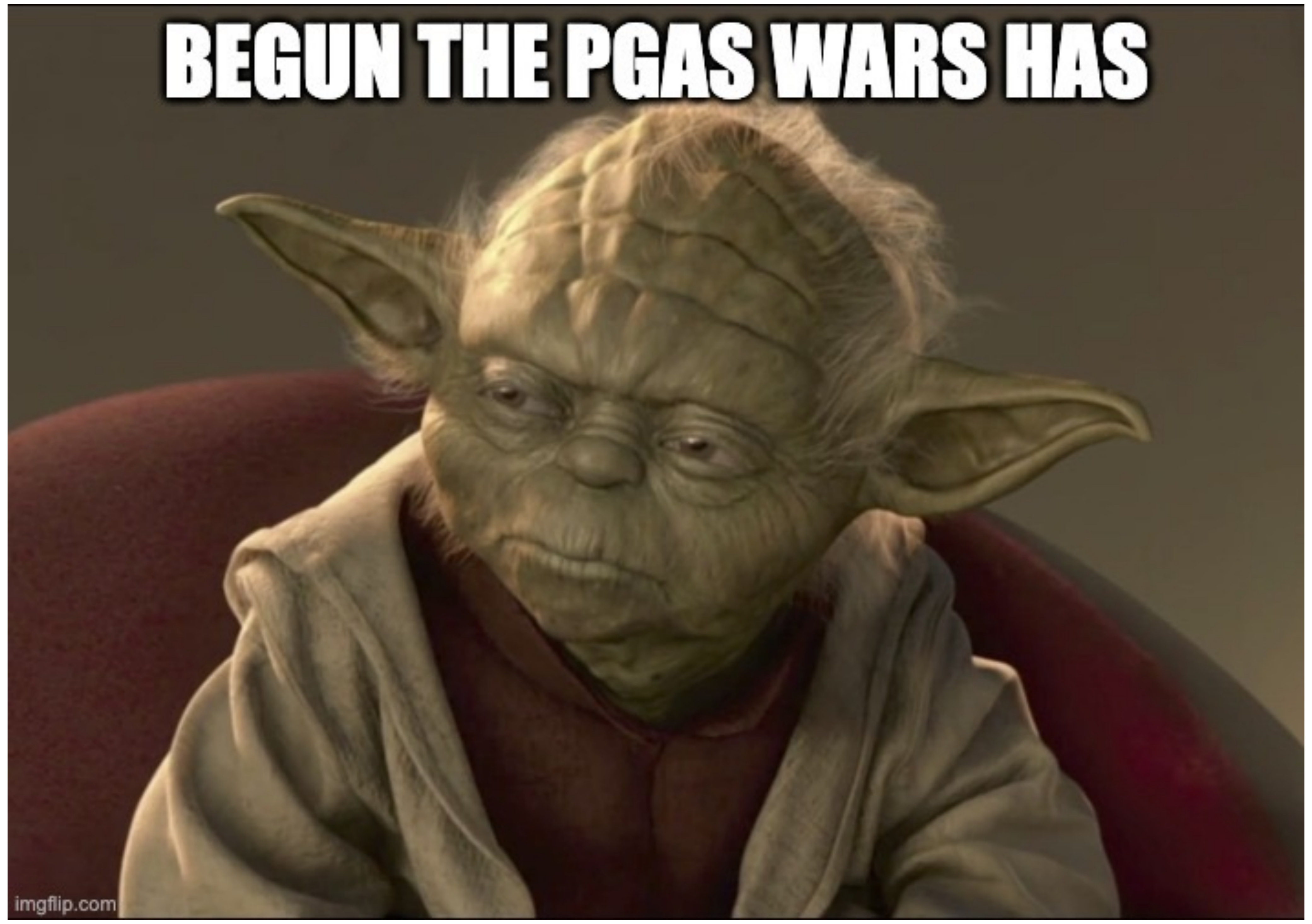
- Most programmers have better things to do than debug *runtime system issues*. HPC ubiquity requires things to just work.
- *Complex software* needs a language-agnostic* and language-interoperable programming model / runtime.
- Very few HPC applications bottleneck in communication, so compromising on performance to get portability is a good idea.


I don't always compute in parallel
but when I do, I prefer MPI.

* C/C++, Fortran (3x), Python, C++, Java, C#, D, Go, Perl, Ruby, Rust, Julia, Ocaml, Haskell, Pascal, Ada, … , but apparently not COBOL.

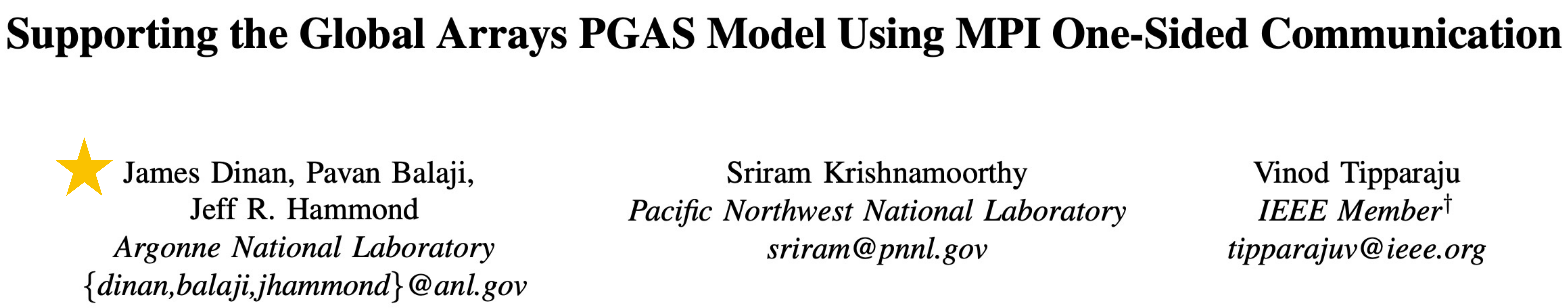
HISTORICAL CONTEXT

- Prior to MPI-3, there was a lot of debate about MPI versus PGAS, which was a two-sided versus one-sided debate.
- MPI Forum aspired to make MPI-3 RMA suitable for use in the following:
  - SHMEM
  - Global Arrays (or ARMCI)
  - Fortran coarrays
  - UPC
- MPI RMA working group aspired to make RMA suitable for use on the following:
  - Bad networks
  - Good networks
  - *Imaginary networks*



STATUS OF RMA USAGE (INCOMPLETE)

Model	**Project**	**Status**
Global Arrays	*ARMCI-MPI*	*In production for NWChem*
OpenSHMEM	*OSHMPI*	*Useful for research, SMPs*
OpenSHMEM	OSHMPI v2	OpenSHMEM 1.4 compliant
Fortran coarrays	OpenCoarrays	GCC 5+
Fortran coarrays	Intel Fortran	Supported in releases since ~2015
Fortran coarrays	CAF 2.0	Published
UPC	GUPC	Evaluated but not using
Grappa	Grappa	Prod. w/ P2P+NBC, no RMA
HPX	HPX	Production w/ P2P, no RMA
Chapel	Chapel	Discussed



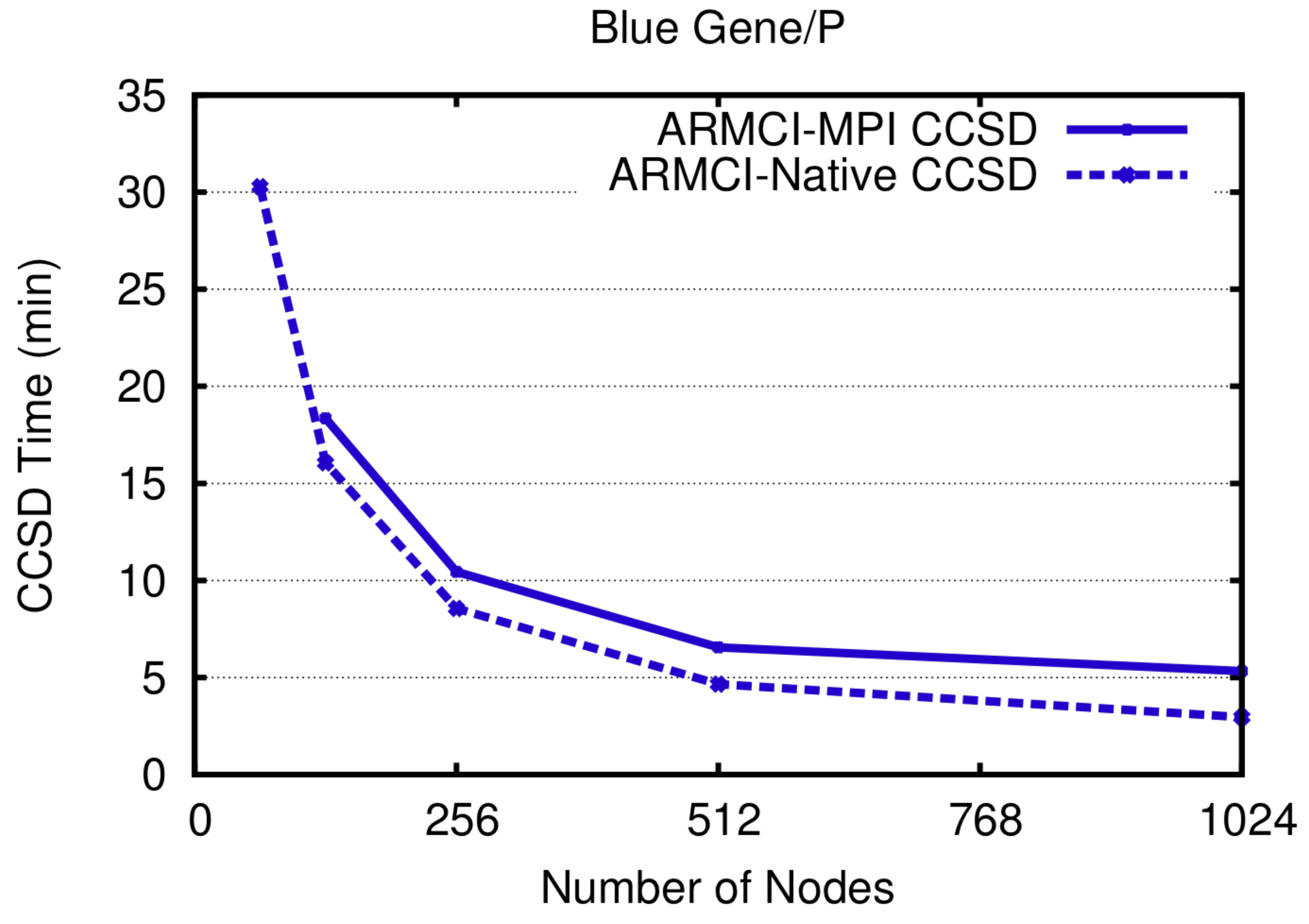
Supporting the Global Arrays PGAS Model Using MPI One-Sided Communication

James Dinan, Pavan Balaji,
Jeff R. Hammond
Argonne National Laboratory
{dinan,balaji,jhammond}@anl.gov

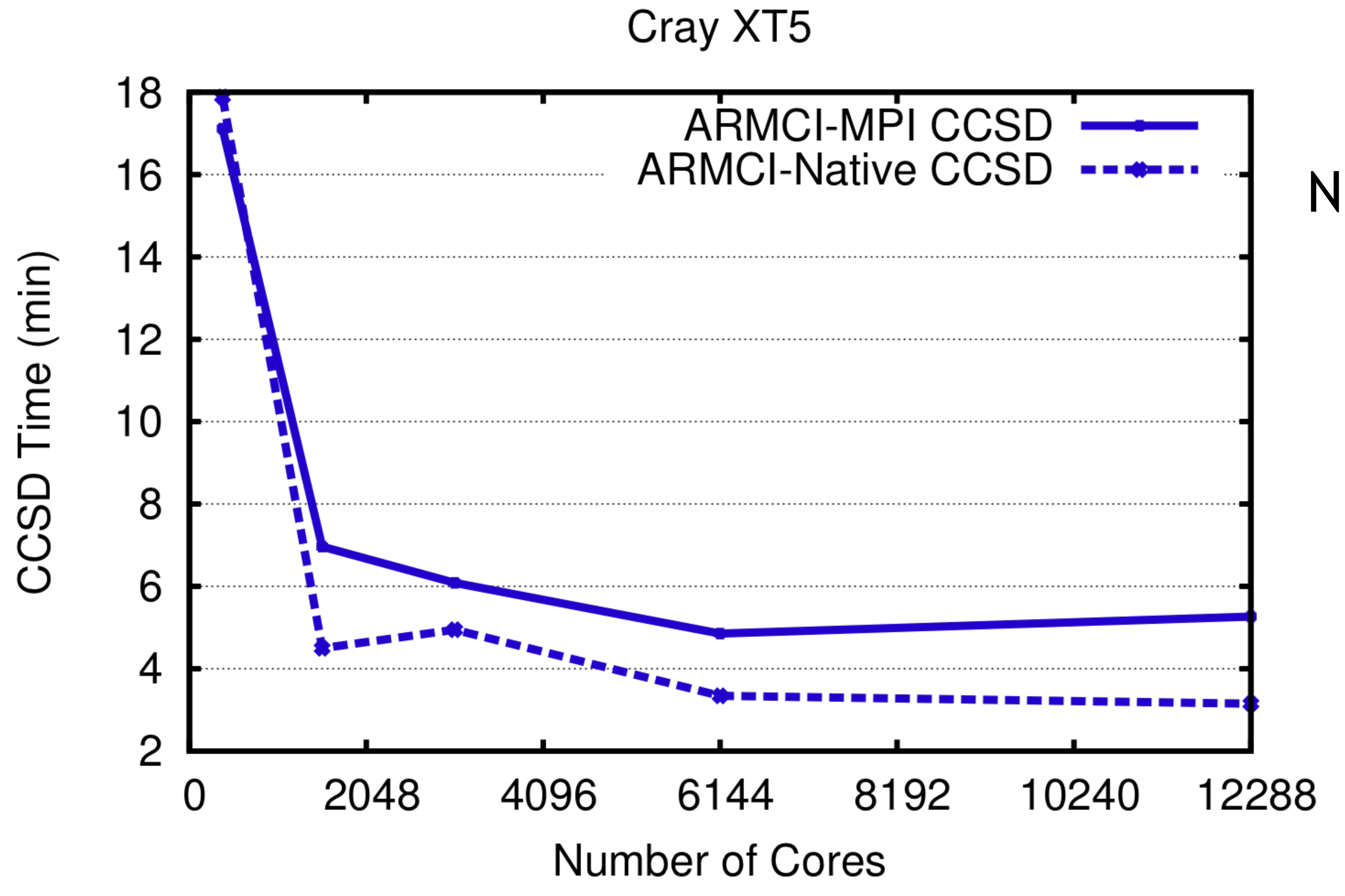
Sriram Krishnamoorthy
Pacific Northwest National Laboratory
sriram@pnnl.gov

Vinod Tipparaju
IEEE Member†
tipparajuv@ieee.org

ARMCI	MPI-2 RMA
No handles for data, just pointers	Opaque handles for data (windows)
Sequential consistency to same location	RMA operations unordered
Separate local and remote completion	Local=remote completion
Nonblocking RMA	Nonblocking essentially impossible
Atomic operations	No atomic operations
Asynchronous progress guarantee	No asynchronous progress guarantee

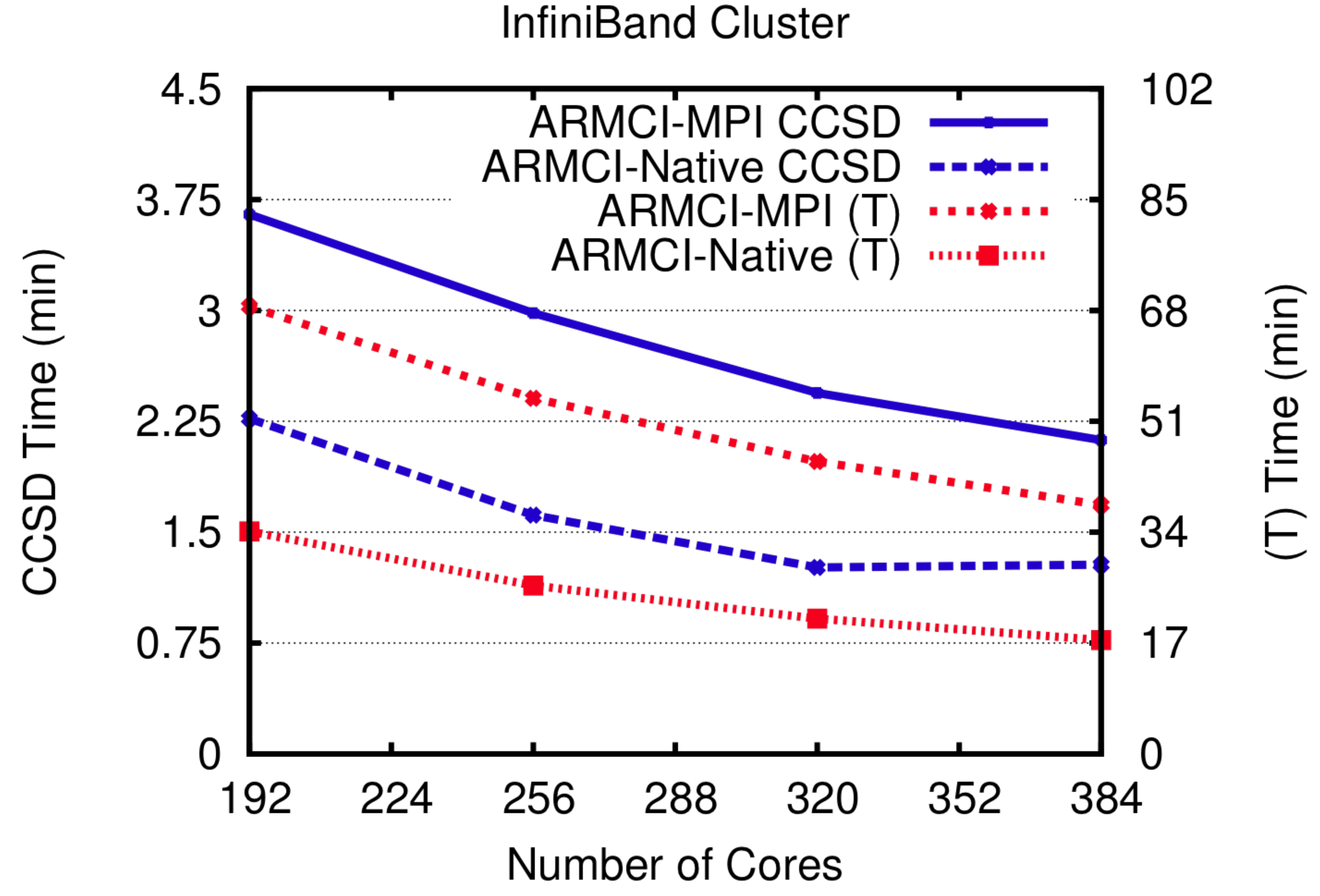


Blue Gene/P

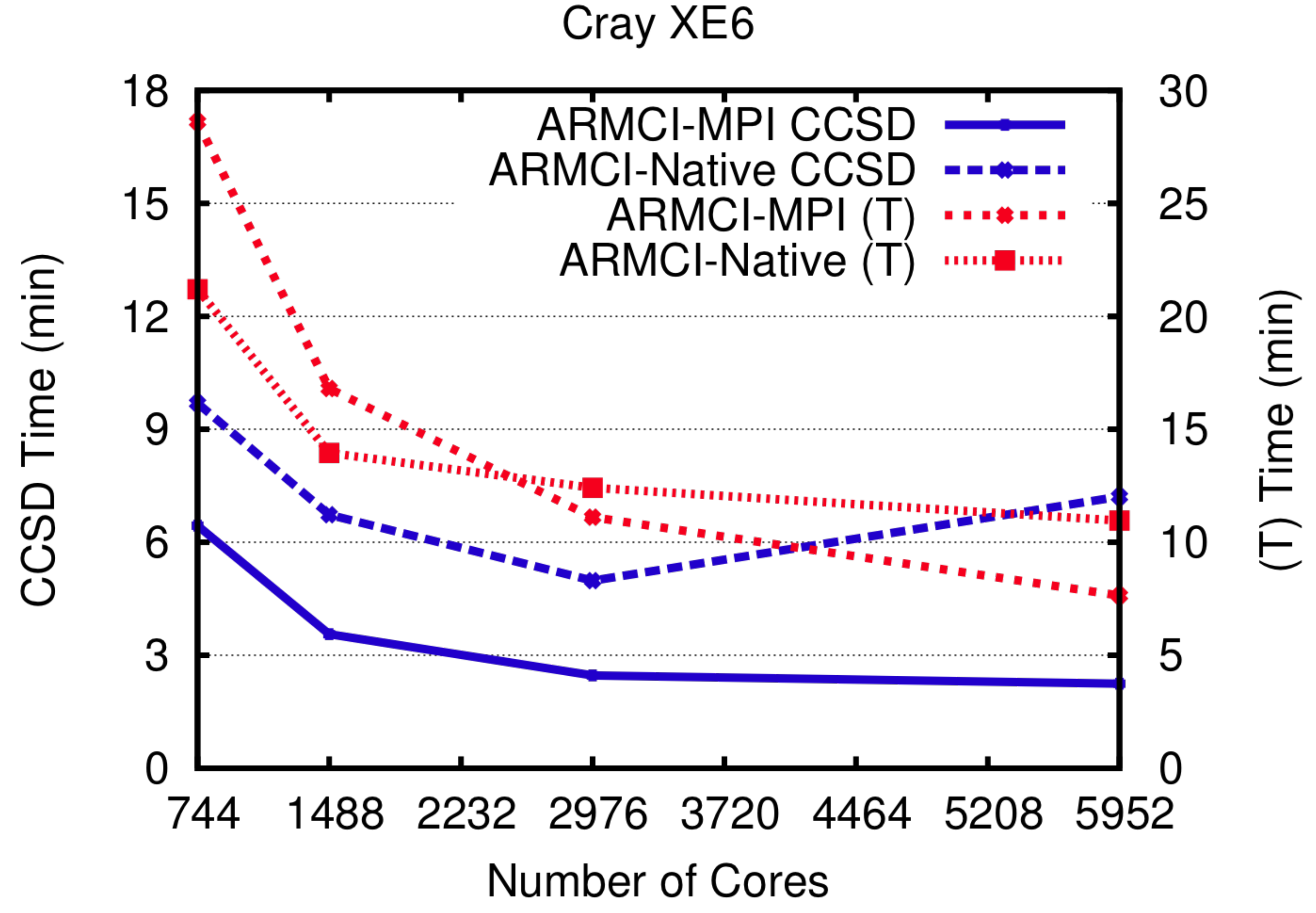


Cray XT5

Native = ~1 year



InfiniBand Cluster



Cray XE6

MPI-3 UPDATE

https://github.com/pmodels/armci-mpi

ARMCI	MPI-3 RMA
No handles for data, just pointers	Opaque handles for data (windows)
Sequential consistency to same location	*Accumulate operations ordered to same location*
Separate local and remote completion	*Separate local and remote completion*
Nonblocking RMA	*Nonblocking feasible*
Atomic operations	*Atomic operations sufficient*
Asynchronous progress guarantee	No asynchronous progress guarantee

NWCHEM SCF PERFORMANCE

```
NWChem 6.3/ARMCI-MPI3/Casper

iter     energy               time
---- ----------------- -----
  1   -2830.4366669992  69.6
  2   -2831.3734512508  78.8
  3   -2831.5712563433  86.9
  4   -2831.5727802438  96.1
  5   -2831.5727956882 110.0
  6   -2831.5727956978 127.8
```

```
NWChem 6.5/ARMCI-DMAPP
(built by NERSC, Nov. 2014)

iter     energy               time
---- ----------------- -----
  1   -2830.4366670018  67.6
  2   -2831.3734512526  85.5
  3   -2831.5713109544 105.4
  4   -2831.5727856636 126.6
  5   -2831.5727956992 161.7
  6   -2831.5727956998 190.9
```

Running on 8 nodes with 24 ppn. Casper uses 2 ppn for comm.

NWCHEM SCF PERFORMANCE

```
NWChem 6.3/ARMCI-MPI3/Casper

iter    energy            time
---- ---------------- -----
 1   -2830.4366669990  69.3
 2   -2831.3734512499  77.1
 3   -2831.5712604368  84.6
 4   -2831.5727804428  93.0
 5   -2831.5727956927 107.3
 6   -2831.5727956977 128.0
```

```
NWChem Dev/ARMCI-MPIPR
(built by NERSC, Sept. 2015)

iter    energy            time
---- ---------------- -----
 1   -2830.4366669999  61.4
 2   -2831.3734512509  69.3
 3   -2831.5713109521  77.8
 4   -2831.5727856618  87.3
 5   -2831.5727956974 103.9
 6   -2831.5727956980 125.7
```

Running on 8 nodes with 24 ppn. **Both** use 2 ppn for comm.

ARMCI-MPI + CASPER

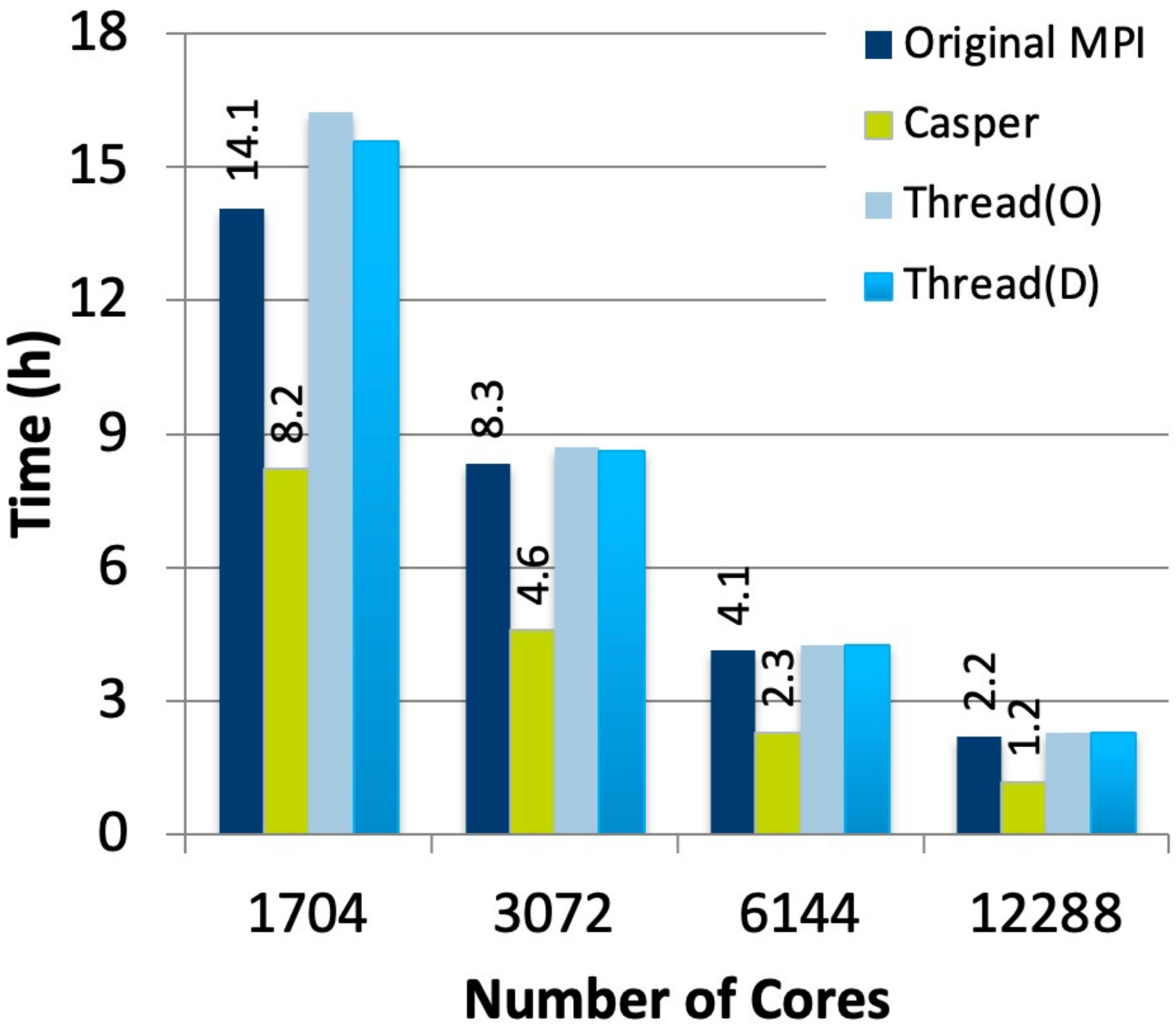
Asynchronous progress is important

Implementation

- Dedicate process(es) for communication, similar to ARMCI.
- Intercept all RMA calls and redirect using shared-memory (requires Win_allocate)

Application usage

- NWChem on 40K+ cores of Cray XC30.
- Bandwidth-limited CCSD(T) for $(H_2O)_{21}$.



M. Si, A. J. Pena, J. Hammond, P. Balaji, M. Takagi, Y. Ishikawa. IPDPS15.
“Casper: An Asynchronous Progress Model for MPI RMA on Many-Core Architectures.”

ARMCI-MPI SUMMARY

Made NWChem universally portable:

- Only ran on Blue Gene/Q because of ARMCI-MPI (but MPI-2 RMA ☹)
- Ran on ARM32 out-of-the-box in 2013
- No weird failures on IB or problems with >>2GB arrays

But

- Asynchronous progress is still a problem. Casper is not always better.
- Latency is too high. ARMCI over two-sided often wins for DFT code.
- Open-MPI correctness issues in RMA are still causing problems for users and developers.

A direct port of Global Arrays would have been better (but much more work)…

Implementing OpenSHMEM using MPI-3 one-sided communication

Jeff R. Hammond[1], Sayan Ghosh[2], and Barbara M. Chapman[2]

ARMCI-MPI

- One window for every allocation (expensive window lookup)
- Reverse-engineered semantics from PNNL implementation
- Weird contortions for ARMCI groups
- Workarounds for asynchronous progress problems

OSHMPI v1

- Symmetric heap suballocated from one window (fast lookup)
- Direct translation from OpenSHMEM specification
- Weird contortions for SHMEM PE subsets
- Fast path for intranode communication
- Ignores SAME_OP_NO_OP nonsense

https://github.com/jeffhammond/oshmpi

OSHMPI PUT MESSAGE-RATE ON IB

Log Rate (Messages/s)
10^8
10^7
10^6
10^5
10^4
10^3
10^2
2^0
2^5
2^{10}
2^{15}
2^{20}
2^{25}
Message size (bytes)
OSHMPI
GASNet
Portals4
MVAPICH2-X
MLNX
Log Rate (Messages/s)
10^7
10^6
10^5
10^4
10^3
10^2
2^0
2^5
2^{10}
2^{15}
2^{20}
2^{25}
Message size (bytes)
OSHMPI
GASNet
Portals4
MVAPICH2-X
MLNX

Figure: Intranode (left) and internode (right).

J. R. Hammond, S. Ghosh, and B. M. Chapman, *First OpenSHMEM Workshop: Experiences, Implementations and Tools*.
“Implementing OpenSHMEM using MPI-3 one-sided communication.”

MPI IMPLEMENTATION OVERHEAD IS (WAS?) HIGH

10^7
10^6
10^5
10^4
10^3
10^2
Log Message Rate (Messages/s)
MPI-3
OpenSHMEM
2^0
2^5
2^{10}
2^{15}
2^{20}
2^{25}
Message size (bytes)
10^8
10^7
10^6
10^5
10^4
10^3
10^2
Log Message Rate (Messages/s)
MPI-3
OpenSHMEM
2^0
2^5
2^{10}
2^{15}
2^{20}
2^{25}
Message size (bytes)

Internode (left), intranode (right). MVAPICH2-X from 2013-2014.

J. R. Hammond, S. Ghosh, and B. M. Chapman, *First OpenSHMEM Workshop: Experiences, Implementations and Tools*.
“Implementing OpenSHMEM using MPI-3 one-sided communication.”

OSHMPI SUMMARY

- **Proved that MPI-3 RMA is a viable back-end for OpenSHMEM (hence v2)**
- Easy to install on every platform
- Very good performance in shared memory only because it bypassed RMA altogether
- Very bad performance in distributed memory because of MPI RMA implementations
- Best SHMEM implementation for benchmarks dominated by collectives ☺

FINAL
THOUGHT

1. IMPLEMENTATIONS ARE THE PROBLEM

- Implementations continue to be bad at performance:
  - Latency is not good
  - Message-rate is not good
  - Bandwidth is inconsistent
  - Asynchronous progress is almost non-existent

- Implementations continue to be bad at correctness:
  - MPICH (and derivates) are correct almost all of the time…
  - NWChem = MPI_Accumulate + MPI_TYPE_SUBARRAY + MPI_SUM + MPI_DOUBLE
  - Insufficient to test correctness in shared-memory

2. RMA IS COMPLICATED AND HARD TO USE

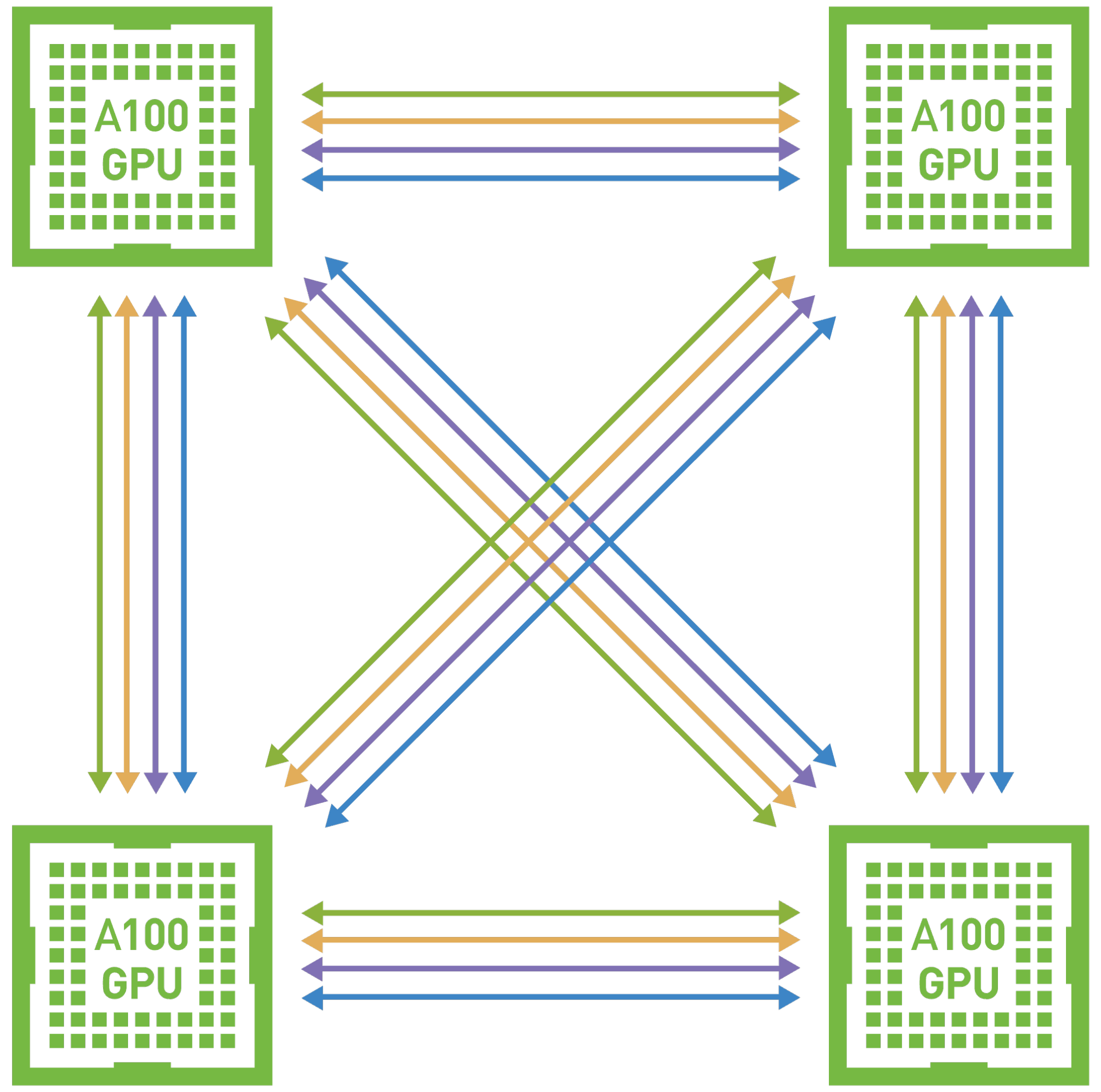
- MPI-1 features were easy to use so scientists assume MPI is easy to use, which is false for RMA
- RMA offers many ways to do the same thing:
  - Windows: allocate, create, dynamic, shared
  - Sync: fence, PSCW, lock, lock_all, flush, flush_all, flush_local, etc.
- The standard should explicitly recommend allocate + lock_all + flush(_local) as the preferred RMA motif
- We need a user guide for RMA somewhere, and a set of benchmarks to determine all the platform-specific dependence of RMA features

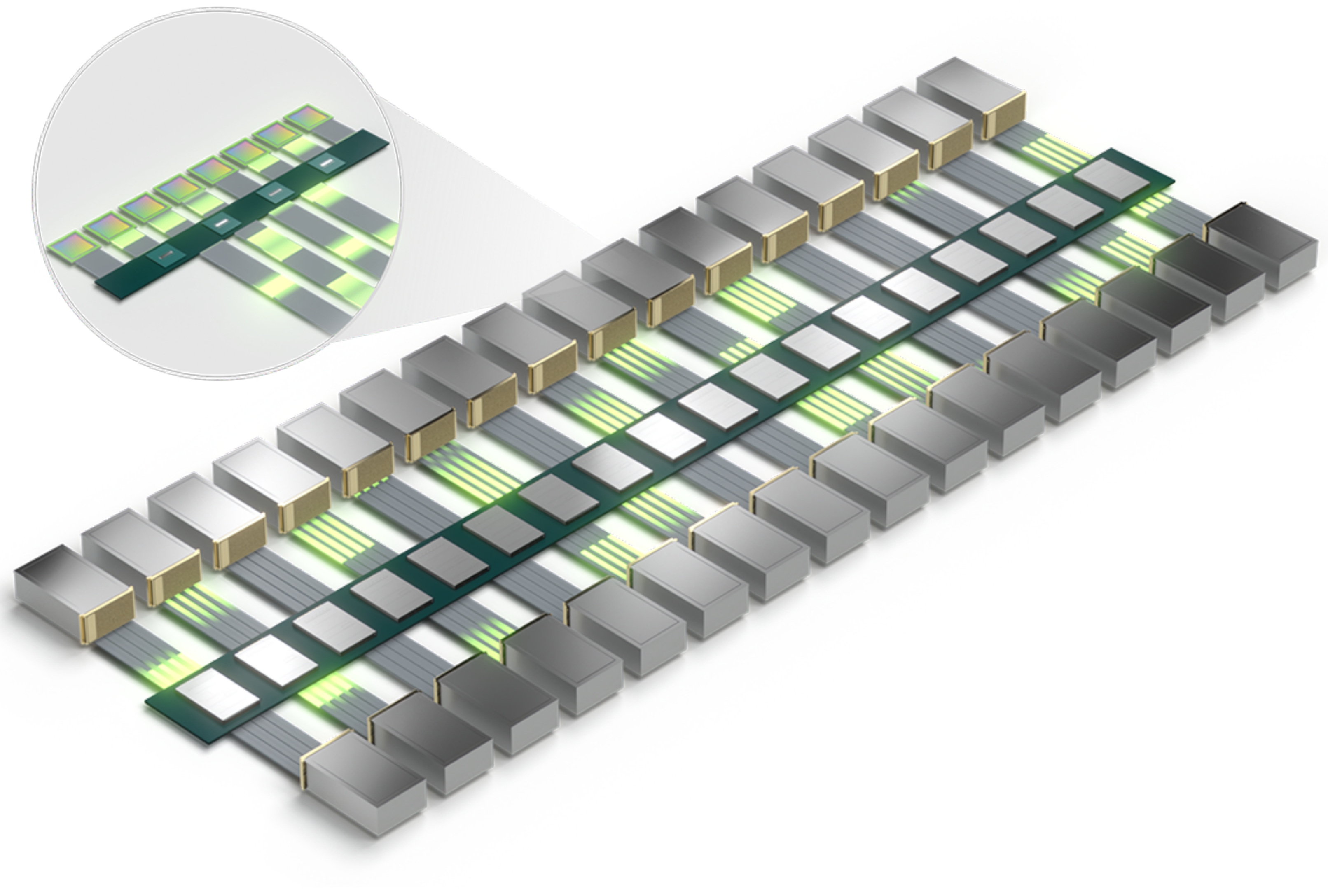
3. RMA NEEDS MINOR FIXES

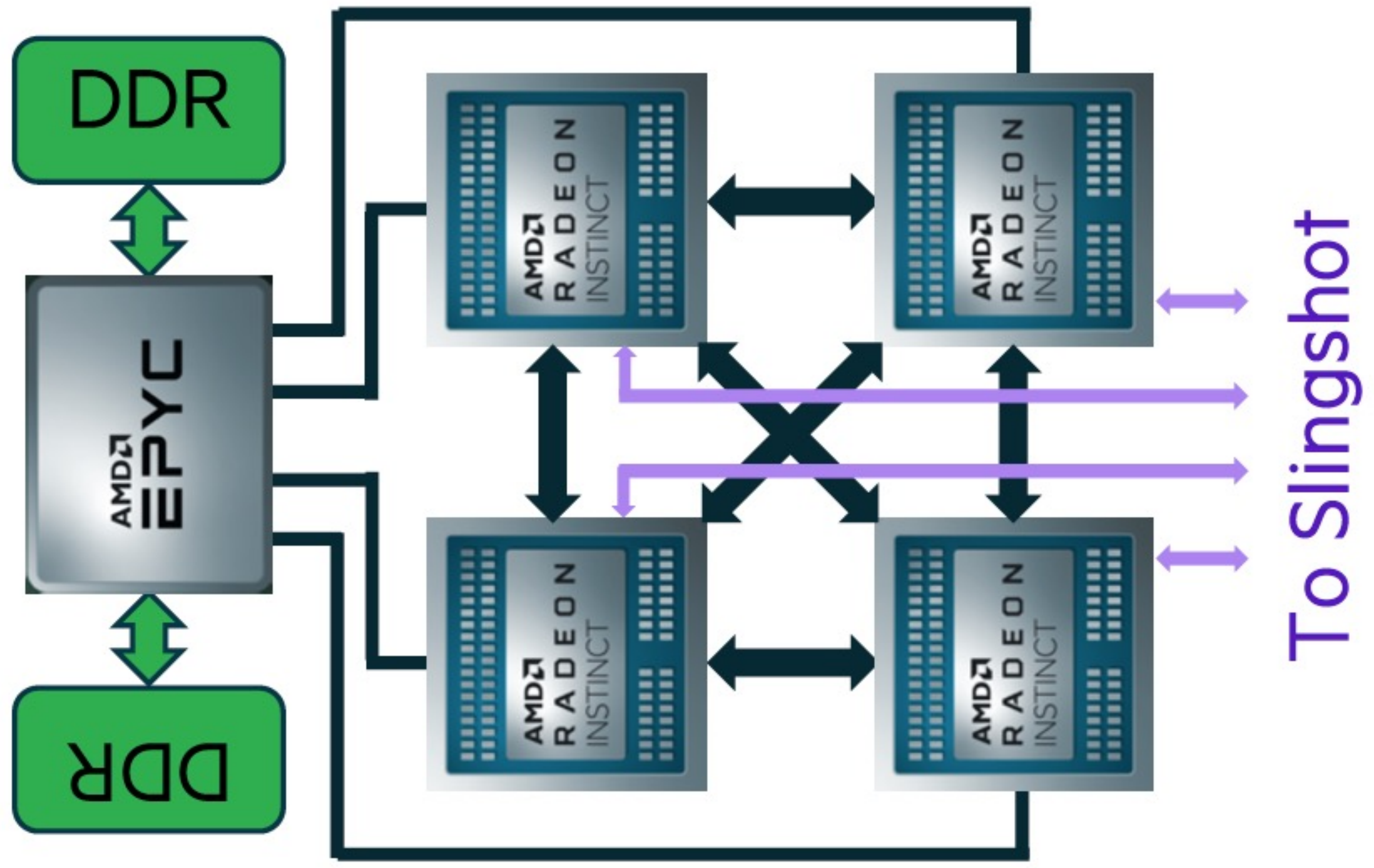
- Users should be allowed to query shared memory in an allocated window: https://github.com/mpi-forum/mpi-issues/issues/23
- There should be a request-based version of everything: https://github.com/mpi-forum/mpi-issues/issues/128
- MPI_PROD should be removed (there are **zero** use cases and **zero** users)
- The default should be ANY_OP, not SAME_OP_NO_OP
- Nonblocking remote flush is a good idea

4. RMA IS THE BEST MODEL FOR GPU COMMUNICATION

- GPUs are really good at moving data of all sizes
- Synchronization is expensive on GPUs due to massive parallelism
- RMA properly separates data movement from synchronization and supports memory allocation and registration







https://www.nvidia.com/en-us/data-center/nvlink/