Towards Remote Memory Access Programming for Data Analytics

With R. Gerstenberger, M. Besta, R. Belli @ SPCL

presented at Lawrence Berkeley National Laboratory, Berkeley, CA, June 2015

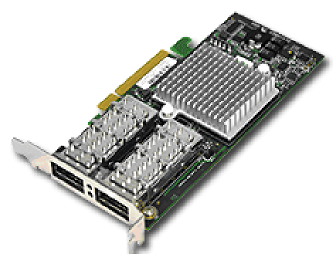
COMMUNICATION IN TODAY'S HPC SYSTEMS

- **The de-facto programming model: MPI-1**
  - Using send/recv messages and collectives

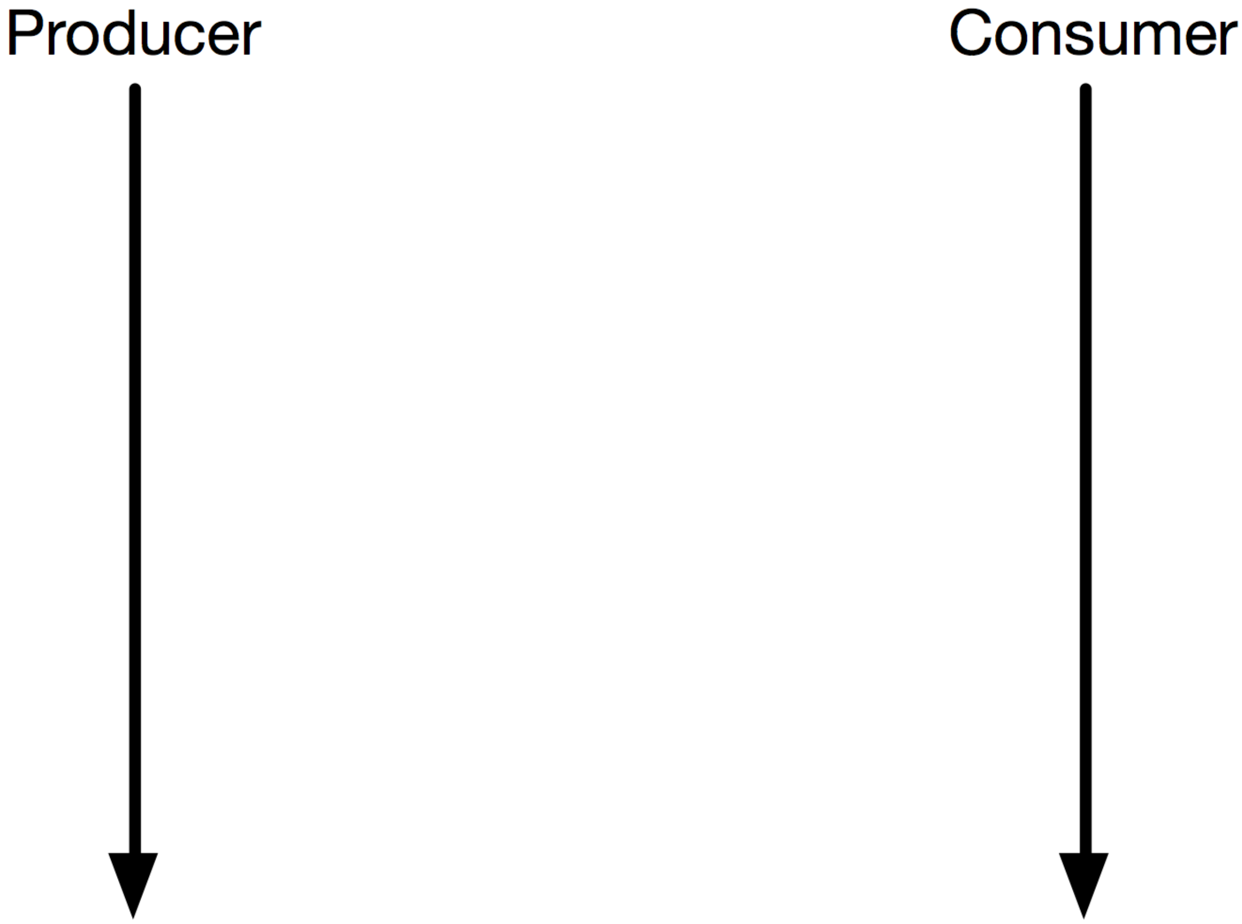
MPI

- **The de-facto network standard: RDMA, SHM**
  - Zero-copy, user-level, os-bypass, fuzz-bang

Random datacenter picture copyrighted by Reuters (yes, they go after academics with claims for 10 year old images)

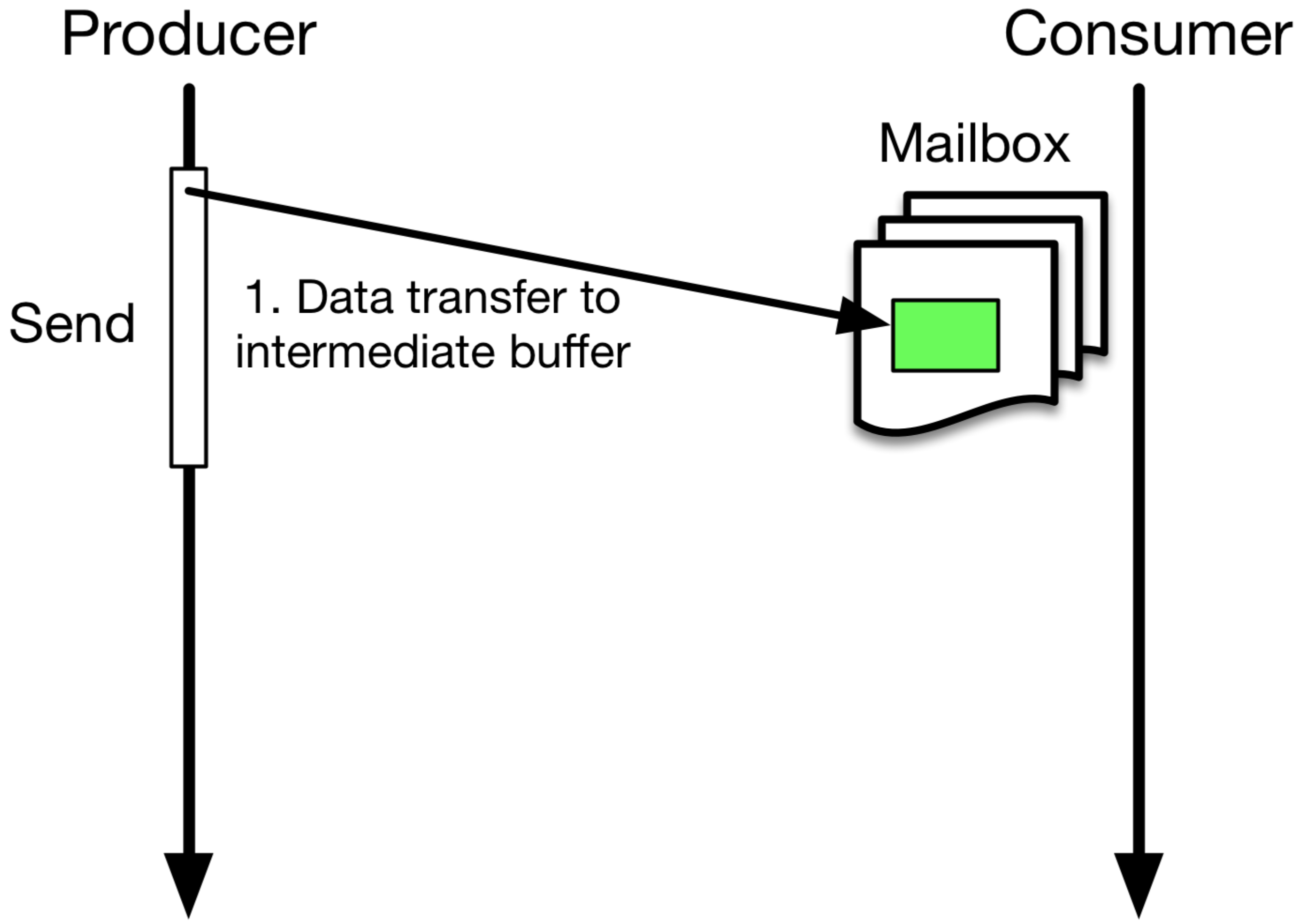


MPI-1 Message Passing – Simple Eager



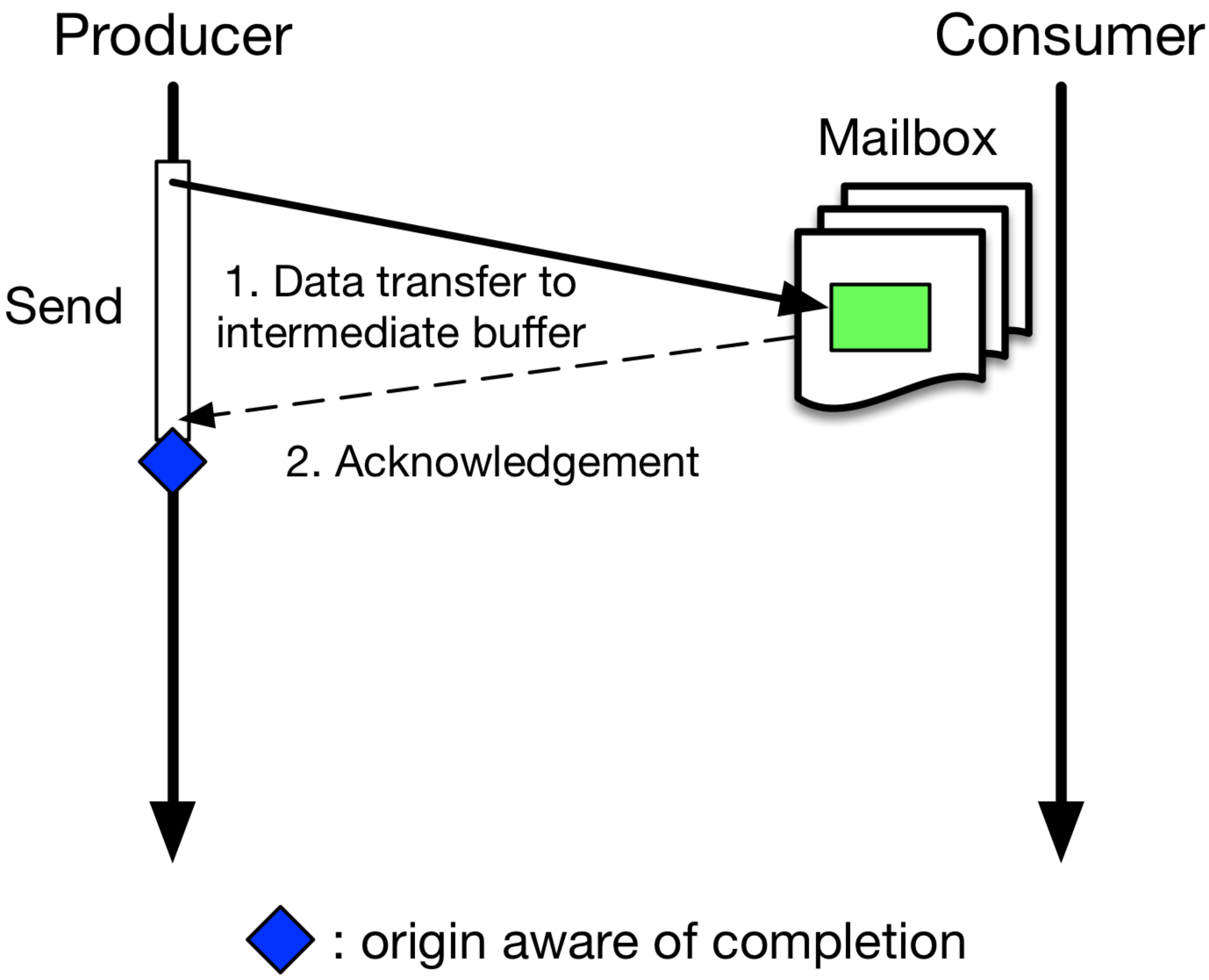
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Eager



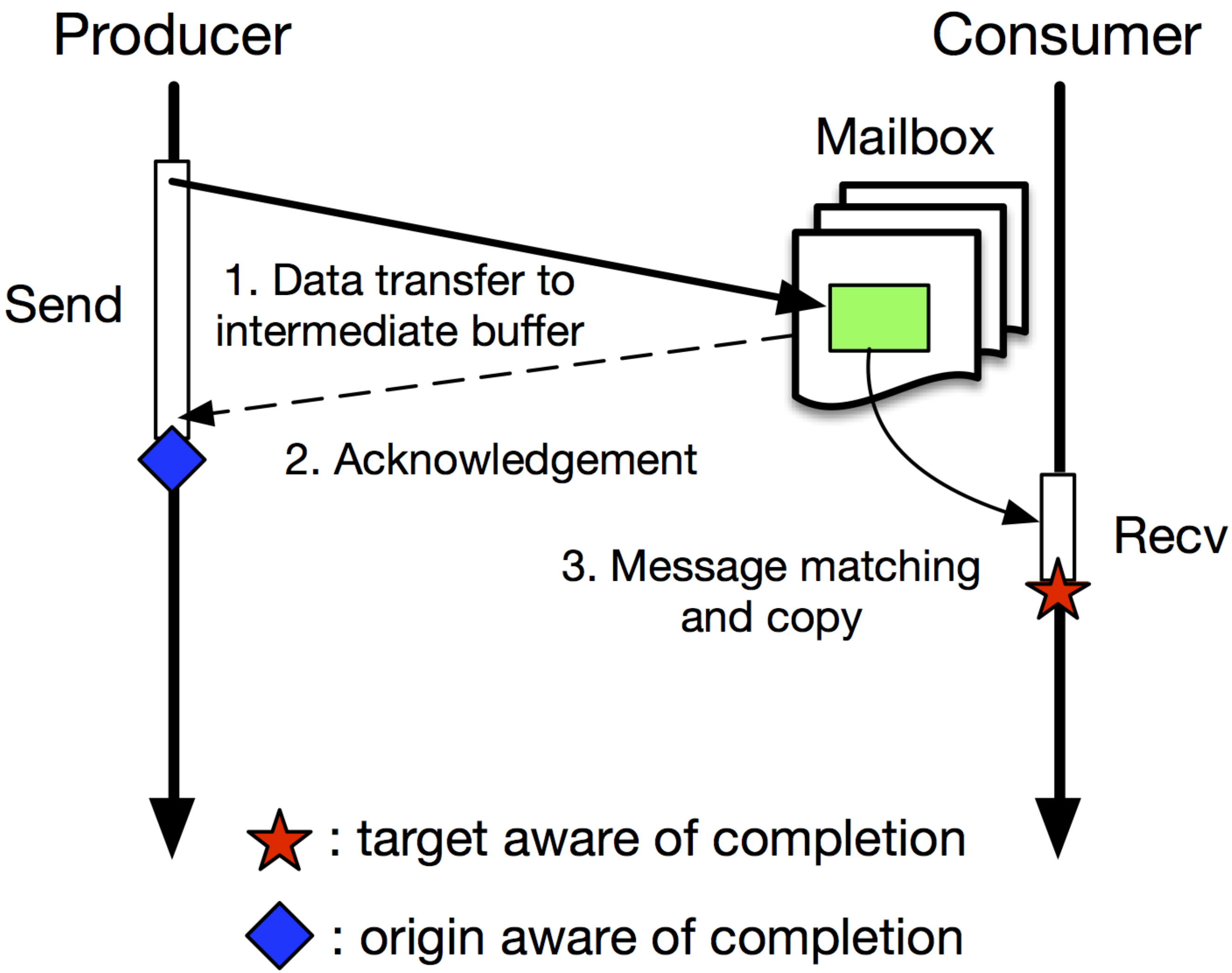
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Eager



[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

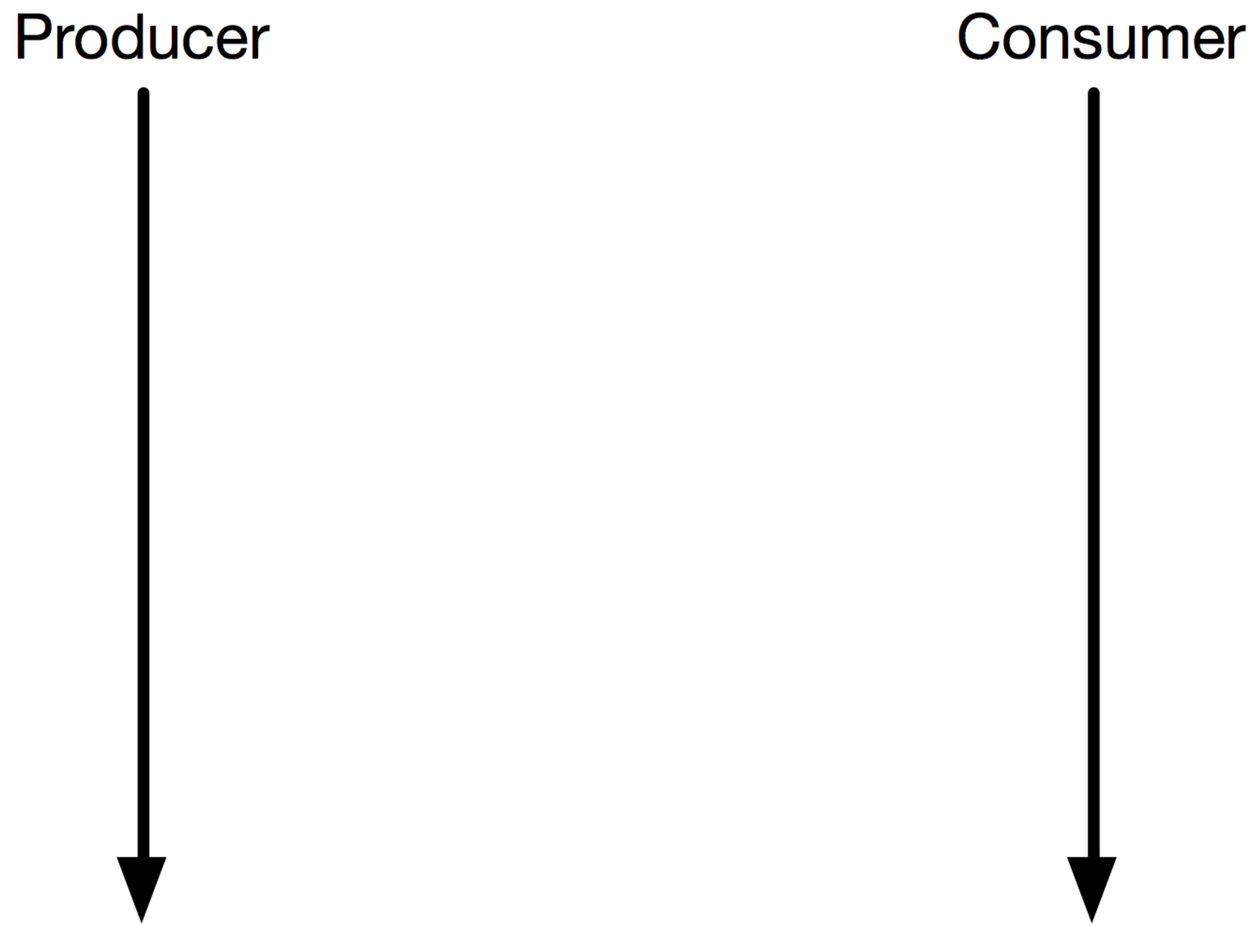
MPI-1 Message Passing – Simple Eager



Critical path: 1 latency + 1 copy

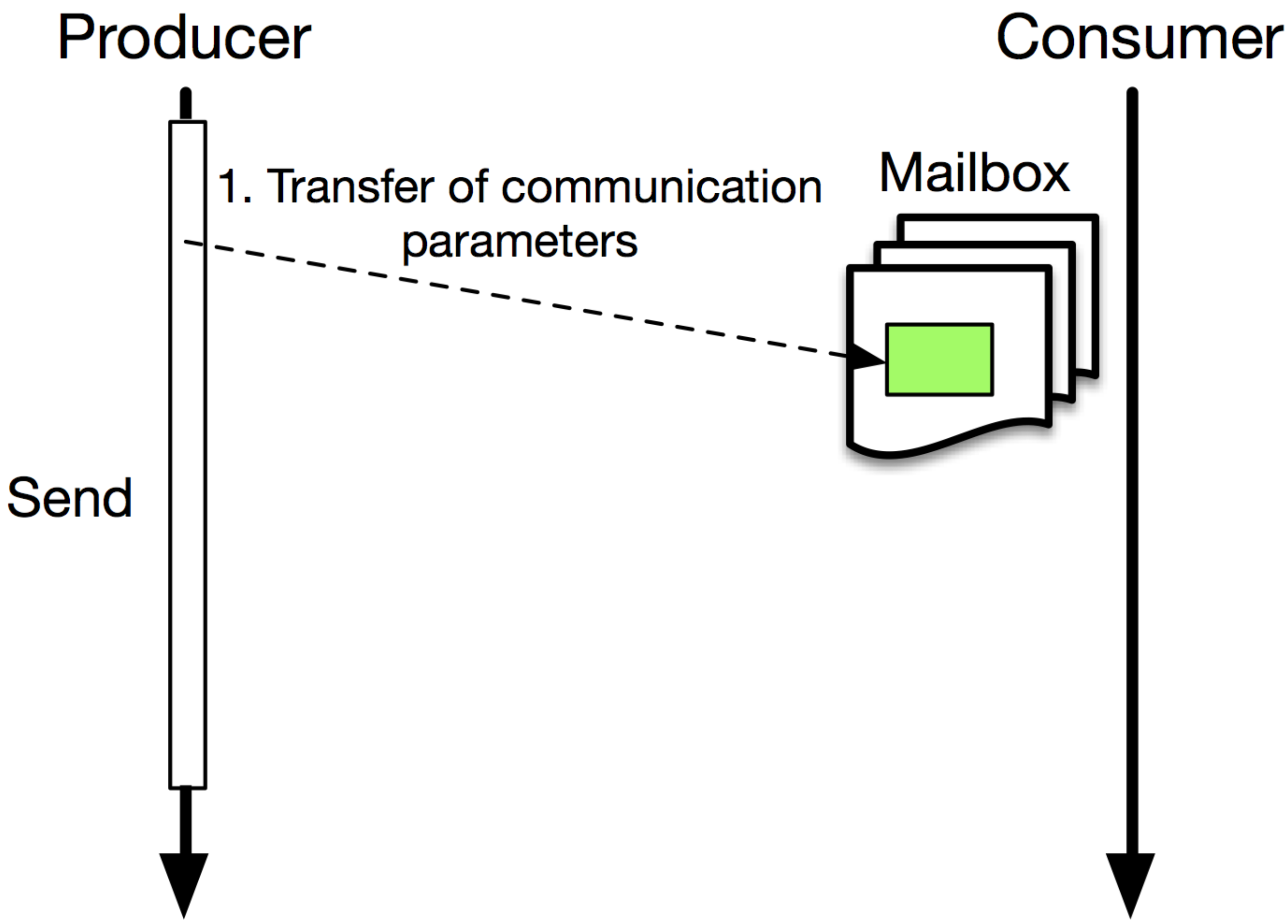
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Rendezvous



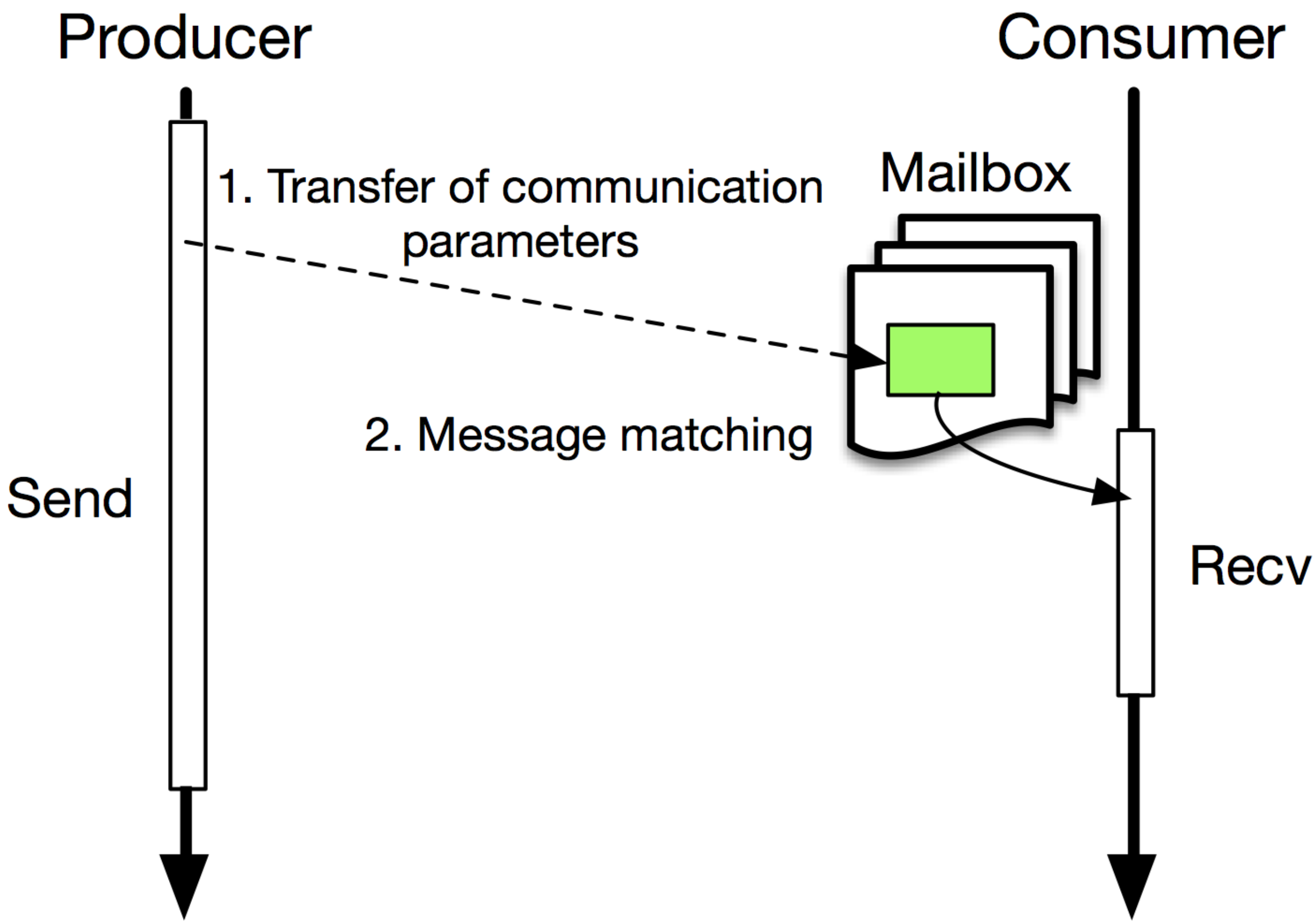
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Rendezvous



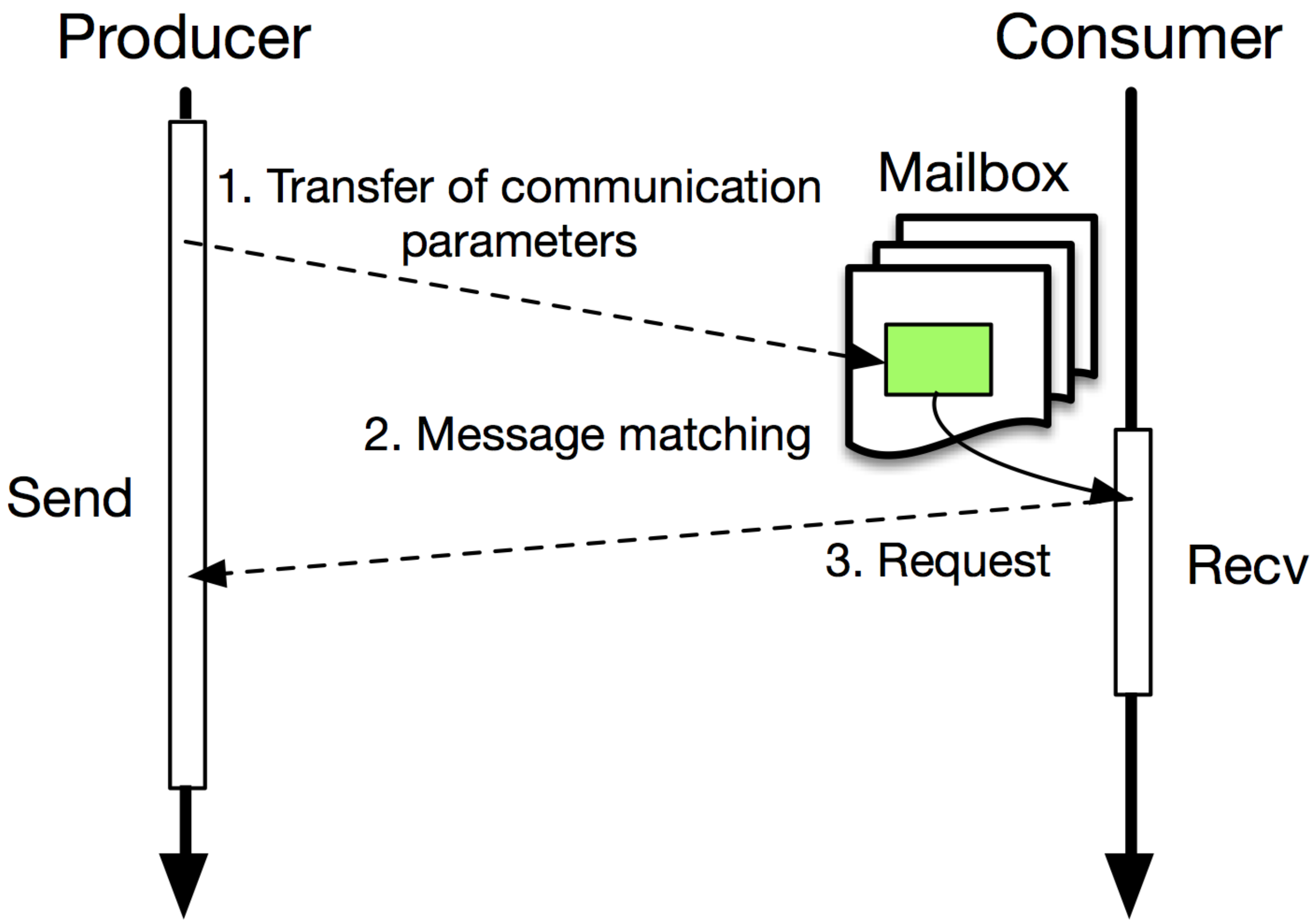
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Rendezvous



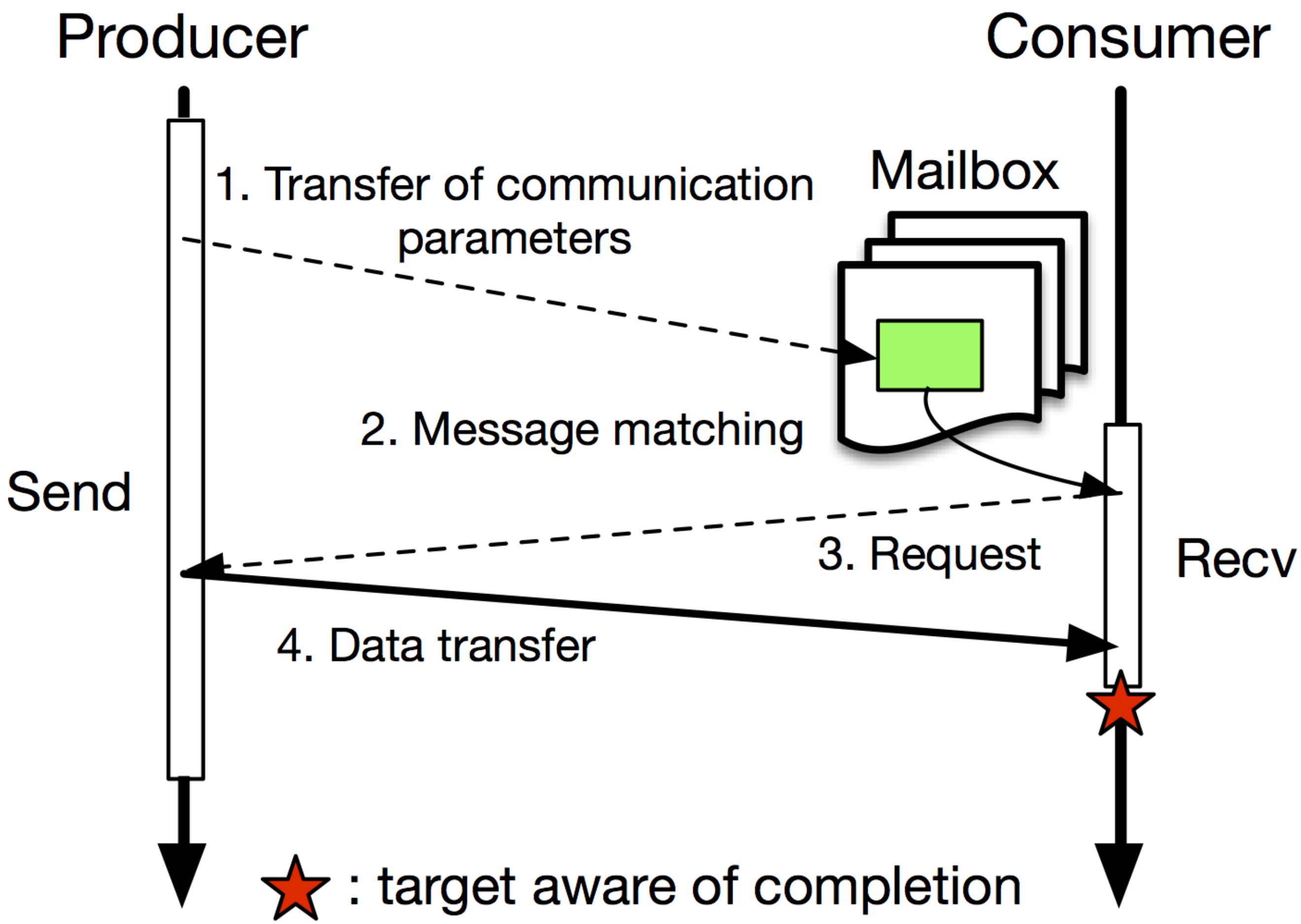
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Rendezvous



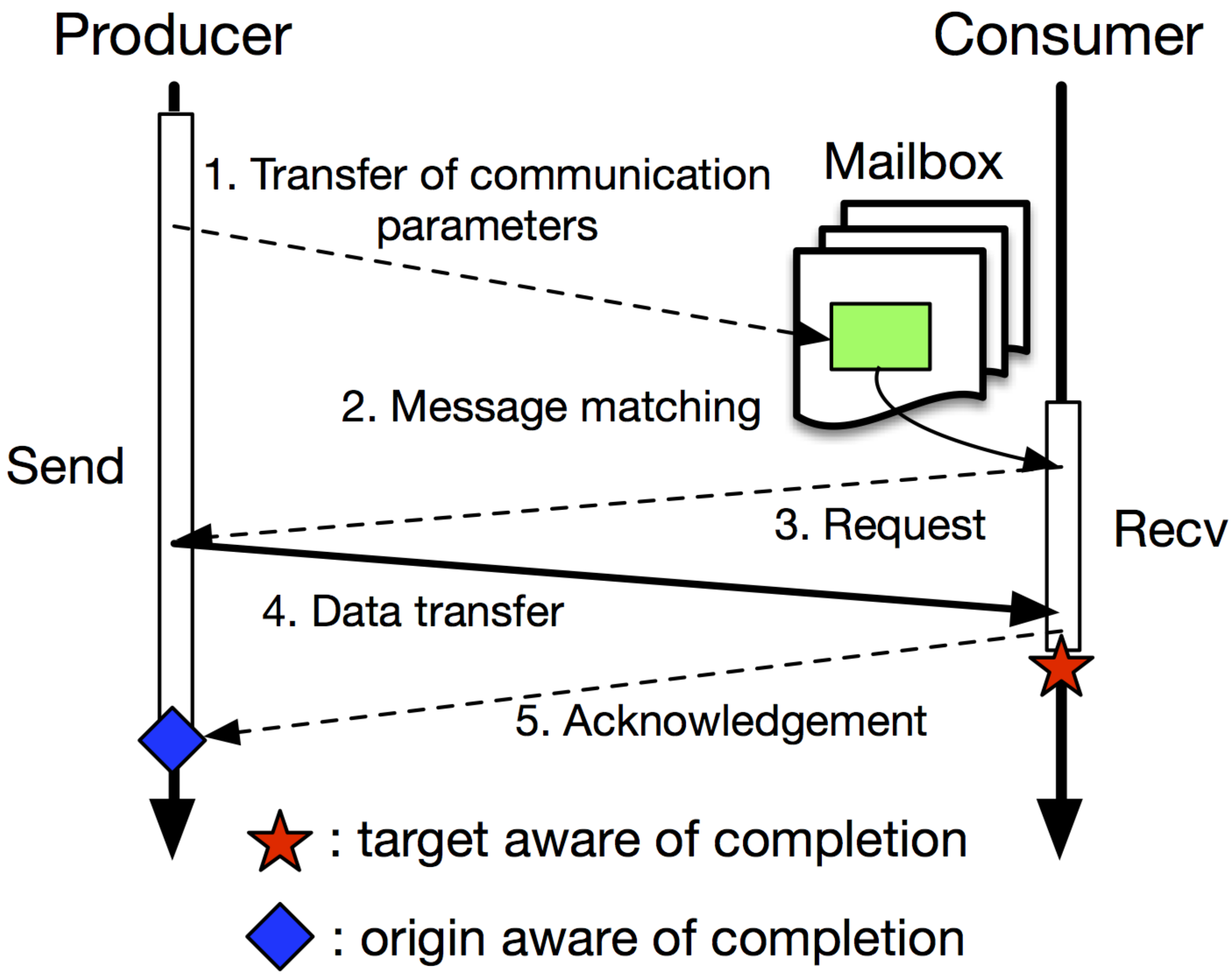
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

MPI-1 Message Passing – Simple Rendezvous



[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

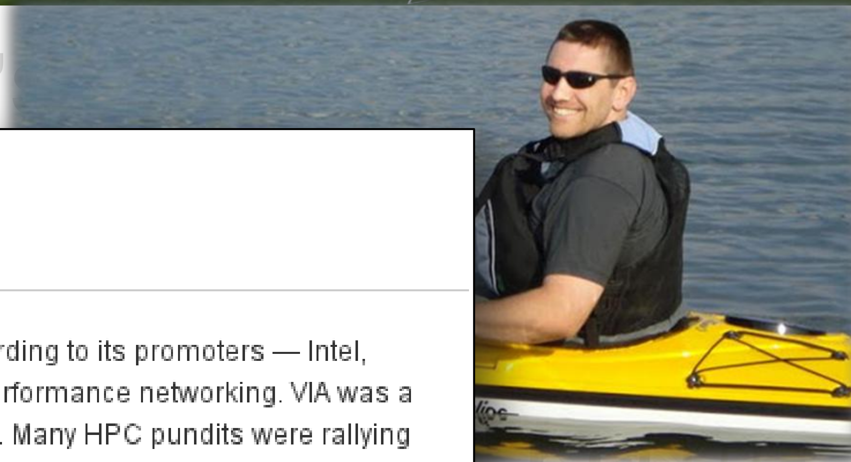
MPI-1 Message Passing – Simple Rendezvous

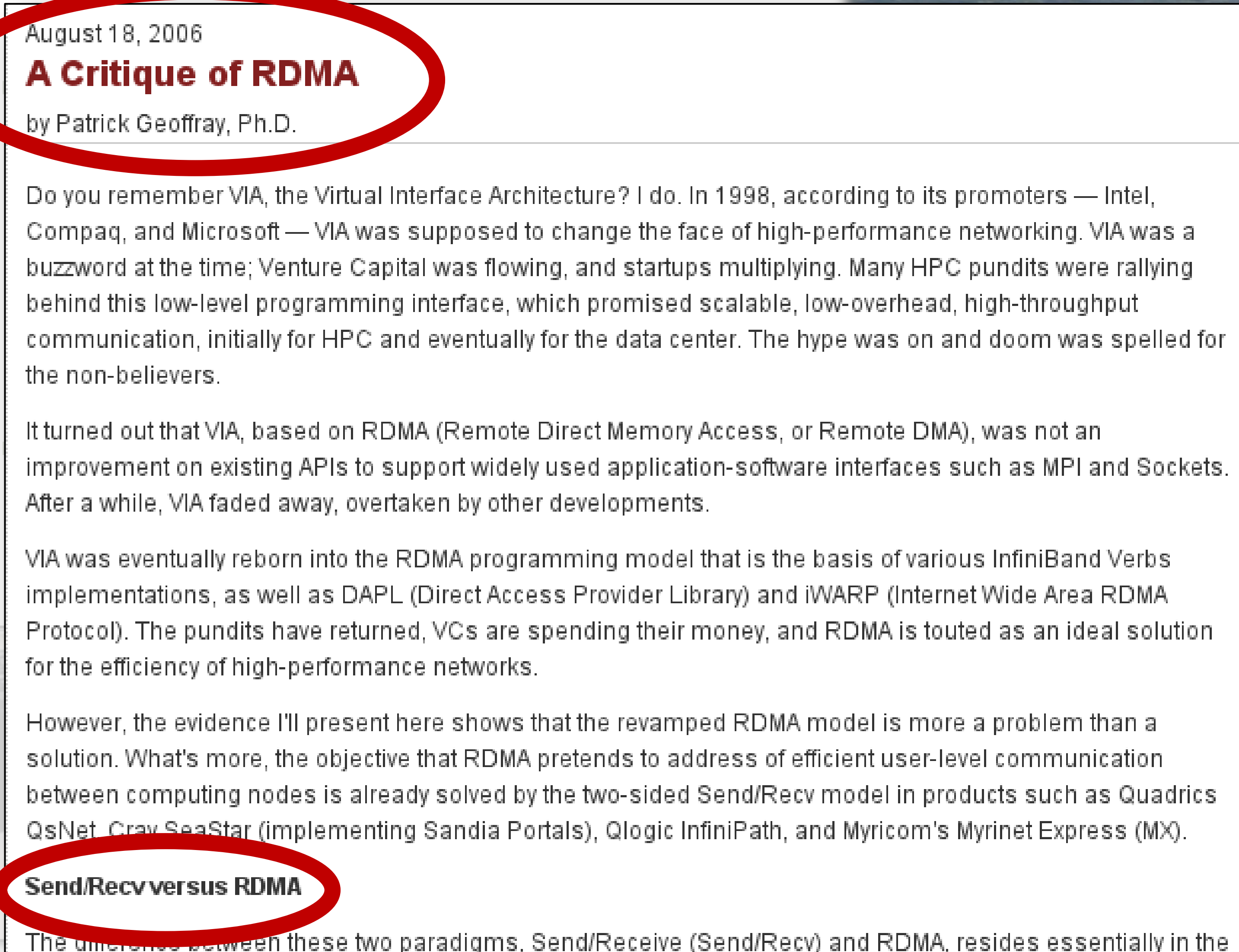


Critical path: 3 latencies

[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

COMMUNICATION IN TODAY'





August 18, 2006

A Critique of RDMA

by Patrick Geoffray, Ph.D.

Do you remember VIA, the Virtual Interface Architecture? I do. In 1998, according to its promoters — Intel, Compaq, and Microsoft — VIA was supposed to change the face of high-performance networking. VIA was a buzzword at the time; Venture Capital was flowing, and startups multiplying. Many HPC pundits were rallying behind this low-level programming interface, which promised scalable, low-overhead, high-throughput communication, initially for HPC and eventually for the data center. The hype was on and doom was spelled for the non-believers.

It turned out that VIA, based on RDMA (Remote Direct Memory Access, or Remote DMA), was not an improvement on existing APIs to support widely used application-software interfaces such as MPI and Sockets. After a while, VIA faded away, overtaken by other developments.

VIA was eventually reborn into the RDMA programming model that is the basis of various InfiniBand Verbs implementations, as well as DAPL (Direct Access Provider Library) and iWARP (Internet Wide Area RDMA Protocol). The pundits have returned, VCs are spending their money, and RDMA is touted as an ideal solution for the efficiency of high-performance networks.

However, the evidence I'll present here shows that the revamped RDMA model is more a problem than a solution. What's more, the objective that RDMA pretends to address of efficient user-level communication between computing nodes is already solved by the two-sided Send/Recv model in products such as Quadrics QsNet, Cray SeaStar (implementing Sandia Portals), Qlogic InfiniPath, and Myricom's Myrinet Express (MX).

Send/Recv versus RDMA

The difference between these two paradigms, Send/Receive (Send/Recv) and RDMA, resides essentially in the

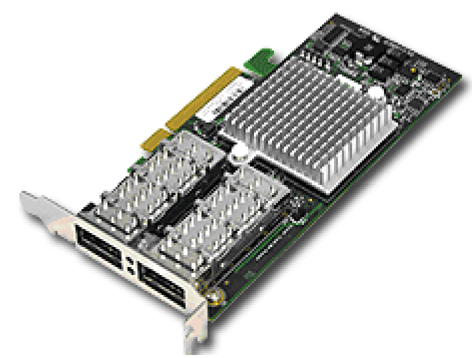
http://www.hpcwire.com/2006/08/18/a_critique_of_rdma-1/

Remote Memory Access Programming

- **Why not use these RDMA features more directly?**
  - A global address space may simplify programming
  - … and accelerate communication
  - … and there could be a widely accepted standard

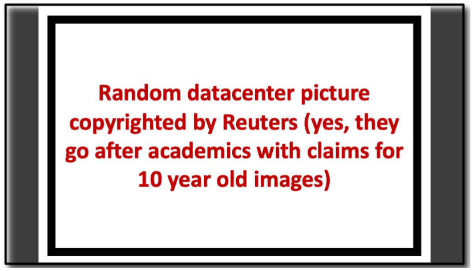
- **MPI-3 RMA ("MPI One Sided") [1] was born**
  - Just one among many others (UPC, CAF, …)
  - Designed to react to hardware trends, learn from others
  - Direct (hardware-supported) remote access
  - New way of thinking for programmers

"Traditionally, HPC programming models are following hardware developments" (IPDPS'15)

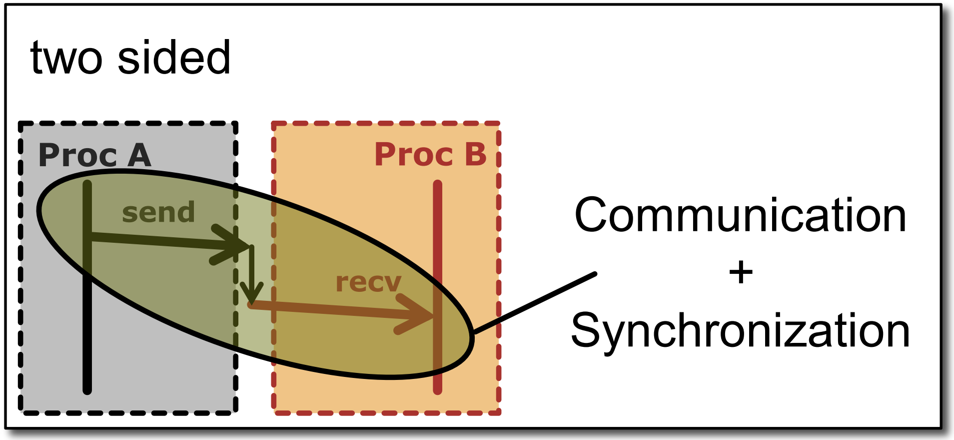


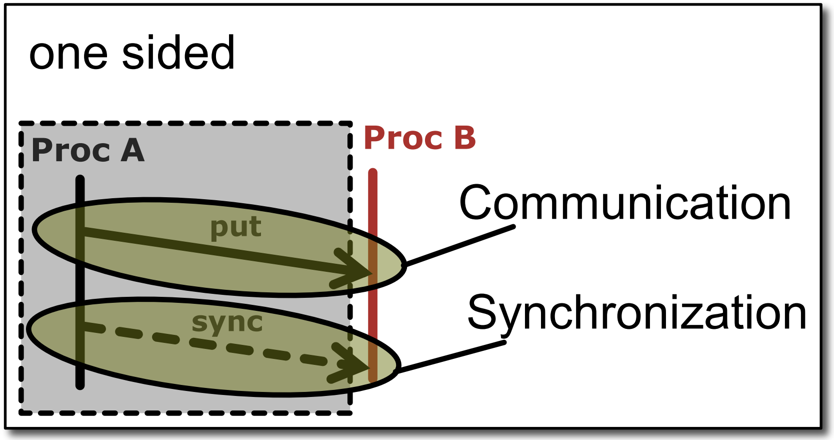
[1] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3 RMA SUMMARY



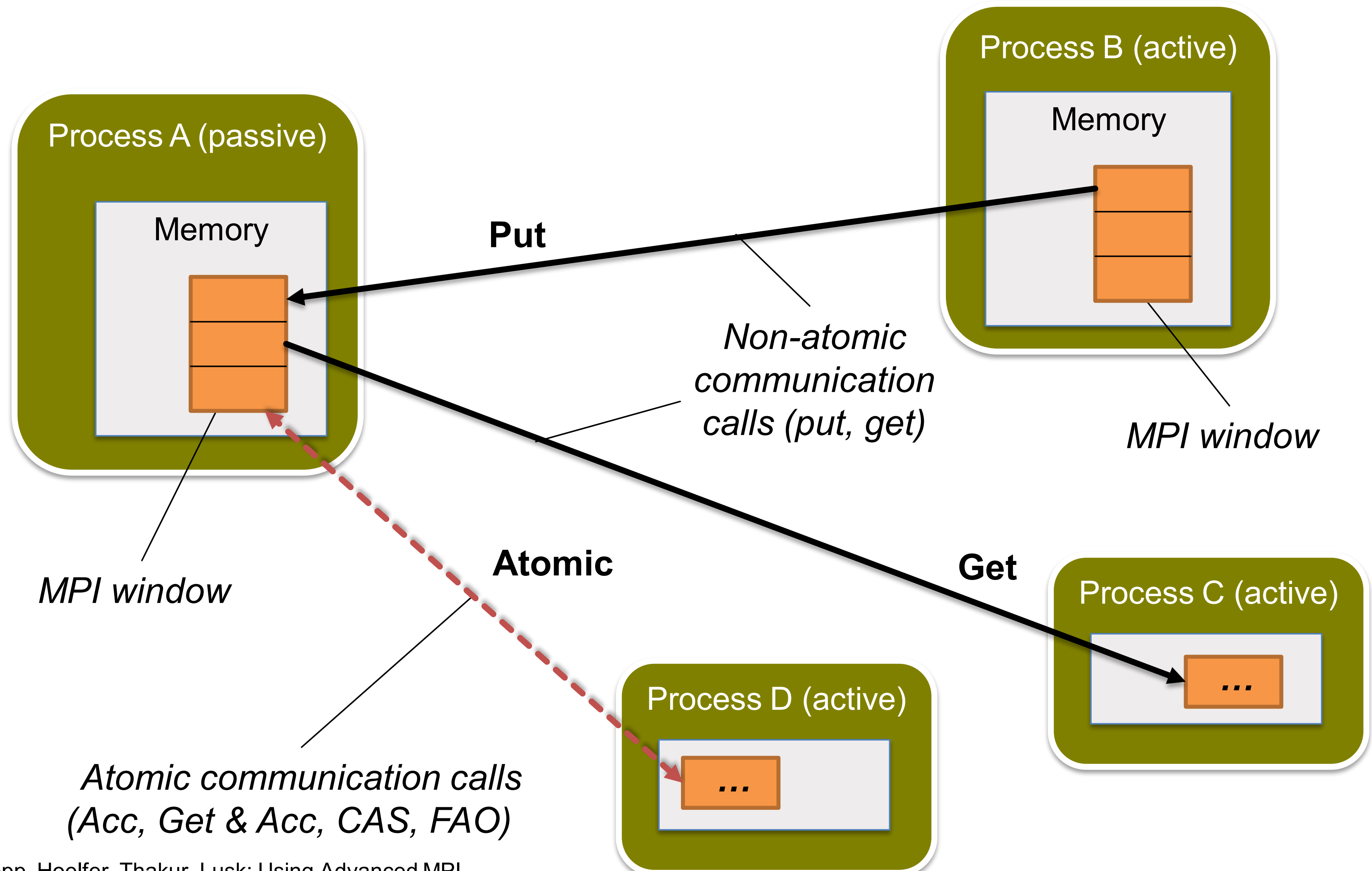
- **MPI-3 updates RMA ("MPI One Sided")**
  - Significant change from MPI-2
- **Communication is „one sided" (no involvement of destination)**
  - Utilize direct memory access
- **RMA decouples communication & synchronization**
  - Fundamentally different from message passing





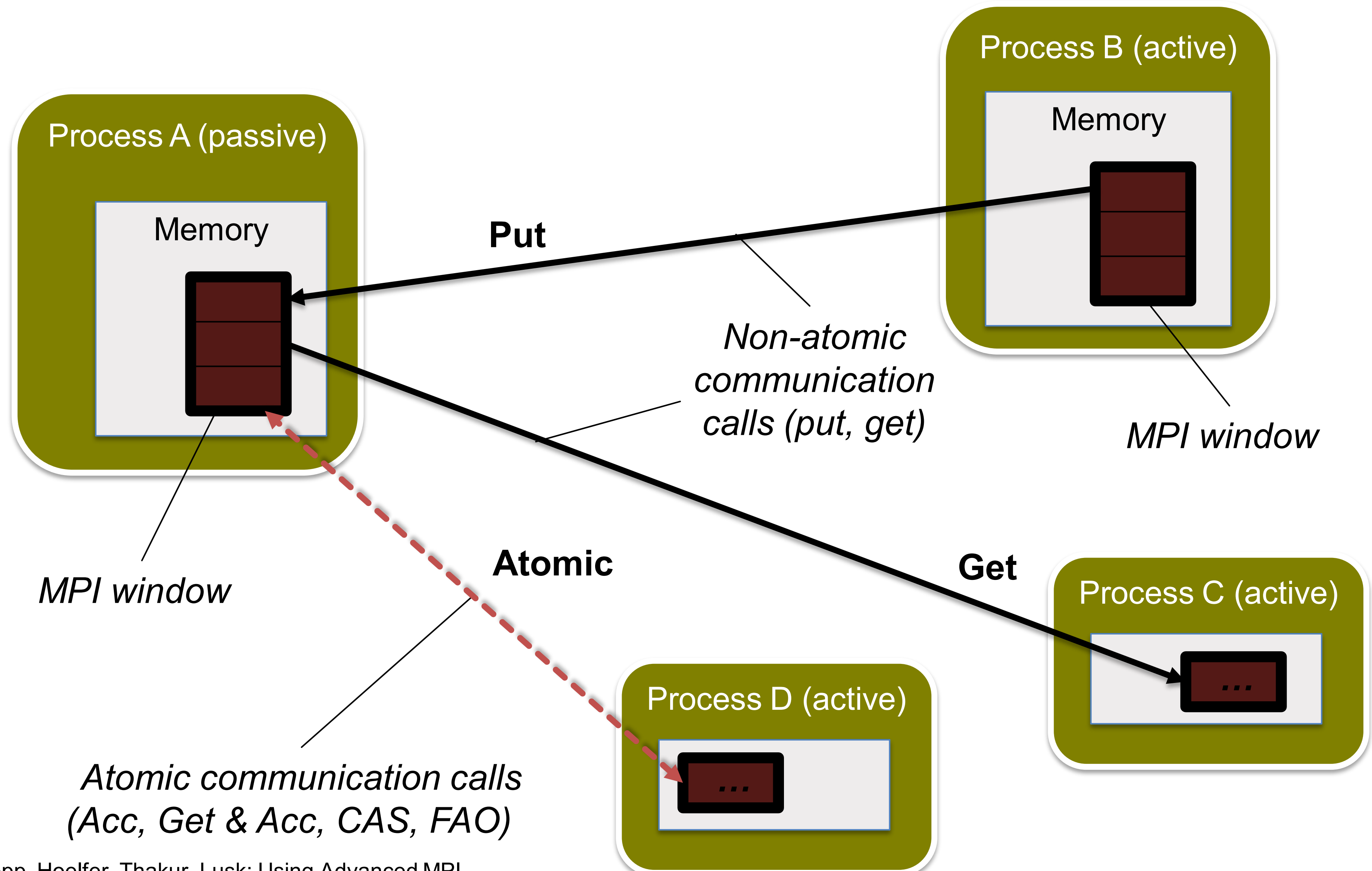
[1] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3 RMA Communication Overview



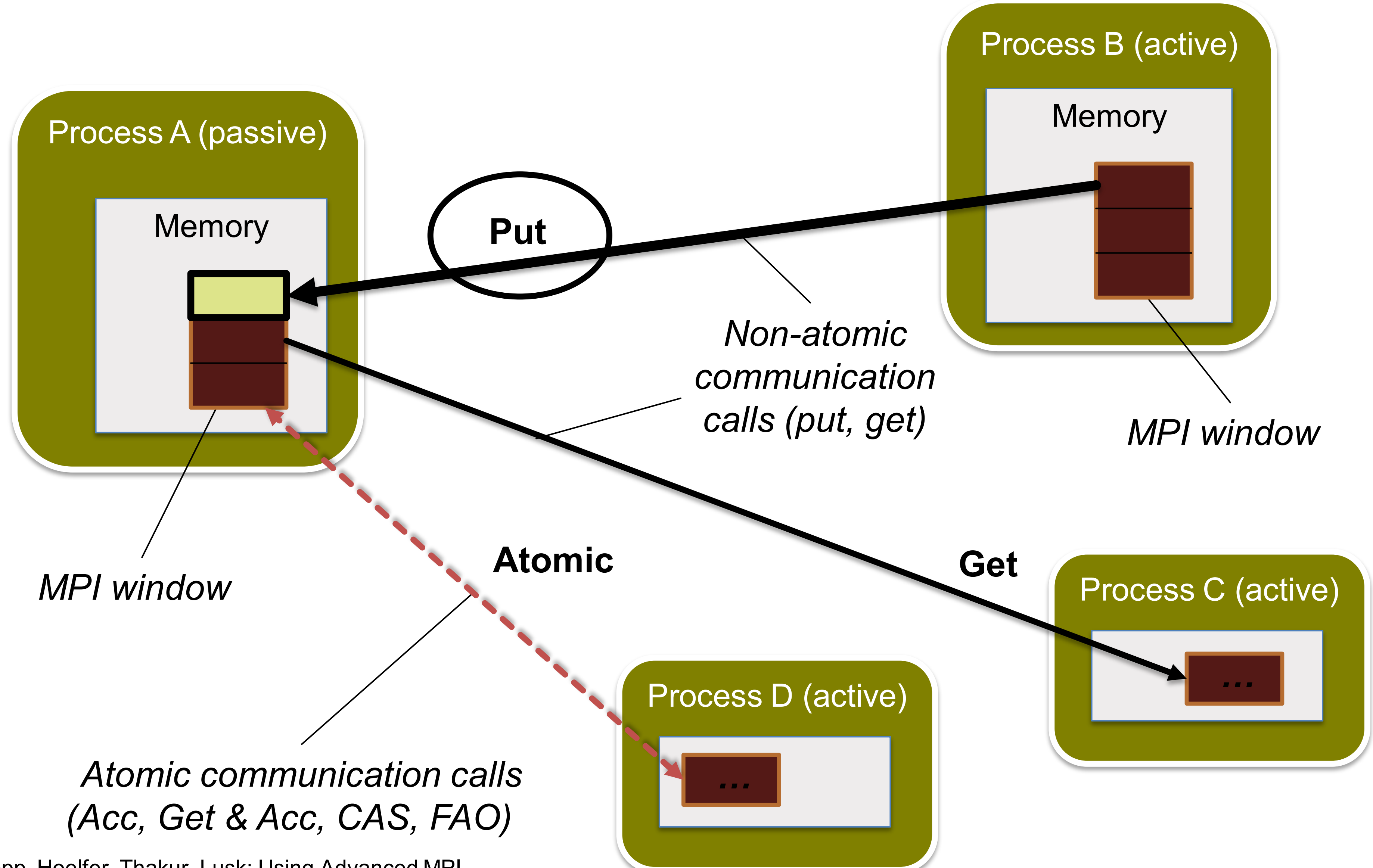
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



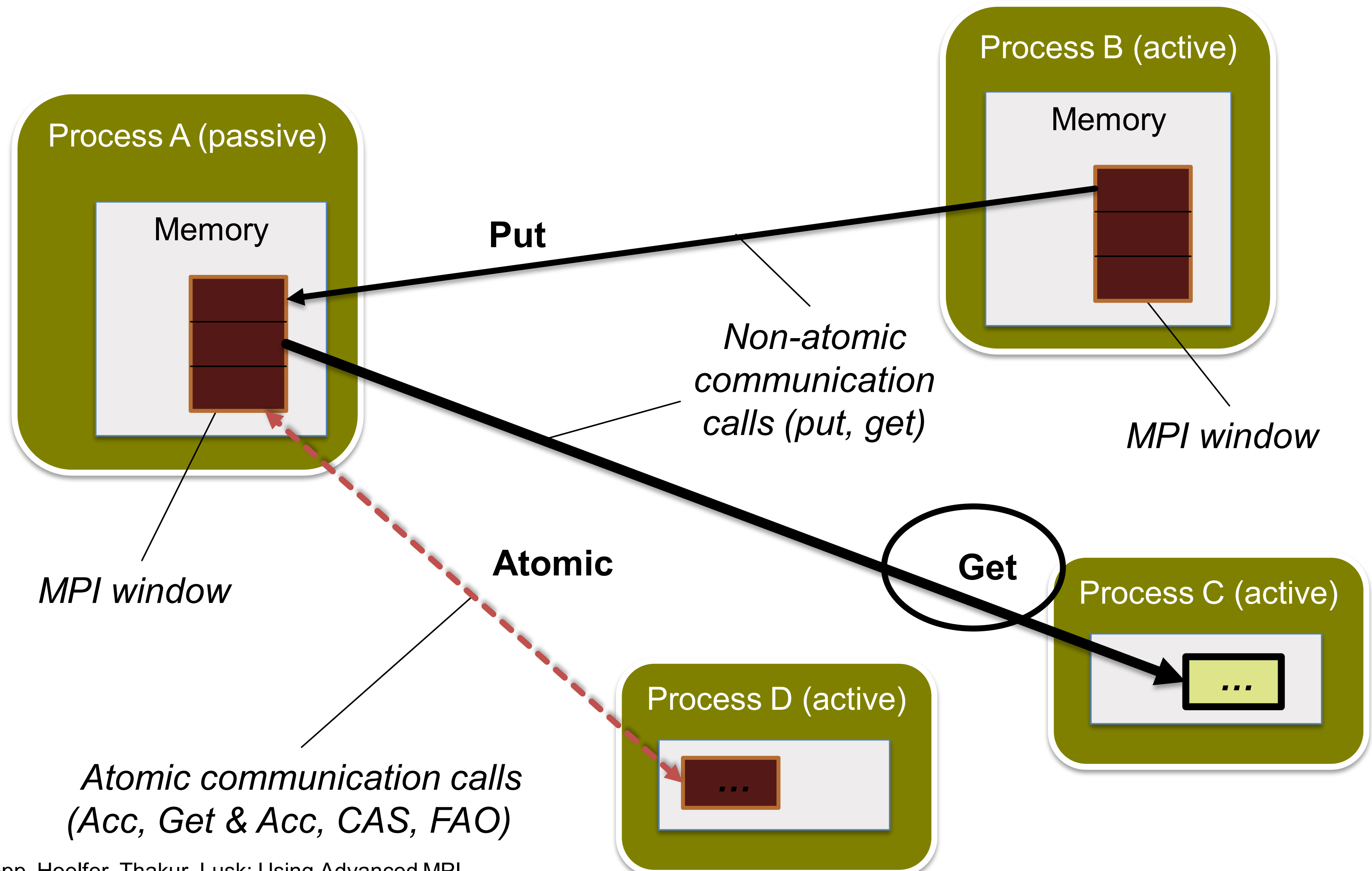
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



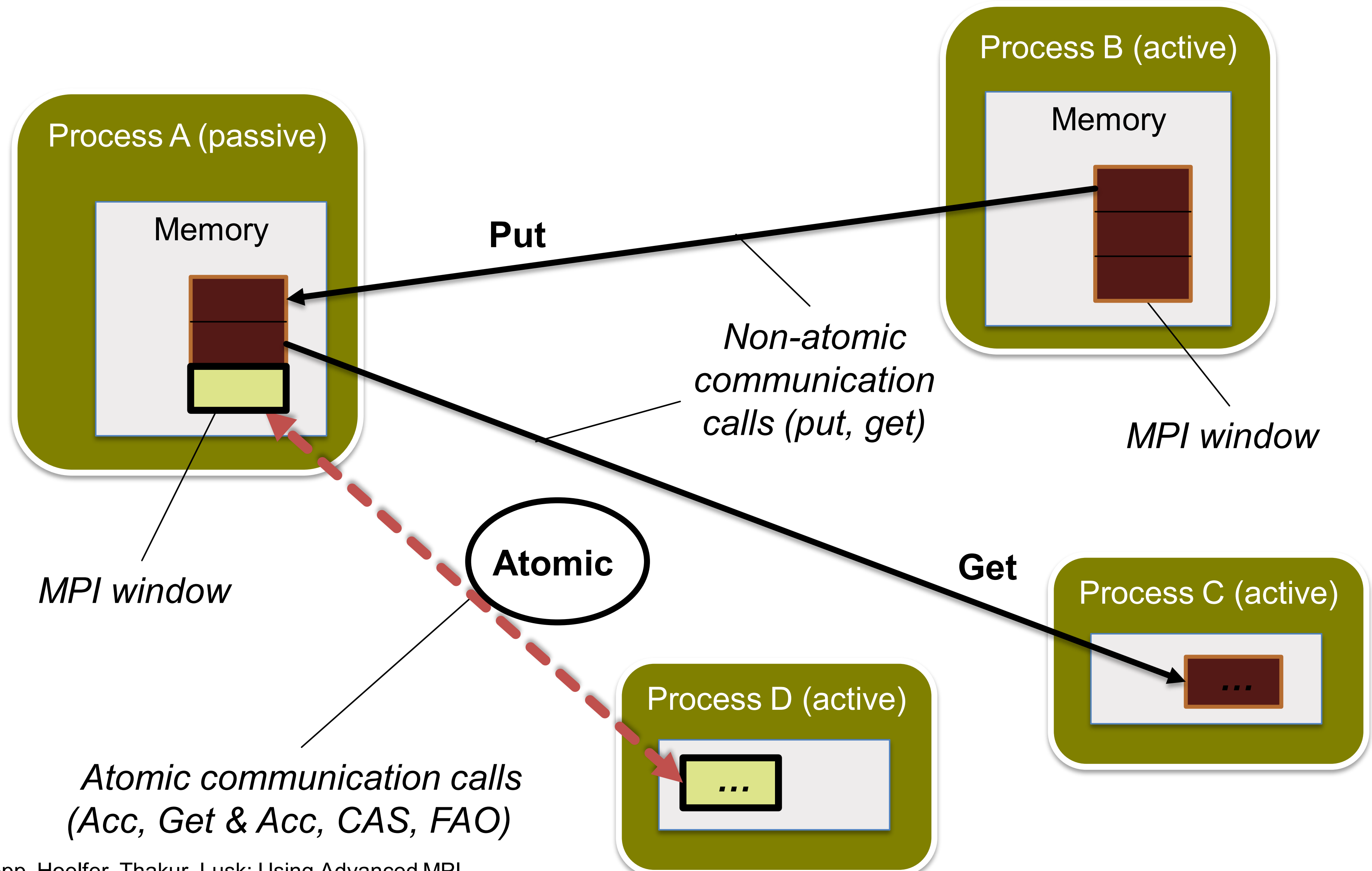
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



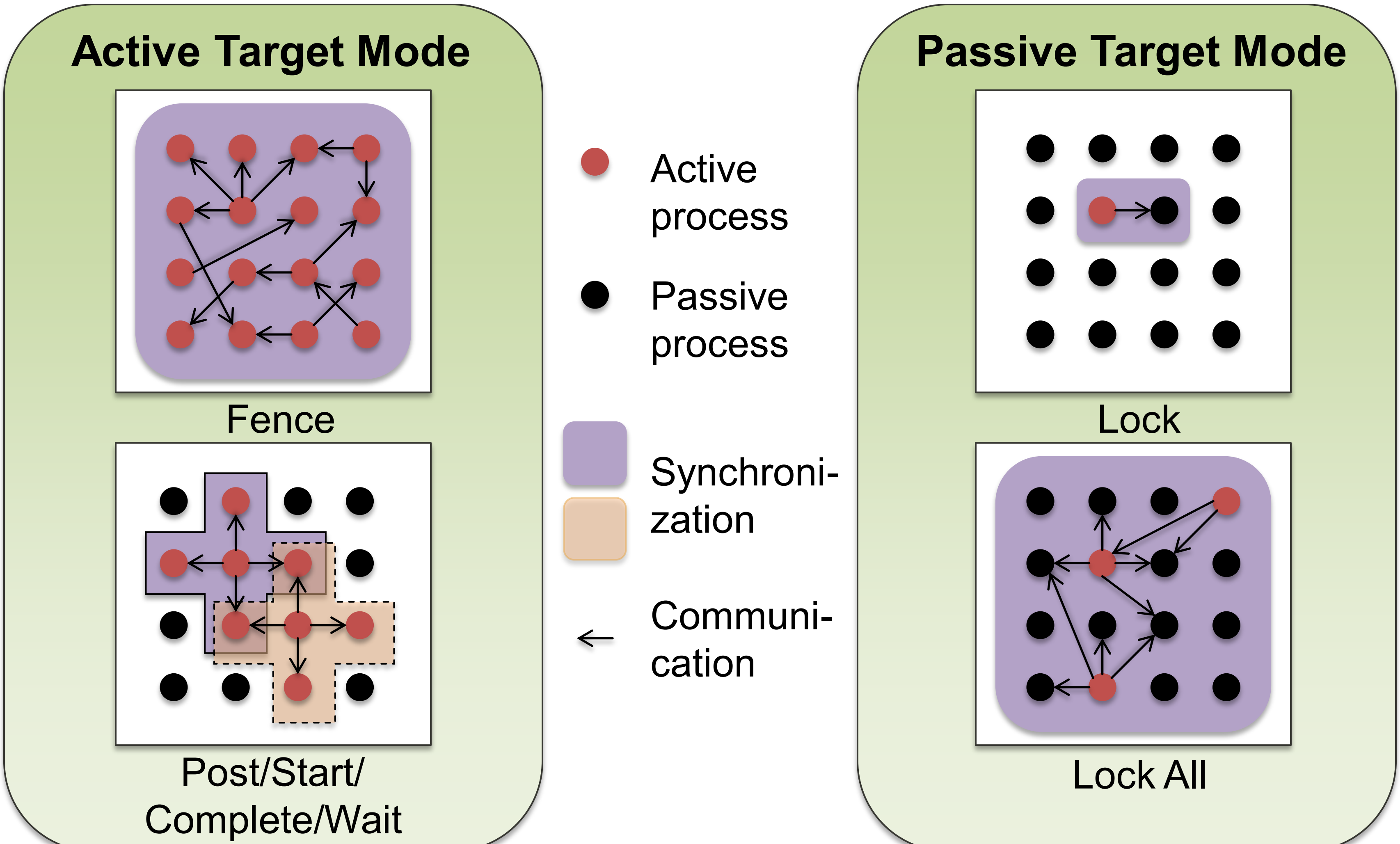
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



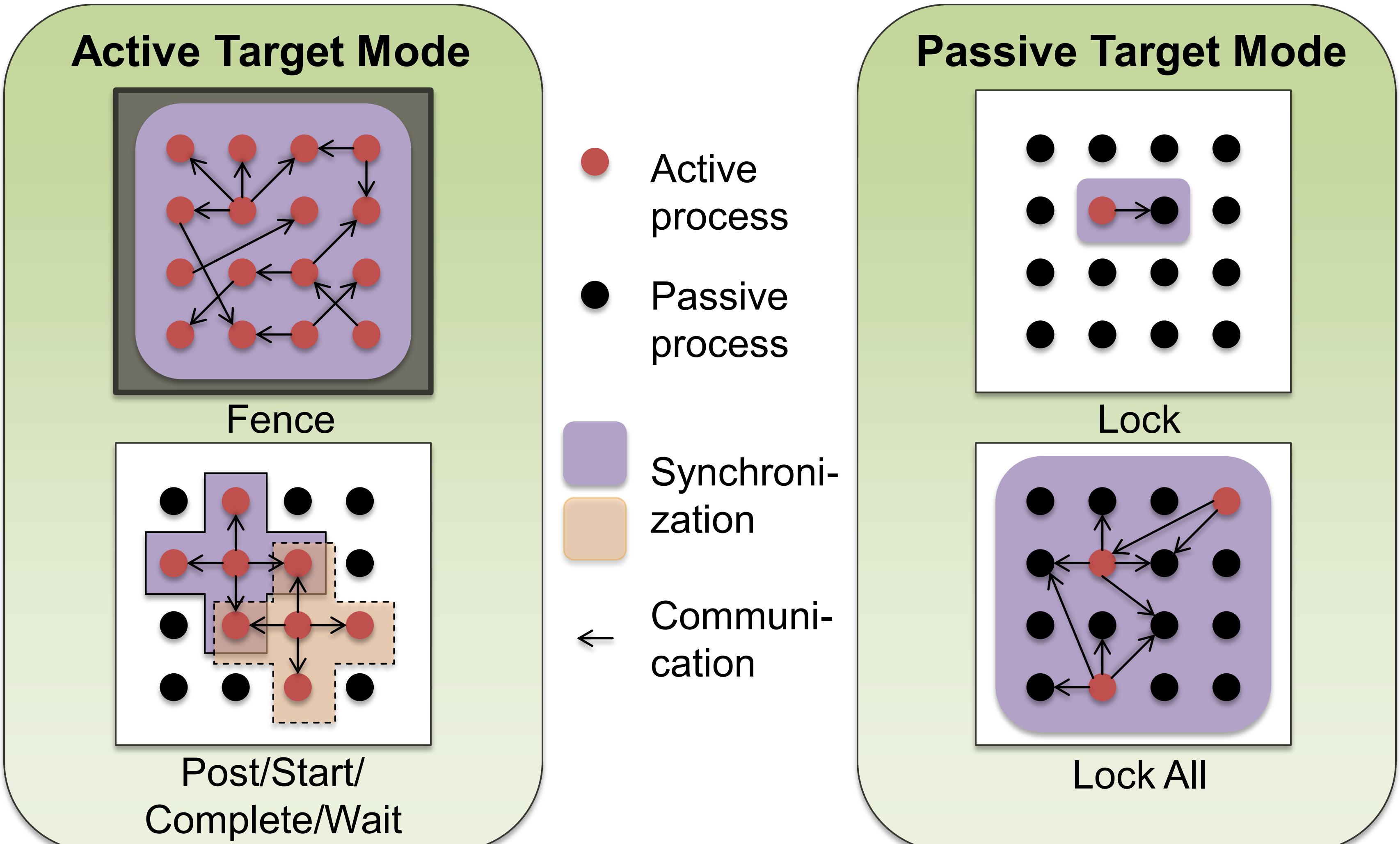
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Synchronization Overview



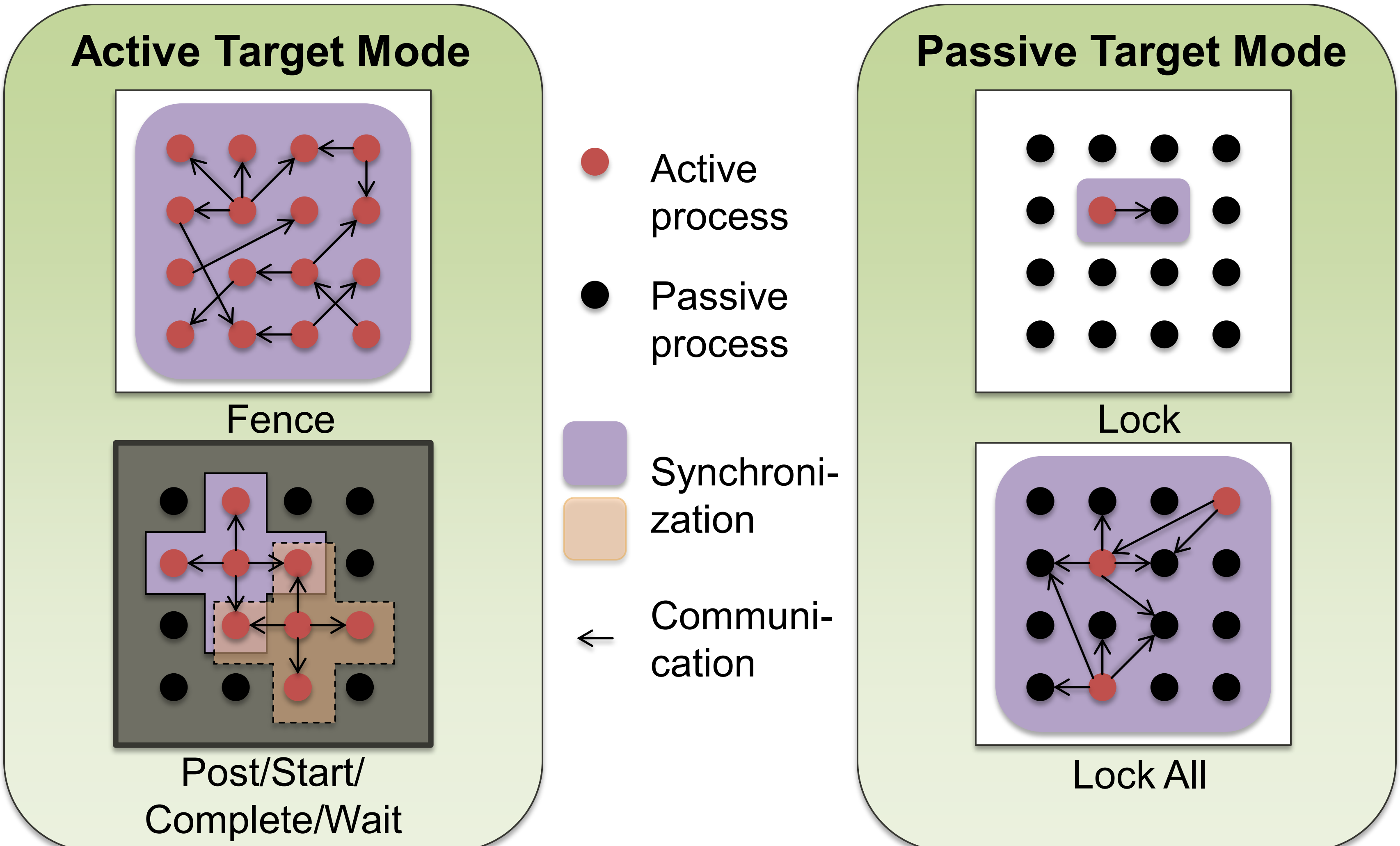
Gropp, Hoefler, Thakur, Lusk: Using Advanced MPI, MIT Press, 2014

MPI-3 RMA Synchronization Overview



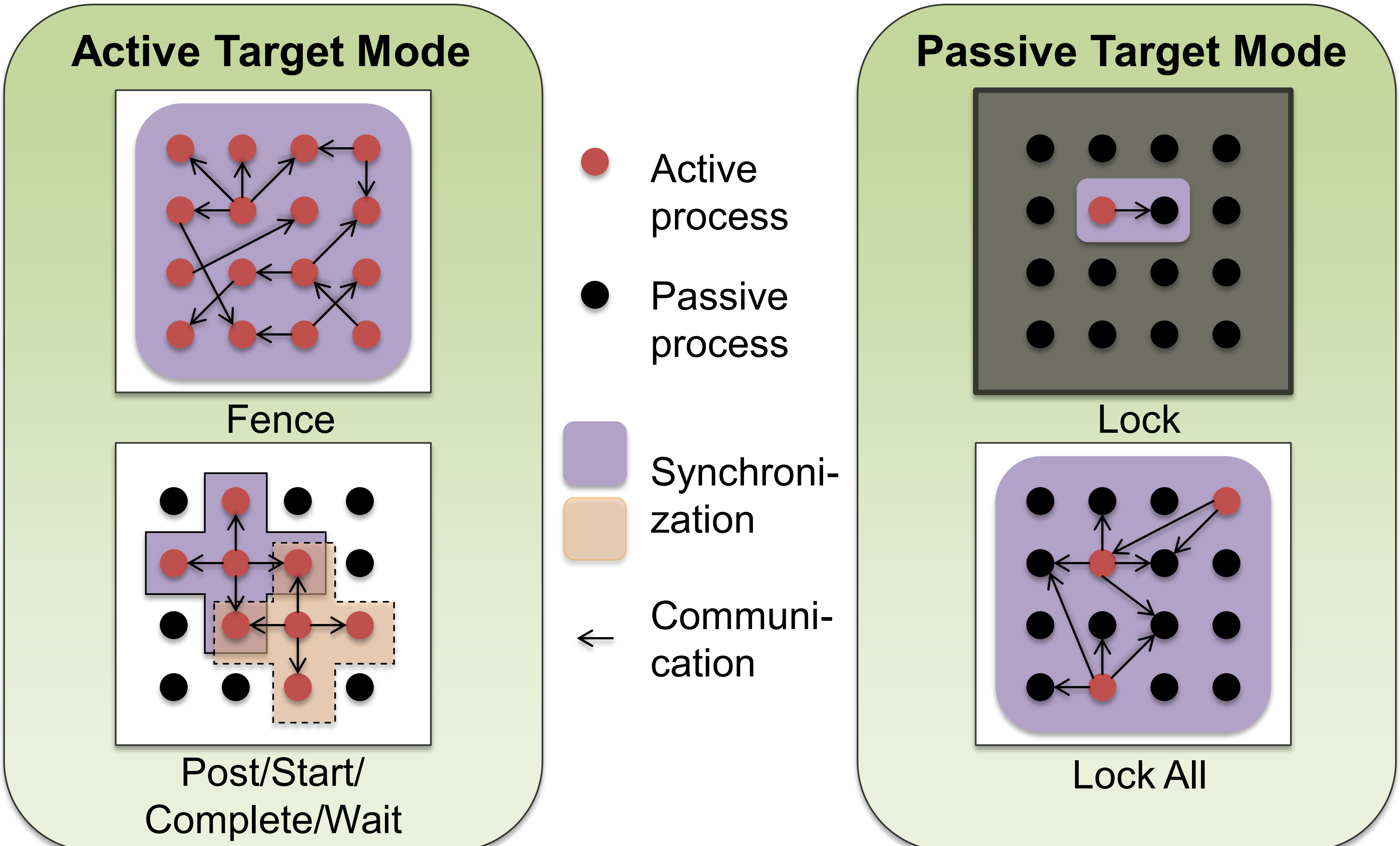
Gropp, Hoefler, Thakur, Lusk: Using Advanced MPI, MIT Press, 2014

MPI-3 RMA Synchronization Overview



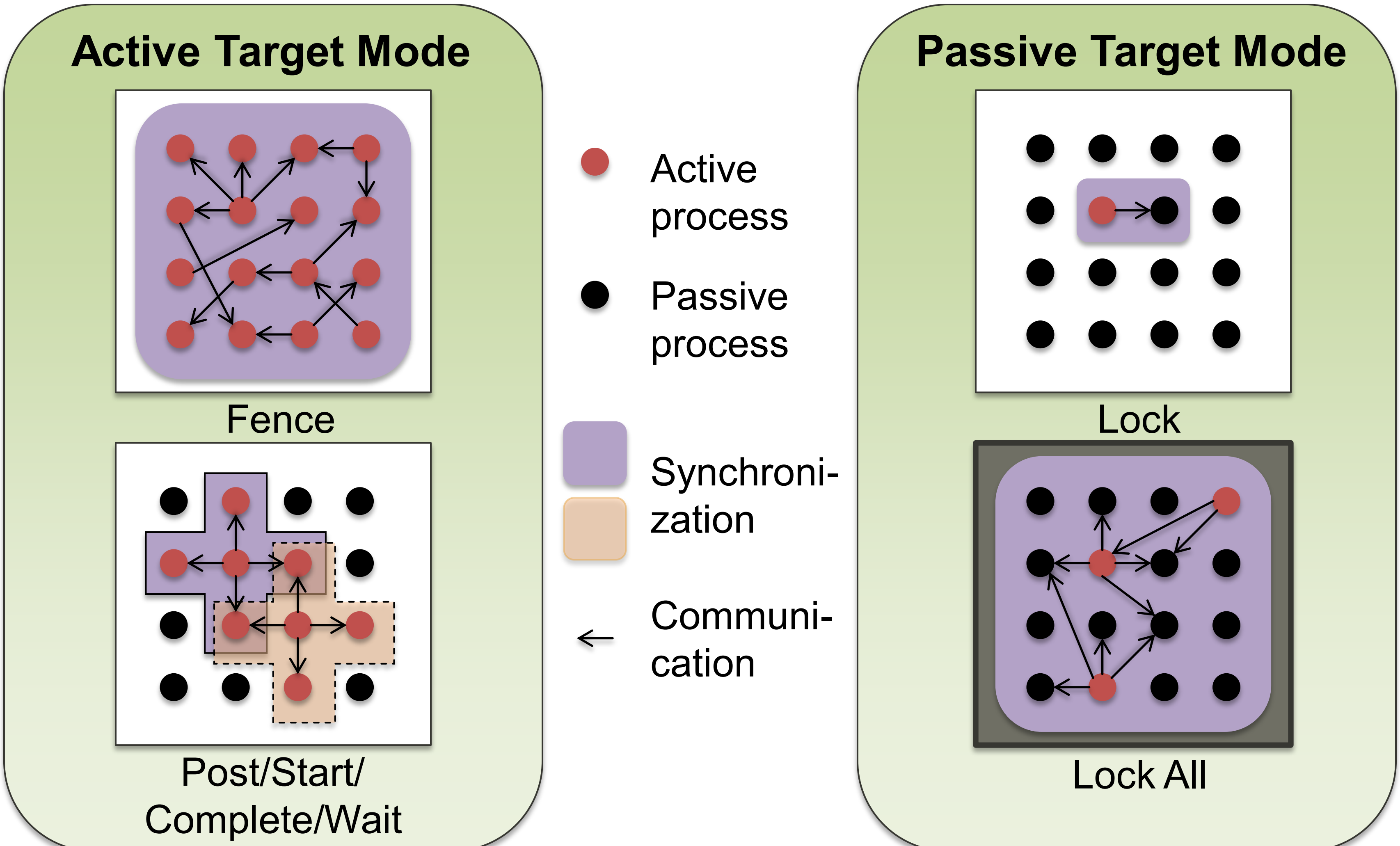
Gropp, Hoefler, Thakur, Lusk: Using Advanced MPI, MIT Press, 2014

MPI-3 RMA Synchronization Overview



Gropp, Hoefler, Thakur, Lusk: Using Advanced MPI, MIT Press, 2014

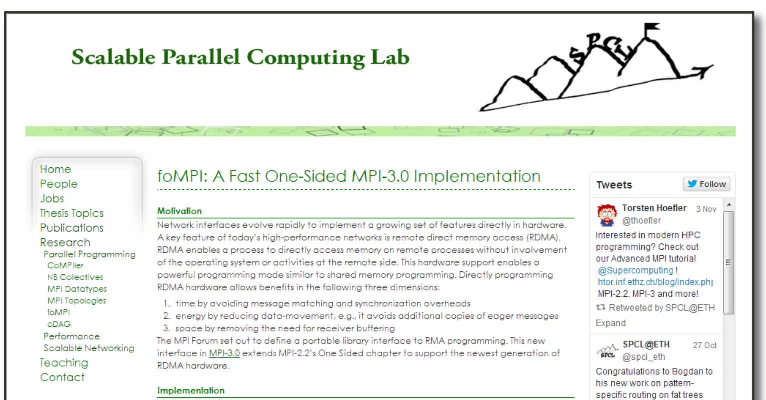
MPI-3 RMA Synchronization Overview

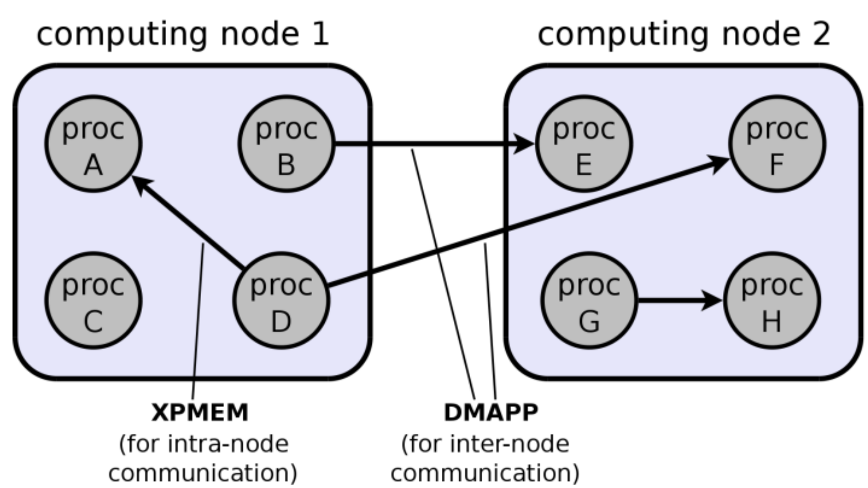


Gropp, Hoefler, Thakur, Lusk: Using Advanced MPI, MIT Press, 2014

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module

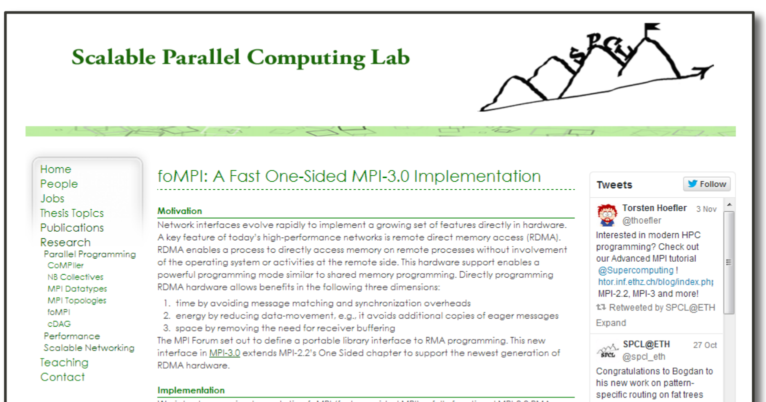


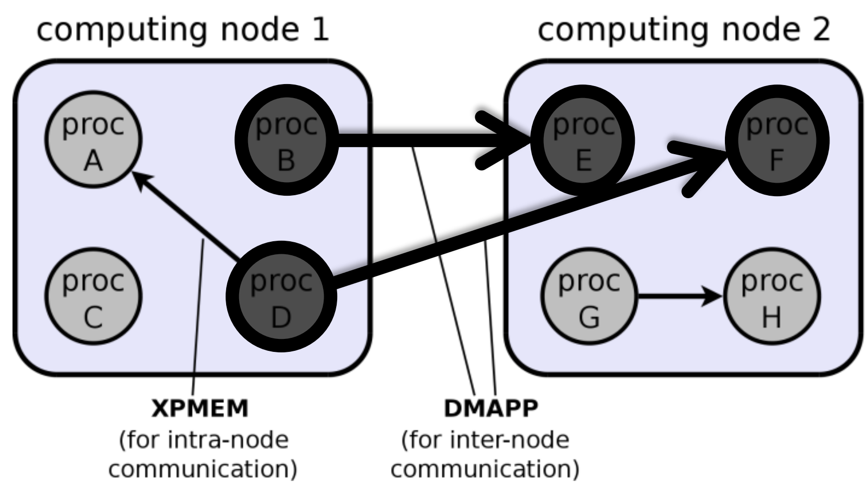


http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module



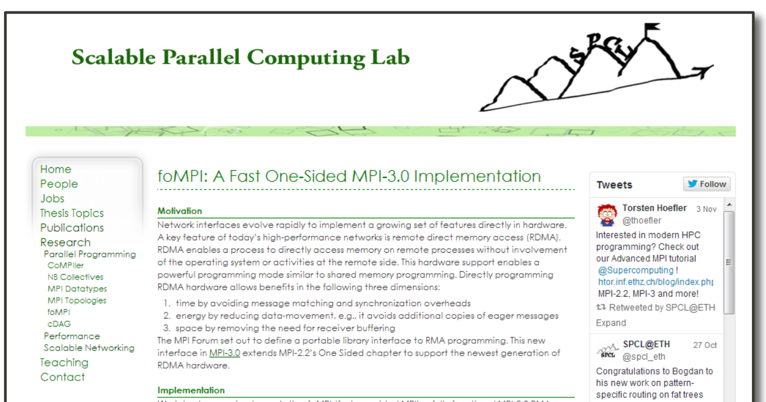


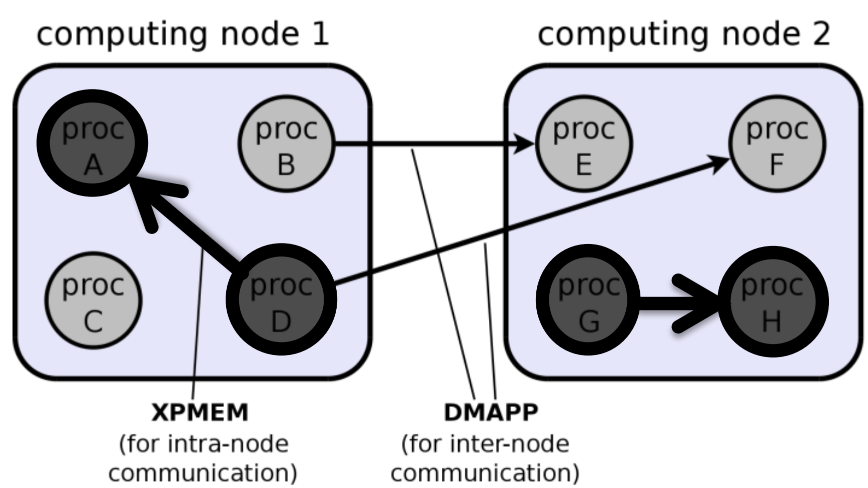
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization

- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM: a portable Linux kernel module

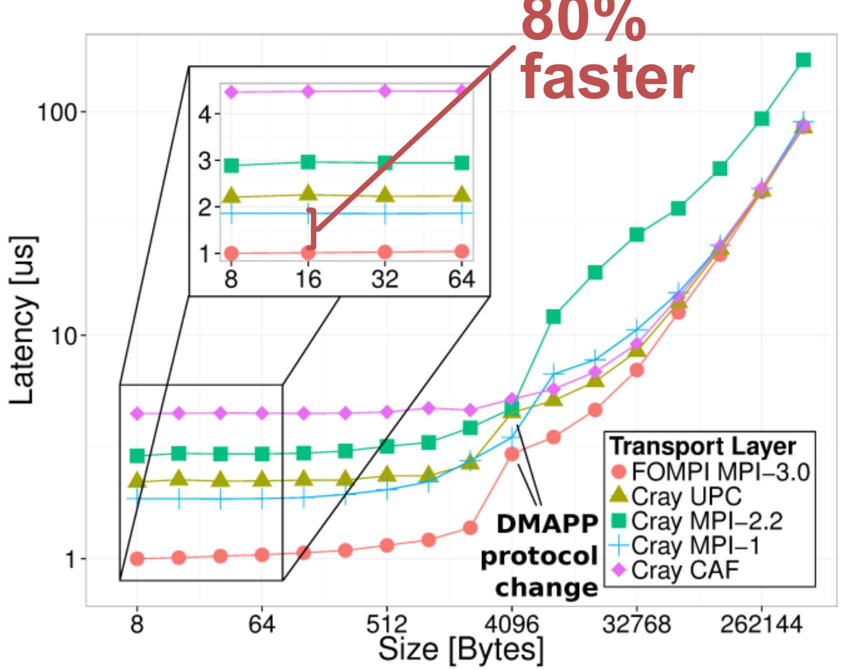




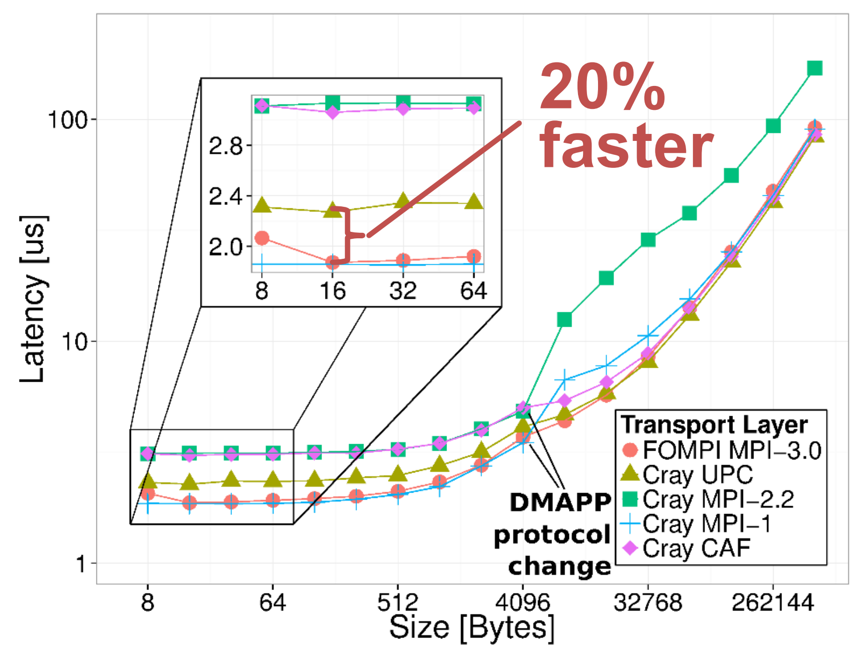
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

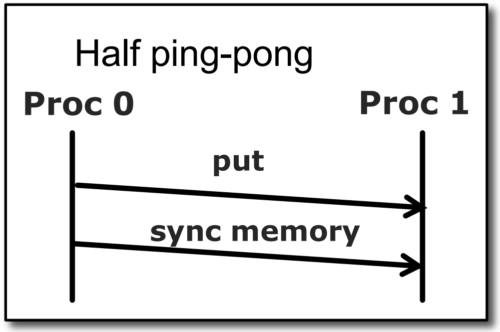
Performance Inter-node: Latency

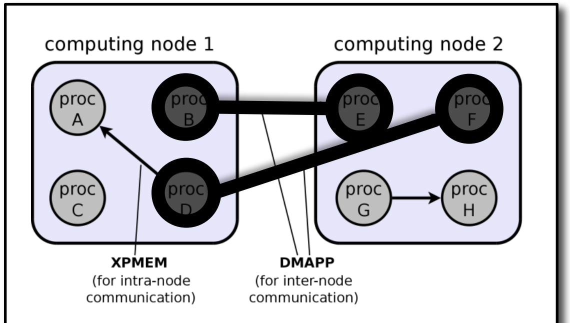
Put Inter-Node



Get Inter-Node

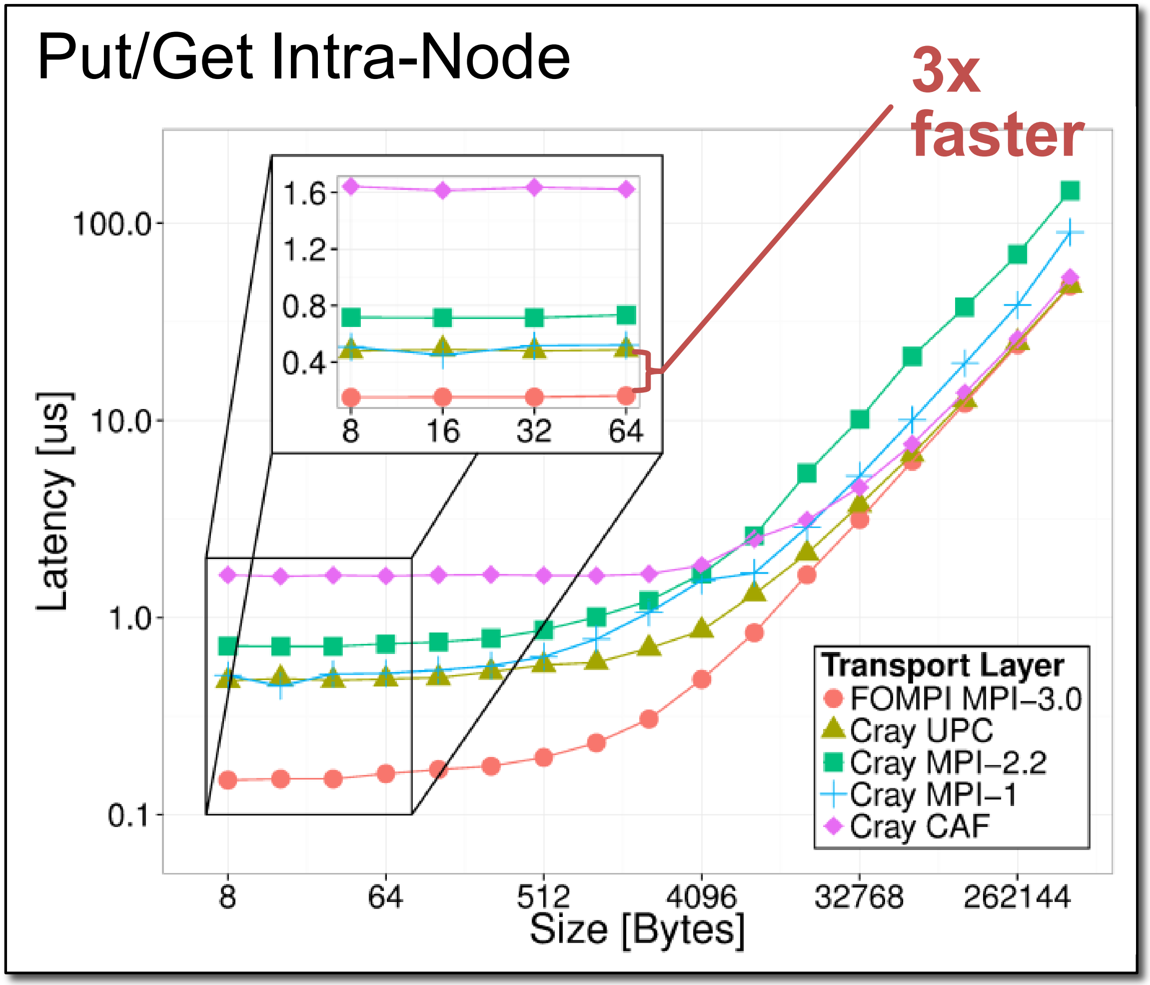


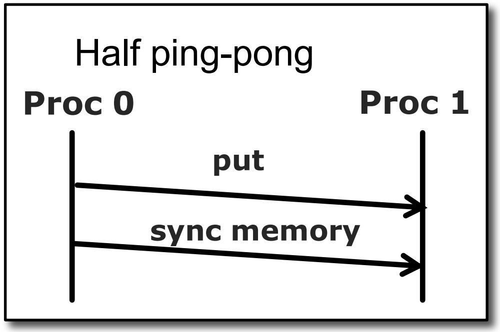


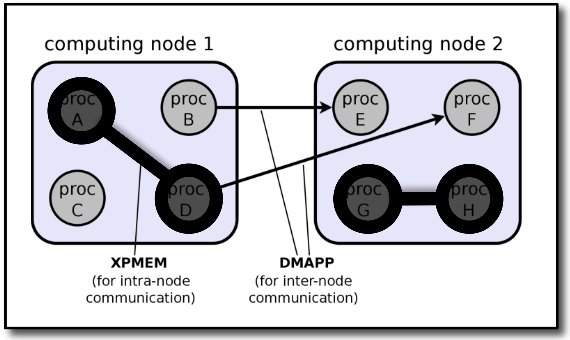


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Performance Intra-node: Latency

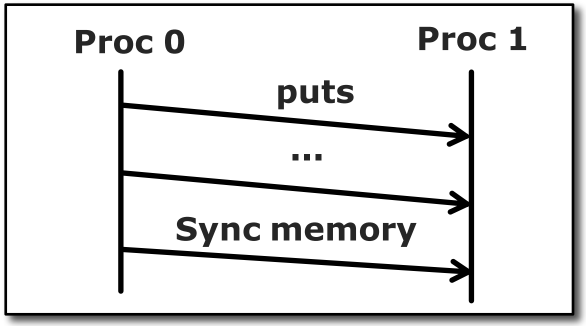


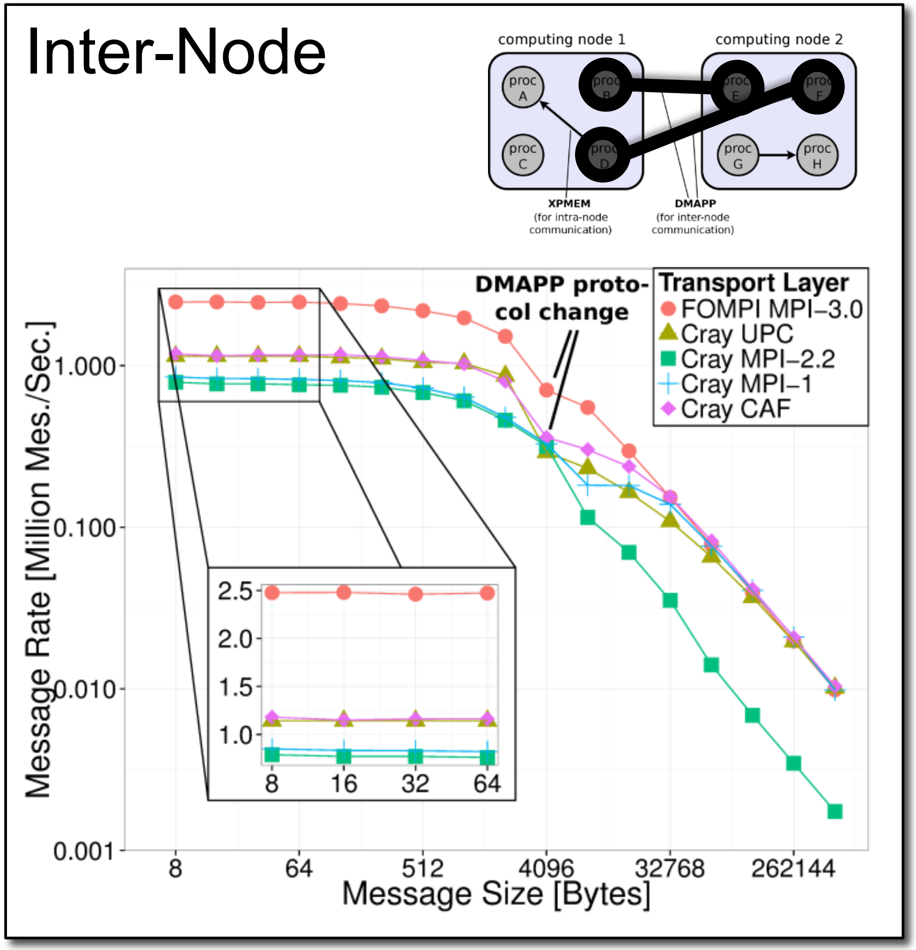


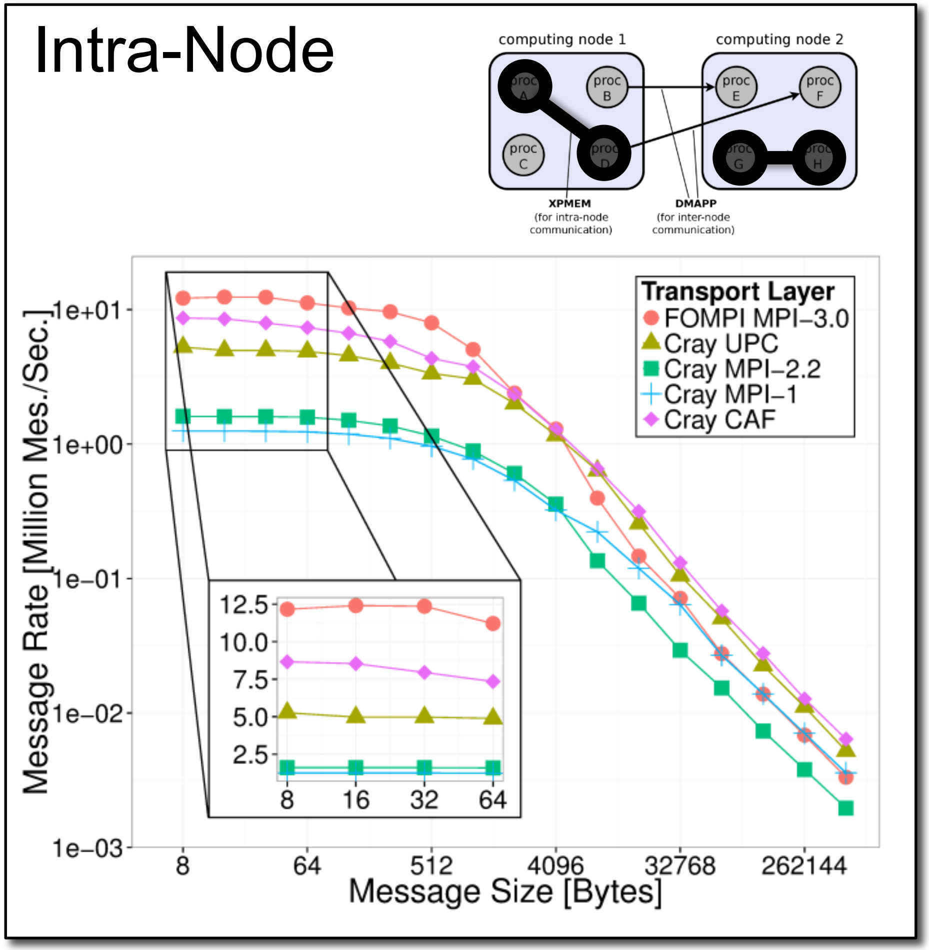


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

PERFORMANCE: MESSAGE RATE

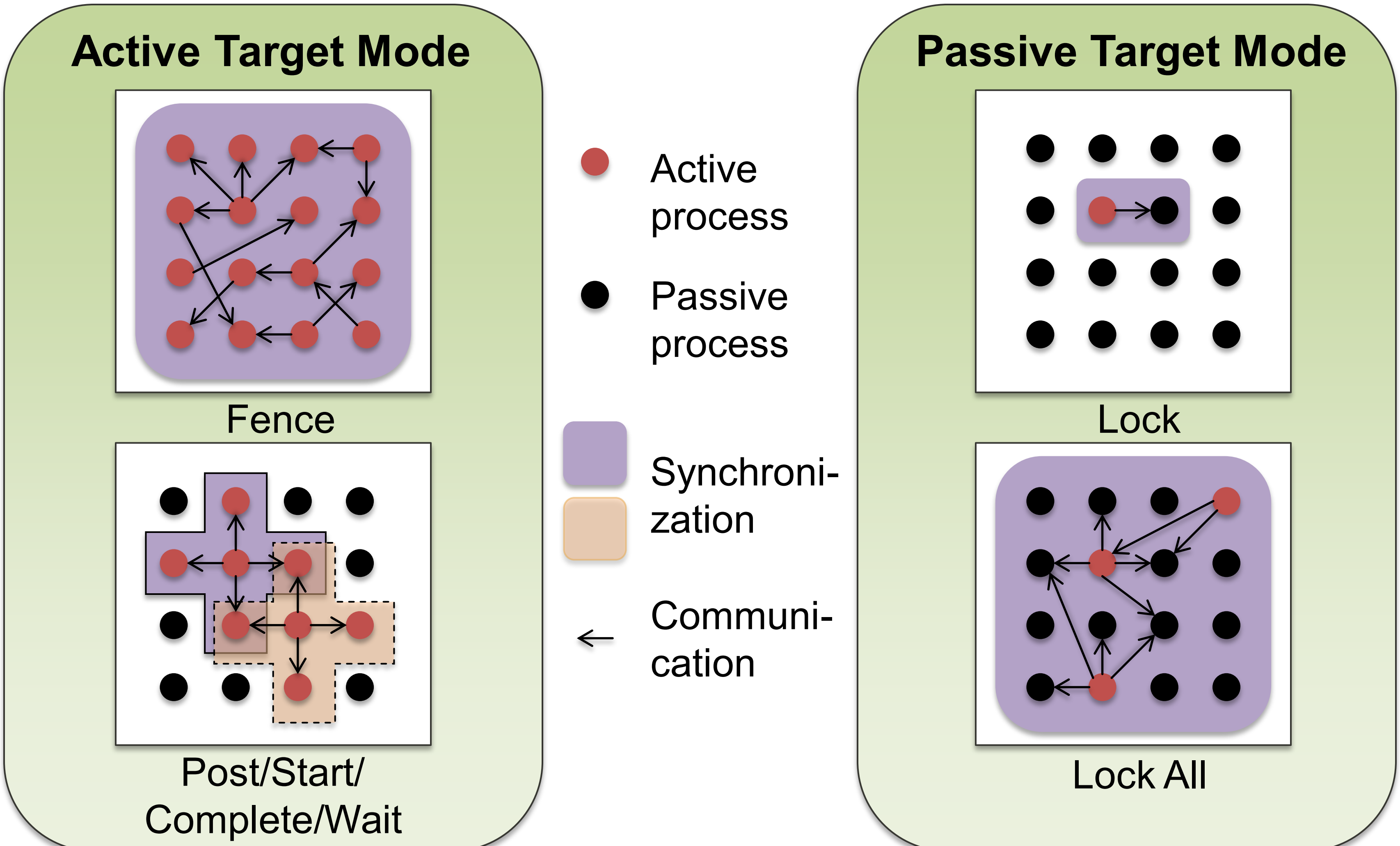




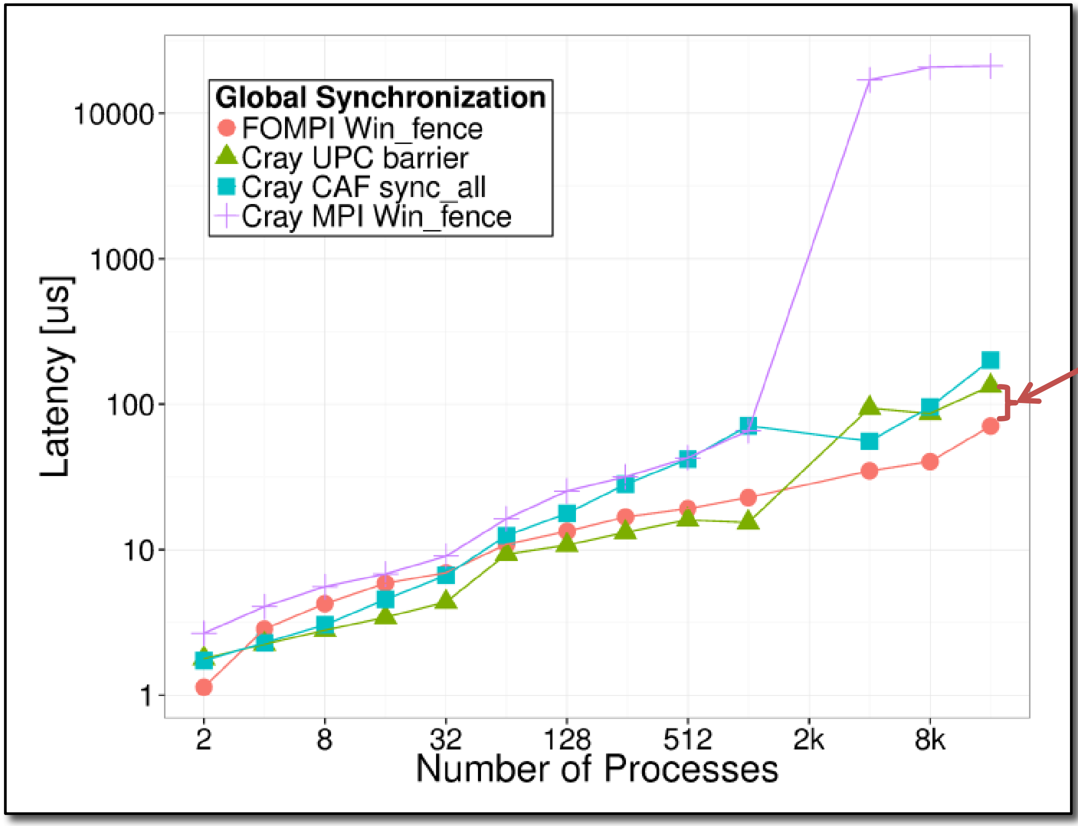


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

PART 3: SYNCHRONIZATION



Scalable Fence Performance



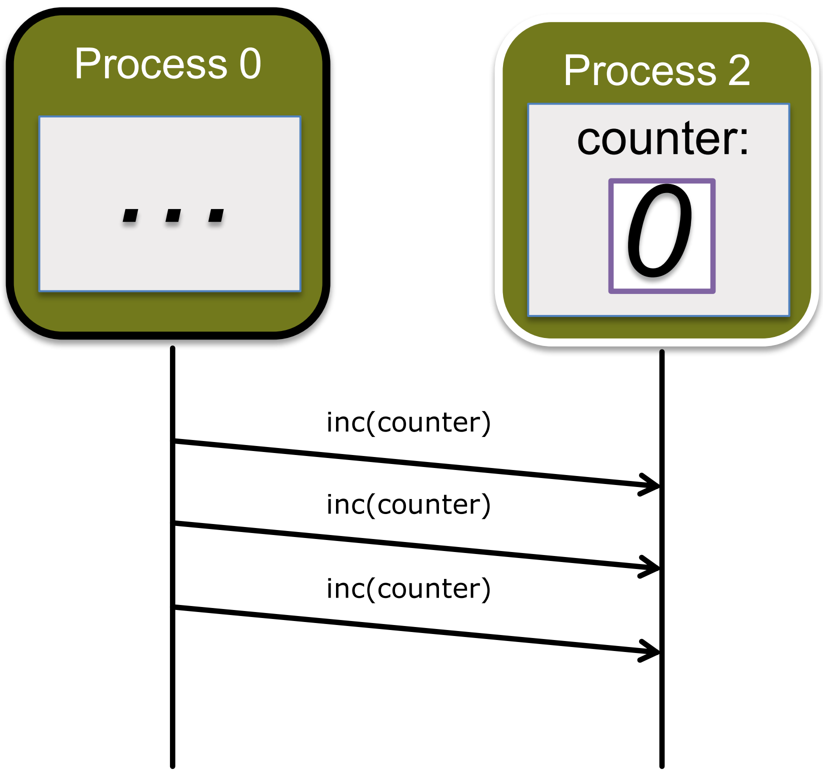
Time bound	$\mathcal{O}(\log p)$
Memory bound	$\mathcal{O}(1)$

Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Flush Synchronization

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

- Guarantees remote completion
- Performs a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path

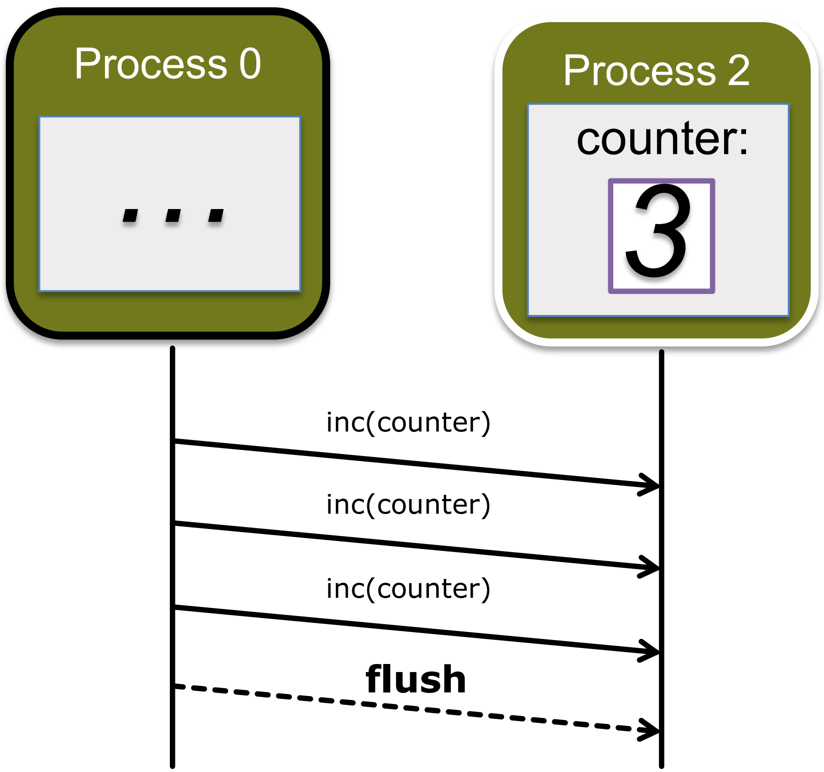


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Flush Synchronization

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

- Guarantees remote completion
- Performs a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path



Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

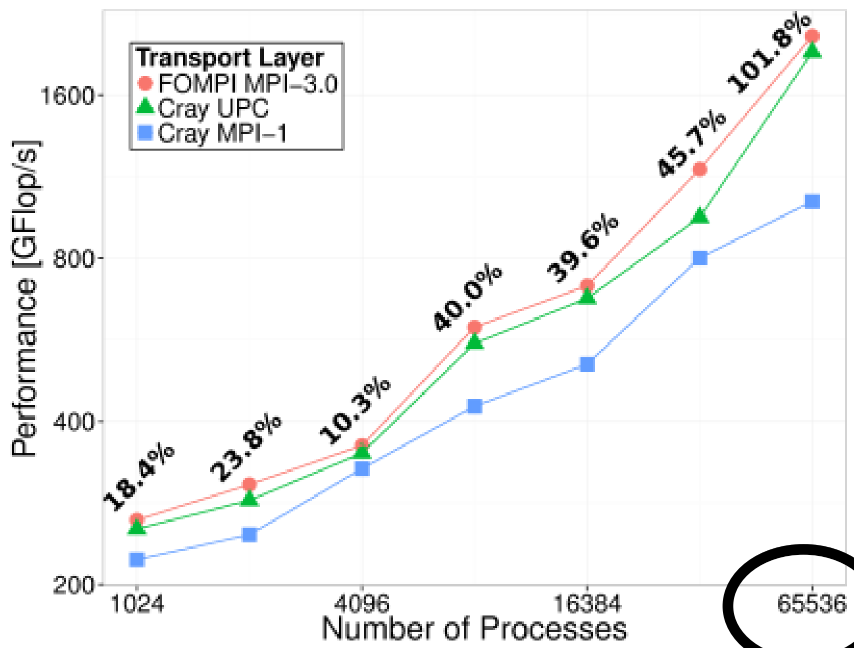
APPLICATION PERFORMANCE

- Evaluation on Blue Waters System
  - 22,640 computing Cray XE6 nodes
  - 724,480 schedulable cores
- One nearly full-scale run ☺

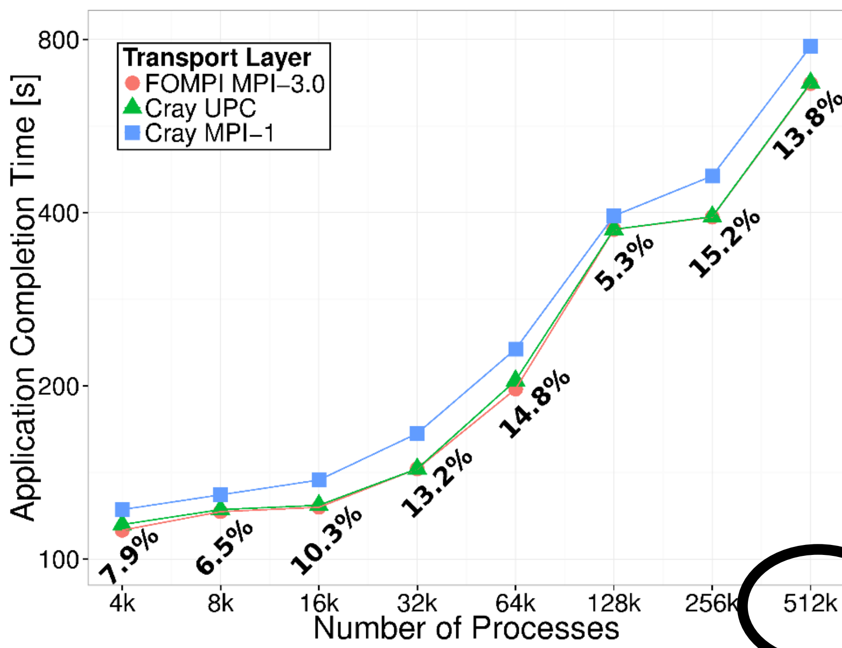
PERFORMANCE: APPLICATIONS

Annotations represent performance gain of foMPI [3] over Cray MPI-1.

NAS 3D FFT [1] Performance



MILC [2] Application Execution Time



scale to 65k procs

scale to 512k procs

[1] Nishtala et al.: Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap. IPDPS'09
[2] Shan et al.: Accelerating applications at scale using one-sided communication. PGAS'12
[3] Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

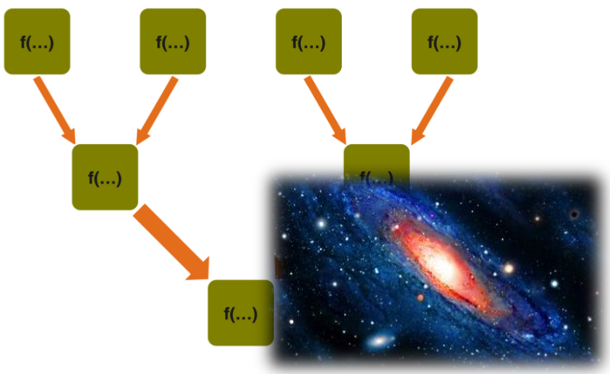
In case you want to learn more

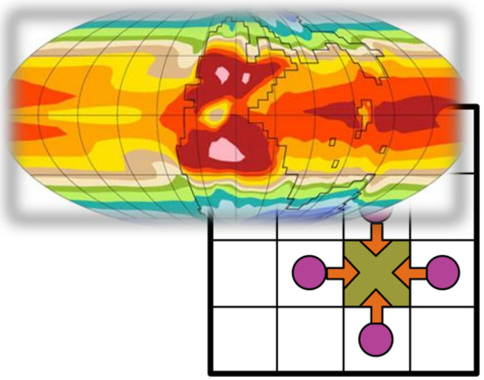
- Available
  - Some ar

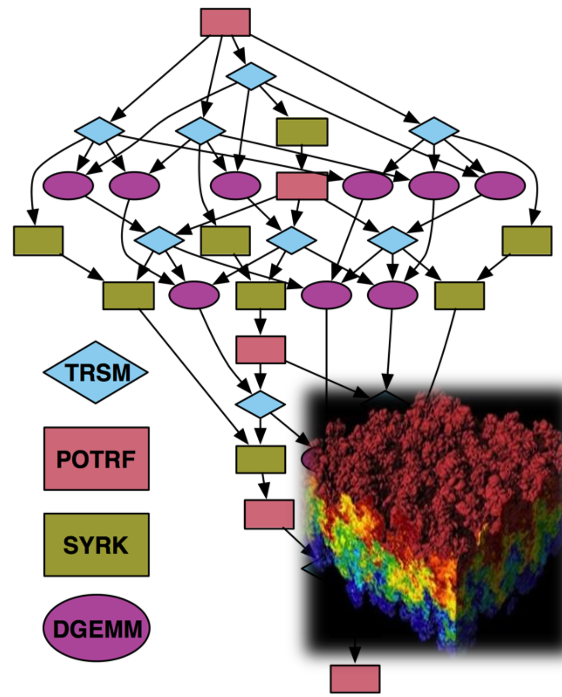
How to implement producer/consumer in passive mode?

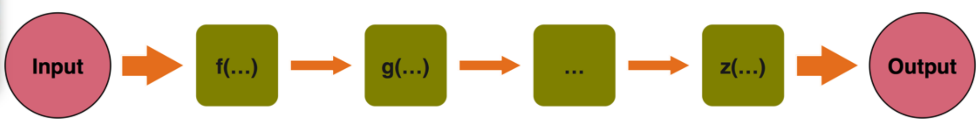
Producer-Consumer Relations

- **Most important communication idiom**
  - Some examples:



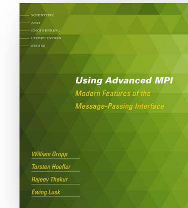




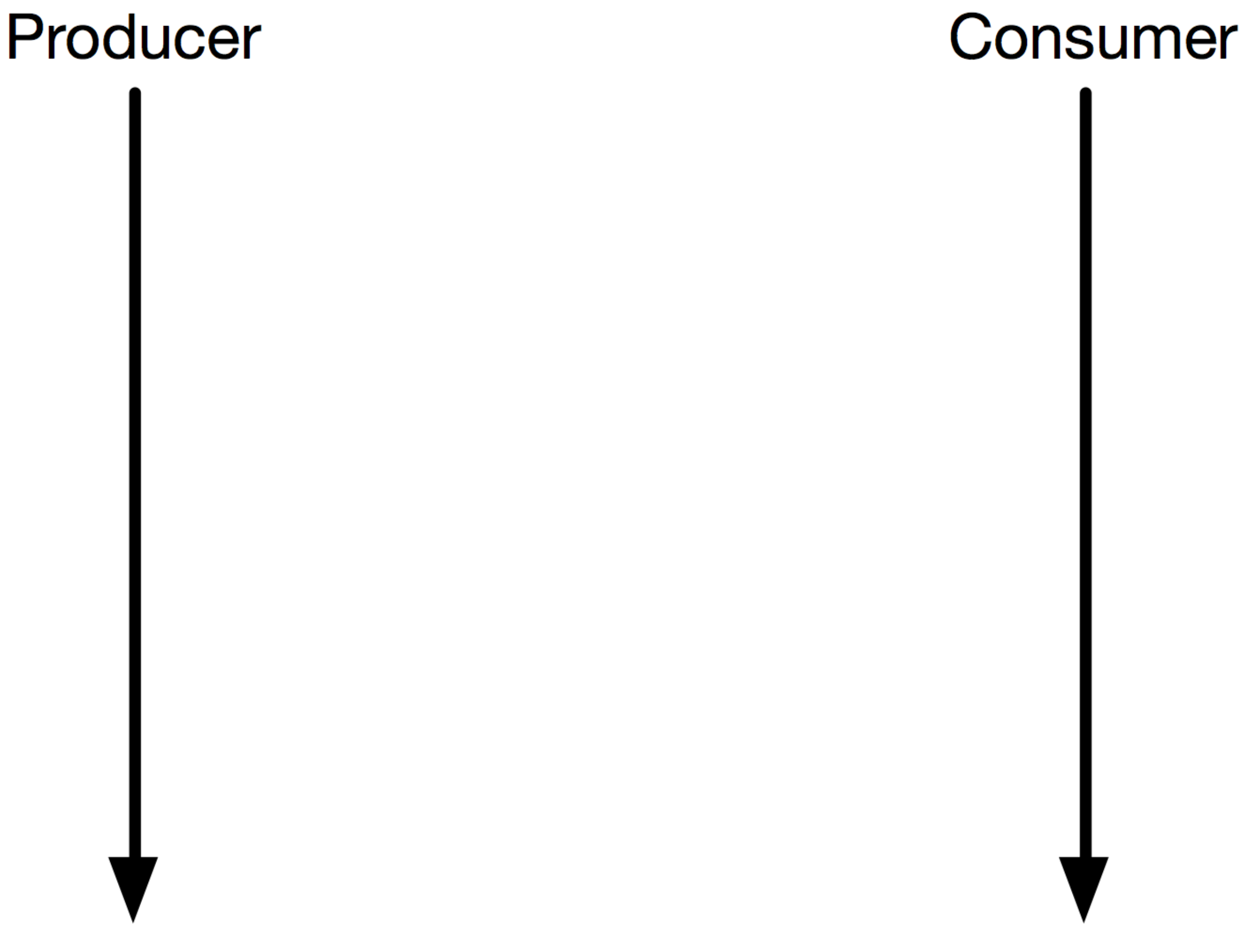


- **Perfectly supported by MPI-1 Message Passing**
  - But how does this actually work over RDMA?

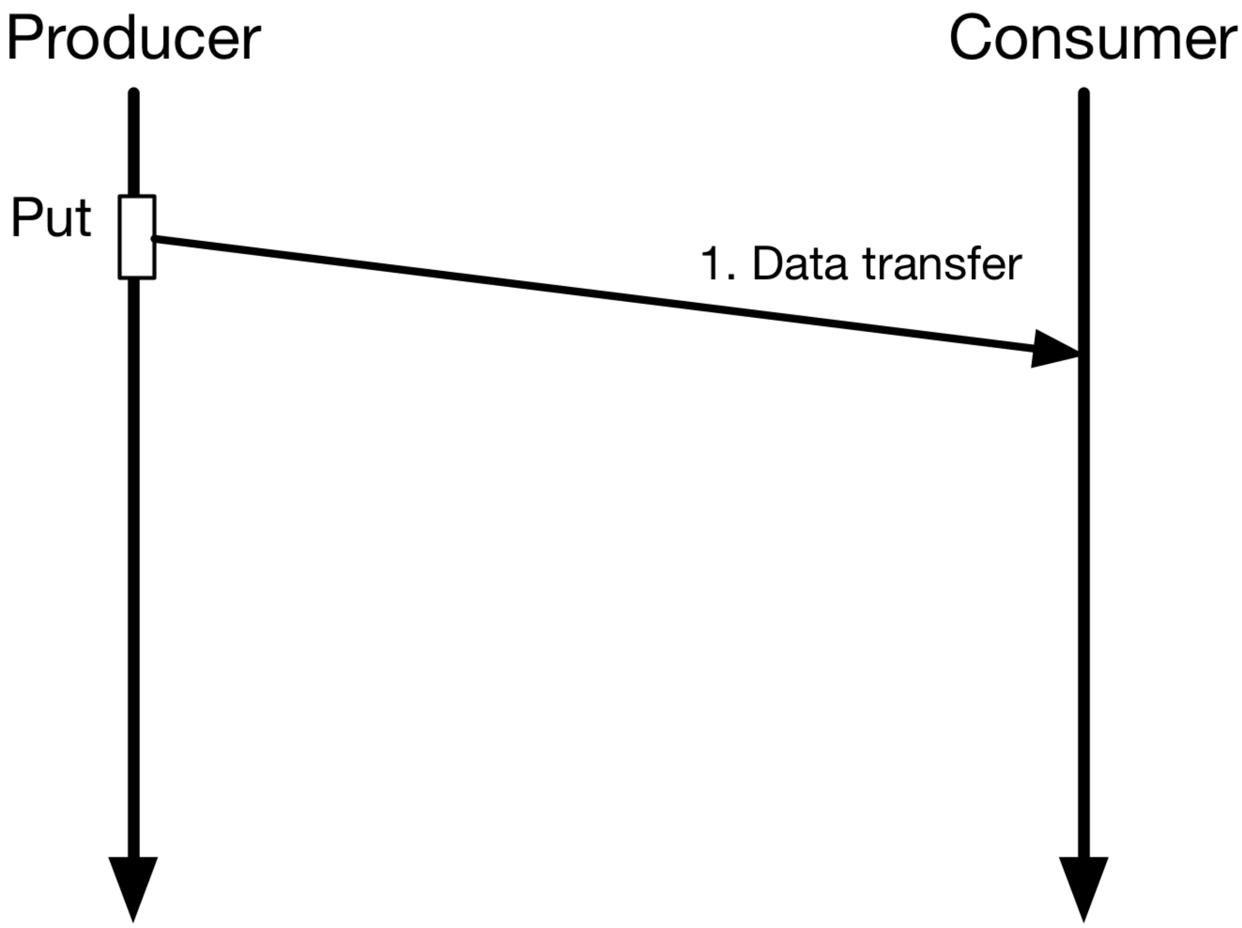
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15



One Sided – Put + Synchronization

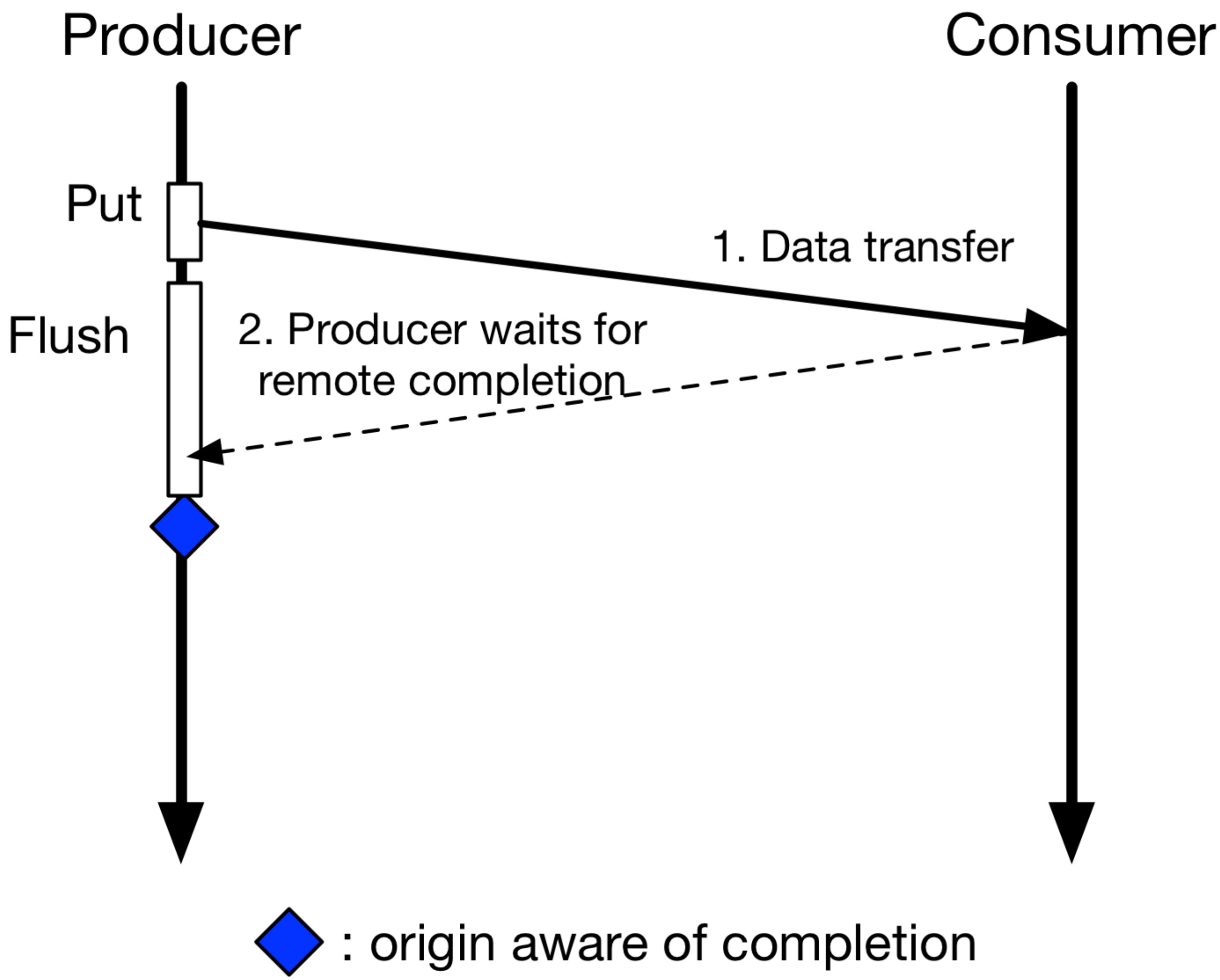


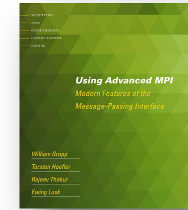
One Sided – Put + Synchronization



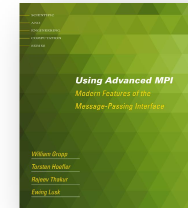
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

One Sided – Put + Synchronization

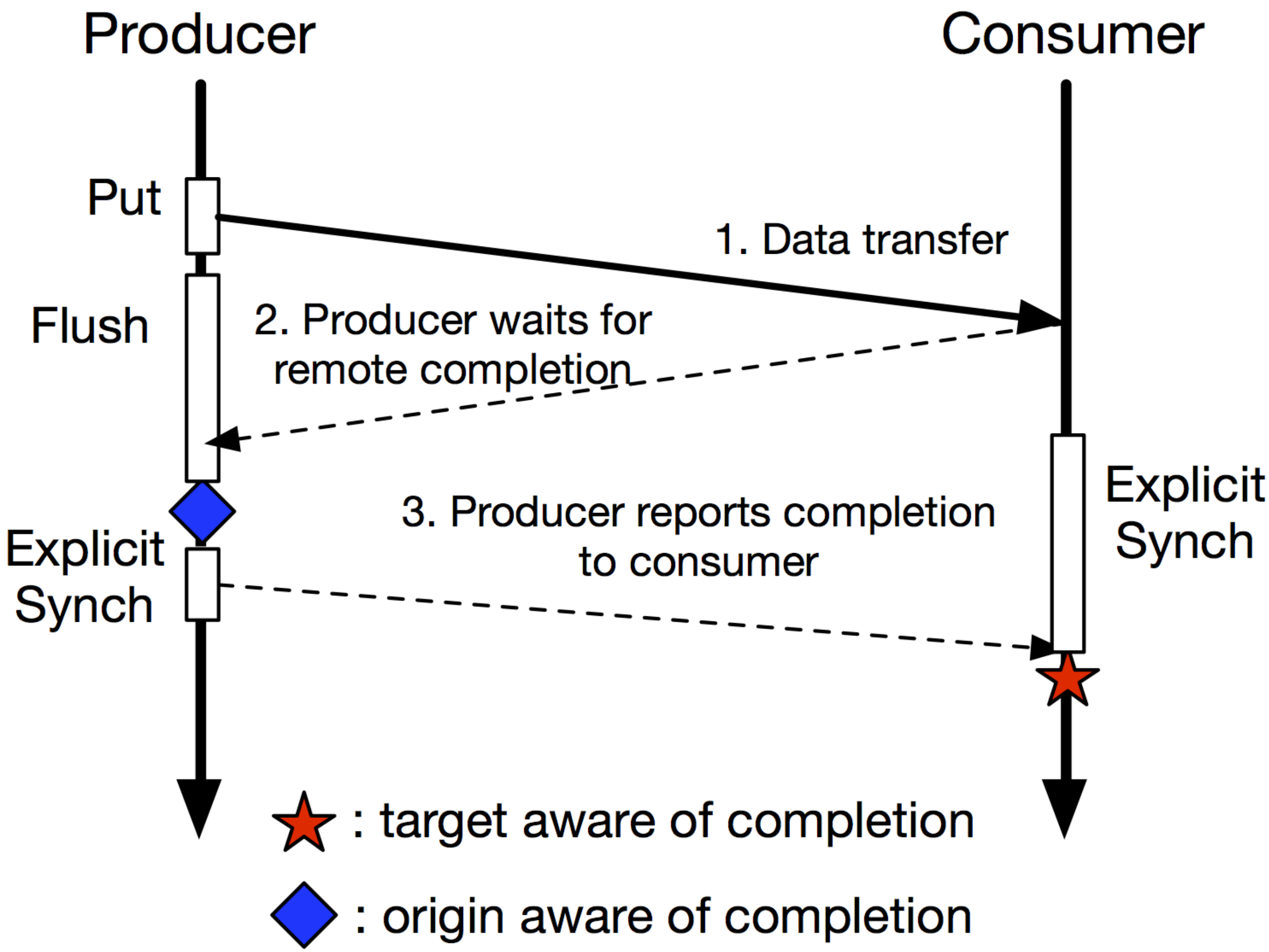




Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15



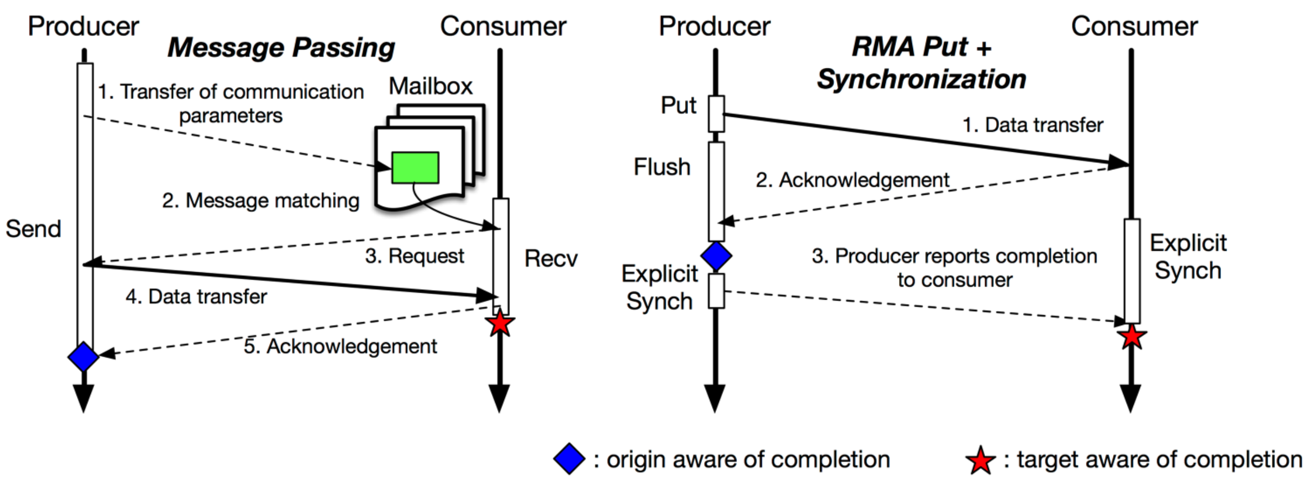
One Sided – Put + Synchronization



Critical path: 3 latencies

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

COMPARING APPROACHES



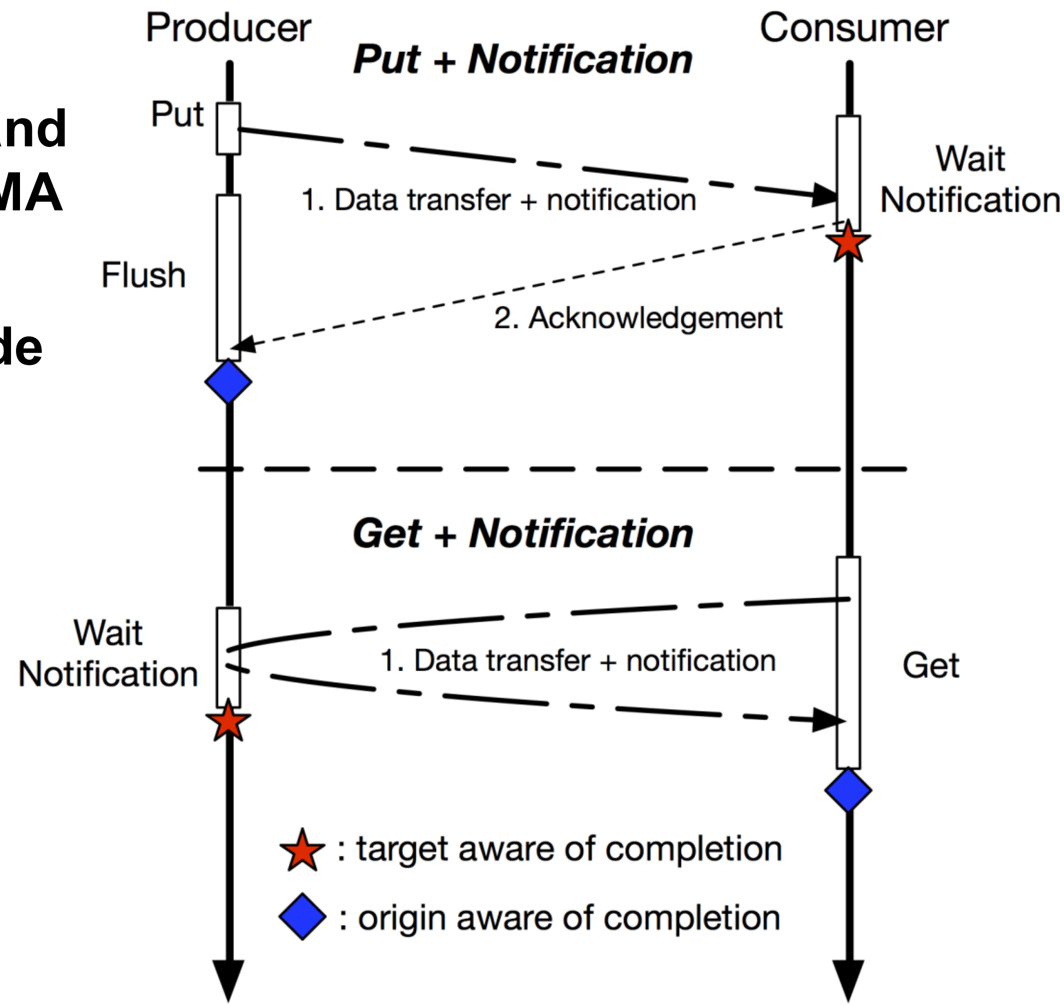
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

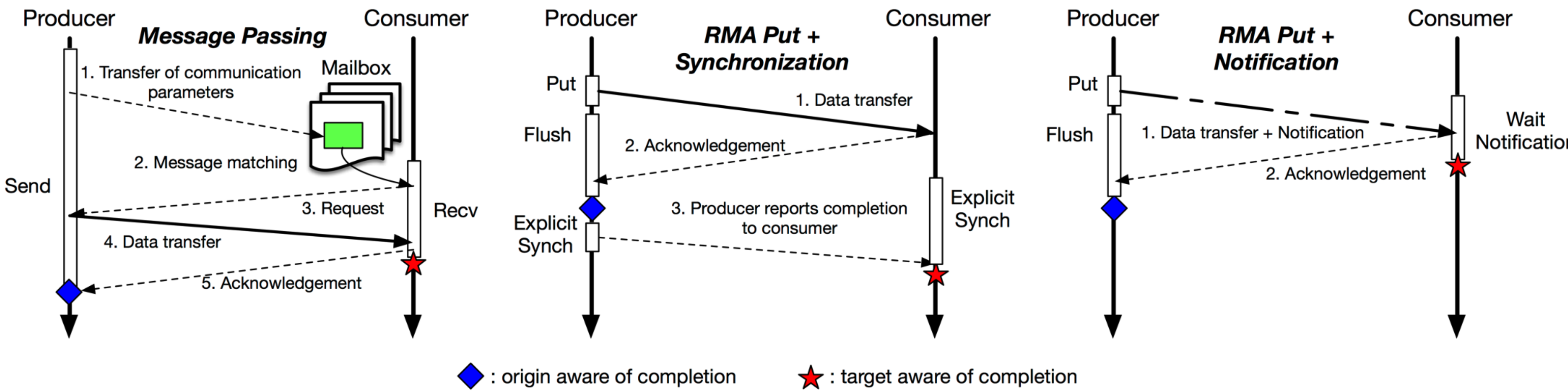
Idea: RMA Notifications

- **First seen in Split-C (1992)**
- **Combine communication and synchronization using RDMA**
- **RDMA networks can provide various notifications**
  - Flags
  - Counters
  - Event Queues



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

COMPARING APPROACHES



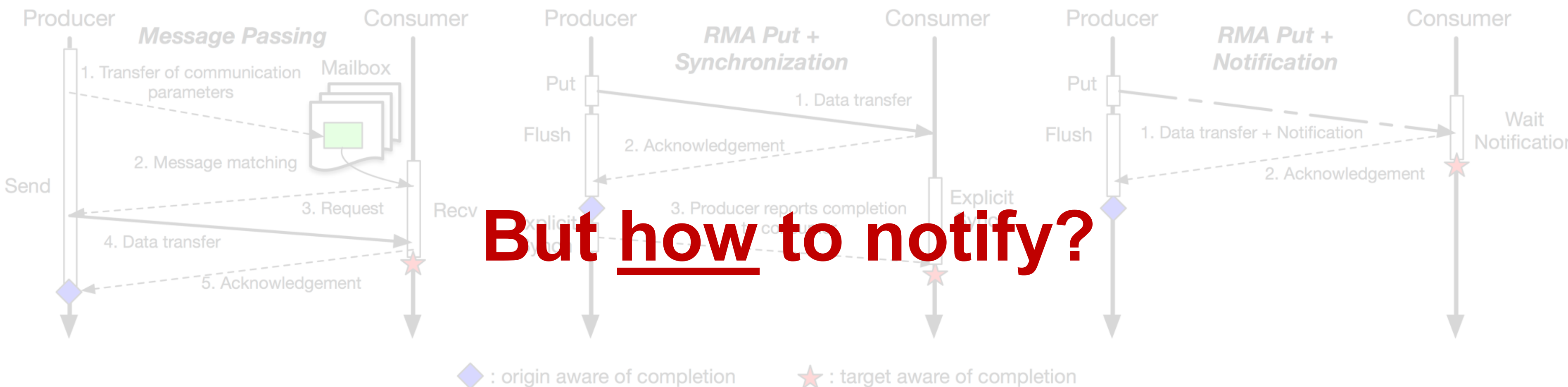
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Notified Access
1 latency

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Comparing Approaches



But how to notify?

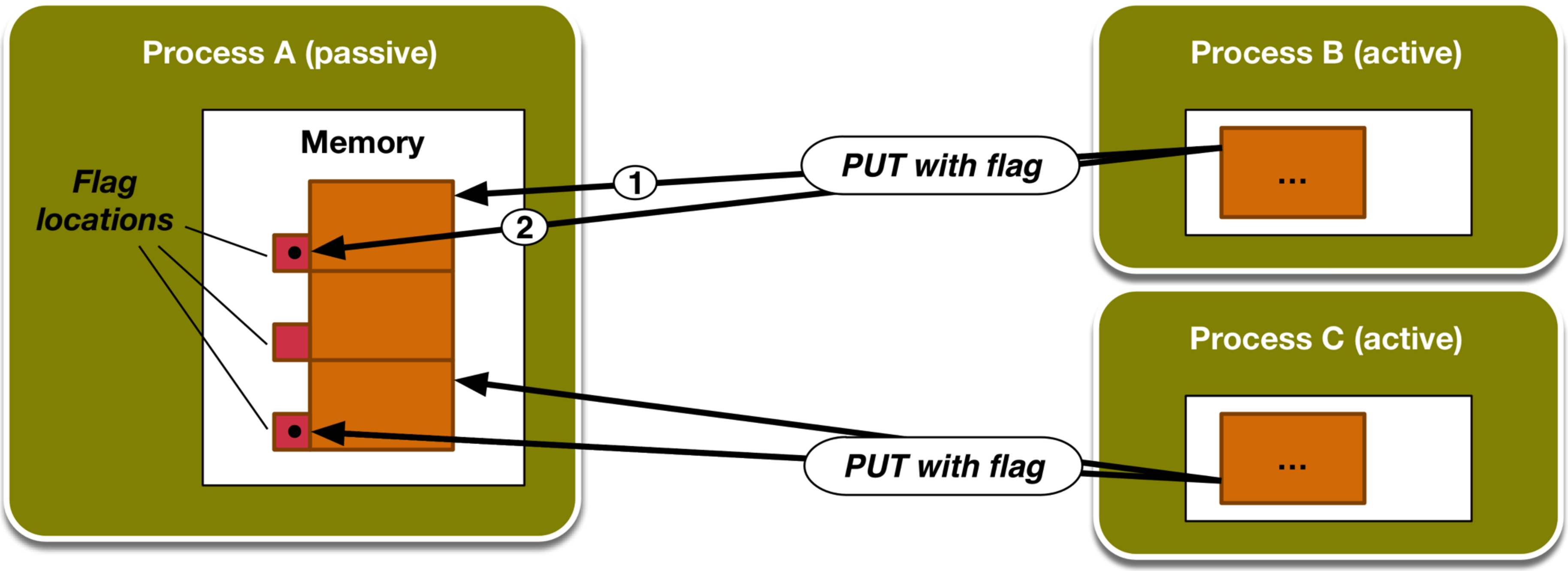
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Notified Access
1 latency

PREVIOUS WORK: OVERWRITING INTERFACE

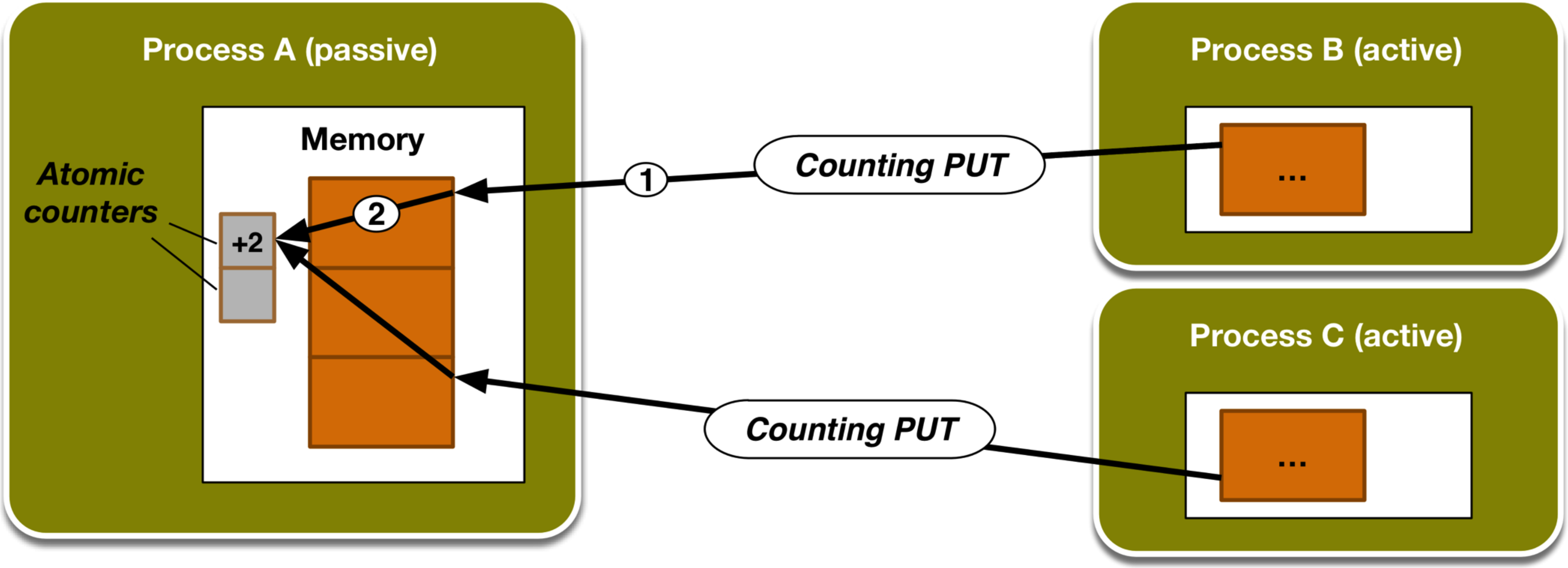
- **Flags (polling at the remote side)**
  - Used in *GASPI, DMAPP, NEON*



- **Disadvantages**
  - Location of the flag chosen at the sender side
  - Consumer needs at least one flag for every process
  - Polling a high number of flags is inefficient

PREVIOUS WORK: COUNTING INTERFACE

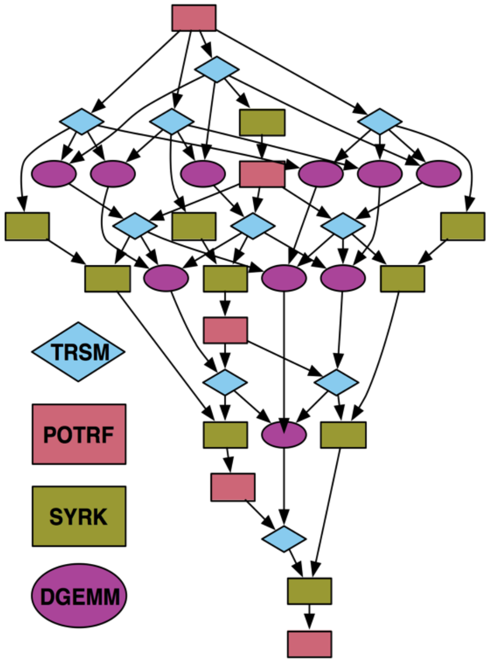
- **Atomic counters (accumulate notifications → scalable)**
  - Used in *Split-C, LAPI, SHMEM - Counting Puts, …*



- **Disadvantages**
  - Dataflow applications may require many counters
  - High polling overhead to identify accesses
  - Does not preserve order (may not be linearizable)

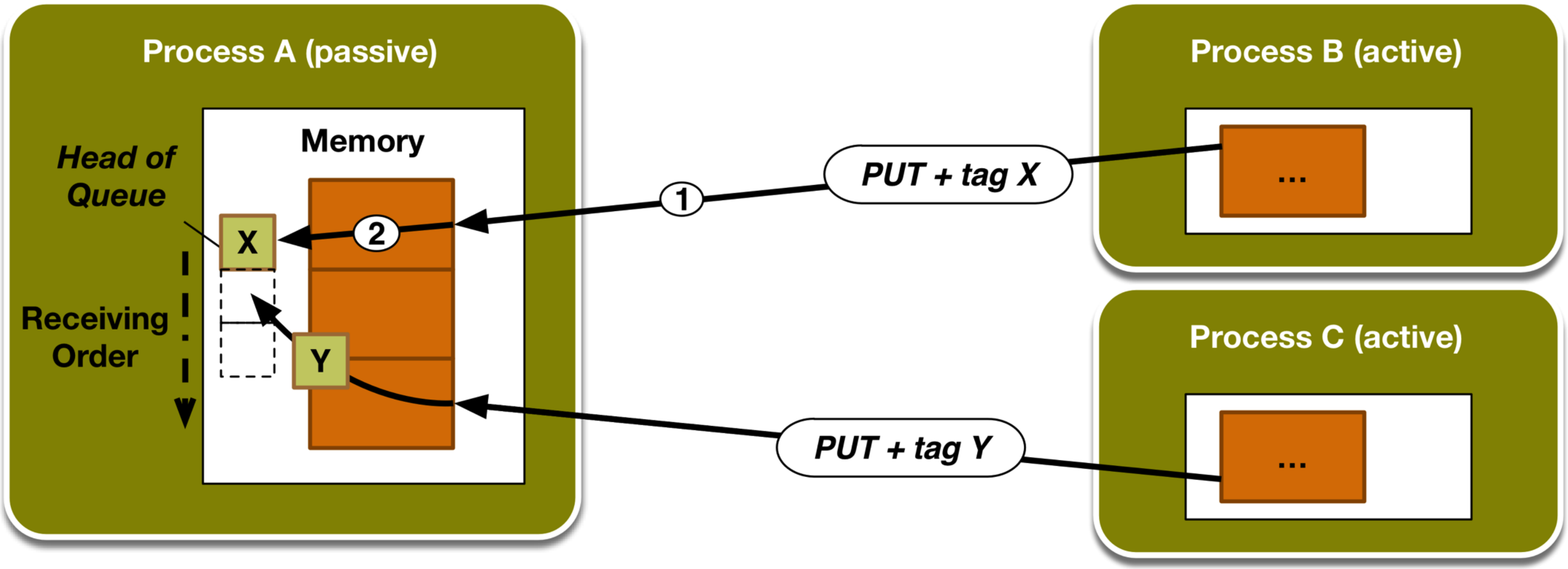
WHAT IS A GOOD NOTIFICATION INTERFACE?

- **Scalable to yotta-scale**
  - Does memory or polling overhead grow with # of processes?
- **Computation/communication overlap**
  - Do we support maximum asynchrony? (better than MPI-1)
- **Complex data flow graphs**
  - Can we distinguish between different accesses locally?
  - Can we avoid starvation?
  - What about load balancing?
- **Ease-of-use**
  - Does it use standard mechanisms?



Our Approach: Notified Access

- **Notifications with MPI-1 (queue-based) matching**
  - Retains benefits of previous notification schemes
  - Poll only head of queue
  - Provides linearizable semantics



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

NOTIFIED ACCESS – AN MPI INTERFACE

- **Minor interface evolution**
  - Leverages MPI two sided <source, tag> matching
  - Wildcards matching with FIFO semantics

Example Communication Primitives

```
int MPI_Put_notify(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                   MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win,
                   int tag);
int MPI_Get_notify(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                   MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win,
                   int tag);
```

Example Synchronization Primitives

```
int MPI_Notify_init(MPI_Win win, int src_rank, int tag, int expected_count, MPI_Request *request);
/*Functions already available in MPI*/
int MPI_Start(MPI_Request *request);
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status);
int MPI_Wait(MPI_Request *request, MPI_Status *status);
```

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Experimental Setting

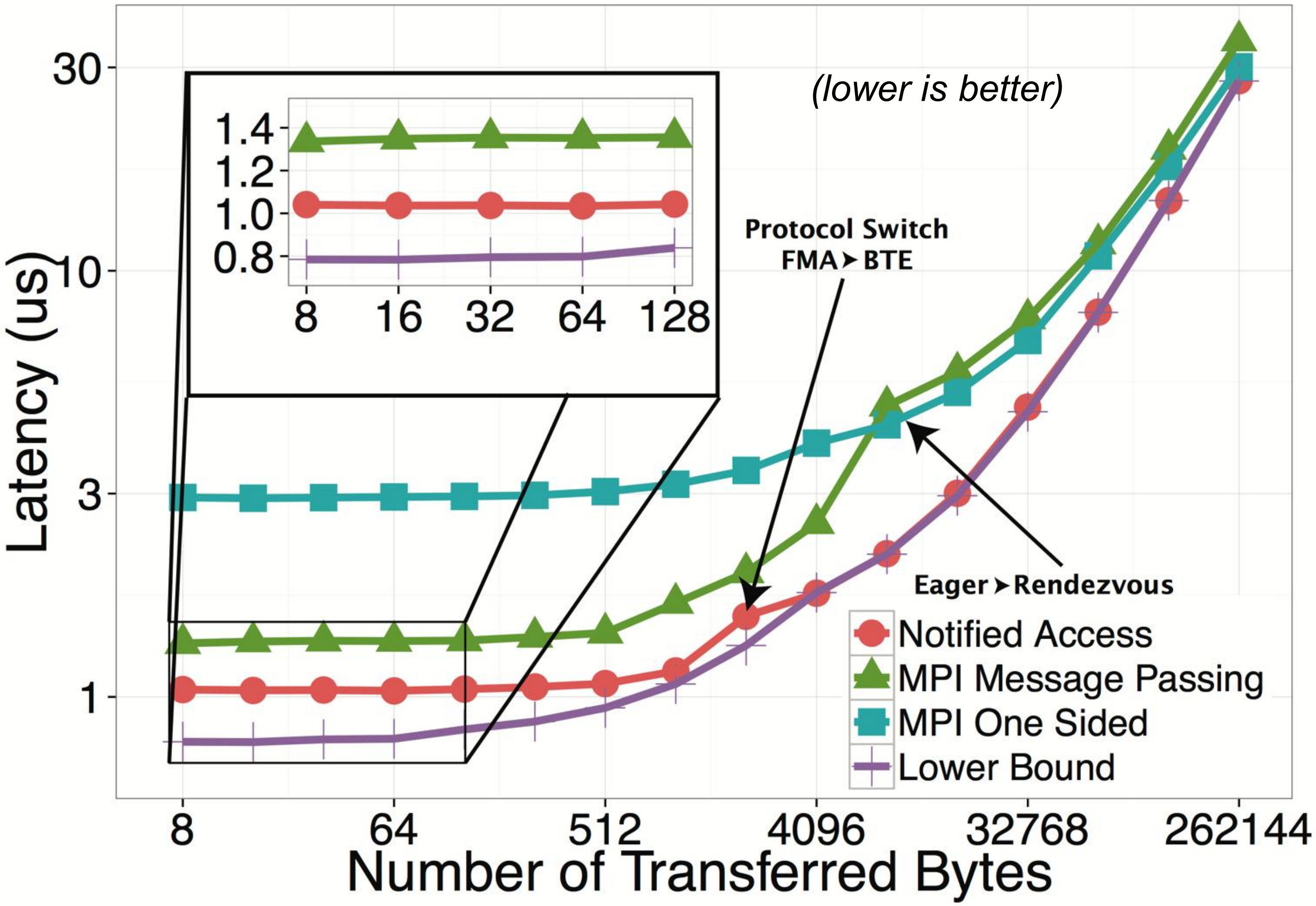
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

- **Piz Daint [1]**
  - Cray XC30, Aries interconnect
  - 5,272 computing nodes (Intel Xeon E5-2670 + NVIDIA Tesla K20X)
  - Theoretical Peak Performance 7.787 Petaflops
  - Peak Network Bisection Bandwidth 33 TB/s

[1] http://www.cscs.ch

PING PONG PERFORMANCE (INTER-NODE)

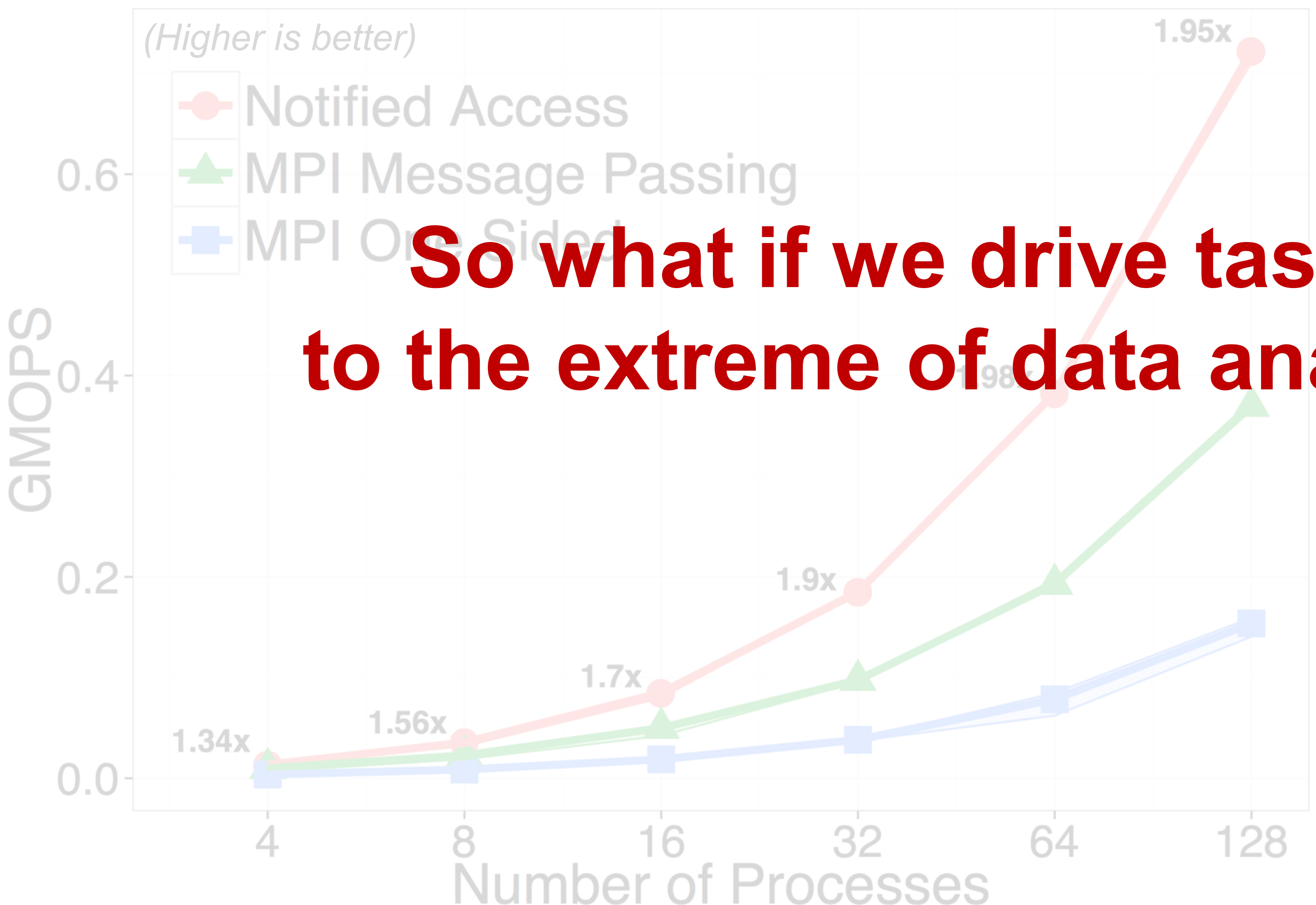
- **1000 repetitions, each timed separately, RDTSC timer**
- **95% confidence interval always within 1% of median**

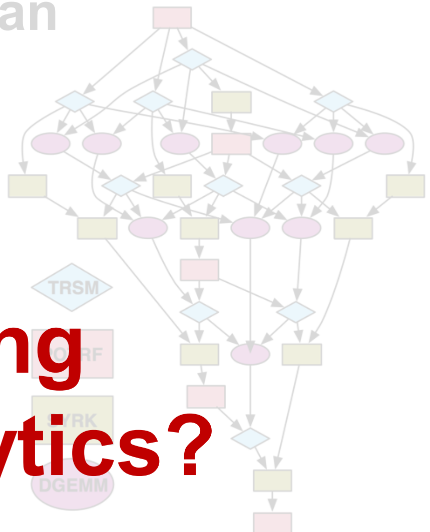


Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Cholesky – Many-to-Many Synchronization

- 1,000 repetitions, each timed separately, RDTSC timer
- 95% confidence interval always within 10% of median



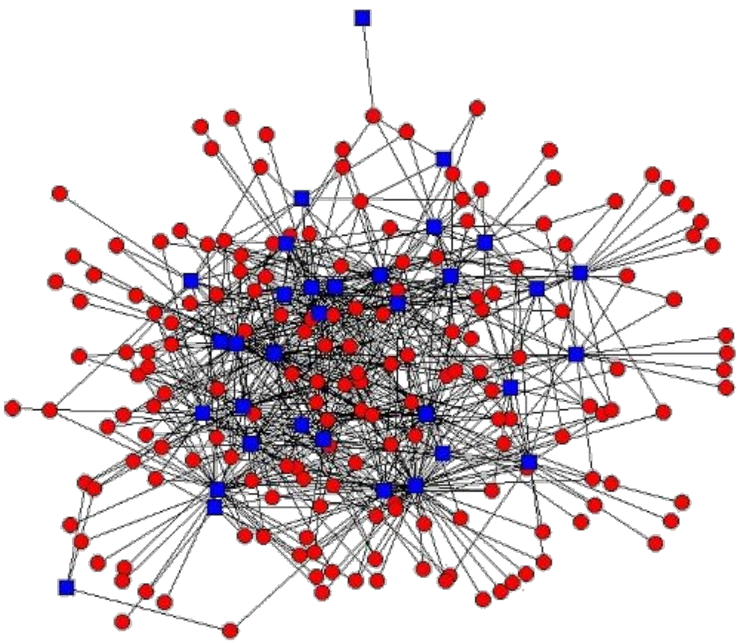


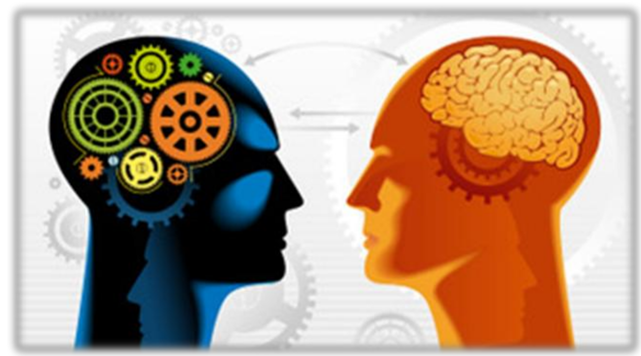
So what if we drive tasking to the extreme of data analytics?

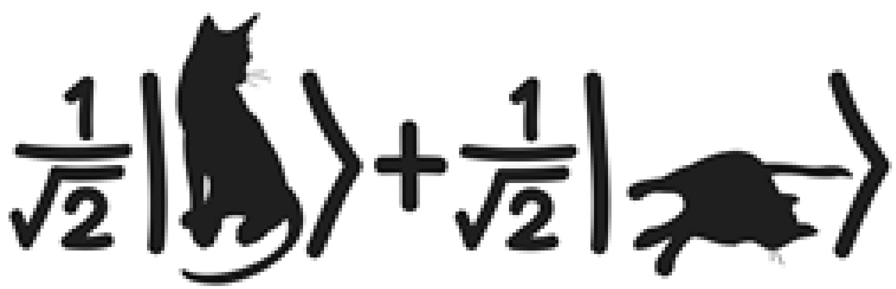
[1]: J. Kurzak, H. Ltaief, J. Dongarra, R. Badia: "Scheduling dense linear algebra operations on multicore processors", CCPE 2010

LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]
  - Machine learning
  - Computational science
  - Social network analysis







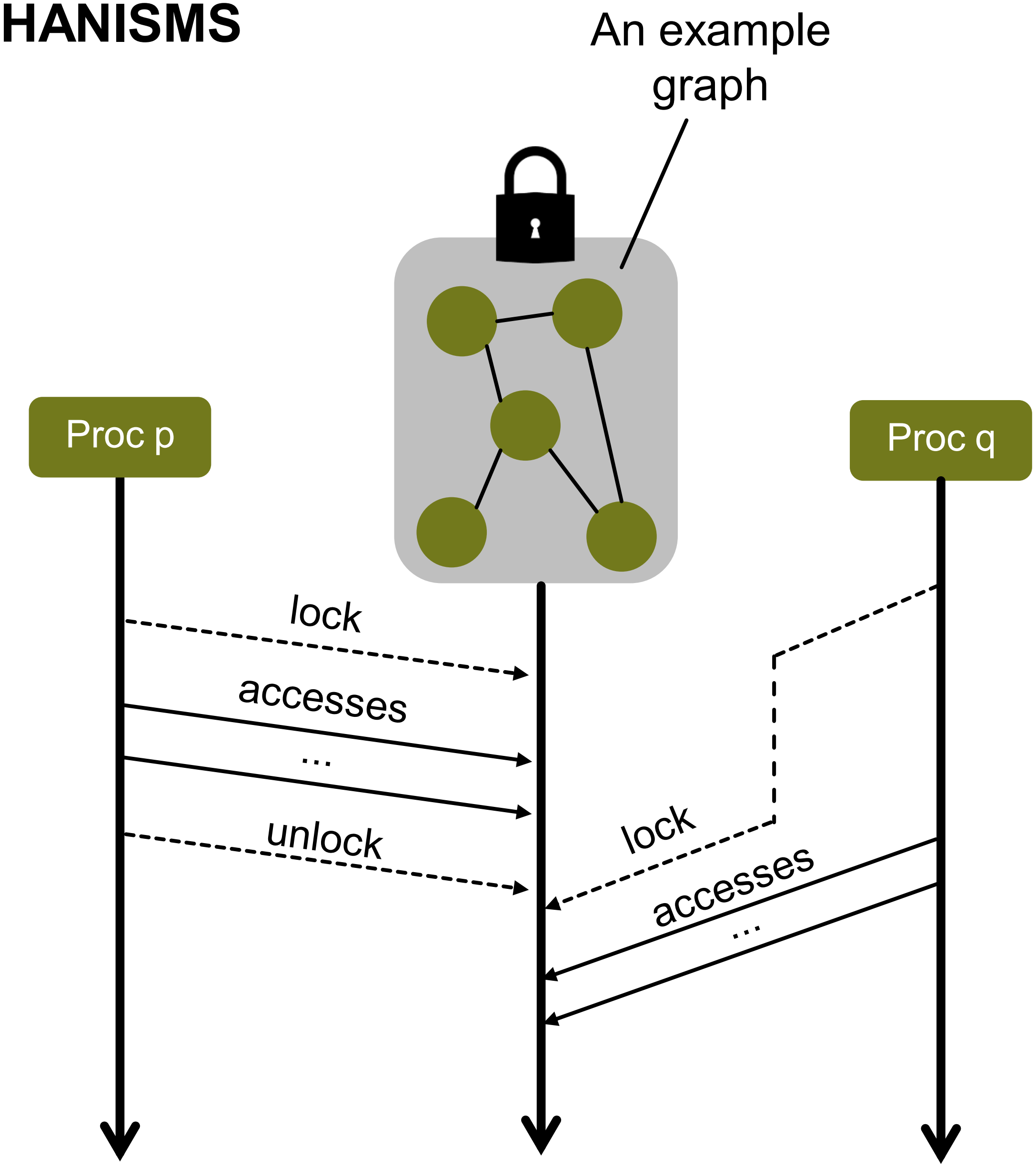
[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Letters, 2007

Synchronization Mechanisms
Coarse locks

Simple protocols

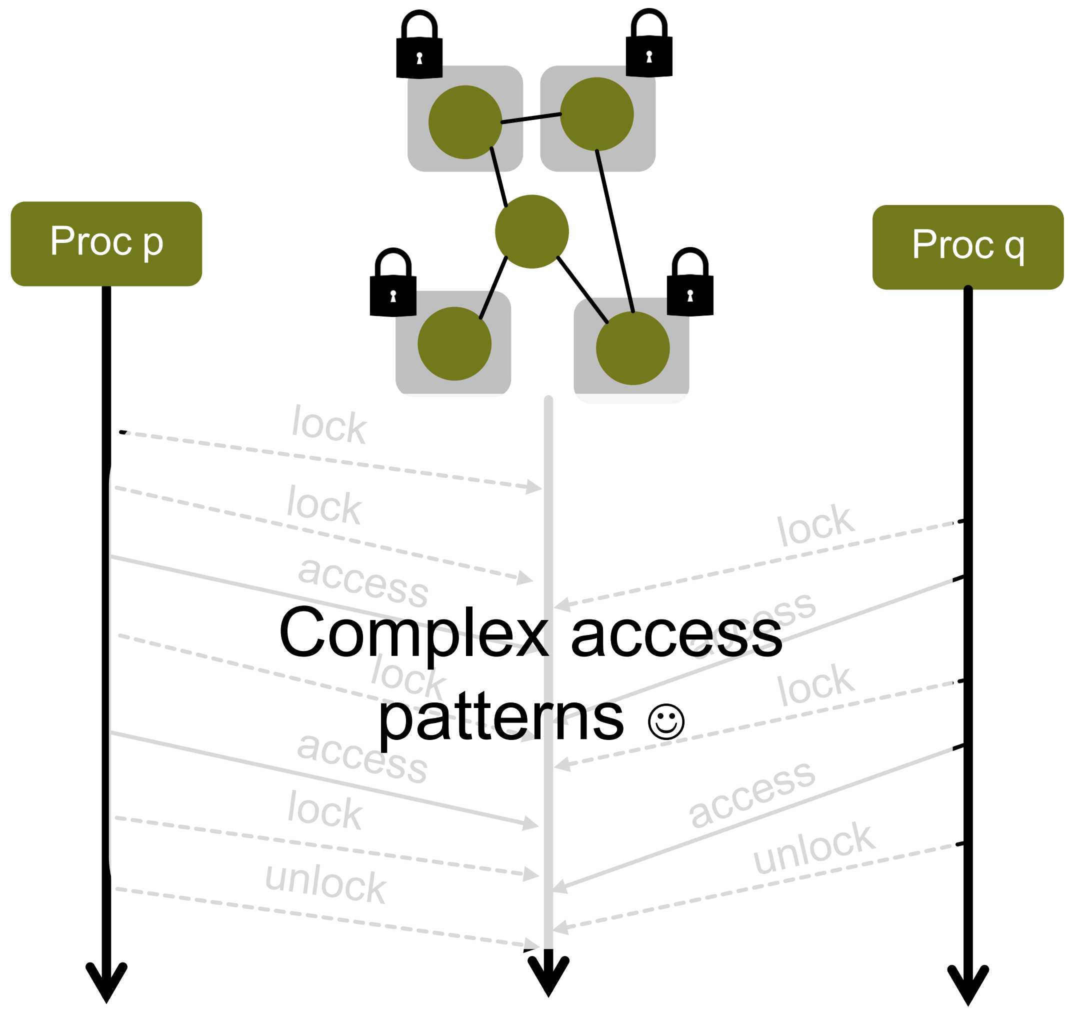
Serialization

Detrimental performance



Synchronization Mechanisms
Fine locks

- ✔ Higher performance possible
- ✘ Complex protocols
- ✘ Risk of deadlocks

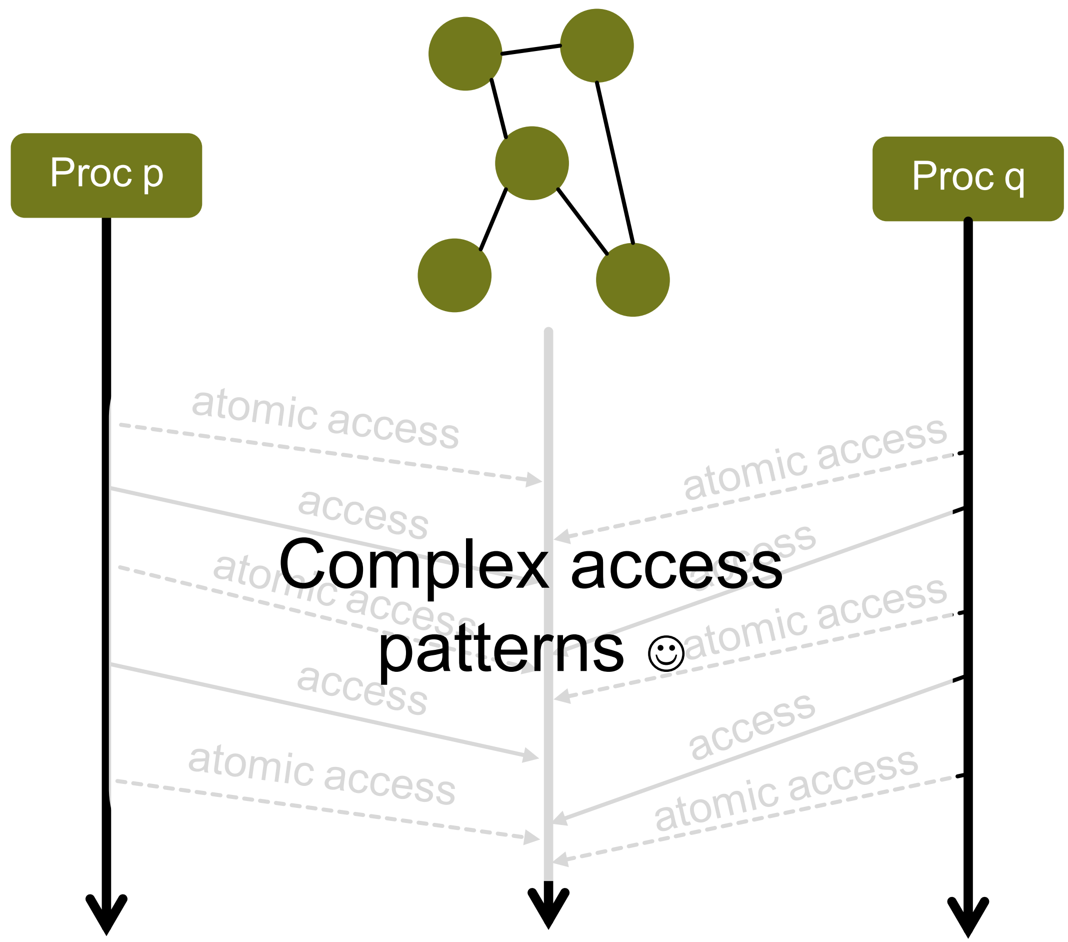


Synchronization Mechanisms
Atomic operations

✔ High performance (may be challenging to get)

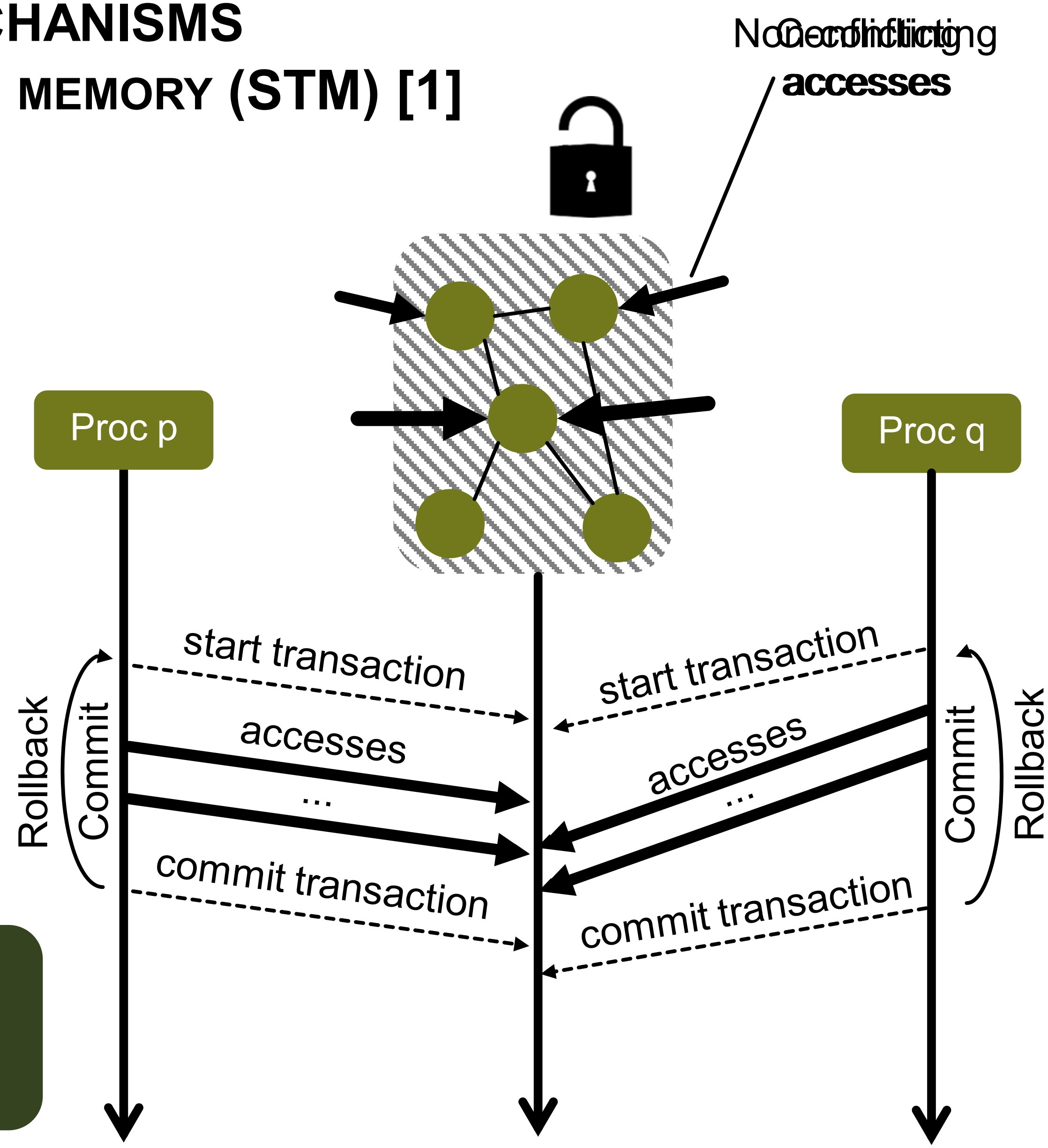
✘ Complex protocols

✘ Subtle issues (ABA, ...)



Synchronization Mechanisms
Software transactional memory (STM) [1]

Conflicts solved with rollbacks and/or serialization.



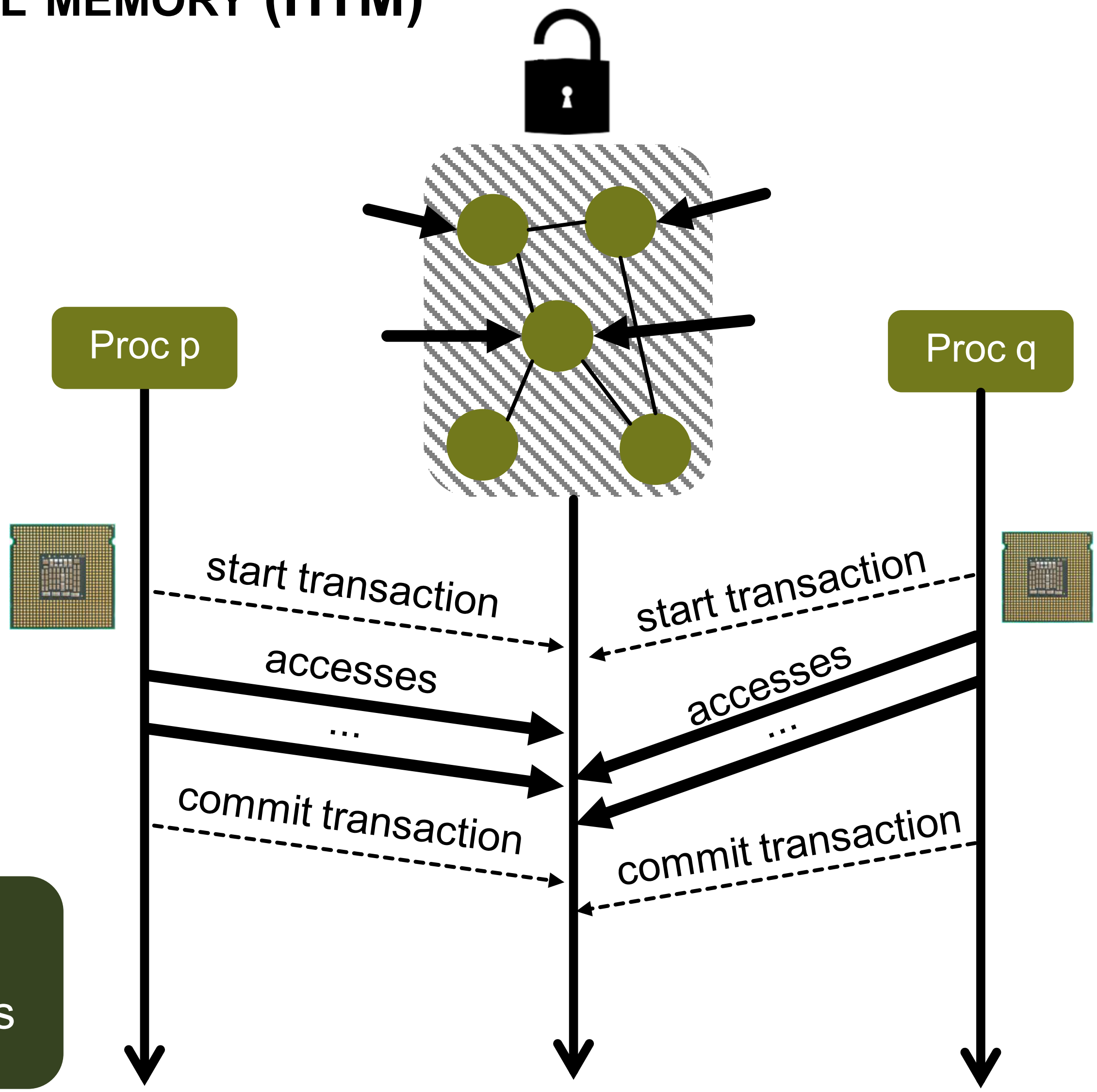
Software overheads

Simple protocols

[1] N. Shavit and D. Touitou. Software transactional memory. PODC'95.

Synchronization Mechanisms
Hardware transactional memory (HTM)

Conflicts solved with rollbacks and/or HW serialization.



High performance? For graphs?

Simple protocols

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Hardware Transactional Memory

Vega

Rock

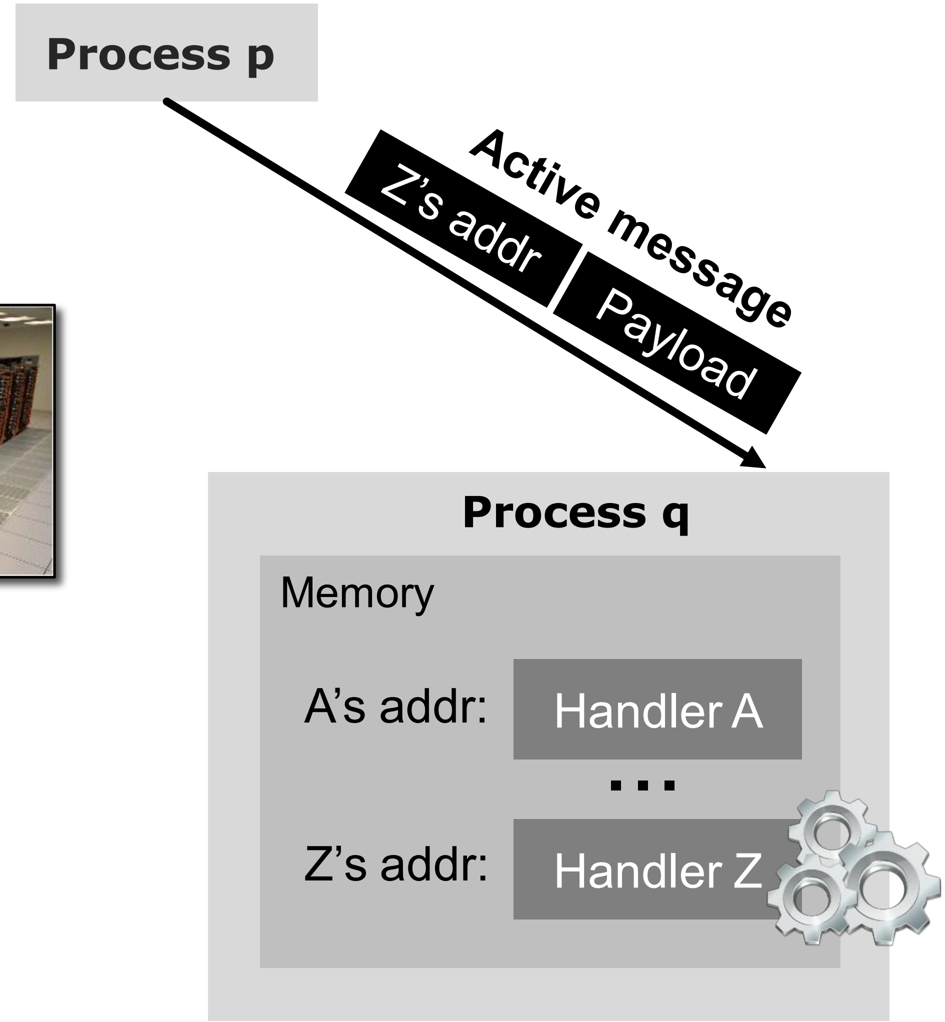
BlueGene/Q

They offer programmability, how about performance?

Haswell

POWER8

ACTIVE MESSAGES (AM)

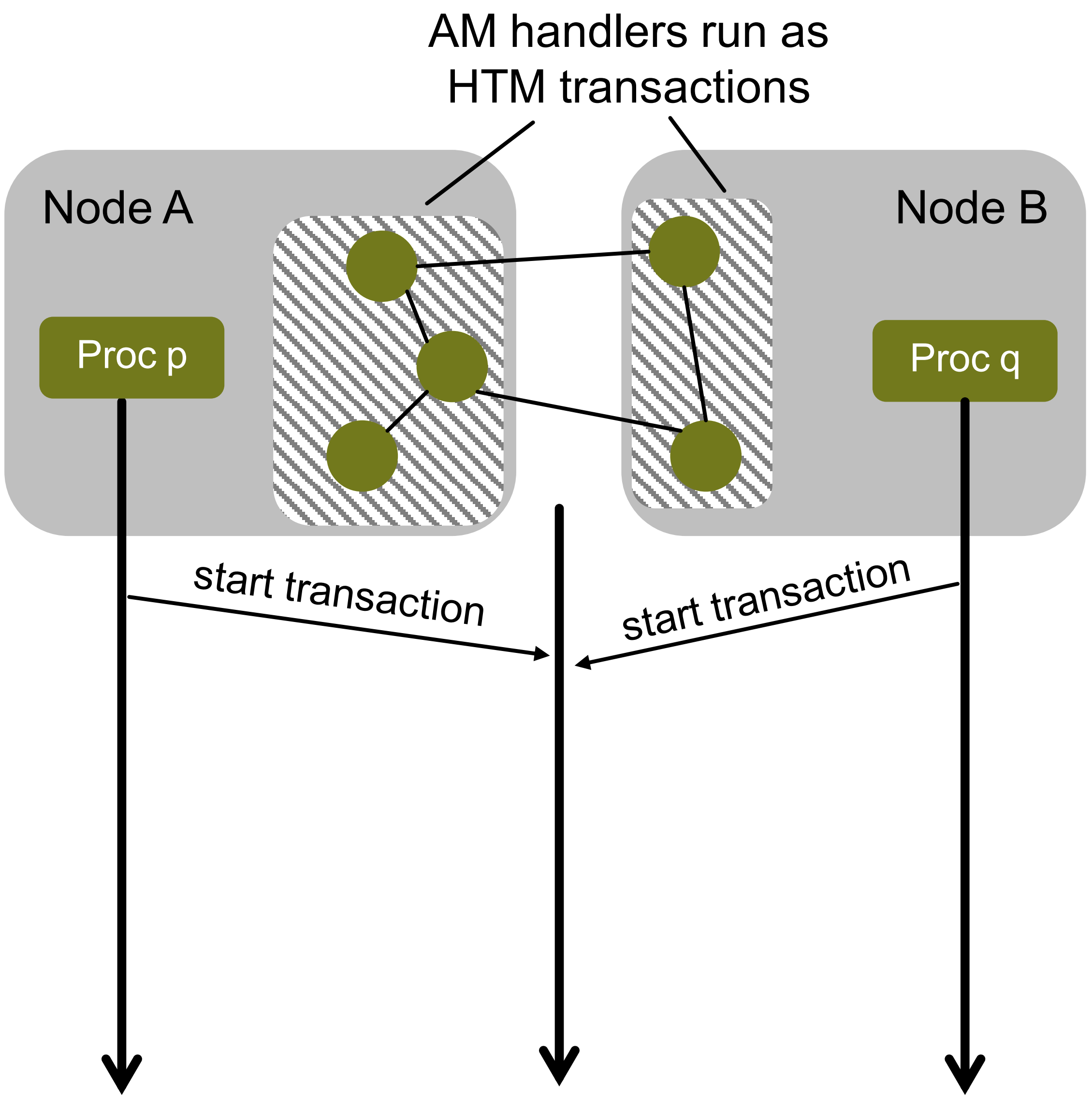


AM++[1]

GASNet [2]

[1] J. J. Willcock et al. AM++: A generalized active message framework. PACT'10.
[2] D. Bonachea, GASNet Specification, v1.1. Berkeley Technical Report. 2002.

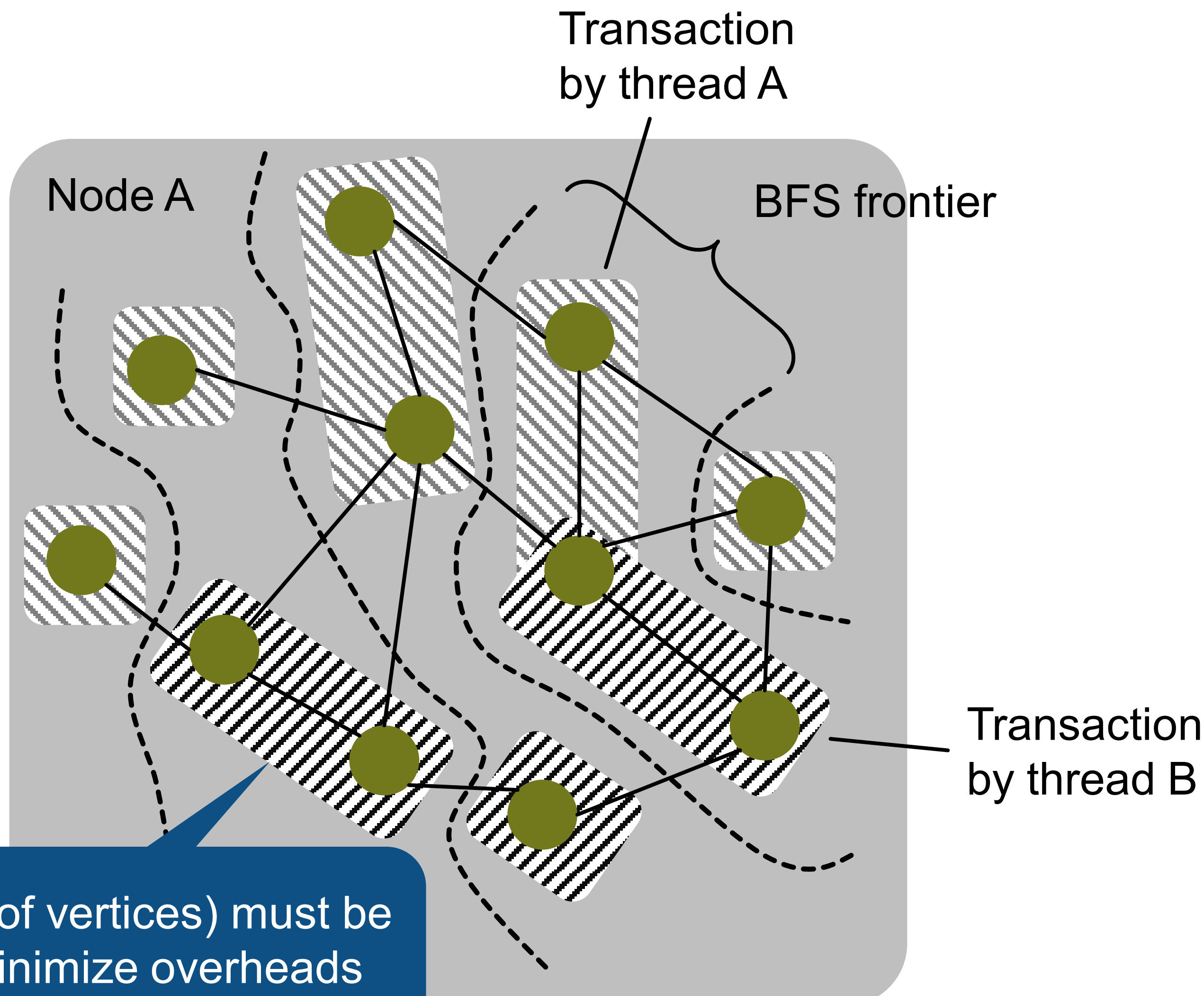
AM + HTM = Atomic Active Messages



Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Accessing Multiple Vertices Atomically

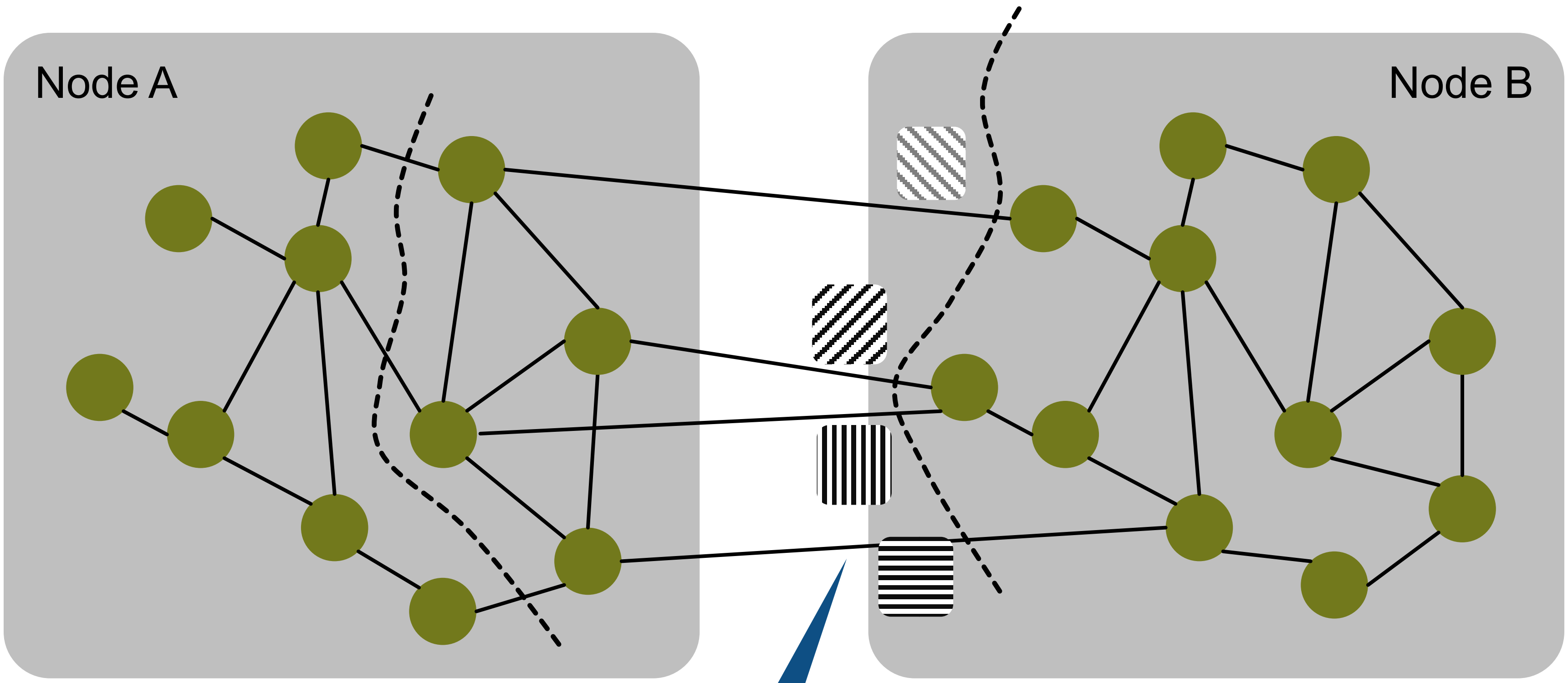
Example: BFS



Size (the number of vertices) must be appropriate to minimize overheads from both commits and rollbacks

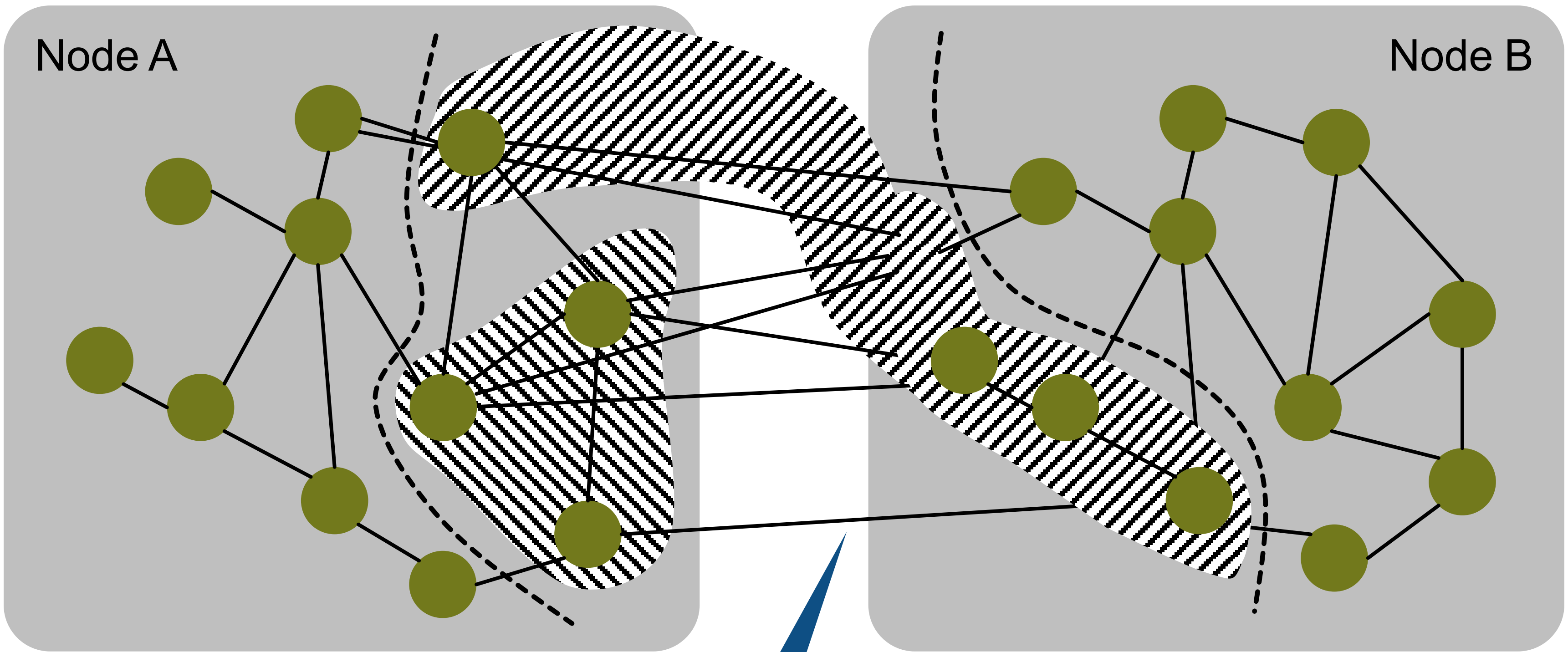
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Transferring Transactions Across Nodes



Transactions must be appropriately coalesced to minimize communication overheads

Executing Transactions on Multiple Nodes



Vertices must be appropriately relocated to enable execution of a hardware transaction

Performance Analysis
Research questions

How can we implement AAM handlers to run most efficiently?

What are performance tradeoffs related to HTM?

What are advantages of HTM over atomics for AAM?

What are the optimal transaction sizes?

PERFORMANCE ANALYSIS
TYPES OF MACHINES

- Evaluation on 3 machines
  - Intel Haswell server
  - InfiniBand cluster
  - IBM BlueGene/Q

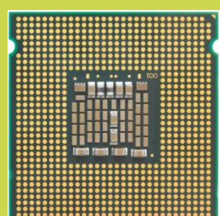
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Atomics

intel
IBM

Haswell HTM

intel



L1 L1

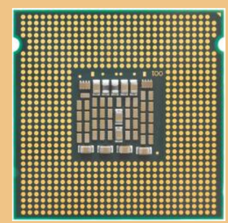
- 32KB per core
- Deployed in L1
- 8-way associative

RTM (Restricted Transactional Memory)

HLE (Hardware Lock Elision)

BlueGene/Q HTM

IBM



L1 L1
L2

- 2MB per core
- Deployed in L2
- 16-way associative

The Long Running Mode

The Short Running Mode

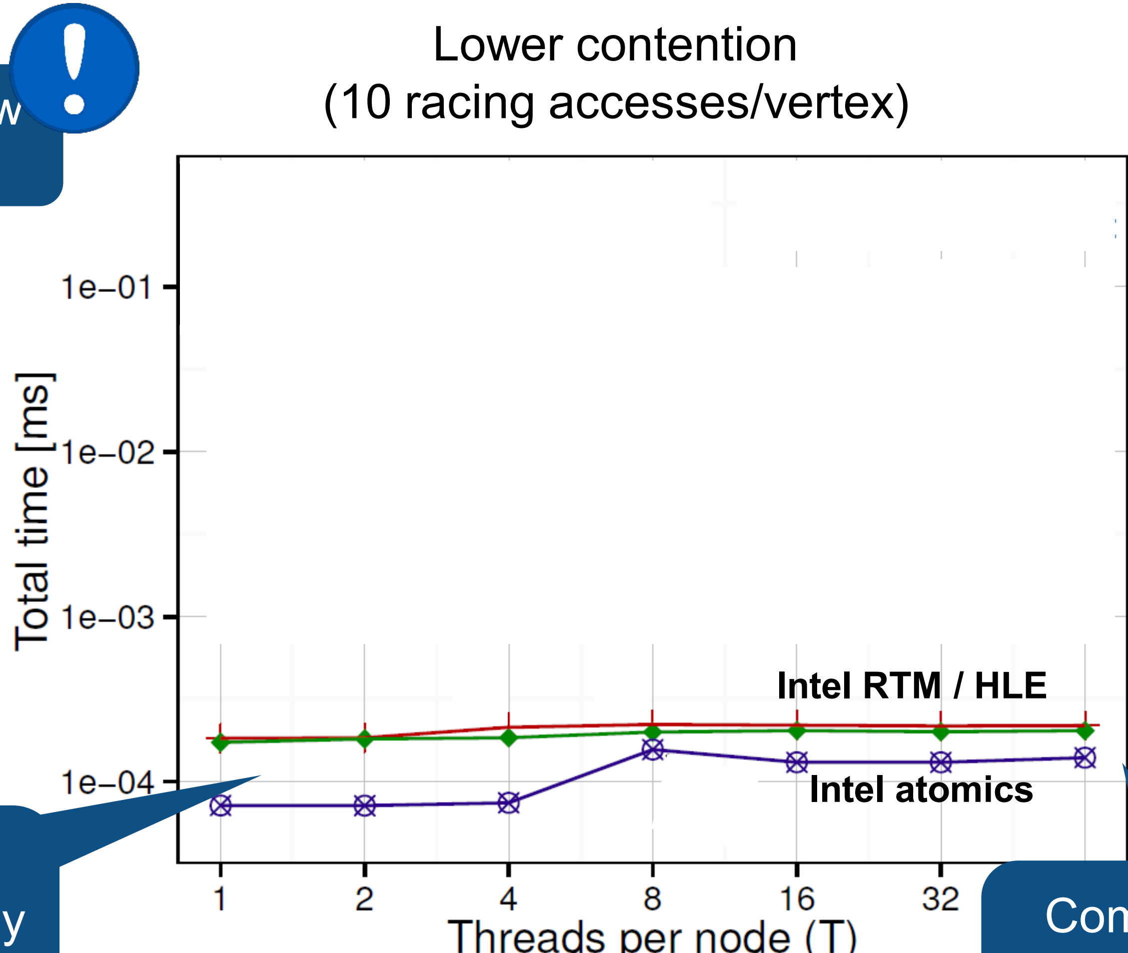
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Used in BFS, SSSP, ...

```
// start handler
if(!v.visited) {
 v.visited = 1;
}
// finish handler
```

Very few aborts



Atomics (CAS) slightly faster than HTM

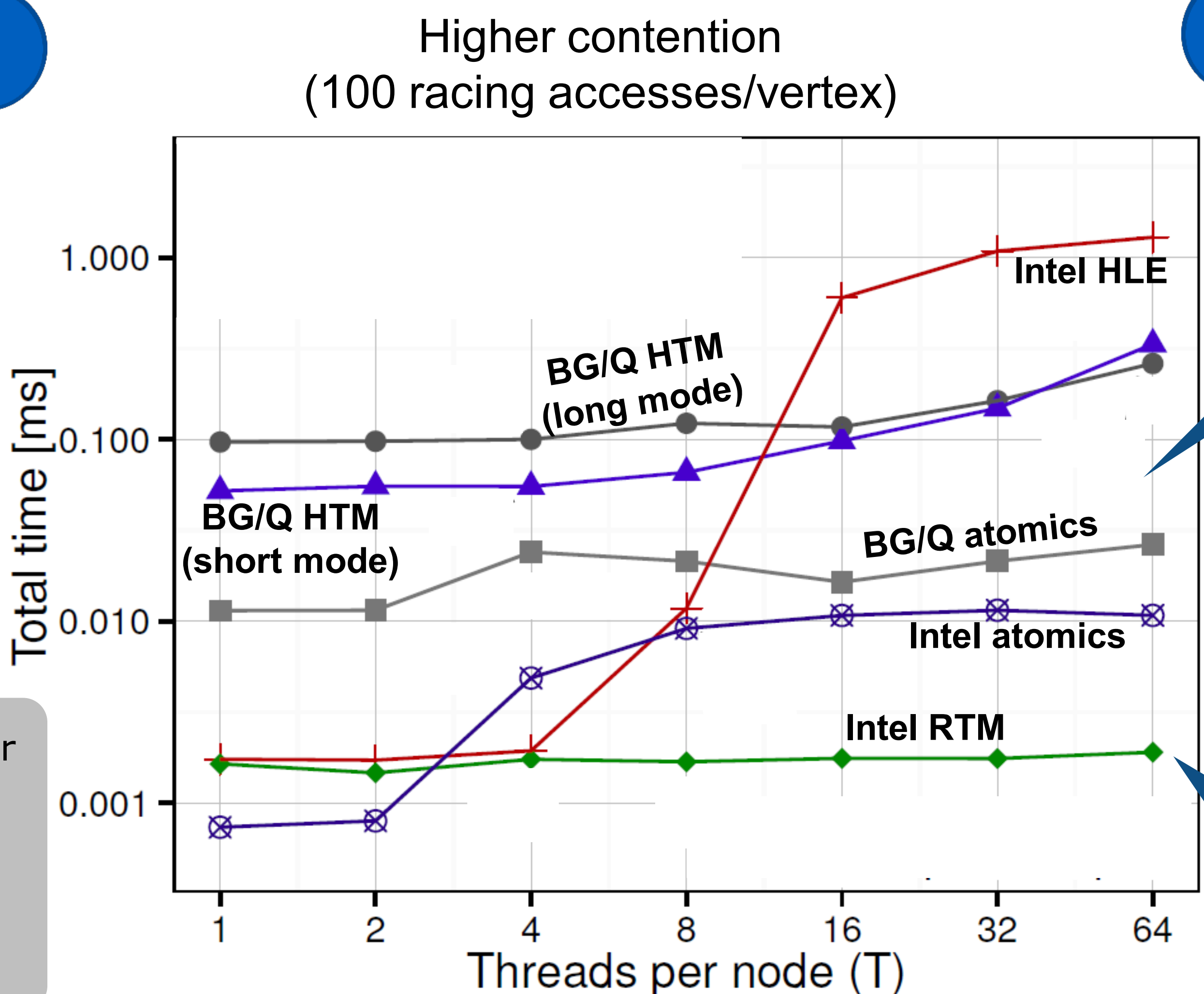
Commit overheads dominate

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Single-Vertex Transactions
Marking a vertex as visited

Used in BFS, SSSP, ...

Still very few aborts



BG/Q HTM still worse (L1 vs L2 matters!)

```
// start handler
if(!v.visited) {
 v.visited = 1;
}
// finish
handler
```

RTM better than atomics

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Single-Vertex Transactions
Incrementing vertex rank

Used in PageRank

IBM
intel

```
// start handler
v.rank++;
// finish handler
```

! Atomics always outperform HTM

! The reason: each transaction always modifies some memory cell, increasing the number of conflicts

Performance Model

Atomics vs transactions

Time to modify *N* vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Overhead per vertex

Startup overheads

Time to modify *N* vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$

Overhead per vertex

Startup overheads

We predict that:

$$B_{AT} < B_{HTM}$$

$$A_{AT} > A_{HTM}$$

Transactions' cost grows slower

Transaction startup overheads dominate

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

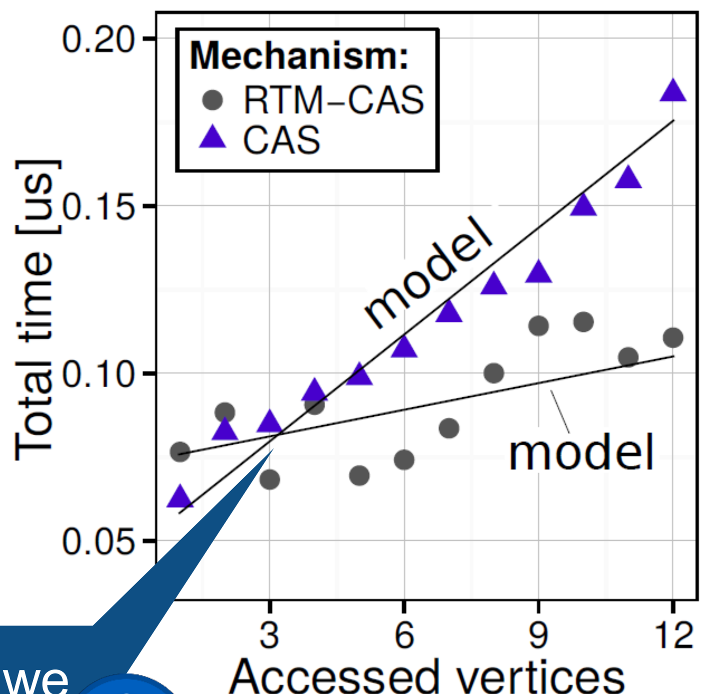
Performance Model
Atomics vs transactions

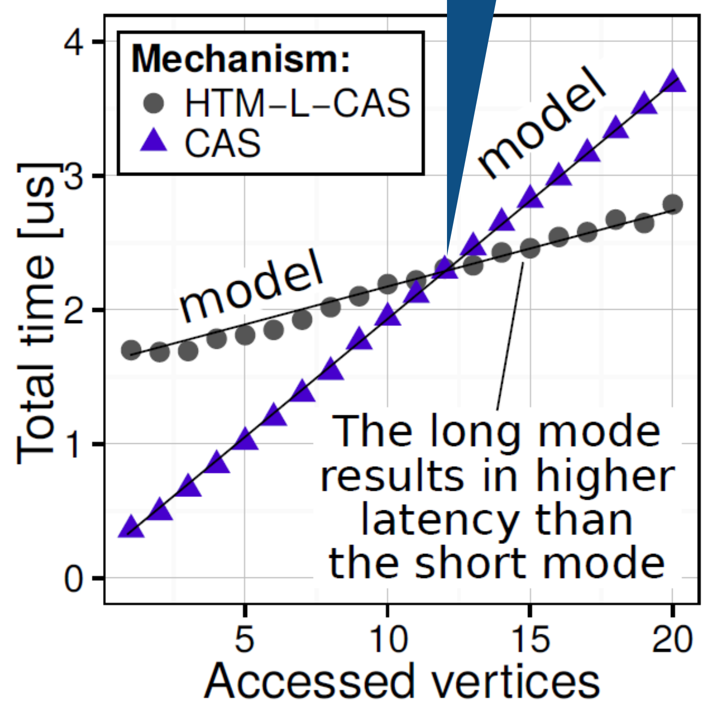
- Can we amortize HTM startup/commit overheads with larger transaction sizes?

Indeed:

$$B_{AT} < B_{HTM}$$

$$A_{AT} > A_{HTM}$$





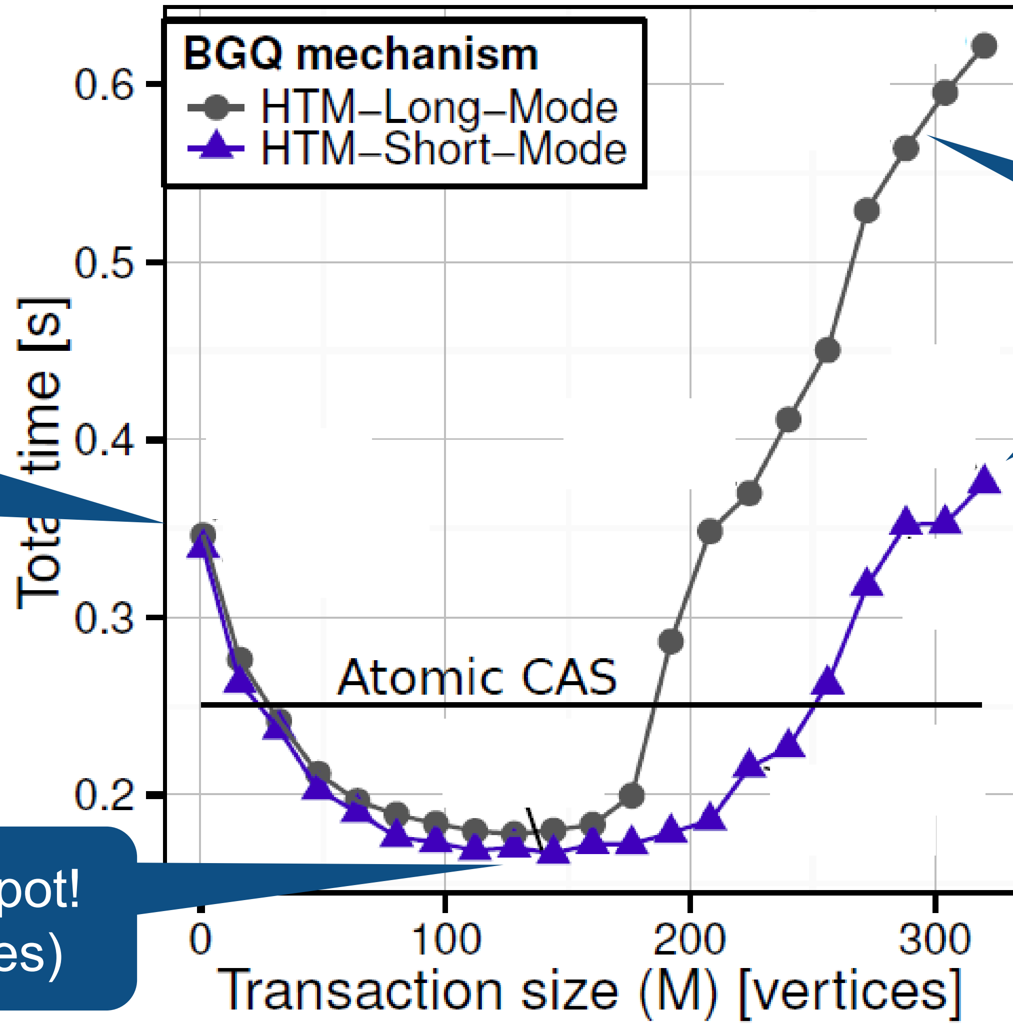
Yes, we can!

Yes, we can!

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

IBM

Multi-Vertex Transactions
Marking vertices as visited



Startup and commit overheads

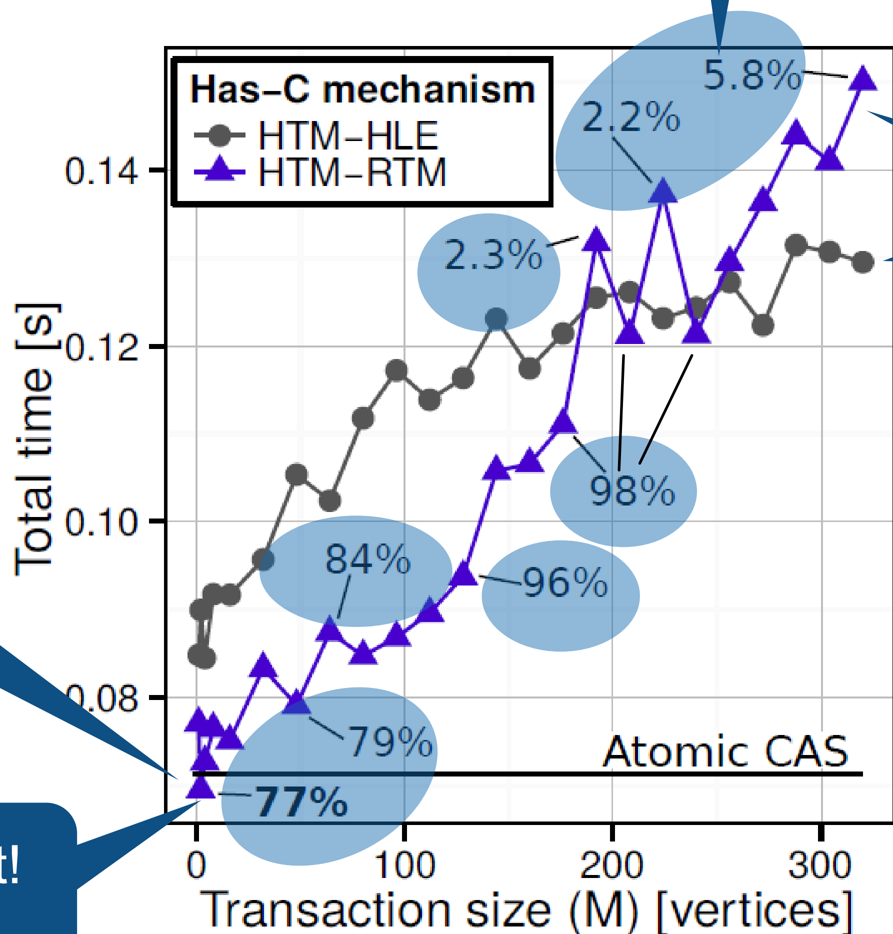
Abort and rollback overheads

The sweetspot! (144 vertices)

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Numbers: % of aborts due to HTM capacity overflows



Abort and rollback overheads

Startup and commit overheads

Majority of aborts are due to HTM capacity overflows (small cache size & associativity)

The sweetspot! (2 vertices)

Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

PERFORMANCE ANALYSIS
QUESTIONS ANSWERED

Larger cache & associativity → fewer aborts & more coarsening

Larger (L2) cache → higher latency

„It really depends" ☺. But... There are some regularities

effectively?

„May fail"

For some algorithms (BFS) HTM is better

„Always succeed"

For others (PageRank) atomics give more performance

M?

AAM establishes a whole hierarchy of algorithms; check the paper ☺

Same for other graphs

Size for BG/Q ~100
>
Size for Haswell ~10

Evaluation
Considered engines

PBGL [4]

Distributed HPC libraries

Galois

[2] Runtimes that exploit amorphous data-parallelism

Graph 500

[1] Hand-tuned algorithm-specific codes

HAMA [3]

Hadoop-based BSP engines

AAM + Graph 500

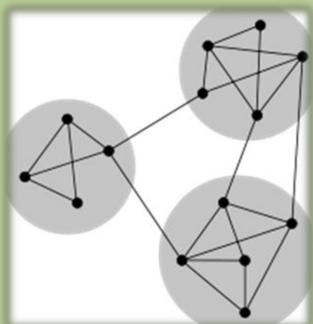
Improving Graph500 design

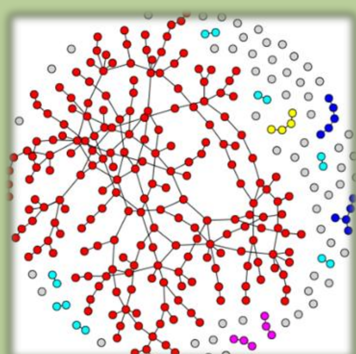
[1] R. Murphy et al. Introducing the Graph 500. CUG'10.
[2] M. Kulkarni et al. Optimistic Parallelism Requires Abstractions. PLDI'07.
[3] S. Seo et al. HAMA: An Efficient Matrix Computation with the MapReduce Framework. CLOUDCOM'10.
[4] D. Gregor and A. Lumsdaine. The parallel BGL: A generic library for distributed graph computations. POOSC'05.

EVALUATION
CONSIDERED TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]





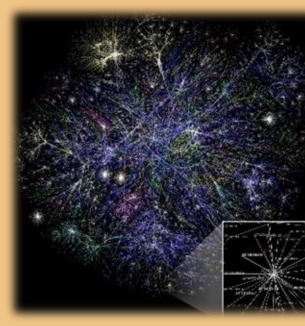
Erdös-Rényi [2]

Real-world SNAP graphs [3]

Road networks

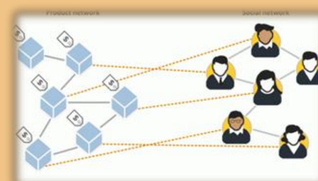
Social networks

Comm. graphs



Web graphs

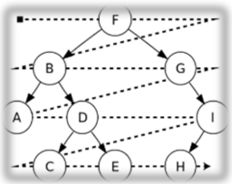
Citation graphs

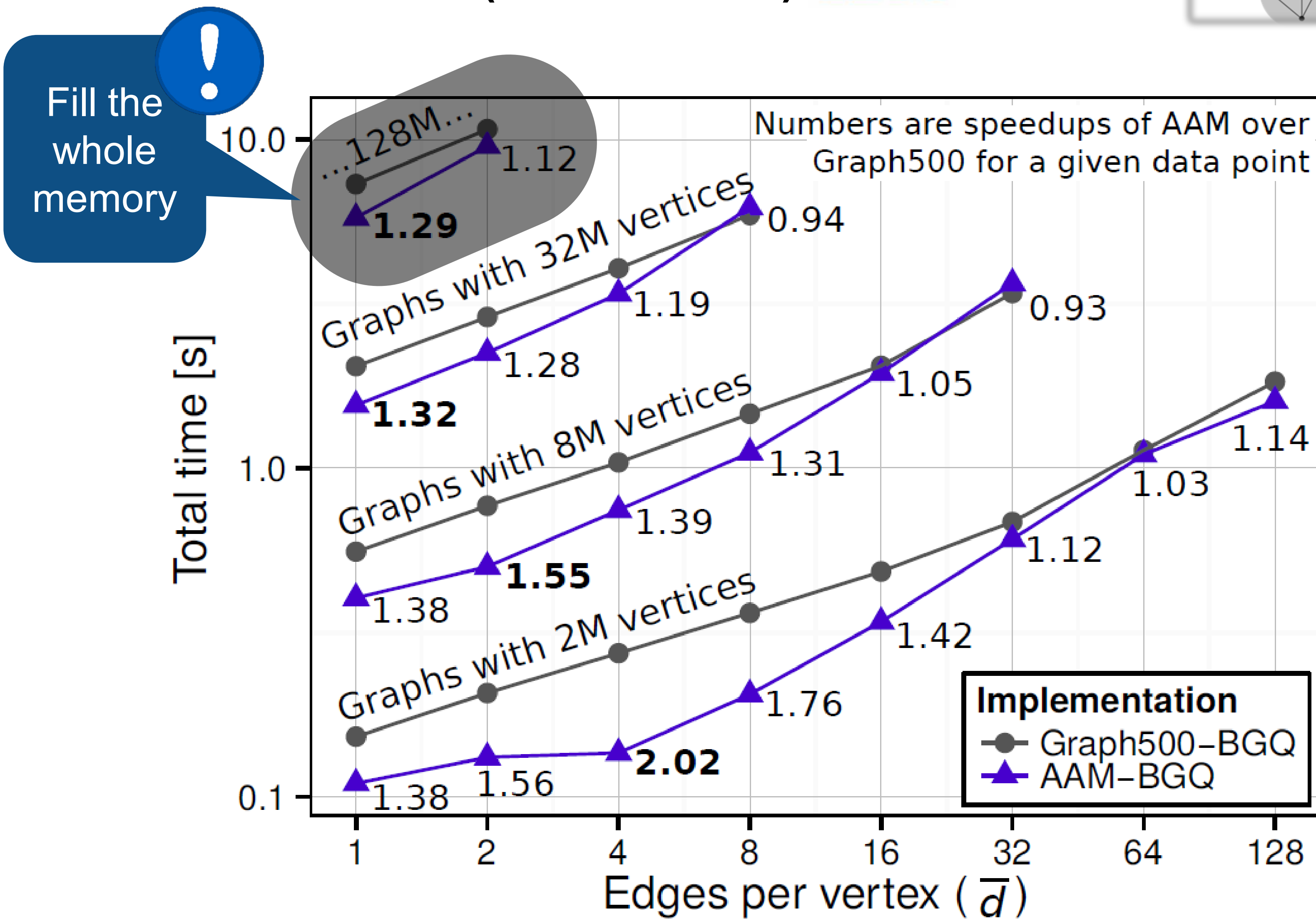


Purchase networks

[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.
[2] P. Erdos and A. Renyi. On the evolution of random graphs. Pub. Math. Inst. Hun. A. Science. 1960.
[3] https://snap.stanford.edu

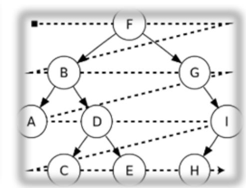
ACCELERATING STATE-OF-THE-ART GRAPH500 + AAM (BLUEGENE/Q) IBM



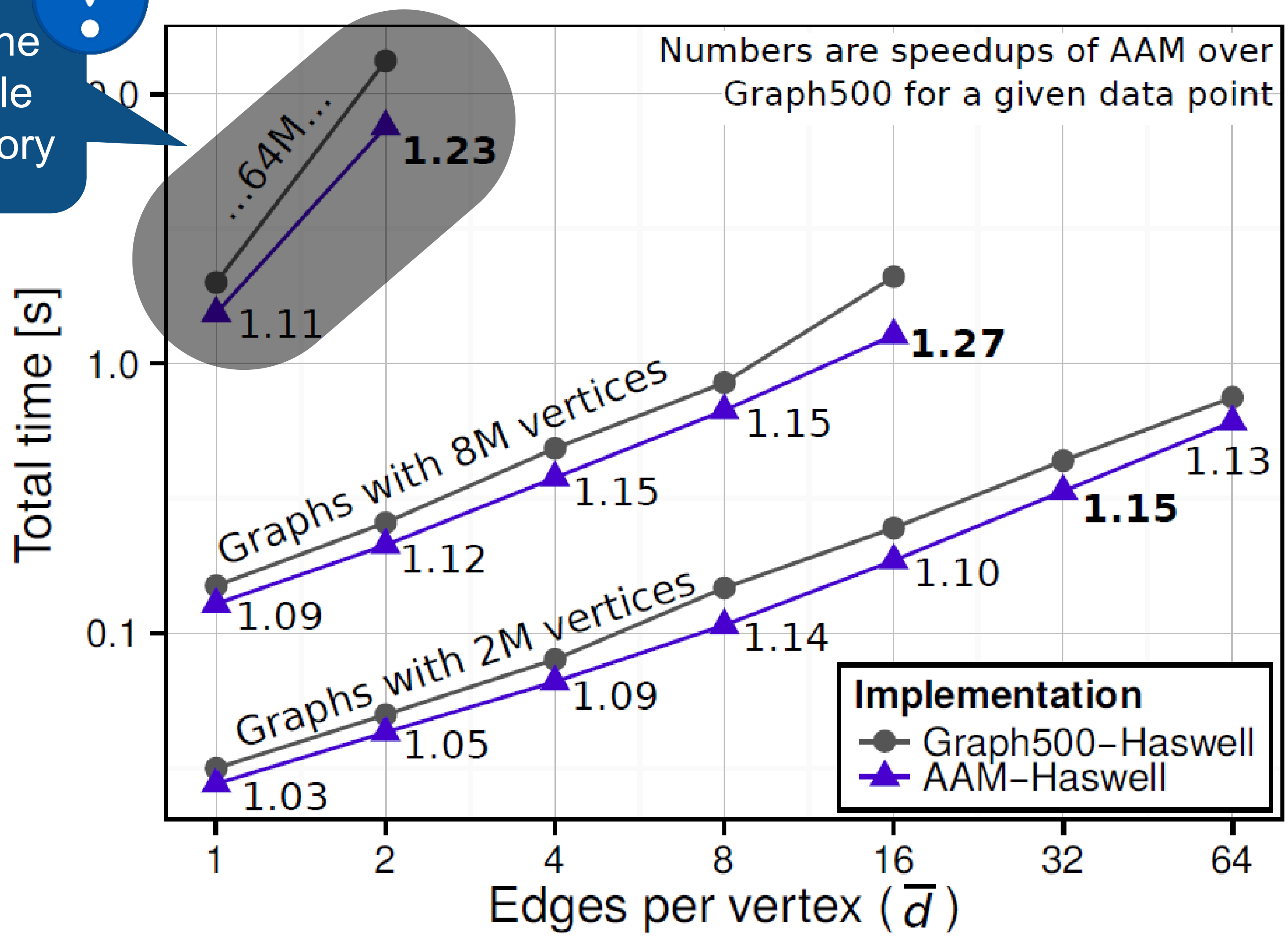


Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (HASWELL)

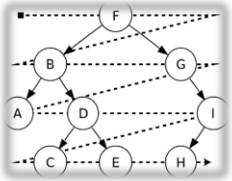


Fill the whole memory



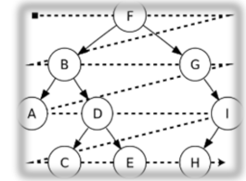
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

OUTPERFORMING STATE-OF-THE-ART



☺ No, you don't have to read it. All details are in the paper. Here: just a summary.

OUTPERFORMING STATE-OF-THE-ART

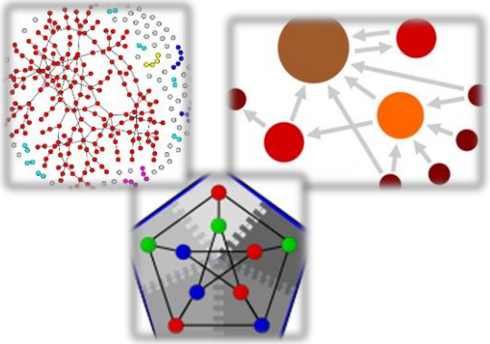


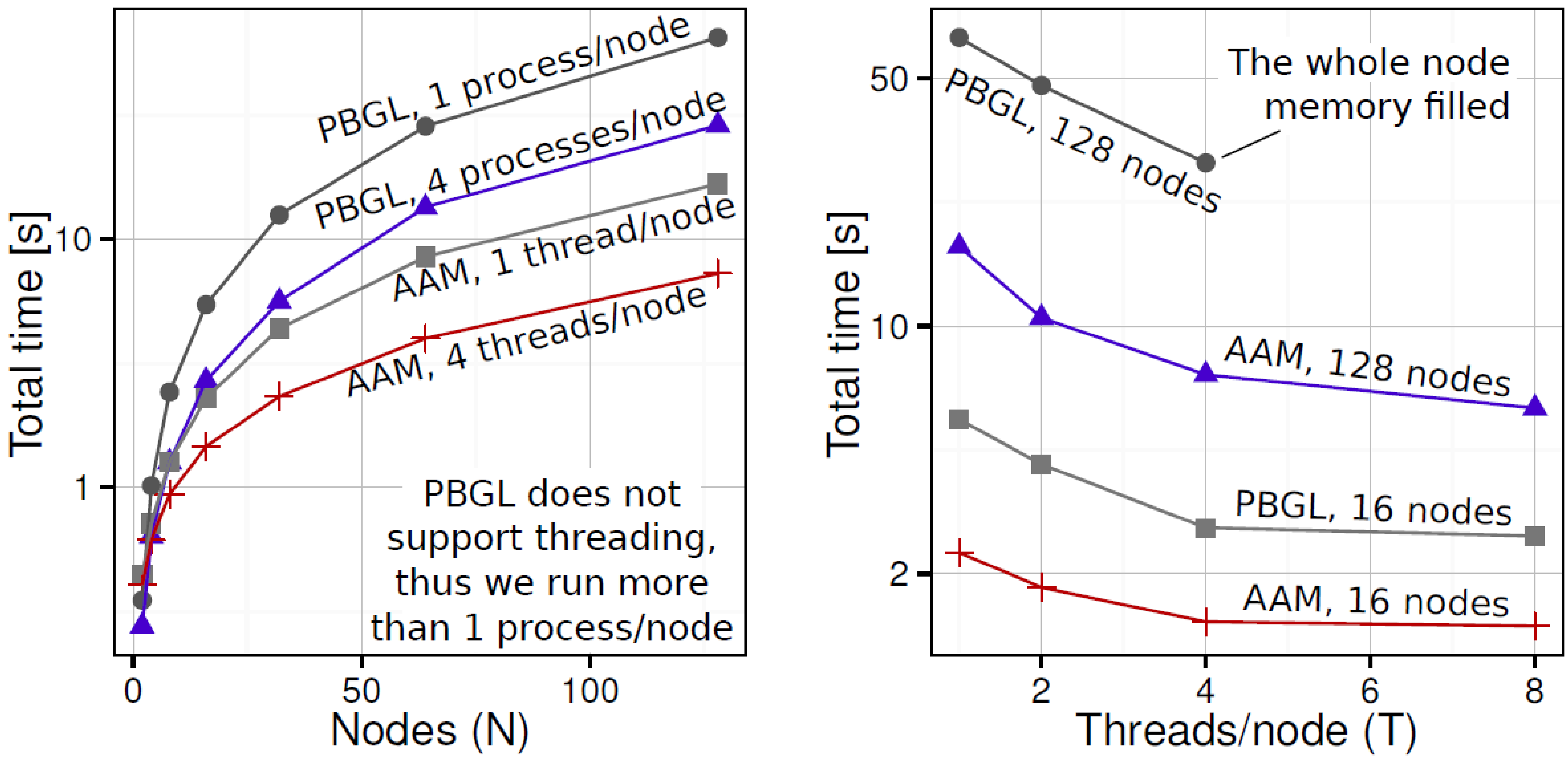
Average overall speedup (geometric mean) over Graph 500: 1.07, Galois: 1.40, HAMA ~1000

1.85x on average, up to 4.3x

Outperforming State-of-the-art

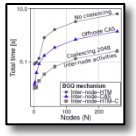
Scalability analysis: distributed-memory

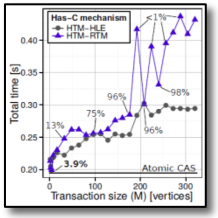


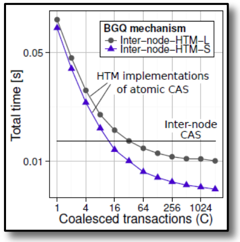


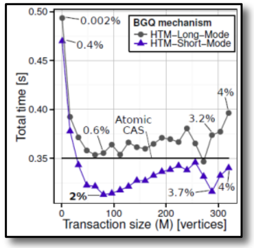
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

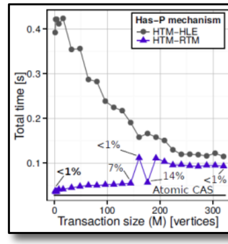
OTHER ANALYSES

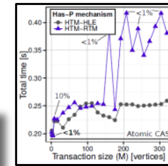


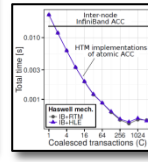


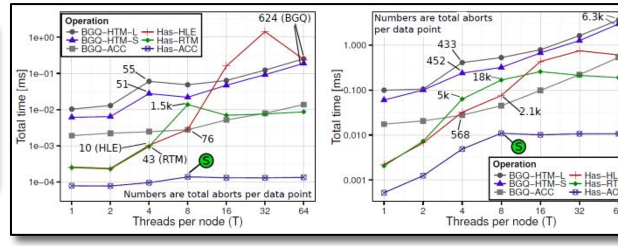


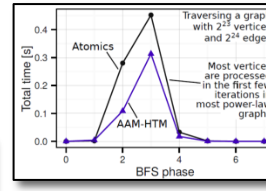


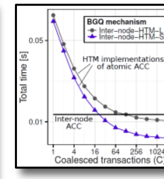


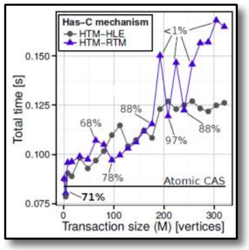


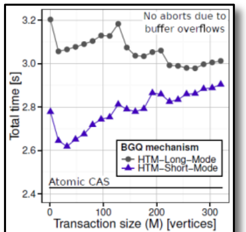


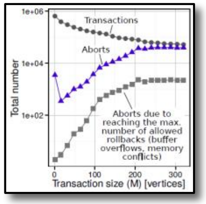


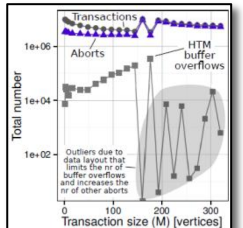


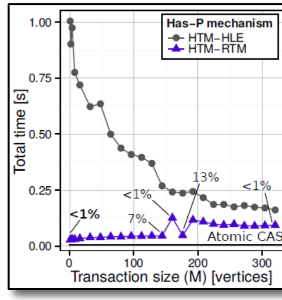




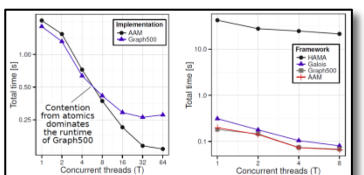


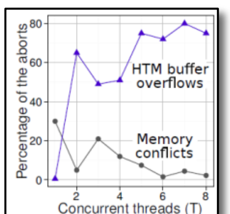


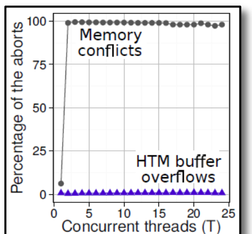


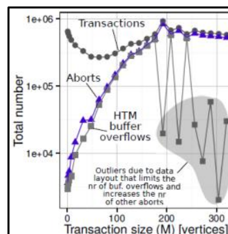


	Aborts due to:		
	Memory conflicts	Buffer overflows	Other reasons
10 ops			
Has-RTM	1,520	1	0
BGQ-HTM-L	624	62	614
BGQ-HTM-S	623	62	613
100 ops			
Has-RTM	18,952	33	0
BGQ-HTM-L	6,374	637	6,360
BGQ-HTM-S	6,392	639	6,380

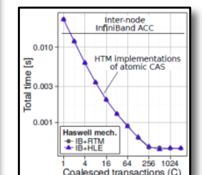


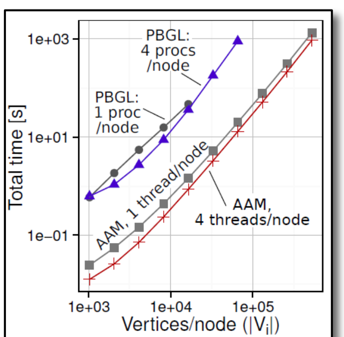


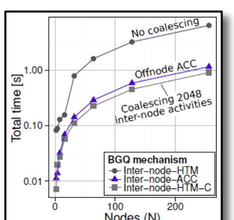




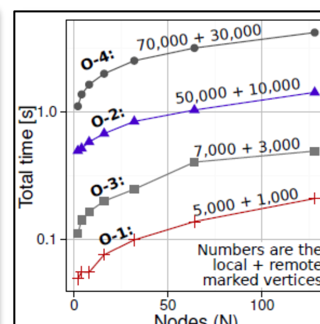
	Aborts due to:		
	Memory conflicts	Buffer overflows	Other reasons
10 ops			
Has-RTM	2	2	0
BGQ-HTM-L	802	3	1
BGQ-HTM-S	1,118	46	180
100 ops			
Has-RTM	2	2	0
BGQ-HTM-L	1,539	5	1
BGQ-HTM-S	2,242	13	2







Input graph properties					BG/Q analysis			Haswell analysis					
Type	ID	Name	$\|V\|$	$\|E\|$	S over g500 ($M = 24$)	M	S over g500	S over g500 ($M = 2$)	S over Galois ($M = 2$)	M	S over g500	S over Galois	S over HAMA
Comm. networks (CNs)	cWT	wiki-Talk	2.4M	5M	2.82	48	3.35	0.91	1.22	6	0.96	1.28	344
	cEU	email-EuAll	265k	420k	3.67	32	4.36	0.76	0.88	4	0.97	1.12	1448
Social networks (SNs)	sLV	soc-LiveJ.	4.8M	69M	1.44	12	1.56	1.05	1.1	3	1.07	1.12	$> 10^4$
	sOR	com-orkut	3M	117M	1.22	20	1.27	1.06	0.69	4	1.13	0.74	$> 10^4$
	sLJ	com-lj	4M	34M	1.44	12	1.54	1.03	1.03	4	1.04	1.04	603
	sYT	com-youtube	1.1M	2.9M	1.67	8	1.84	0.96	1.1	5	0.98	1.11	670
	sDB	com-dblp	317k	1M	1.33	8	1.80	≈1	2.5	2	≈1	2.53	2160
	sAM	com-amazon	334k	925k	1.14	8	1.62	1.04	1.64	2	1.04	1.64	1426
Purchase network (PNs)	pAM	amazon0601	403k	3.3M	1.45	8	1.91	≈1	1.25	3	1.03	1.30	618
Road networks (RNs)	rCA	roadNet-CA	1.9M	5.5M	≈1	2	1.59	1.33	1.74	8	1.38	1.80	$> 10^4$
	rTX	roadNet-TX	1.3M	3.8M	≈1	2	1.53	1.29	1.89	6	1.42	2.08	$> 10^4$
	rPA	roadNet-PA	1M	3M	≈1	2	1.52	≈1	2.00	9	1.07	2.16	$> 10^4$
Citation graphs (CGs)	ciP	cit-Patents	3.7M	16.5M	1.16	8	1.57	1.01	1.26	2	1.01	1.26	1875
Web graphs (WGs)	wGL	web-Google	875k	5.1M	1.78	12	2.08	0.98	1.26	6	1.06	1.35	365
	wBS	web-BerkStan	685k	7.6M	1.91	24	1.91	0.93	1.31	5	1.07	1.40	755
	wSF	web-Stanford	281k	2.3M	1.89	24	1.89	0.98	1.54	5	1.07	1.58	1077

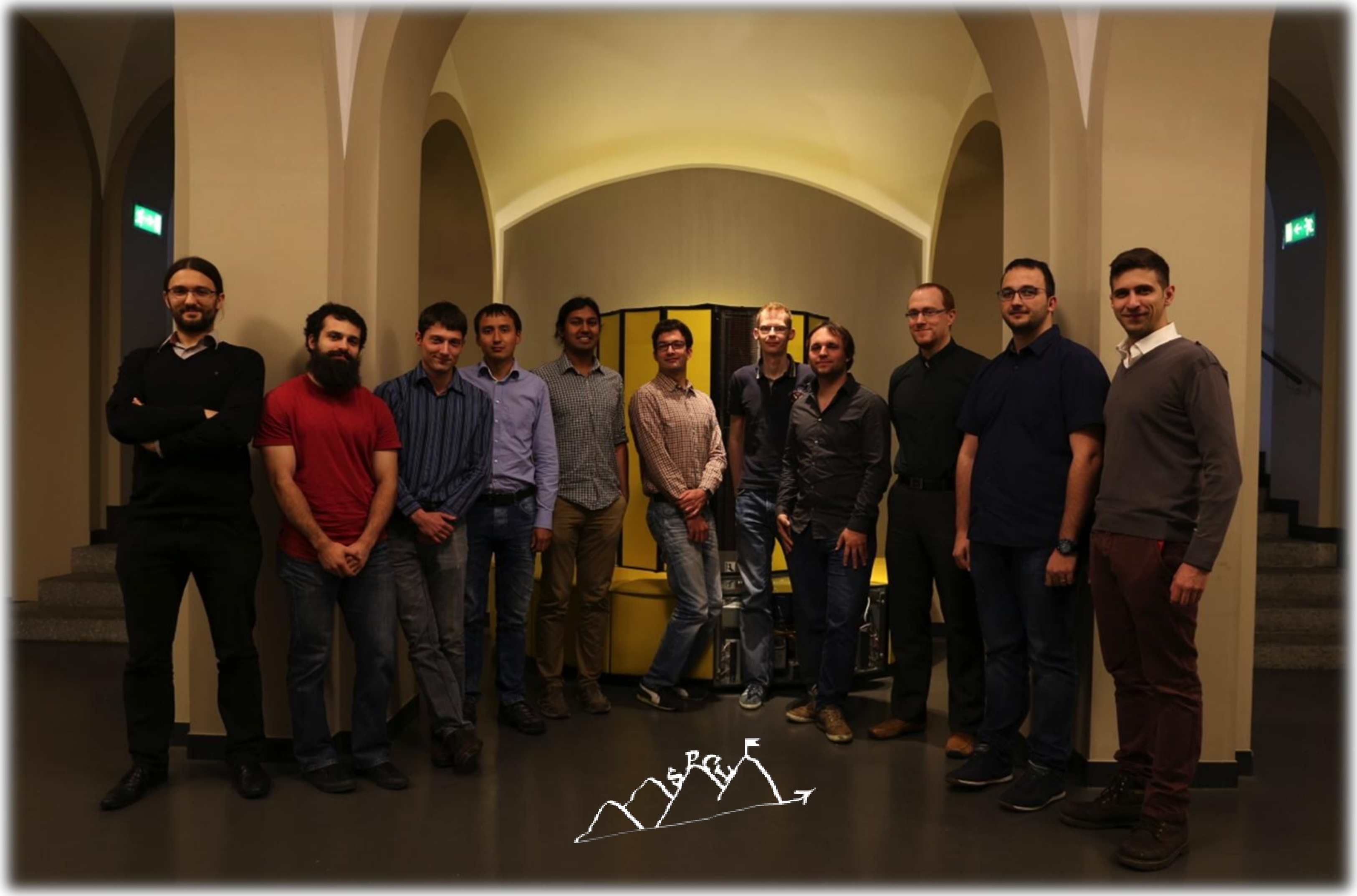


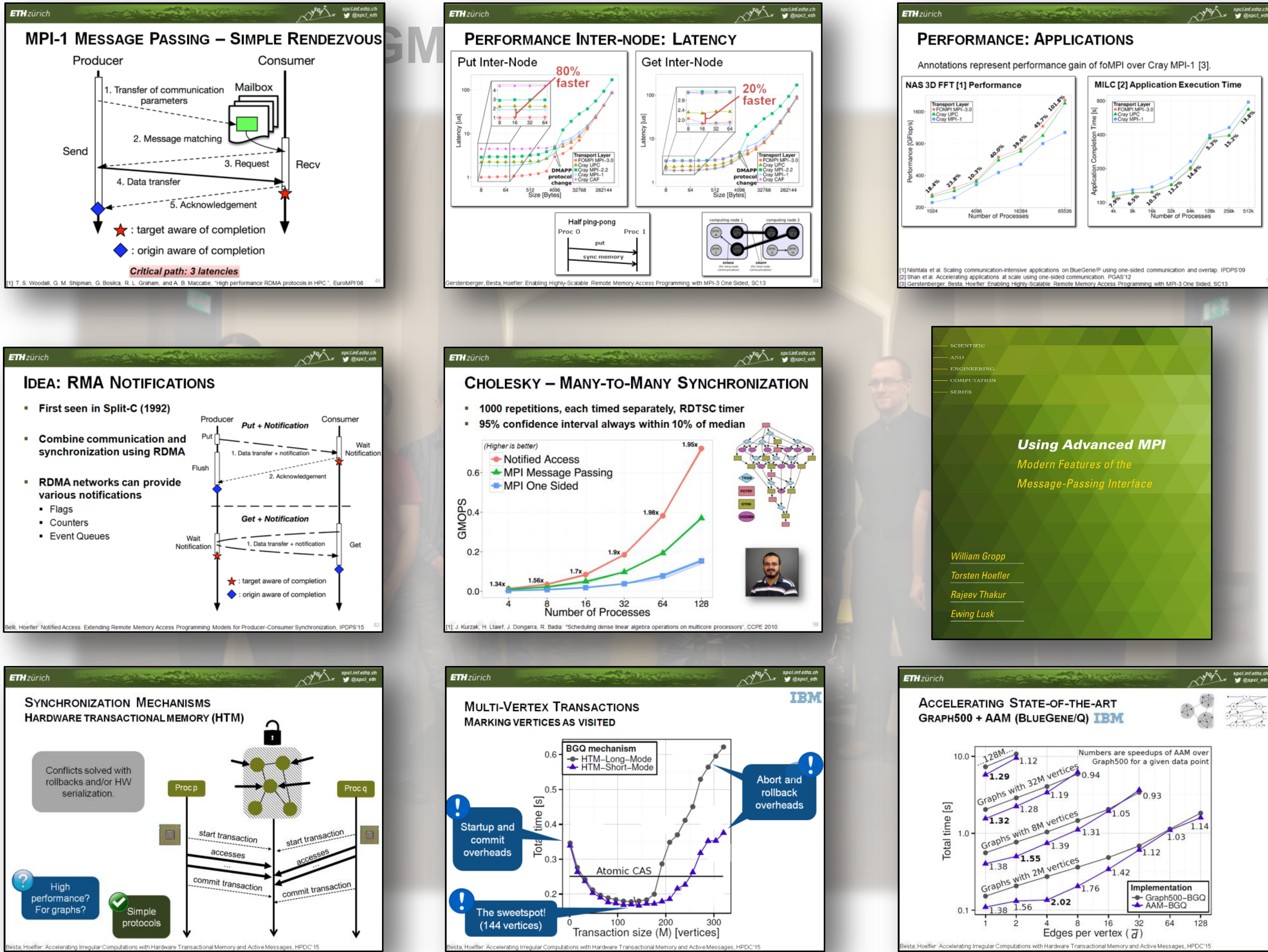
DISCUSSION AND CONCLUSIONS

- **MPI-3 RMA [1] standardizes weak remote memory**
  - Builds on existing practice (UPC, CAF, ARMCI etc.)
  - Rich set of synchronization mechanisms
- **Notified Access [2] can support producer/consumer**
  - Maintains benefits of RDMA
- **HTM can accelerate parallel applications [3]**
  - Uses optimistic coarsened irregular parallelism
- **Atomic Active Messages use HTM on DM systems**
  - First steps towards software-emulated TM over RDMA
  - Thinking about hardware support
- **Significant speedups over highly-tuned graph frameworks**
  - Haswell: 7% over Graph 500, 40% over Galois, 1000x over Hama
- **What next? Discussions?**
  - GraphBlas using RMA+HTM?

[1] T. Hoefler, J. Dinan, R. Thakur, B. Barrett, P. Balaji, W. Gropp, K. Underwood:: Remote Memory Access Programming in MPI-3, TOPC
[2] Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15
[3] Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

ACKNOWLEDGMENTS




MPI-1 Message Passing – Simple Rendezvous
Producer
Consumer
1. Transfer of communication parameters
Mailbox
2. Message matching
Send
3. Request
Recv
4. Data transfer
5. Acknowledgement
: target aware of completion
: origin aware of completion
Critical path: 3 latencies
[1] T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06
Performance Inter-node: Latency
Put Inter-Node
80% faster
Get Inter-Node
20% faster
Half ping-pong
Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13
Performance: Applications
Annotations represent performance gain of foMPI over Cray MPI-1 [3].
NAS 3D FFT [1] Performance
MILC [2] Application Execution Time
Idea: RMA Notifications
First seen in Split-C (1992)
Combine communication and synchronization using RDMA
RDMA networks can provide various notifications
Flags
Counters
Event Queues
Put + Notification
Get + Notification
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15
Cholesky – Many-to-Many Synchronization
1000 repetitions, each timed separately, RDTSC timer
95% confidence interval always within 10% of median
(Higher is better)
Notified Access
MPI Message Passing
MPI One Sided
GMOPS
Number of Processes
[1] J. Kurzak, H. Ltaief, J. Dongarra, R. Badia: "Scheduling dense linear algebra operations on multicore processors", CCPE 2010
SCIENTIFIC AND ENGINEERING COMPUTATION SERIES
Using Advanced MPI
Modern Features of the Message-Passing Interface
William Gropp
Torsten Hoefler
Rajeev Thakur
Ewing Lusk
Synchronization Mechanisms
Hardware Transactional Memory (HTM)
Conflicts solved with rollbacks and/or HW serialization.
Proc p
Proc q
start transaction
accesses
commit transaction
High performance? For graphs?
Simple protocols
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15
Multi-Vertex Transactions
Marking vertices as visited
IBM
BGQ mechanism
HTM–Long–Mode
HTM–Short–Mode
Startup and commit overheads
Abort and rollback overheads
The sweetspot! (144 vertices)
Atomic CAS
Transaction size (M) [vertices]
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15
Accelerating state-of-the-art
Graph500 + AAM (BlueGene/Q) IBM
Numbers are speedups of AAM over Graph500 for a given data point
Graphs with 32M vertices
Graphs with 8M vertices
Graphs with 2M vertices
Total time [s]
Edges per vertex
Implementation
Graph500–BGQ
AAM–BGQ
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15