T. HOEFLER

RDMA, Scalable MPI-3 RMA, and Next-Generation Post-RDMA Interconnects

HPC Advisory Council Swiss Workshop, Lugano, Switzerland

WITH HELP OF ROBERT GERSTENBERGER, MACIEJ BESTA, S. DI GIROLAMO, K. TARANOV, R. E. GRANT, R. BRIGHTWELL AND ALL OF SPCL

The Development of High-Performance Networking Interfaces

Scalable Coherent Interface   Myrinet GM+MX   Virtual Interface Architecture   OFED   libfabric

Ethernet+TCP/IP   Fast Messages   Quadrics QsNet   Cray Gemini   IB Verbs   Portals 4

1980   1990   2000   2010   2020

sockets   (active) message based   protocol offload   remote direct memory access (RDMA)

coherent memory access   OS bypass   zero copy   triggered operations

InfiniBand Trade Association Launches the RoCE Initiative to Advance RDMA over Converged Ethernet Solutions

RoCE delivers significant performance and efficiency gains to cloud, storage, virtualization and hyper-converged infrastructures

businessinsider.com

Microsoft to Drive RDMA Into Datacenters and Clouds

November 18, 2013 by Timothy Prickett Morgan

RDMA over Ethernet - the Rocky road to convergence

17 November 2015 | By Brandon Hoff

June 2017

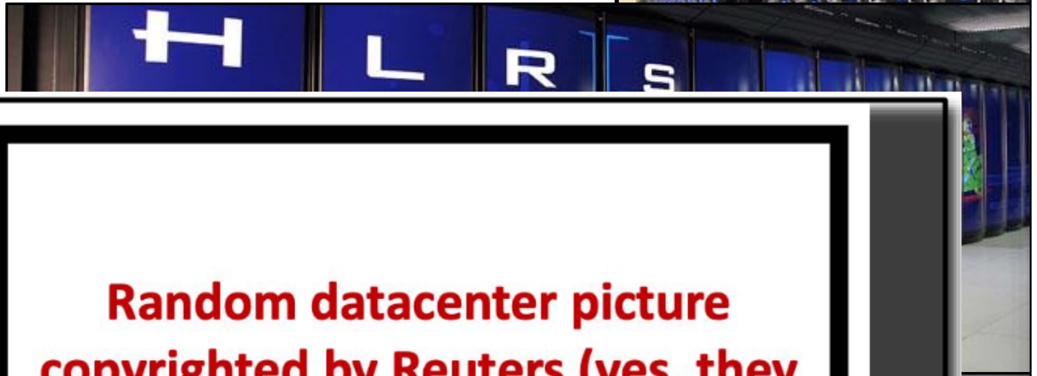
TOP500® SUPERCOMPUTER SITES

95 / top-100 systems use RDMA

>285 / top-500 systems use RDMA

RDMA as MPI-3.0 Remote Memory Access Transport

- MPI-3.0 supports RMA (“MP
  - Designed to react to hardware
  - Majority of HPC networks supp



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

research highlights

Communications of the ACM, October 2018, Vol. 61 No. 10, Pages 106-113
DOI:10.1145/3264413

Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided

By Robert Gerstenberger,* Maciej Besta, and Torsten Hoefler

Abstract
Modern high-performance networks offer remote direct memory access (RDMA) that exposes a process' virtual address space to other processes in the network. The *Message Passing Interface* (MPI) specification has recently been extended with a programming interface called MPI-3 *Remote Memory Access* (MPI-3 RMA) for efficiently exploiting state-of-the-art RDMA features. MPI-3 RMA enables a powerful programming model that alleviates many message passing downsides. In this work, we design and develop bufferless protocols that demonstrate how to implement this interface and support scaling to millions of cores with negligible memory consumption while providing highest performance and minimal overheads. To arm programmers, we provide a spectrum of performance models for RMA functions that enable rigorous mathematical analysis of application performance and facilitate the development of codes that solve given tasks within specified time and energy budgets. We validate the usability of our library and models with several application studies with up to half a million processes. In a wider sense, our work illustrates how to use RMA principles to accelerate computation- and data-intensive codes.

1. INTRODUCTION
Supercomputers have driven the progress of various soci-

Supercomputers consist of massively parallel nodes, each supporting up to hundreds of hardware threads in a single shared-memory domain. Up to tens of thousands of such nodes can be connected with a high-performance network, providing large-scale distributed-memory parallelism. For example, the Blue Waters machine has >700,000 cores and a peak computational bandwidth of >13 petaflops.

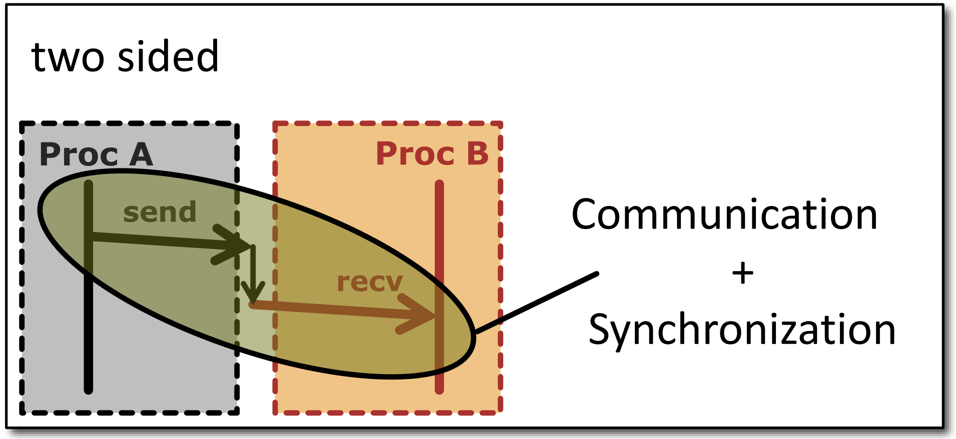
Programming such large distributed computers is far from trivial: an ideal programming model should tame the complexity of the underlying hardware and offer an easy abstraction for the programmer to facilitate the development of high-performance codes. Yet, it should also be able to effectively utilize the available massive parallelism and various heterogeneous processing units to ensure highest scalability and speedups. Moreover, there has been a growing need for the support for *performance modeling*: a rigorous mathematical analysis of application performance. Such formal reasoning facilitates developing codes that solve given tasks within the assumed time and energy budget.

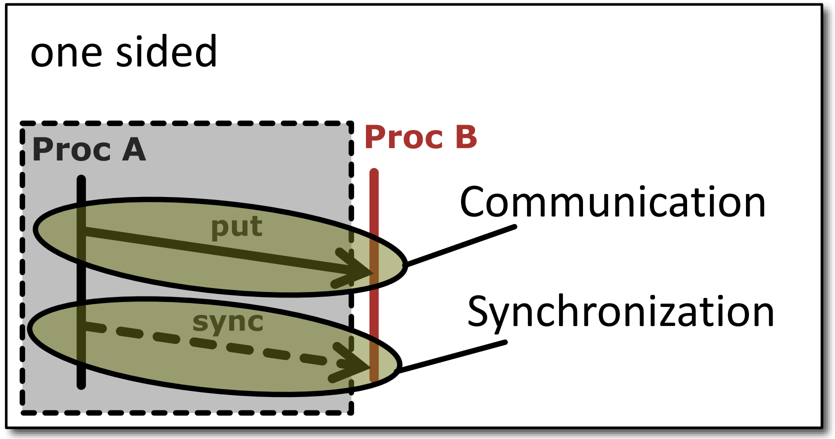
The *Message Passing Interface* (MPI)[11] is the *de facto* standard API used to develop applications for distributed-memory supercomputers. MPI specifies message passing as well as remote memory access semantics and offers a rich set of features that facilitate developing highly scalable and portable codes; message passing has been the prevalent model so far. MPI's message passing specification does not prescribe specific ways how to exchange messages and thus enables flex-

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

MPI-3.0 Remote Memory Access

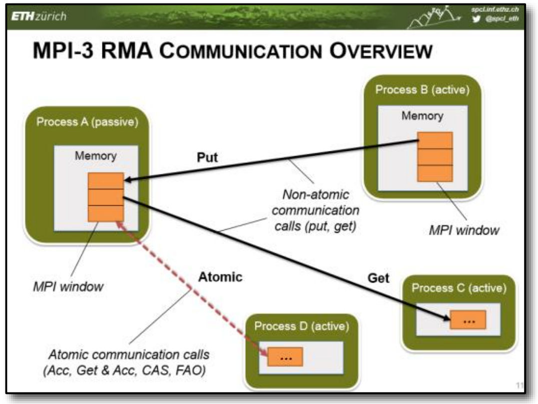
- MPI-3.0 supports RMA ("MPI One Sided")
  - Designed to react to hardware trends
  - Majority of HPC networks support RDMA
- Communication is „one sided" (no involvement of destination)
- RMA decouples communication & synchronization
  - Different from message passing



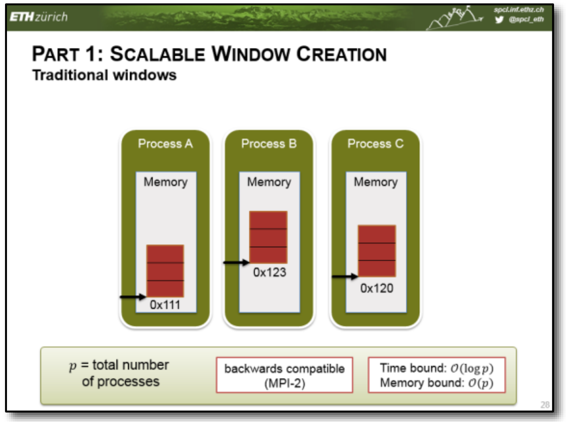


[1] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

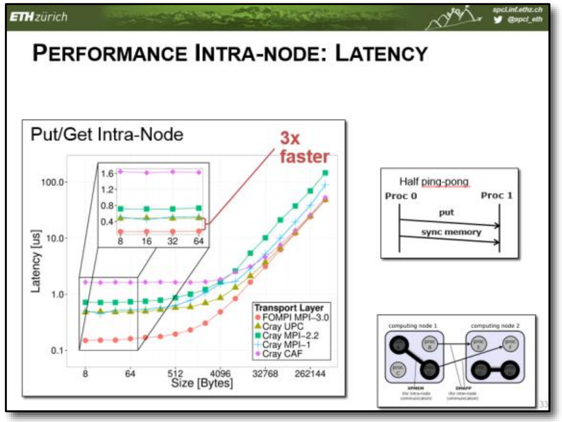
Presentation Overview



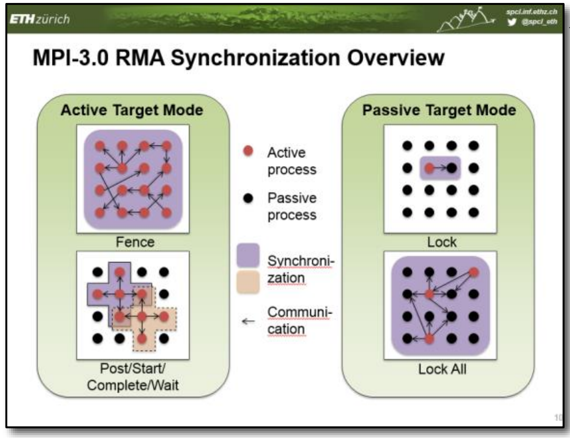
1. Overview of three MPI-3 RMA concepts



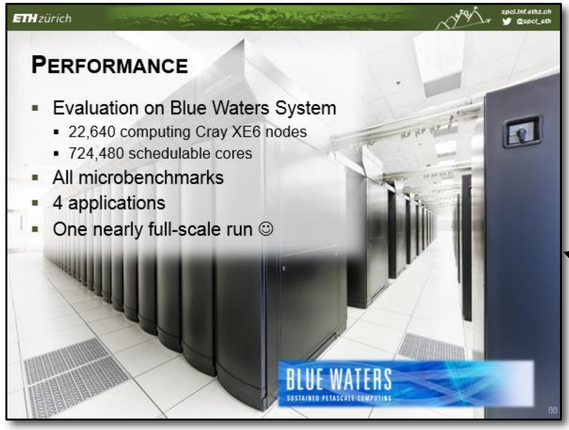
2. MPI window creation



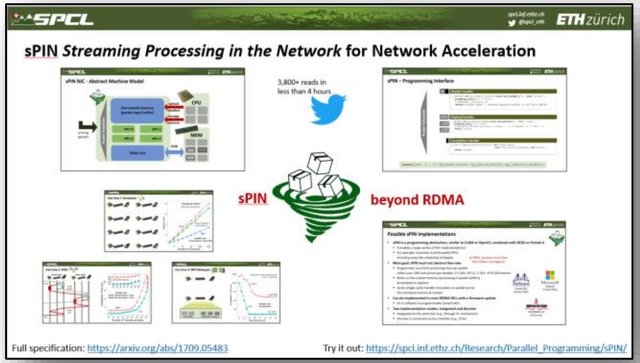
3. Communication



4. Synchronization



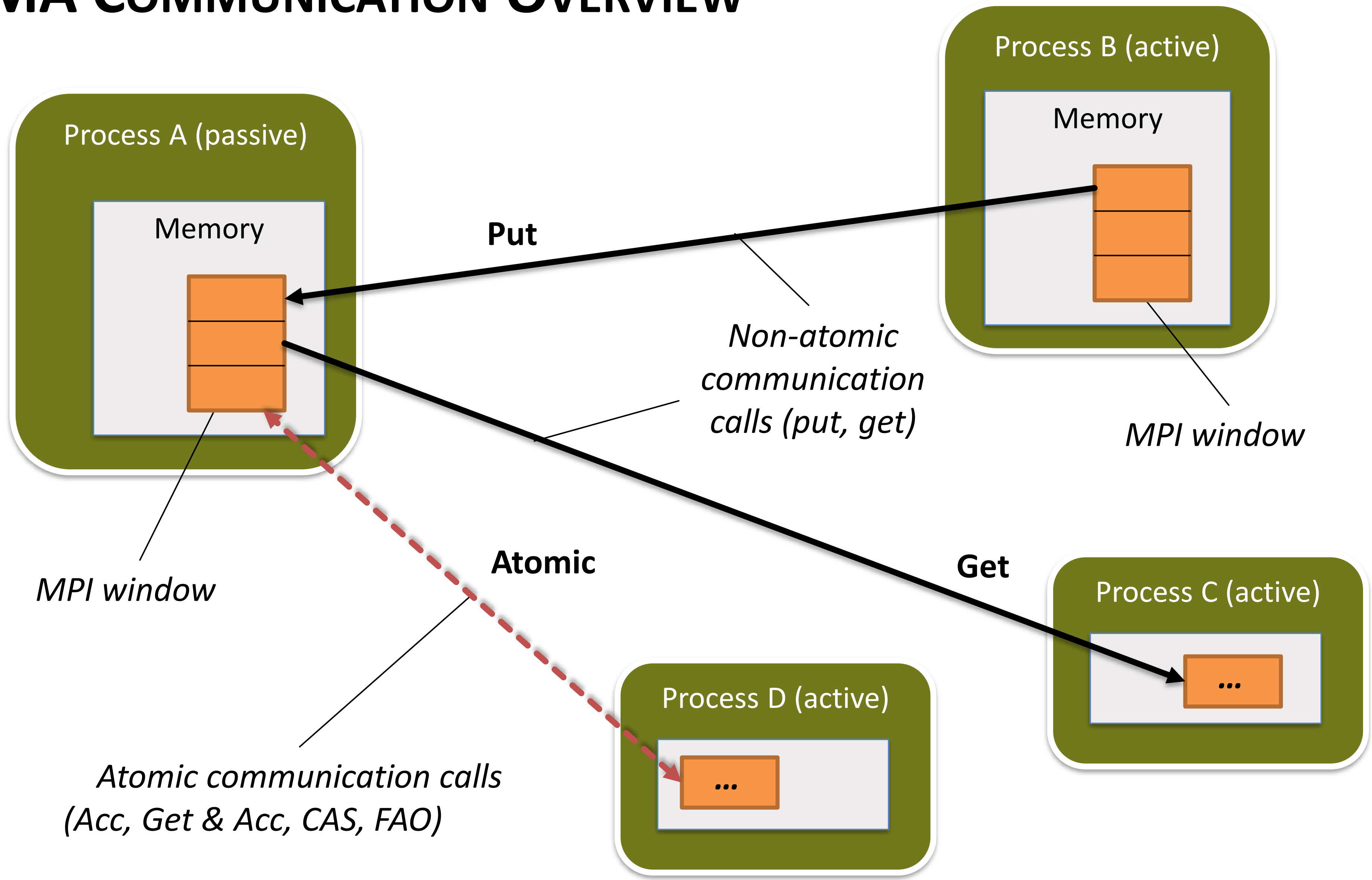
5. Application evaluation



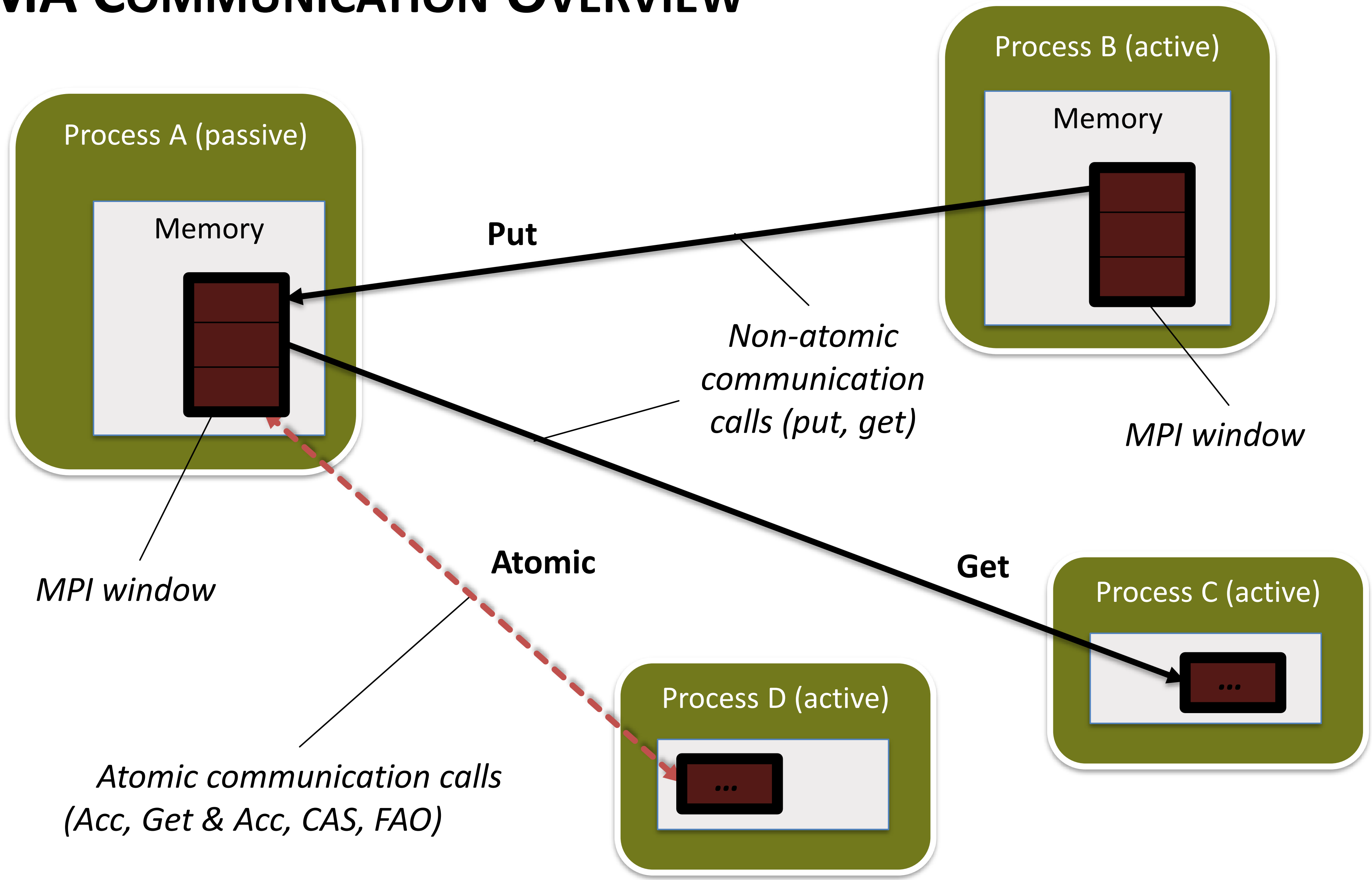
6. Post-RDMA networking

Outlook!

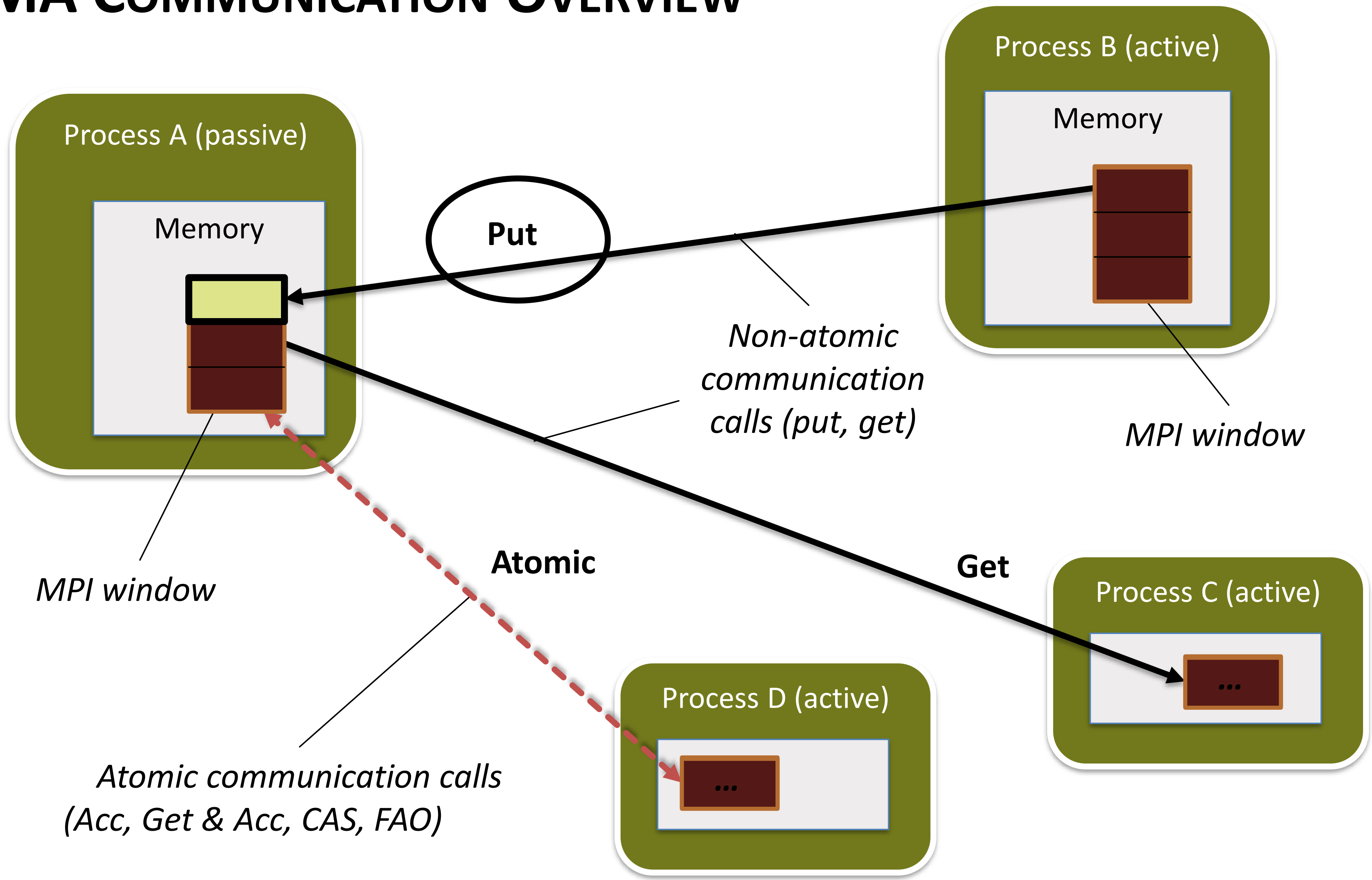
MPI-3 RMA Communication Overview



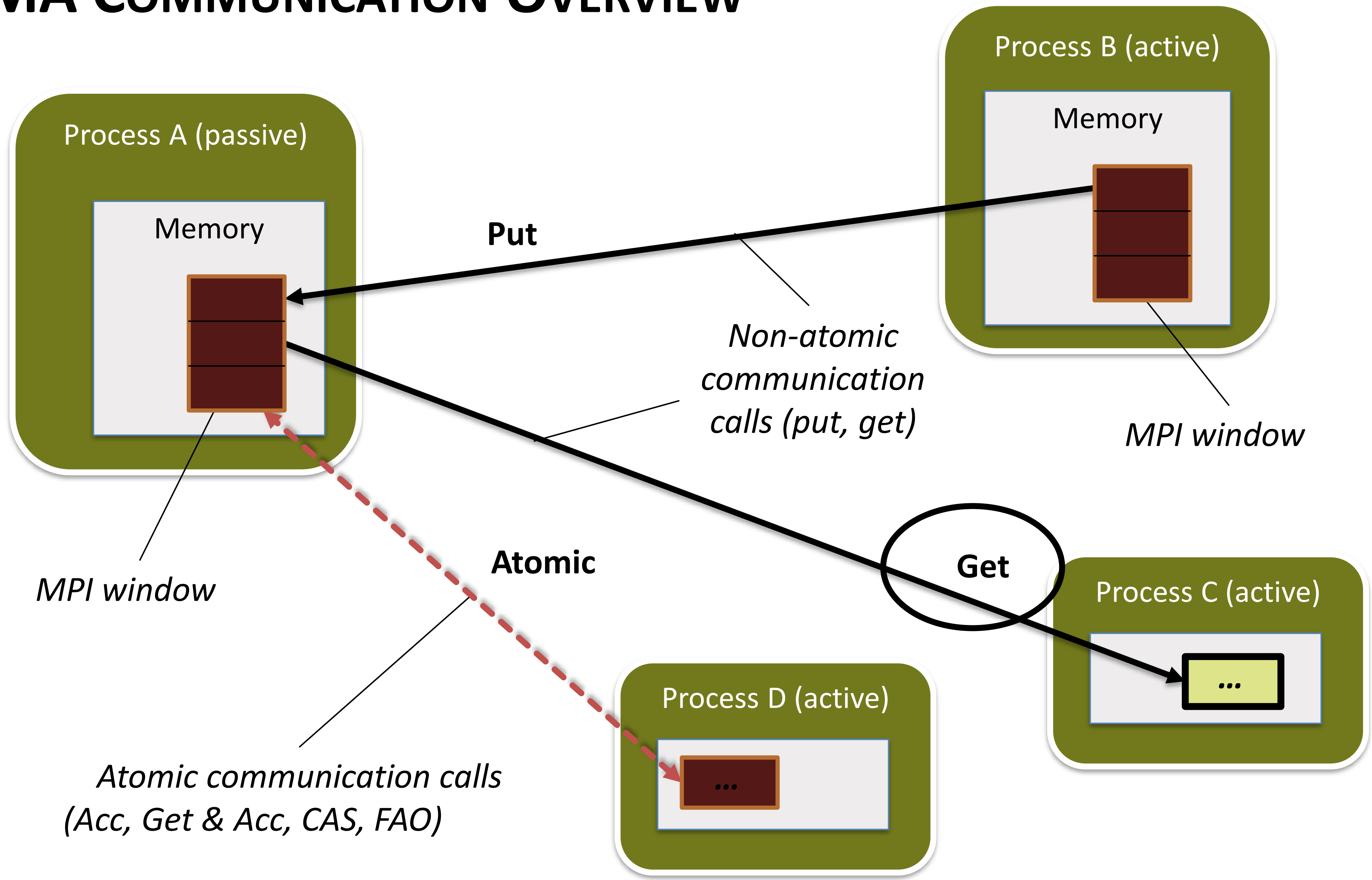
MPI-3 RMA Communication Overview



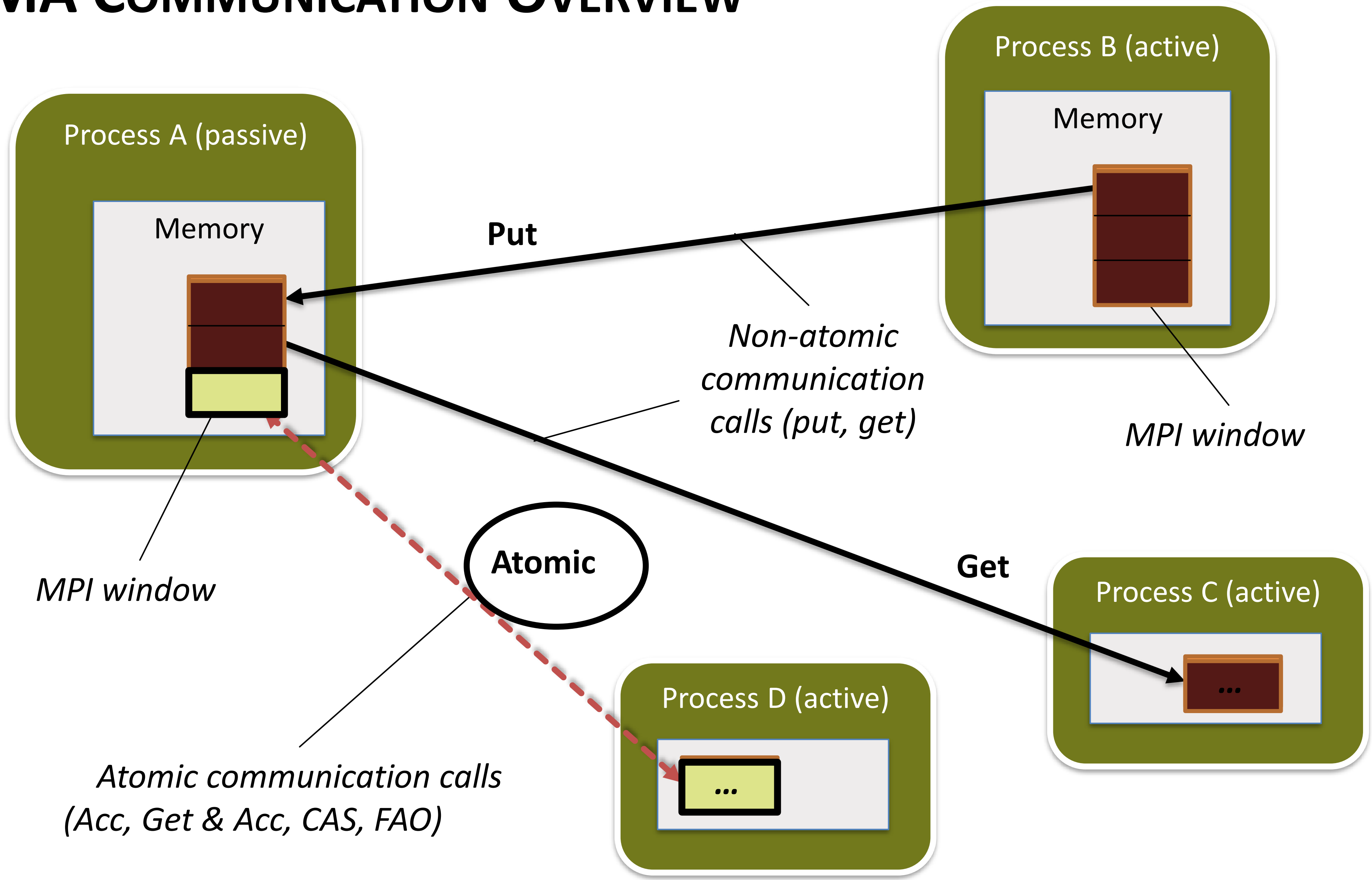
MPI-3 RMA Communication Overview



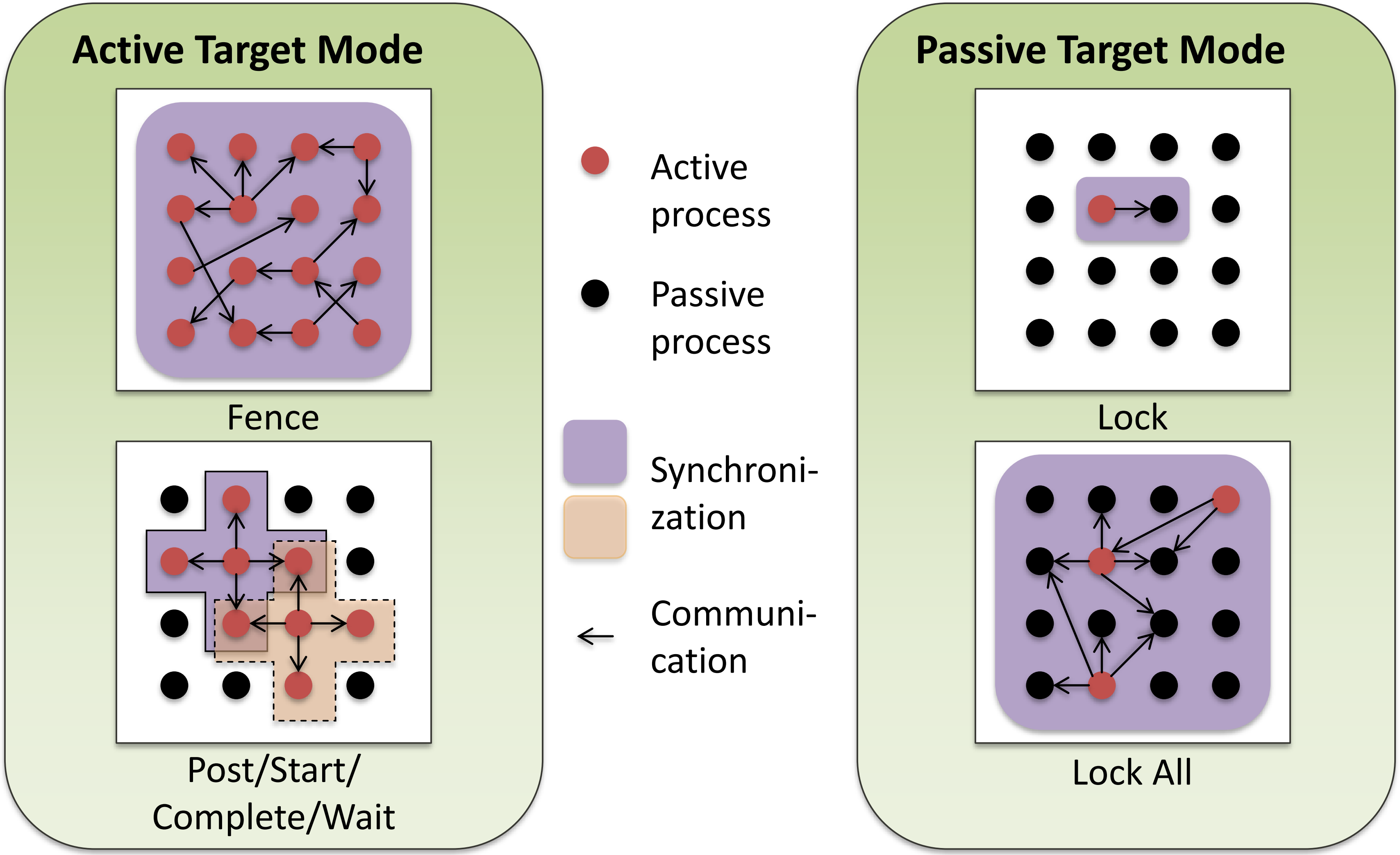
MPI-3 RMA Communication Overview



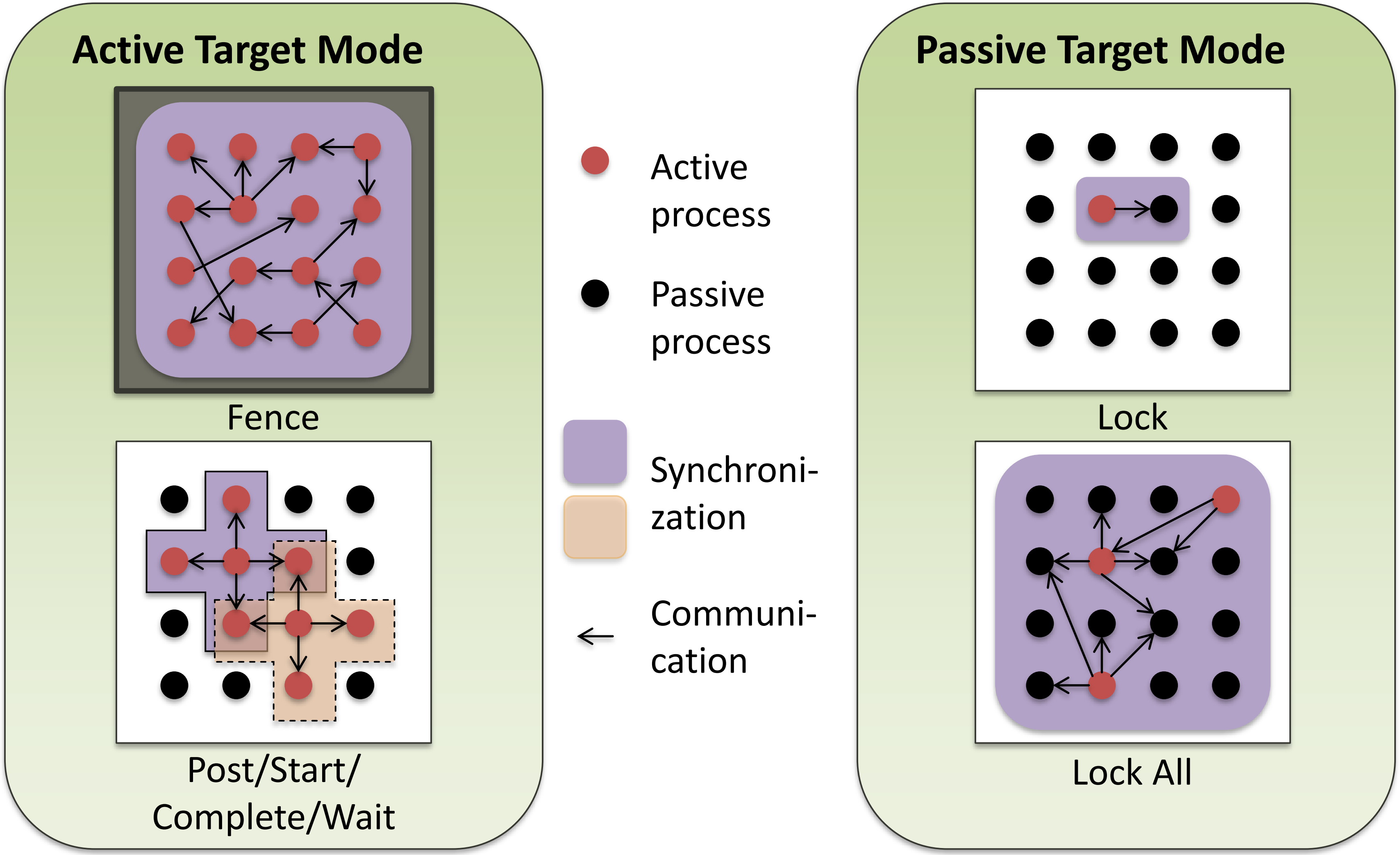
MPI-3 RMA Communication Overview



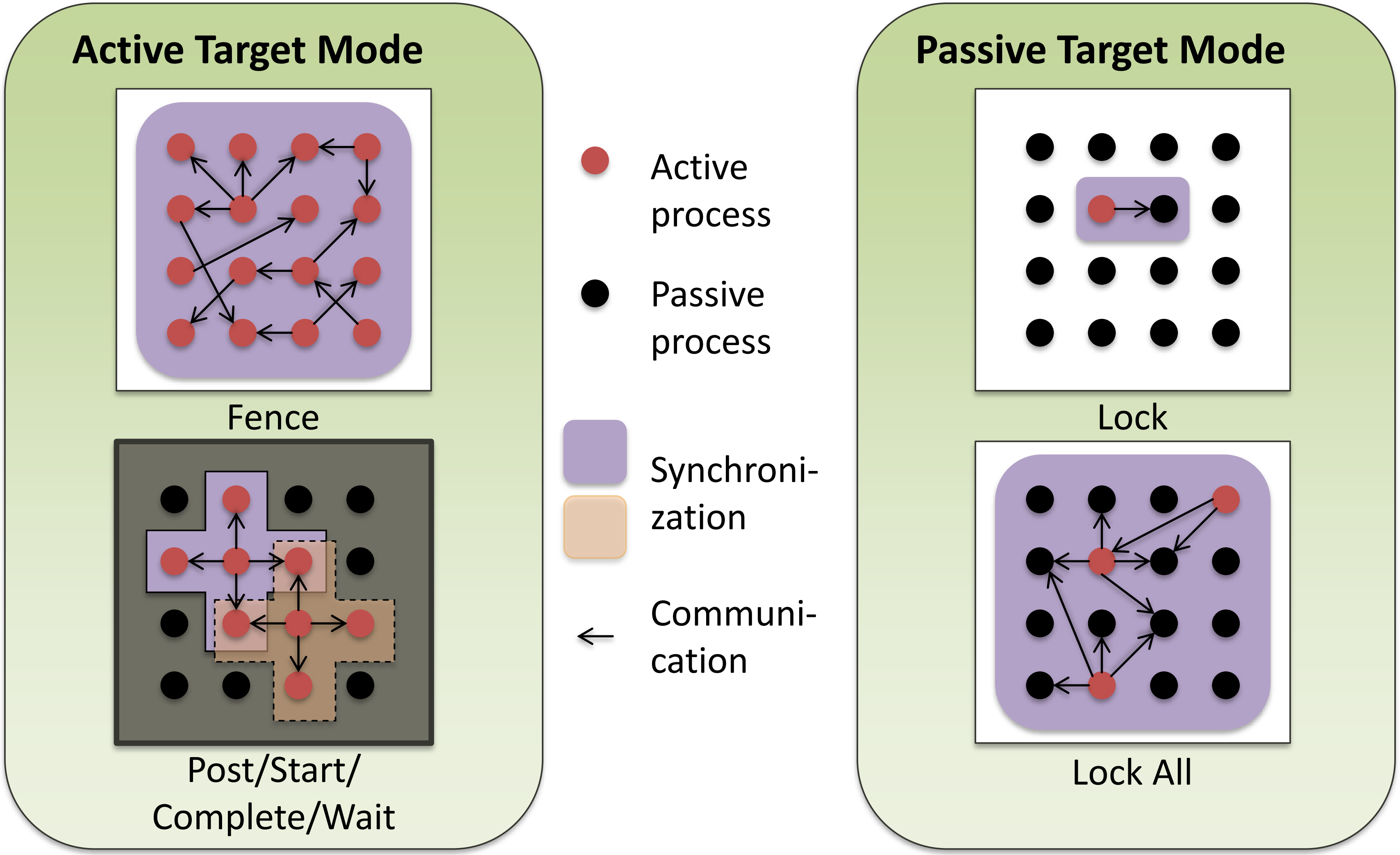
MPI-3.0 RMA Synchronization Overview



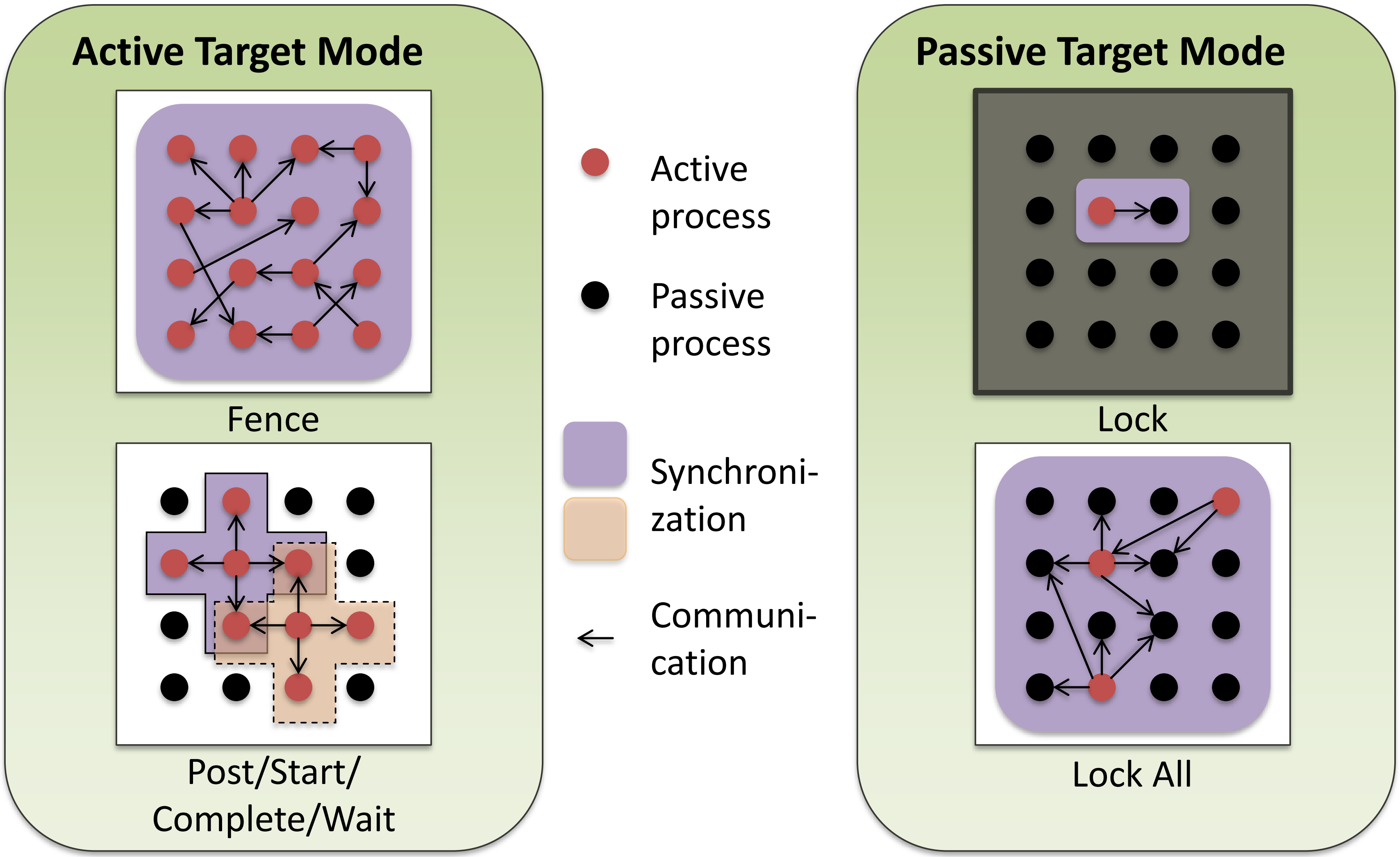
MPI-3.0 RMA Synchronization Overview



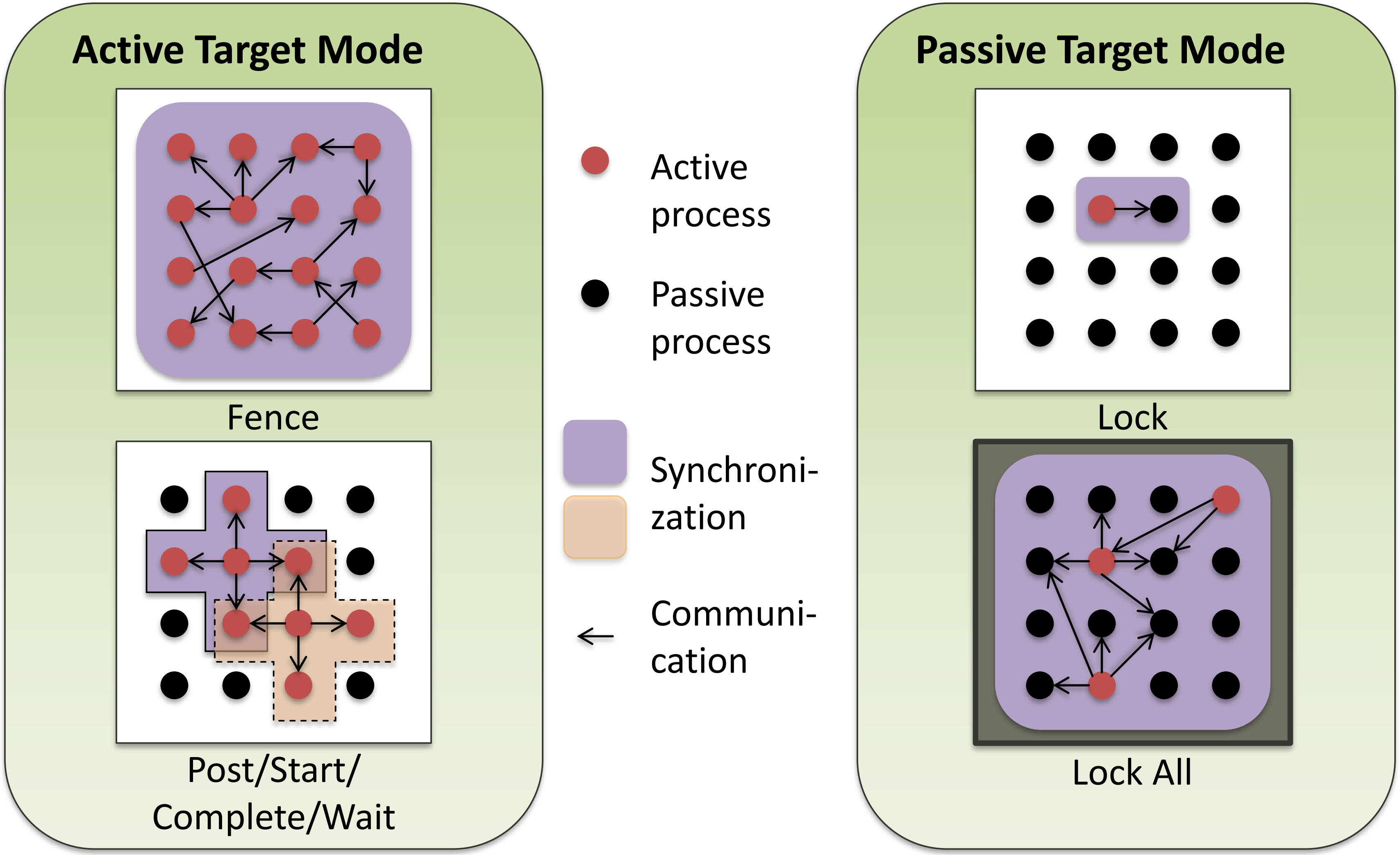
MPI-3.0 RMA Synchronization Overview



MPI-3.0 RMA Synchronization Overview



MPI-3.0 RMA Synchronization Overview



Scalable Protocols & Reference Implementation

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

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

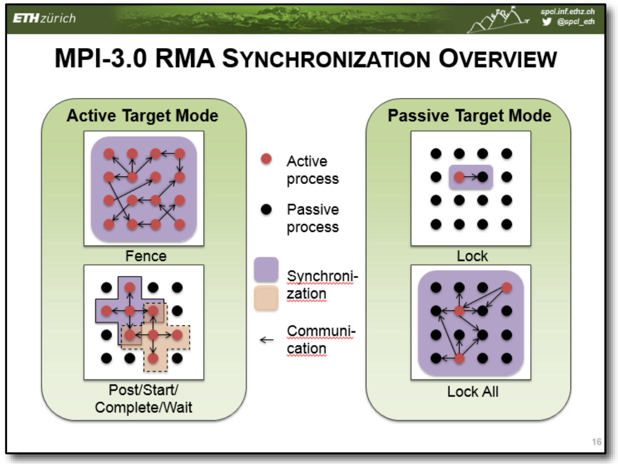
Scalable Protocols & Reference Implementation

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

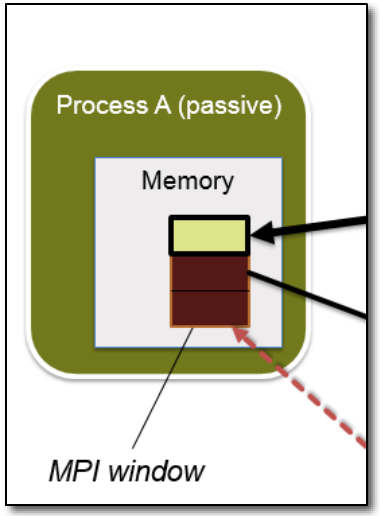
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Protocols & Reference Implementation

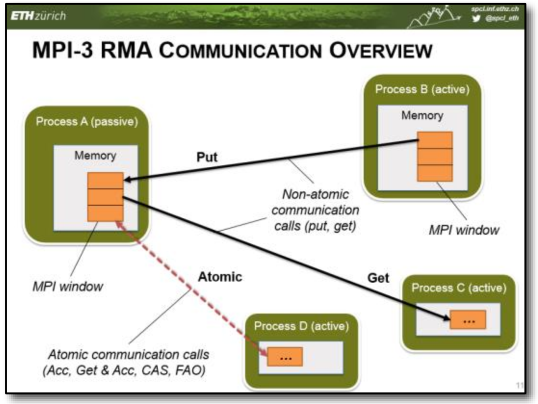
- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization



Synchronization



Window creation

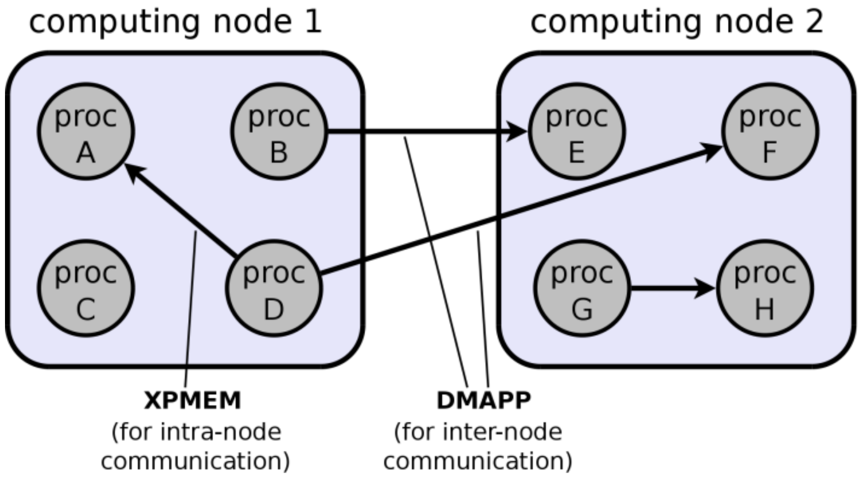


Communication

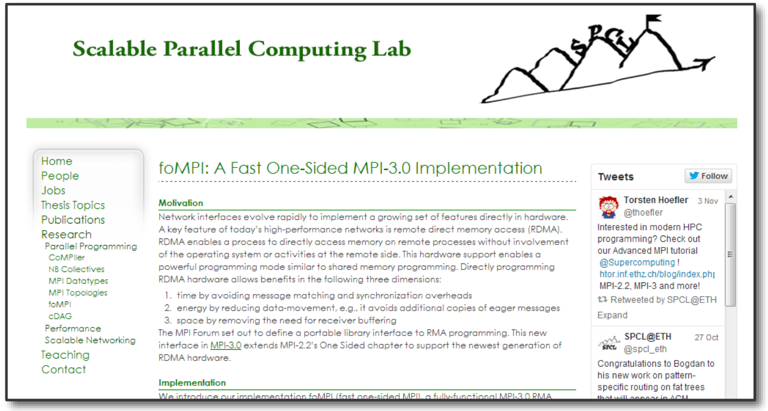
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

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



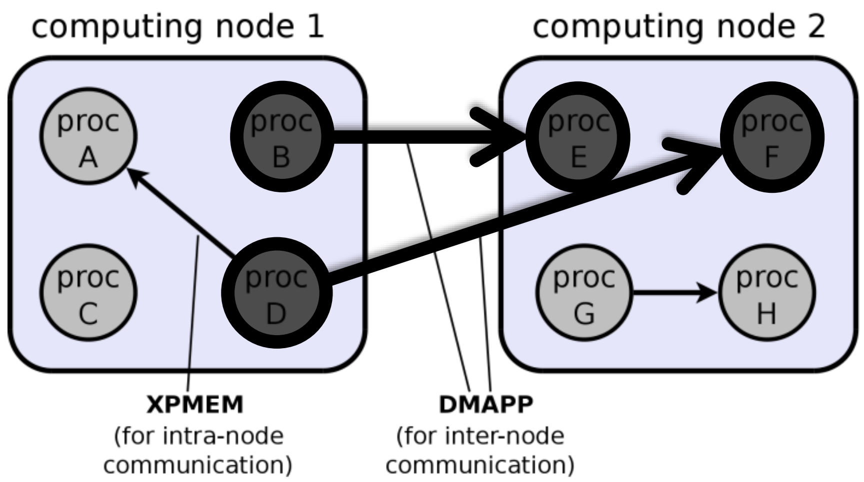
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

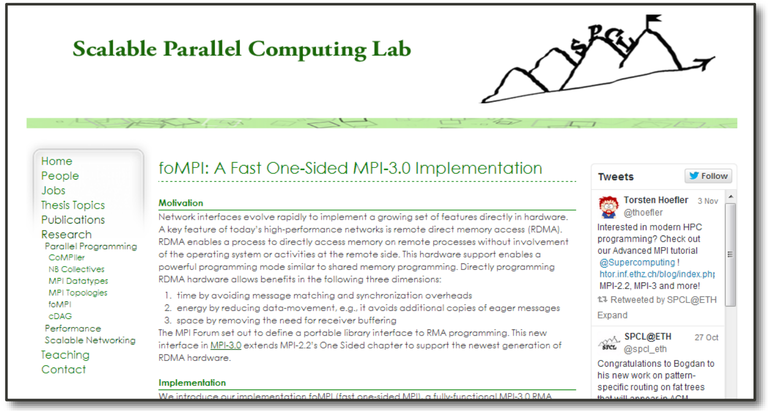
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

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





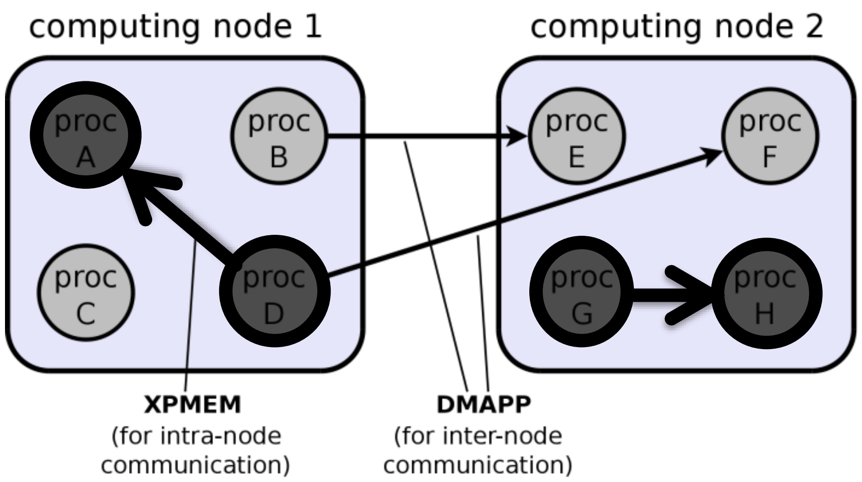
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

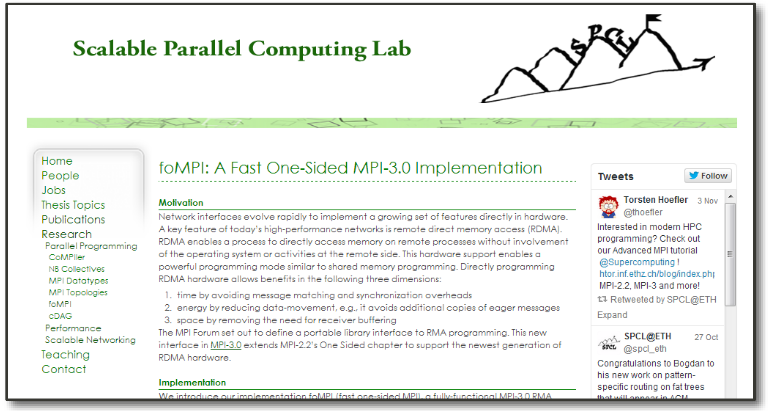
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

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



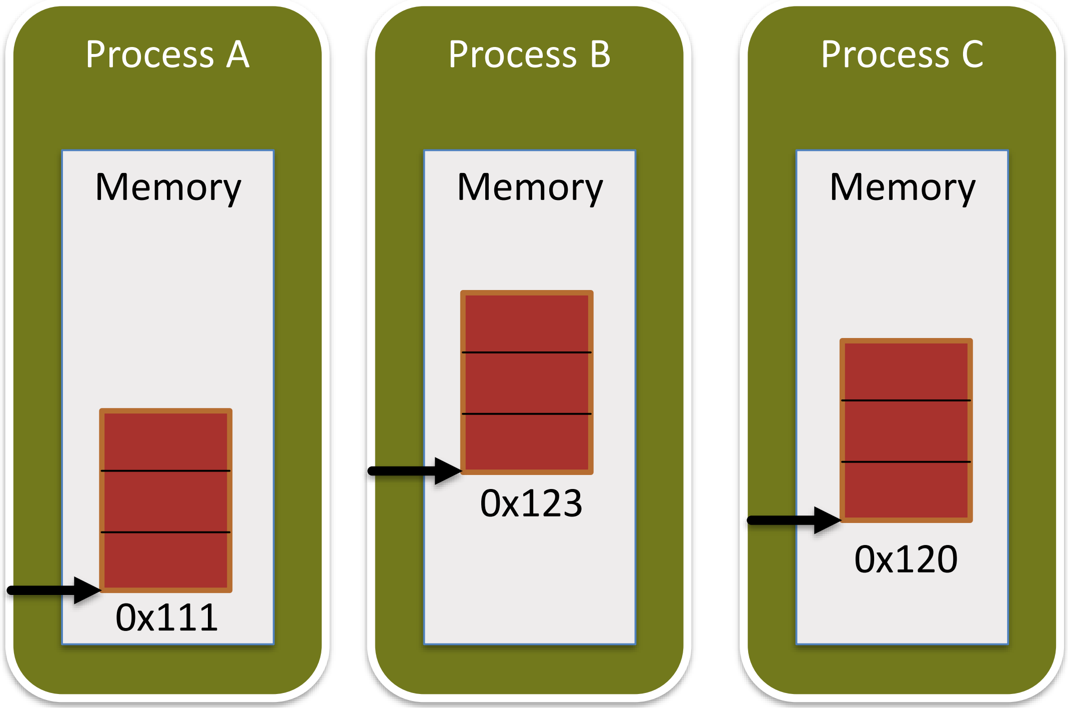


http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Part 1: Scalable Window Creation

Traditional windows



p = total number of processes

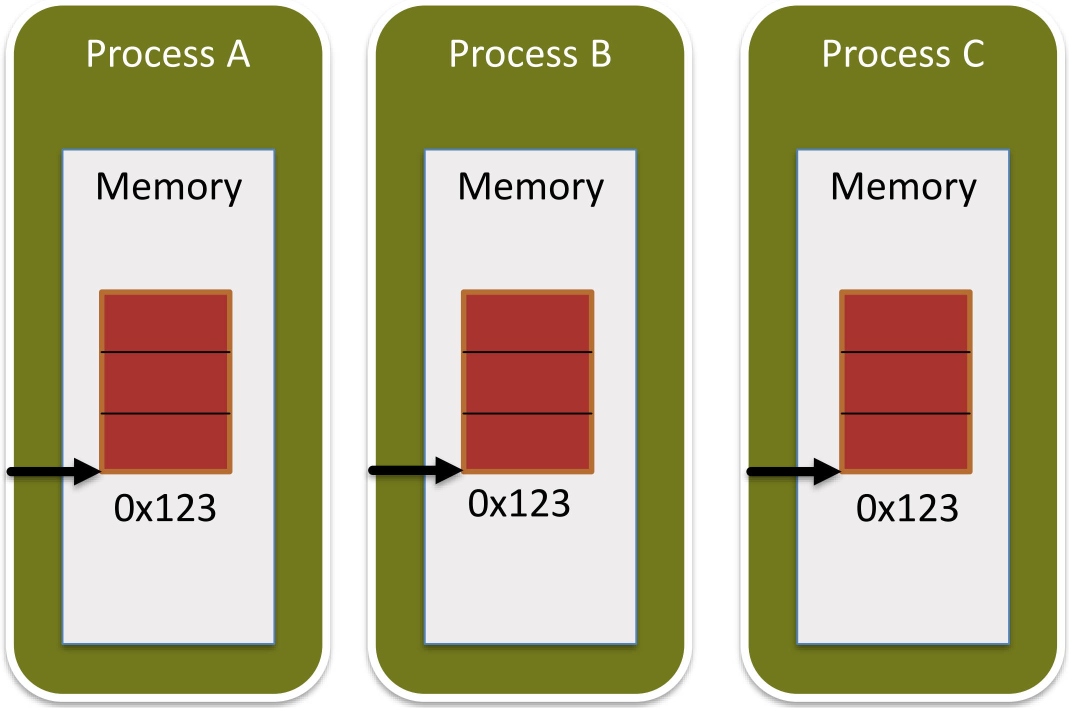
backwards compatible (MPI-2)

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

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PART 1: SCALABLE WINDOW CREATION

Allocated windows



p = total number of processes

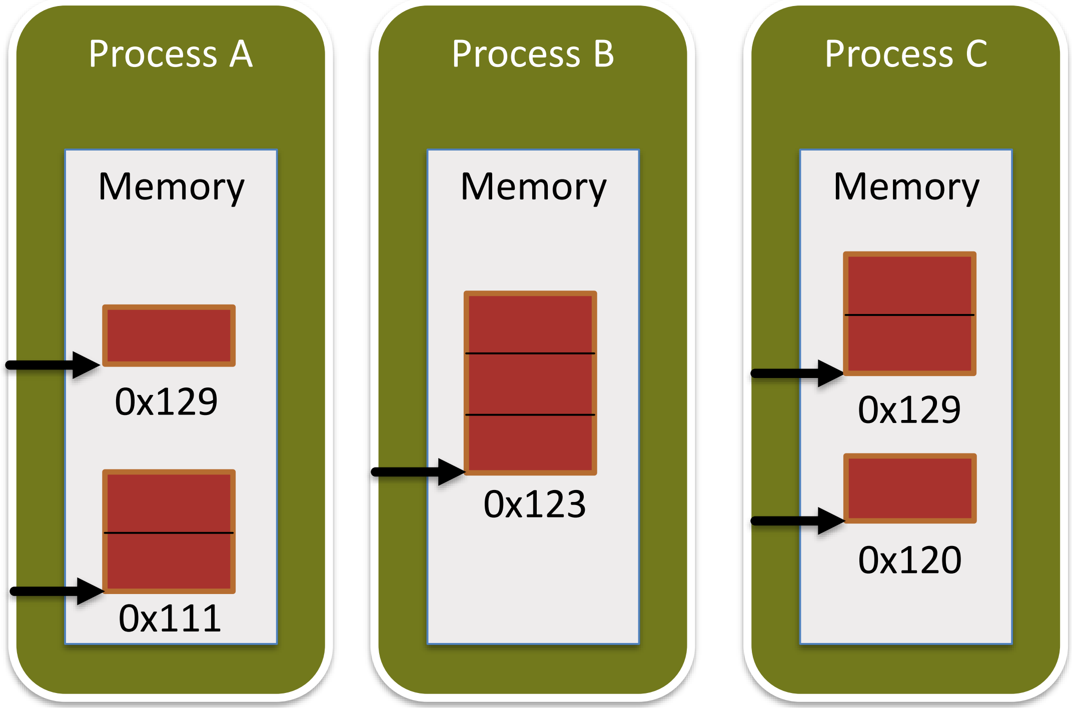
Allows MPI to allocate memory

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

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Part 1: Scalable Window Creation

Dynamic windows



p = total number of processes

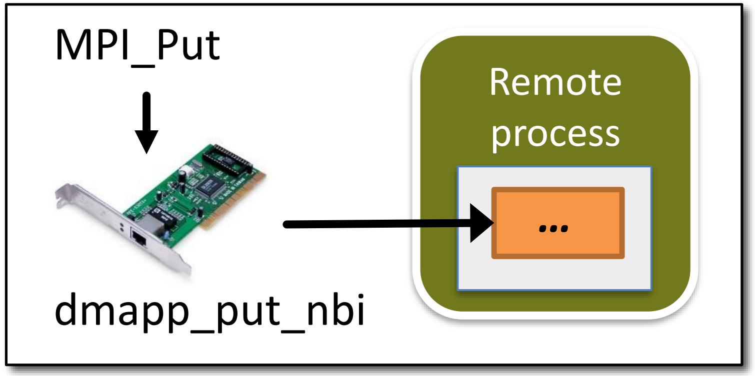
Local attach/detach
Most flexible

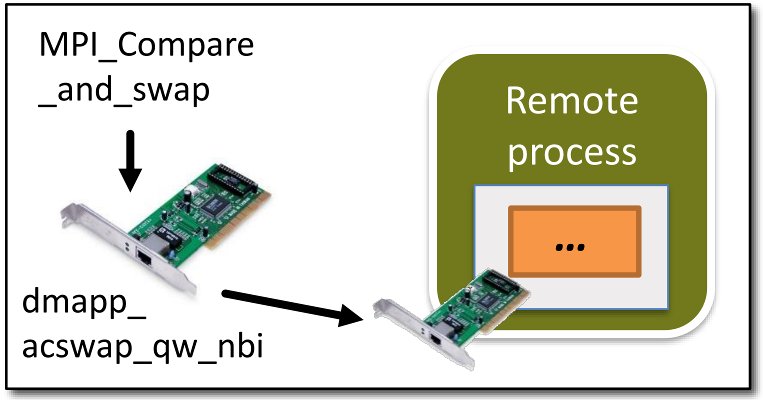
Time bound: $\mathcal{O}(p)$
Memory bound: $\mathcal{O}(p)$

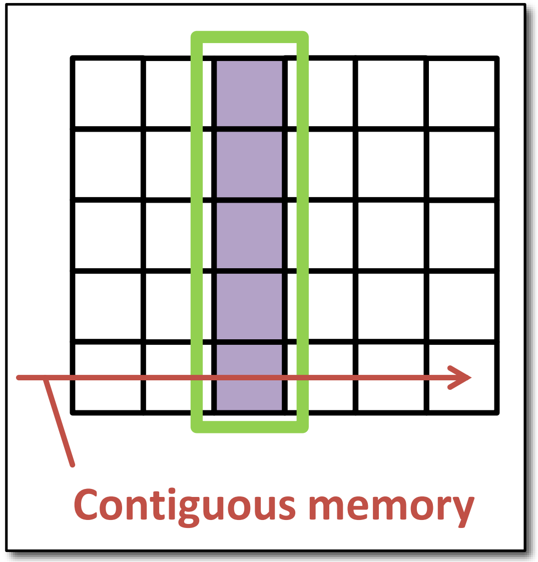
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Part 2: Communication

- Put and Get:
  - Direct DMAPP put and get operations or local (blocking) memcpy (XPMEM)
- Accumulate:
  - DMAPP atomic operations for 64 bit types
  - ...or fall back to remote locking protocol
- MPI datatype handling with MPITypes library [1]
  - Fast path for contiguous data transfers of common intrinsic datatypes (e.g., MPI_DOUBLE)

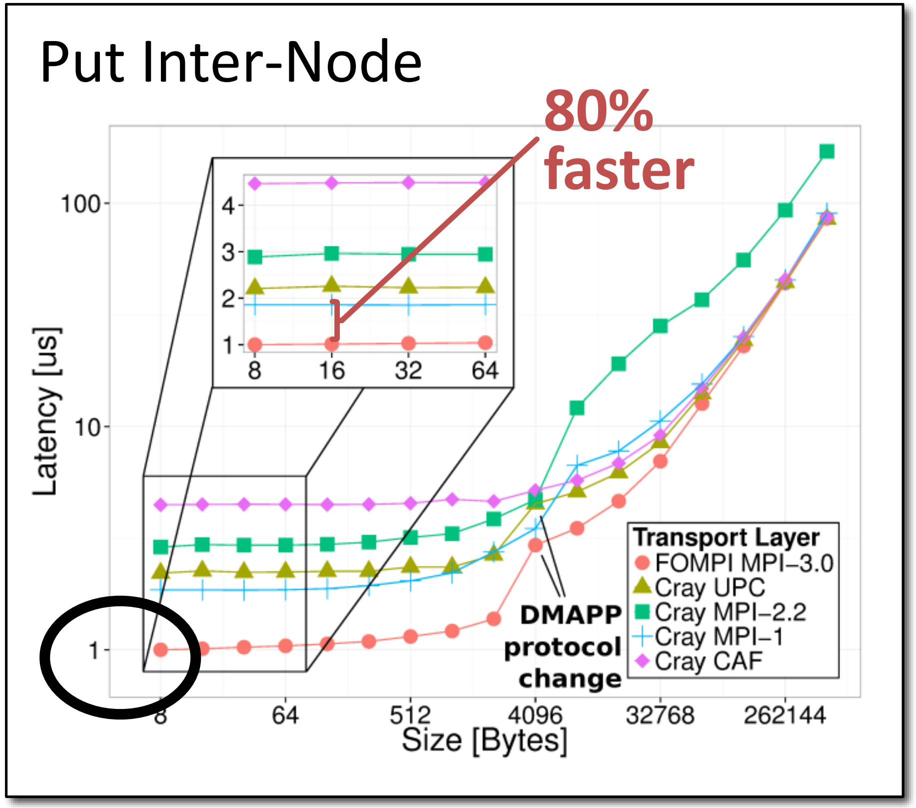


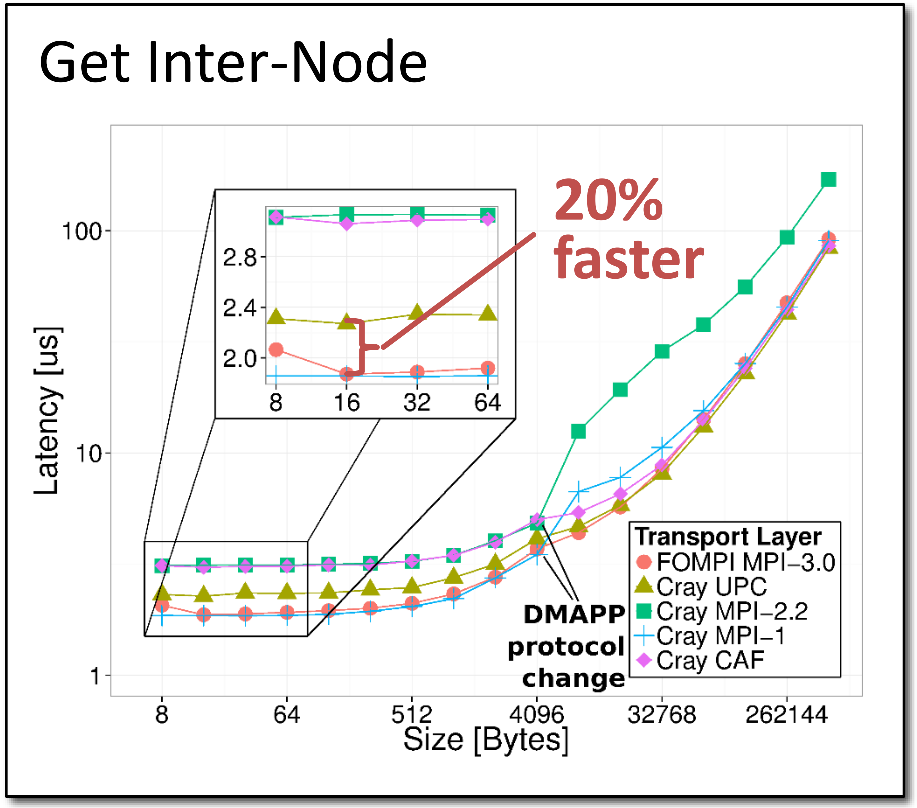


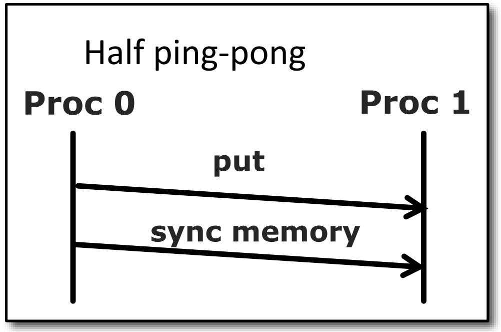


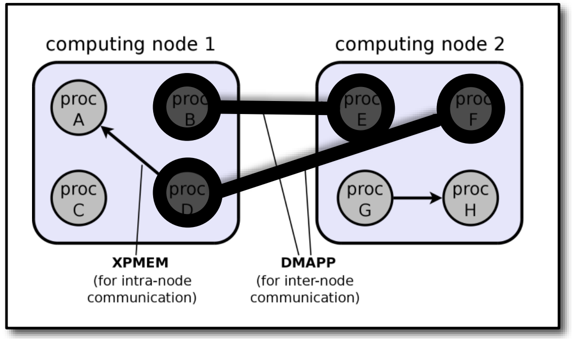
[1] Ross, Latham, Gropp, Lusk, Thakur. Processing MPI datatypes outside MPI. EuroMPI/PVM’09

Performance Inter-node: Latency

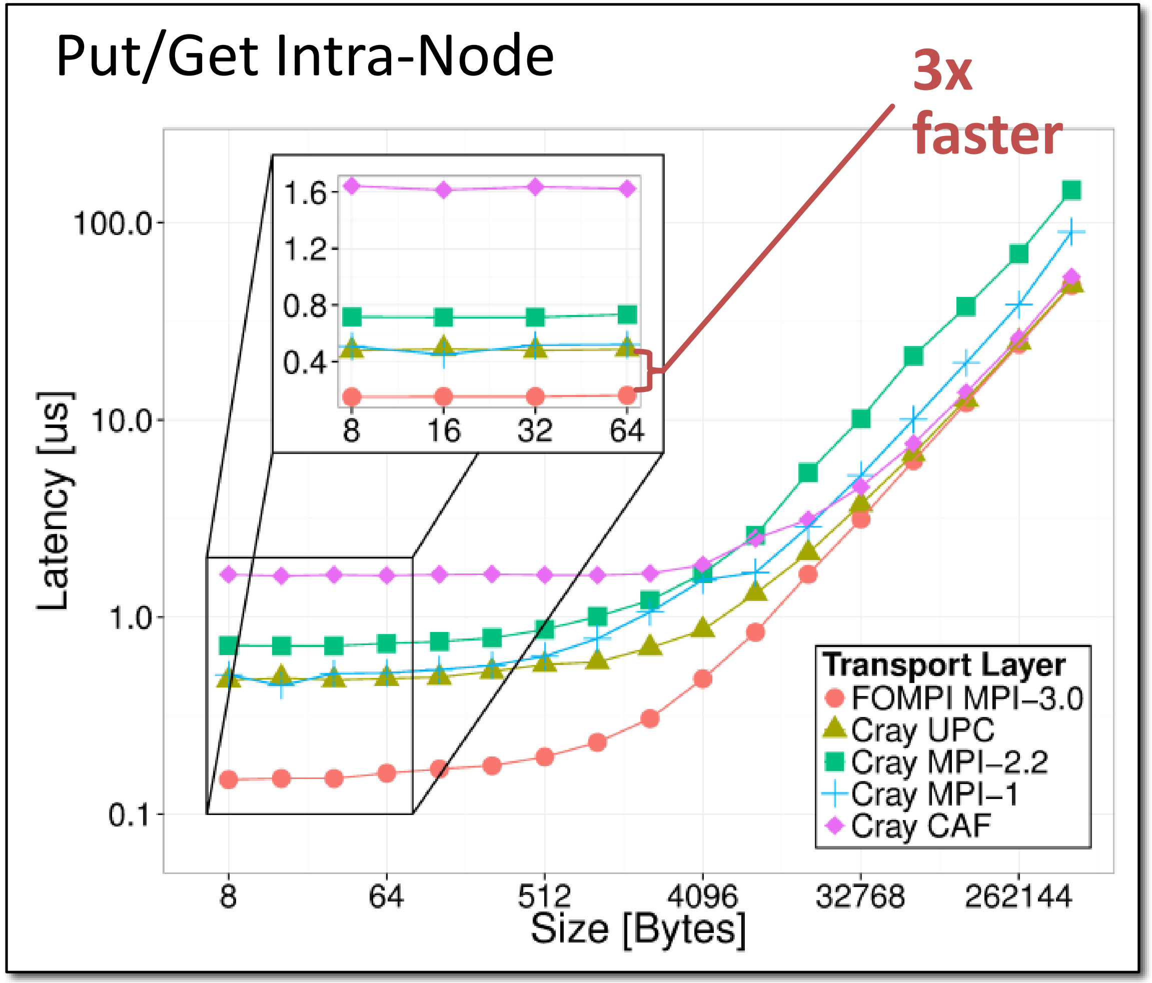


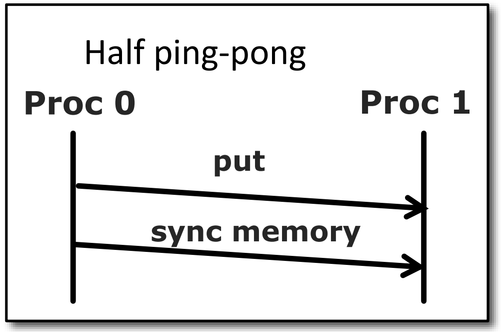


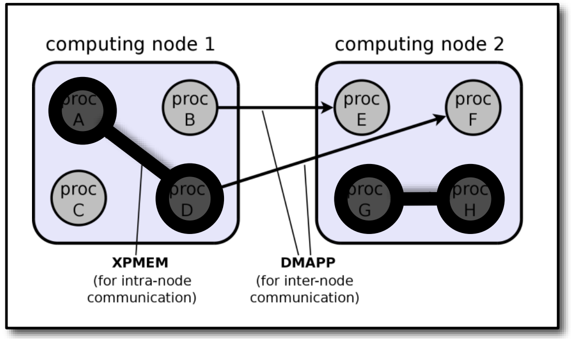




Performance Intra-node: Latency

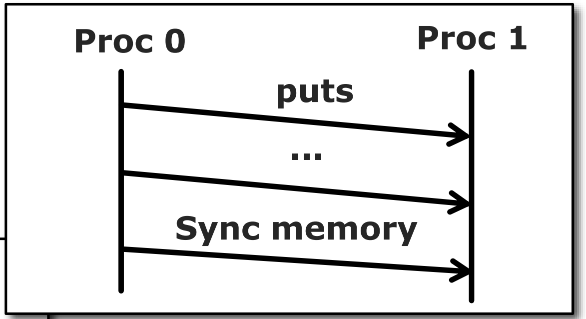


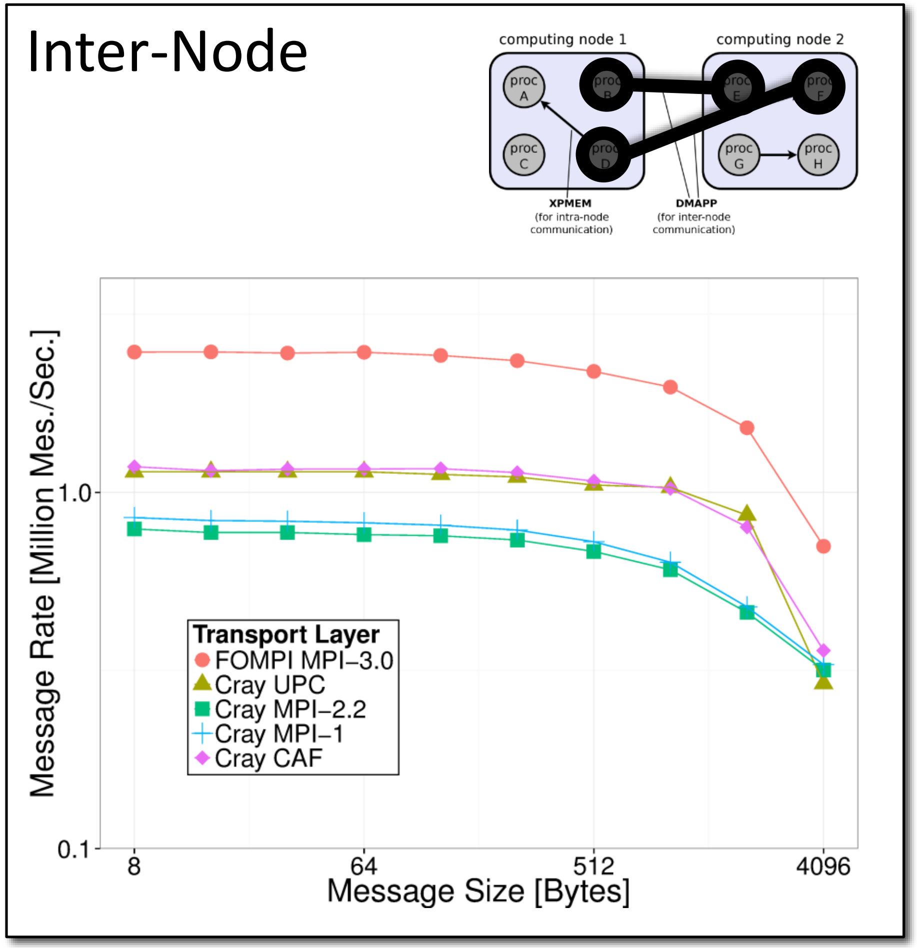


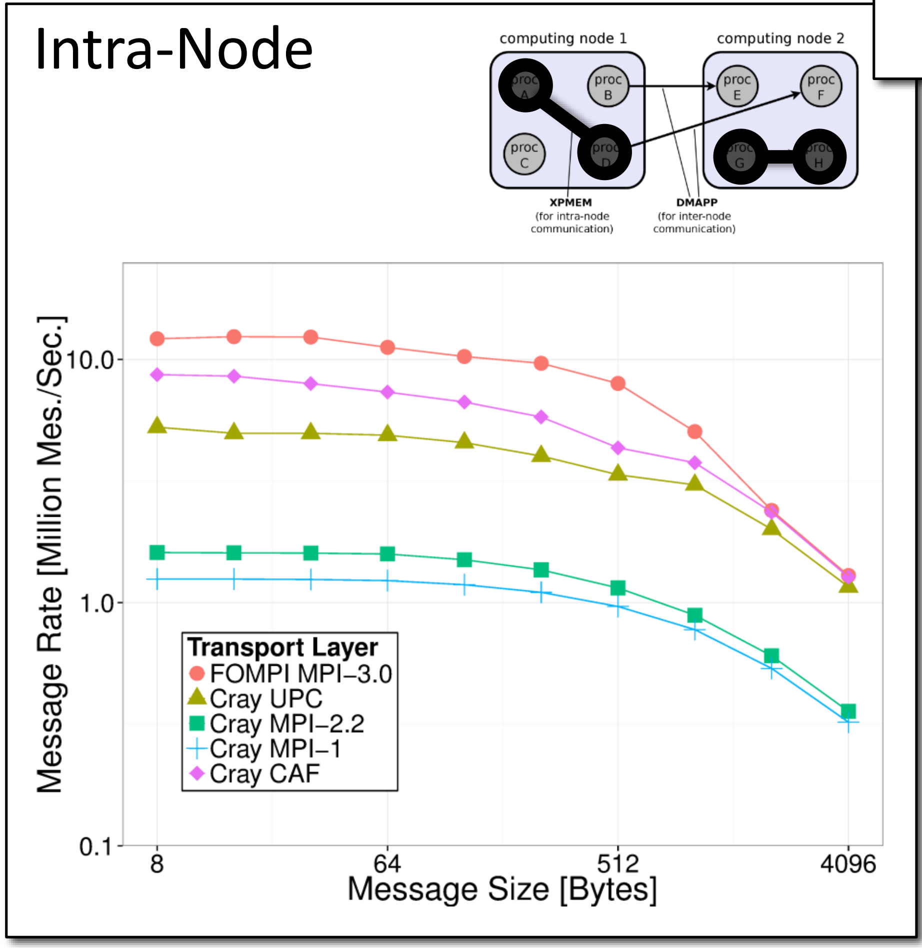


Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Performance: Message Rate

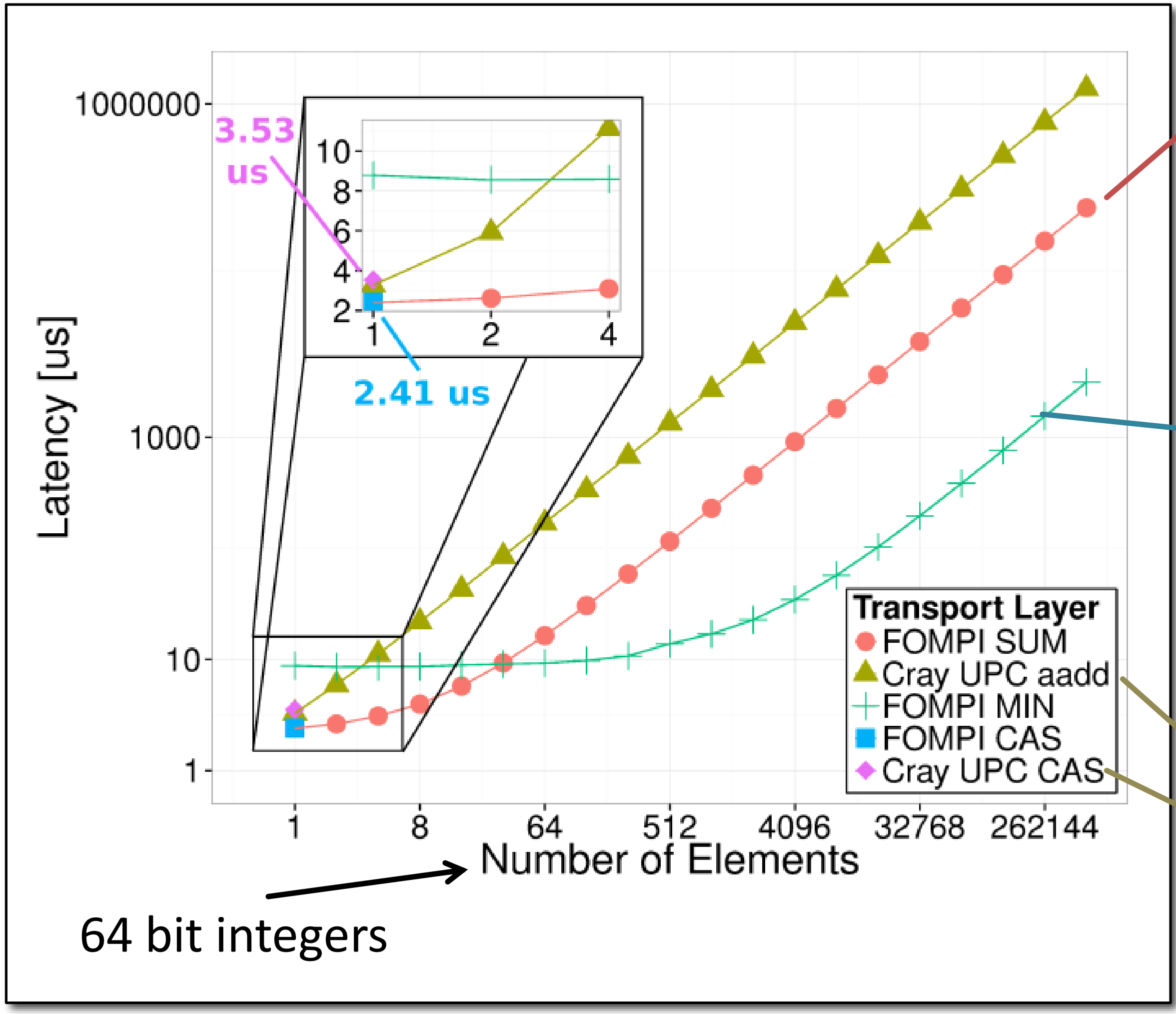






Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PERFORMANCE: ATOMICS



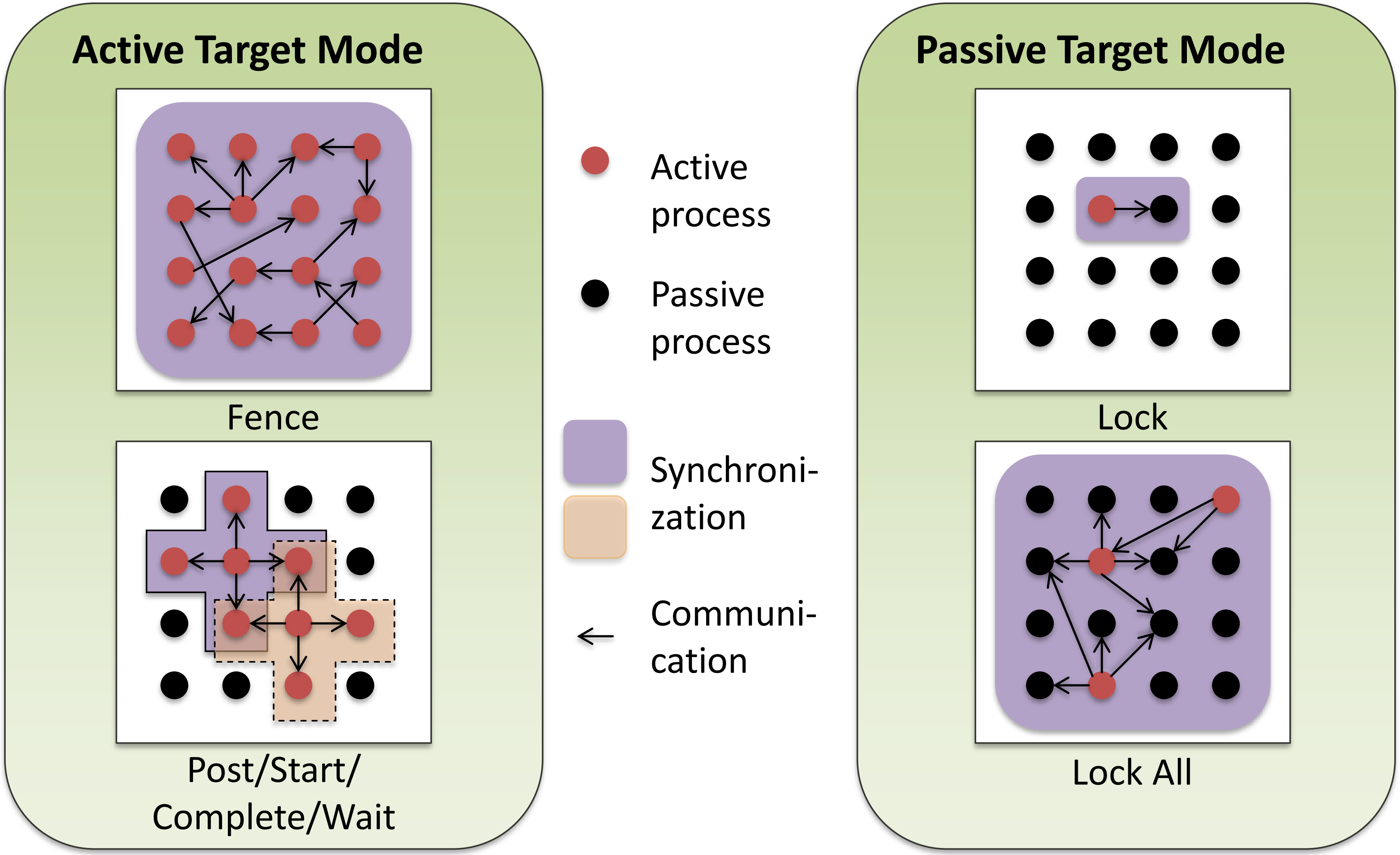
hardware-accelerated protocol:
lower latency

fall back protocol:
higher bandwidth

proprietary

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

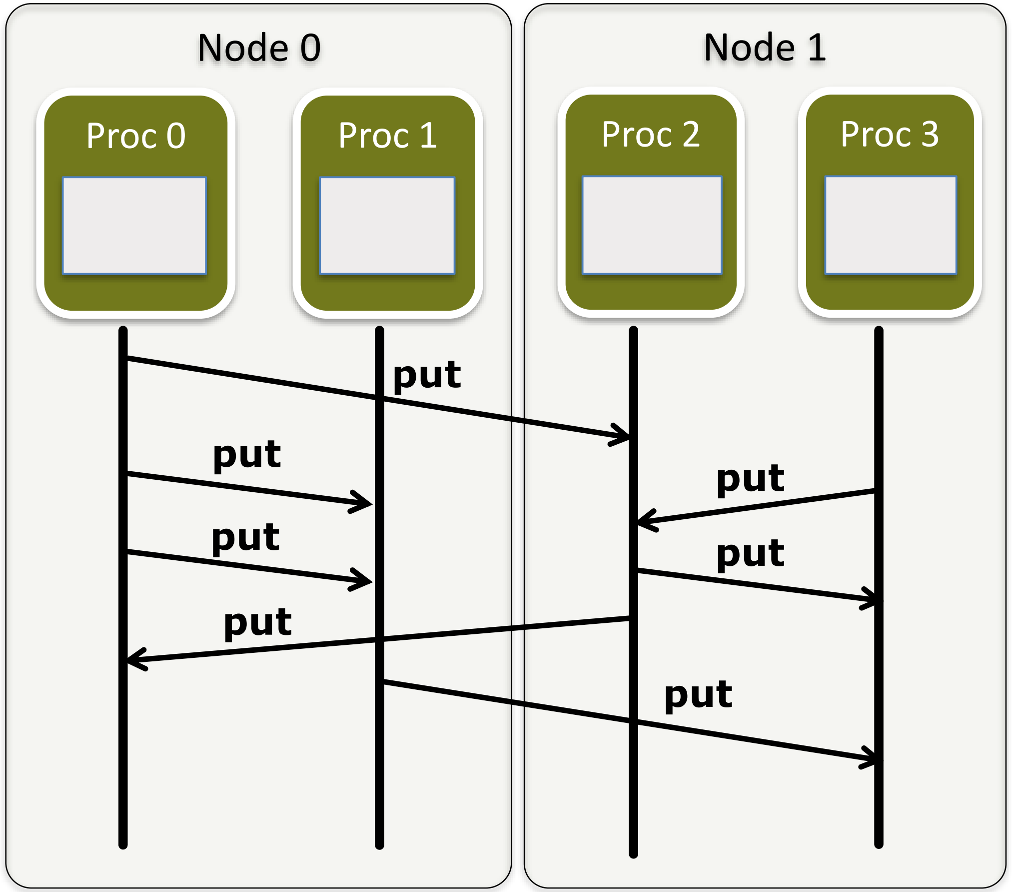
Part 3: Synchronization



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Fence Implementation

- Collective call
- Completes all outstanding memory operations

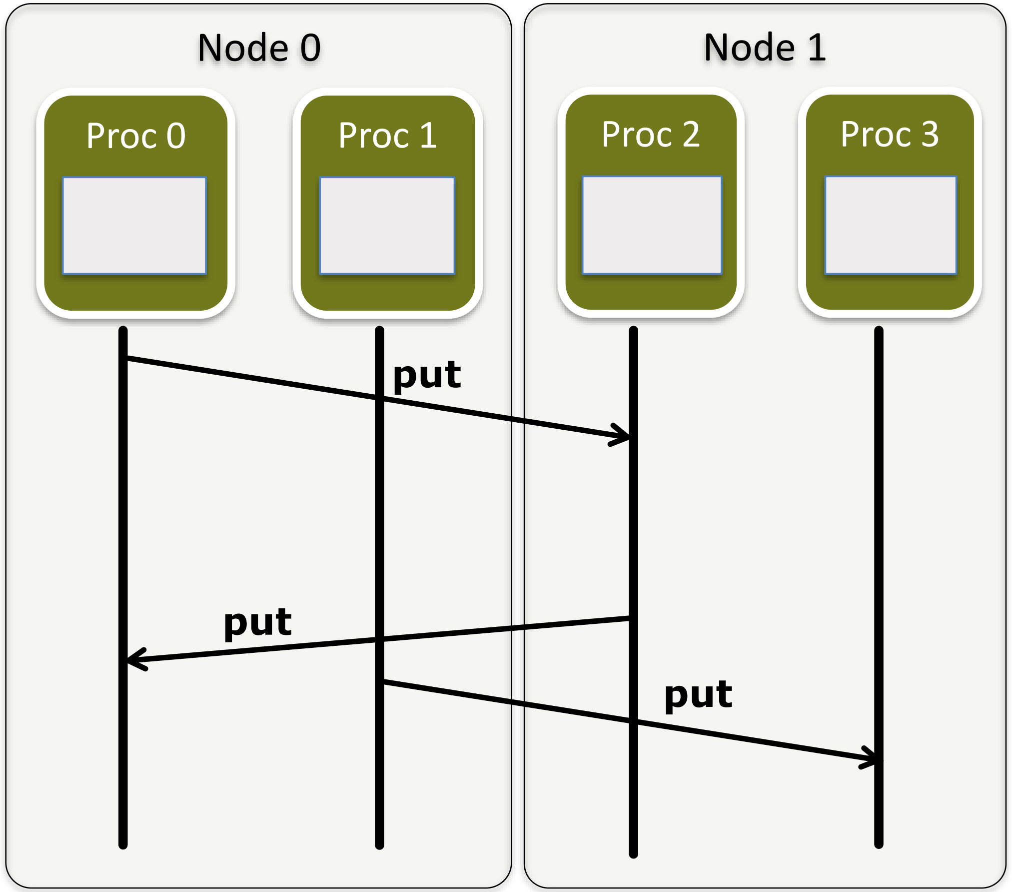



```
int MPI_Win_fence(…) {
  asm( mfence );
  dmapp_gsync_wait();
  MPI_Barrier(...);
  return MPI_SUCCESS;
}
```

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Fence Implementation

- Collective call
- Completes all outstanding memory operations



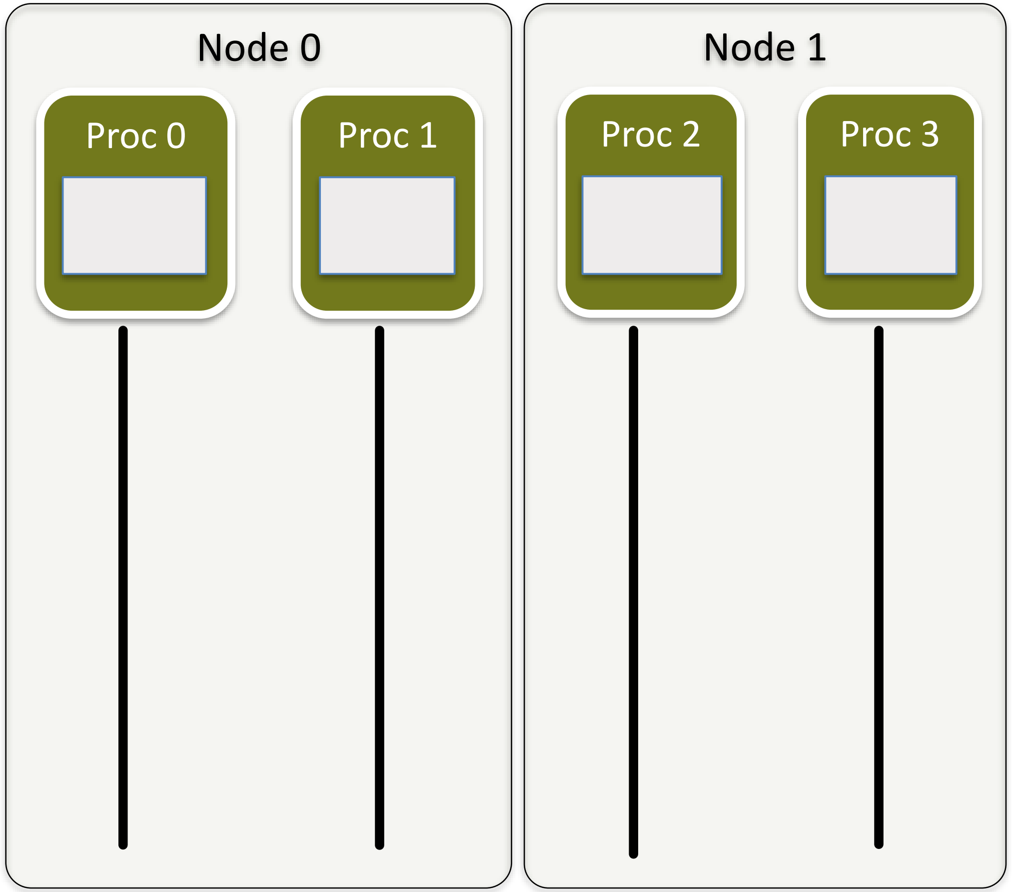

```
int MPI_Win_fence(…) {
  asm( mfence );
  dmapp_gsync_wait();
  MPI_Barrier(...);
  return MPI_SUCCESS;
}
```

Local completion (XPMEM)

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Fence Implementation

- Collective call
- Completes all outstanding memory operations



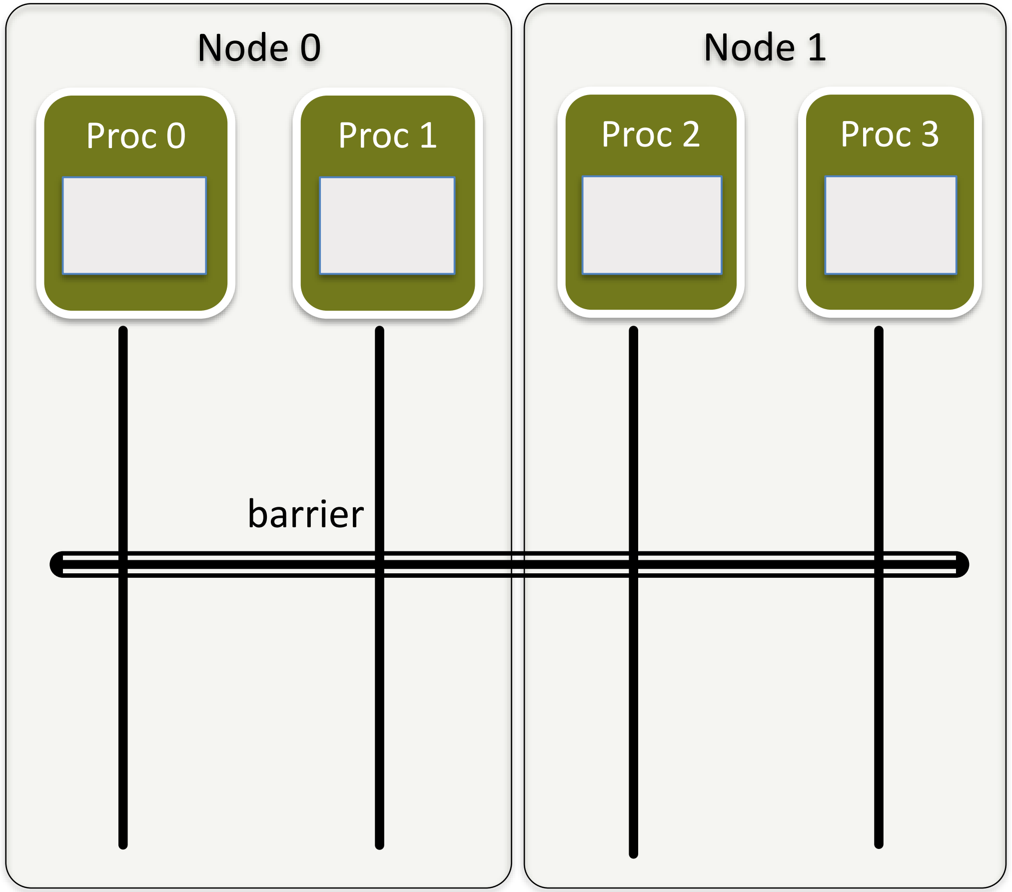

```
int MPI_Win_fence(…) {
  asm( mfence );
  dmapp_gsync_wait();
  MPI_Barrier(...);
  return MPI_SUCCESS;
}
```

Local completion
(DMAPP)

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Fence Implementation

- Collective call
- Completes all outstanding memory operations

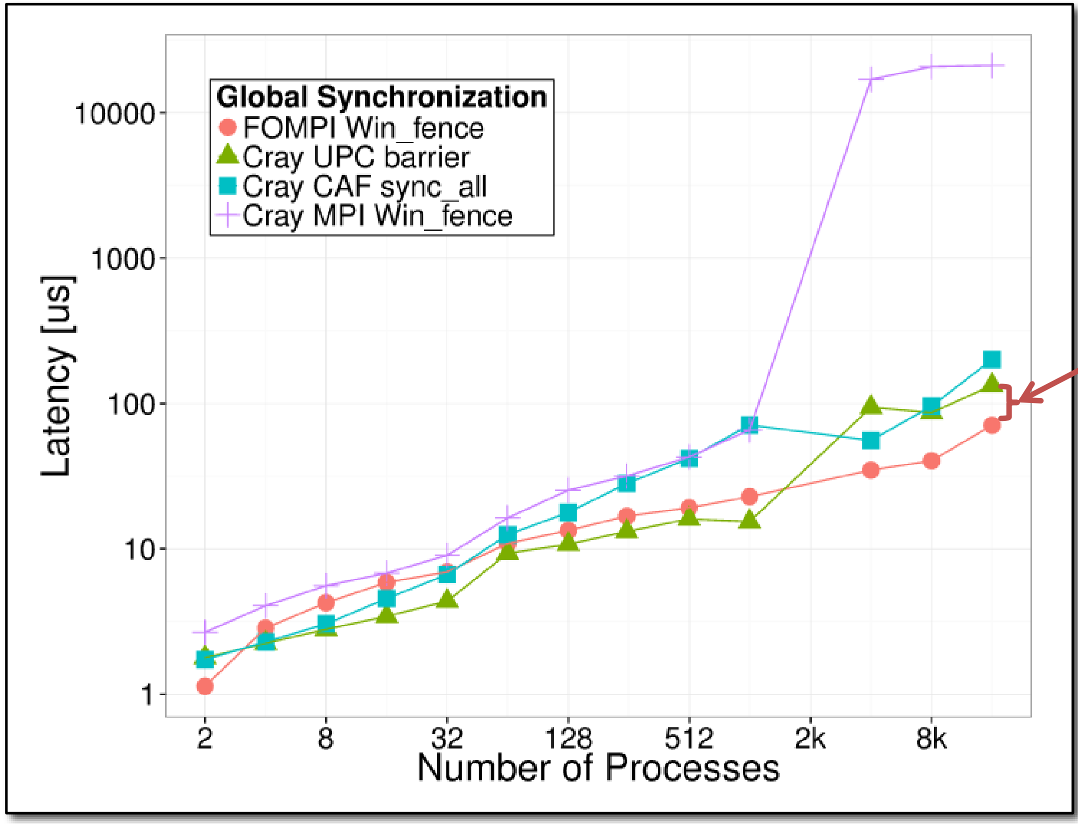



```
int MPI_Win_fence(…) {
  asm( mfence );
  dmapp_gsync_wait();
  MPI_Barrier(...);
  return MPI_SUCCESS;
}
```

Global completion

Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Fence Performance

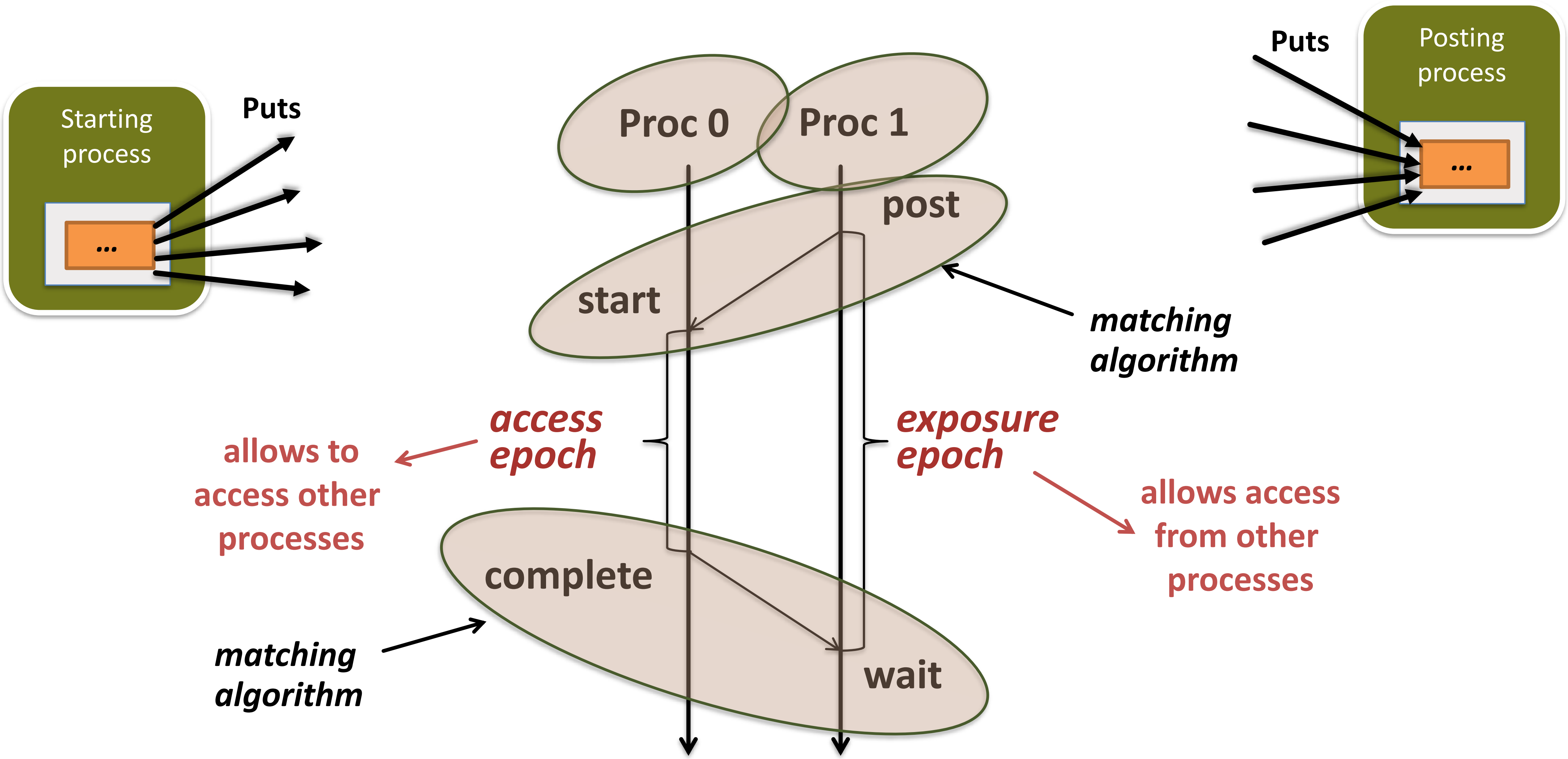


90% faster

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

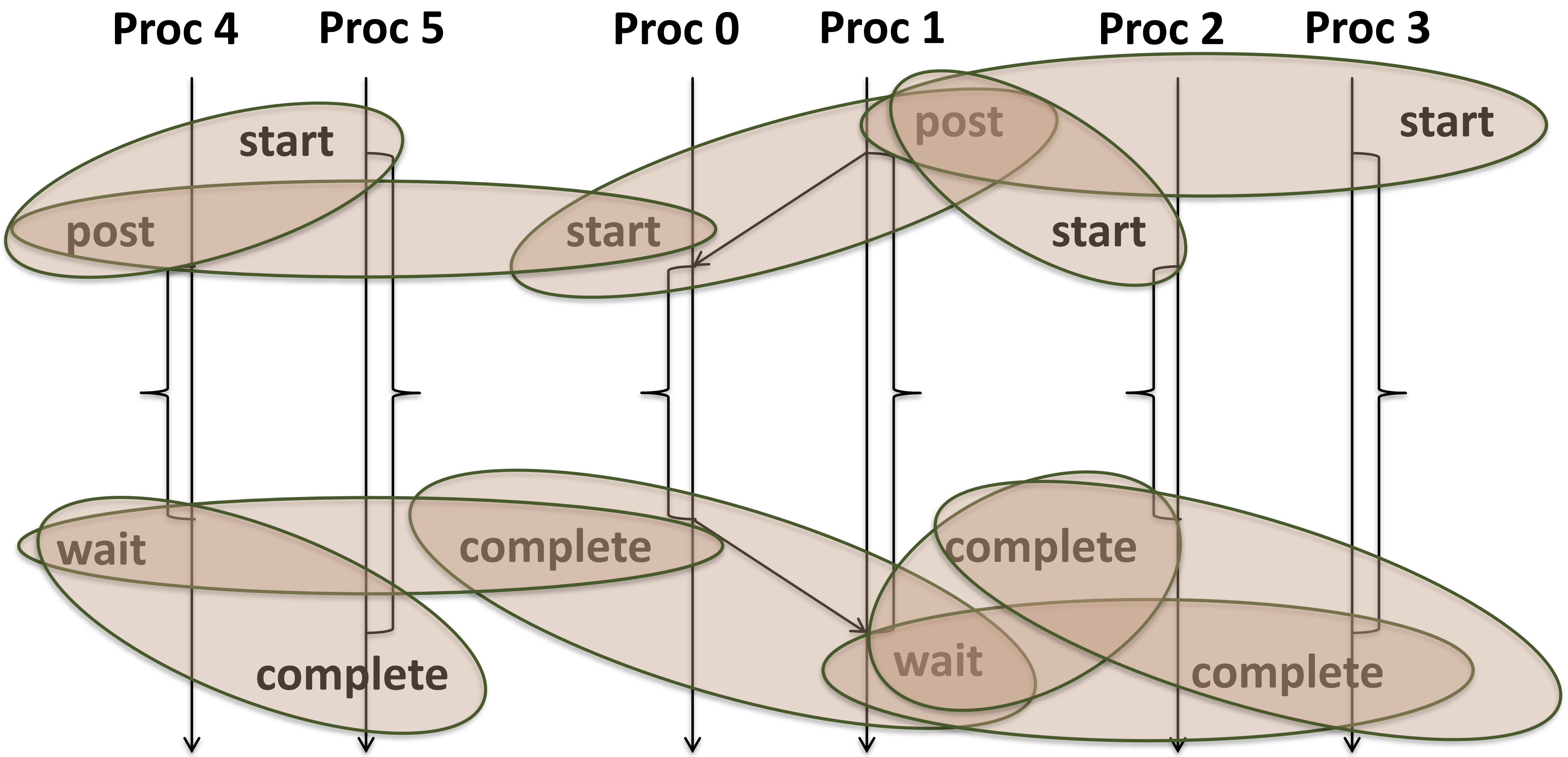
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Synchronization



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

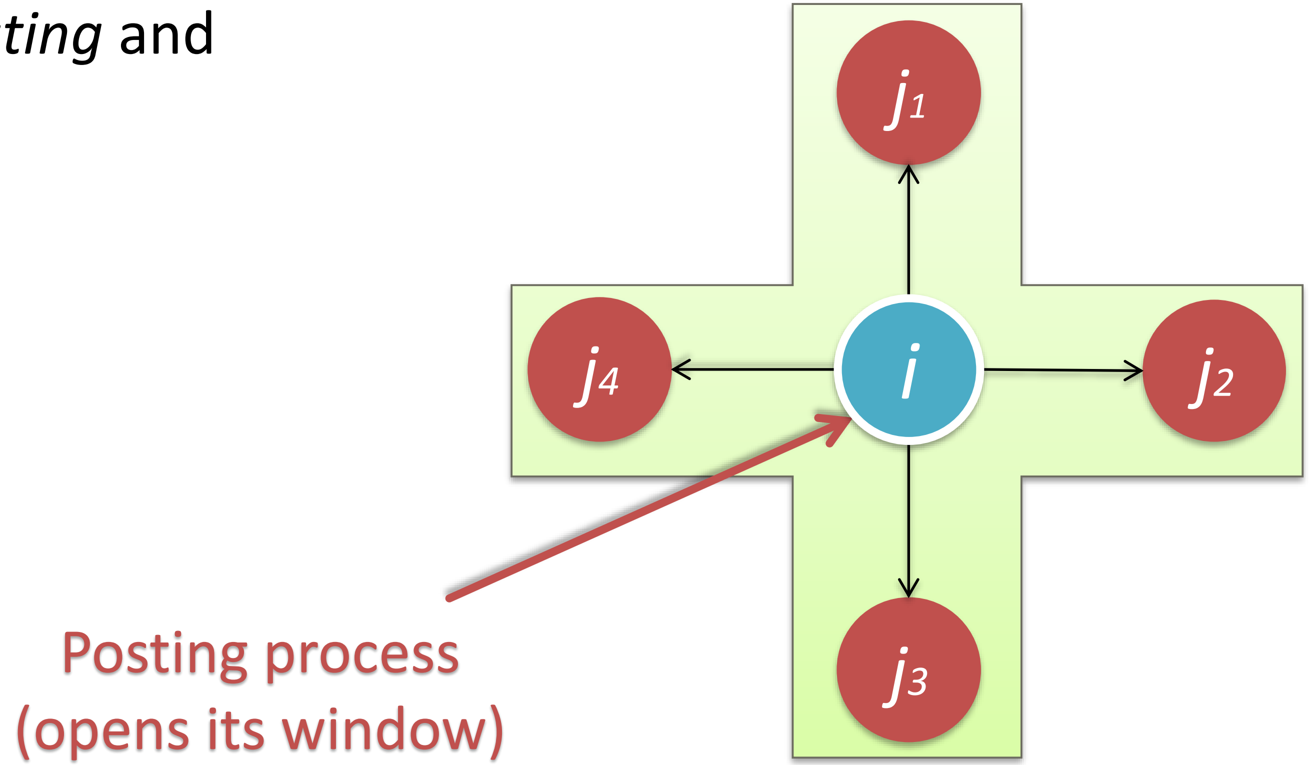
PSCW SYNCHRONIZATION



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

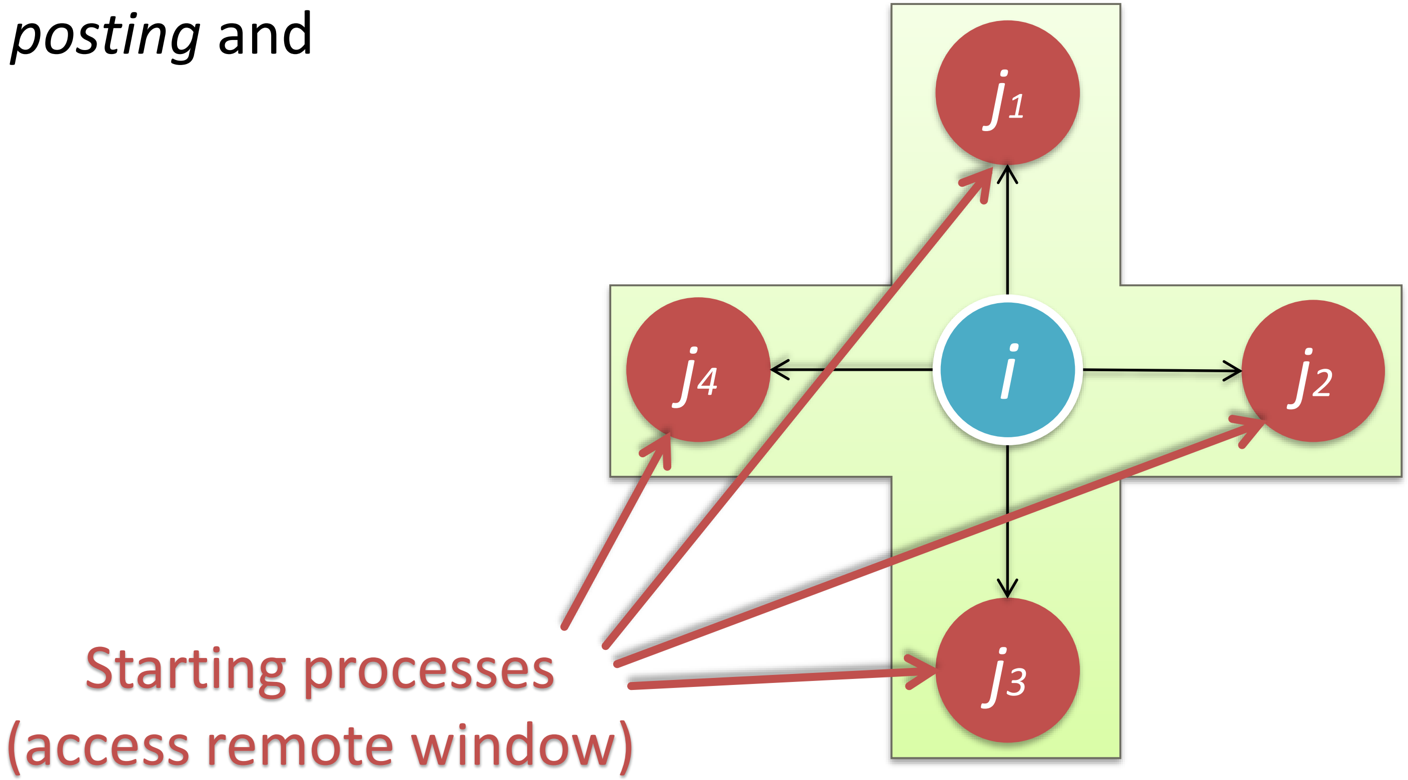
- In general, there can be n *posting* and m *starting* processes
- In this example there is one *posting* and 4 *starting* processes



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

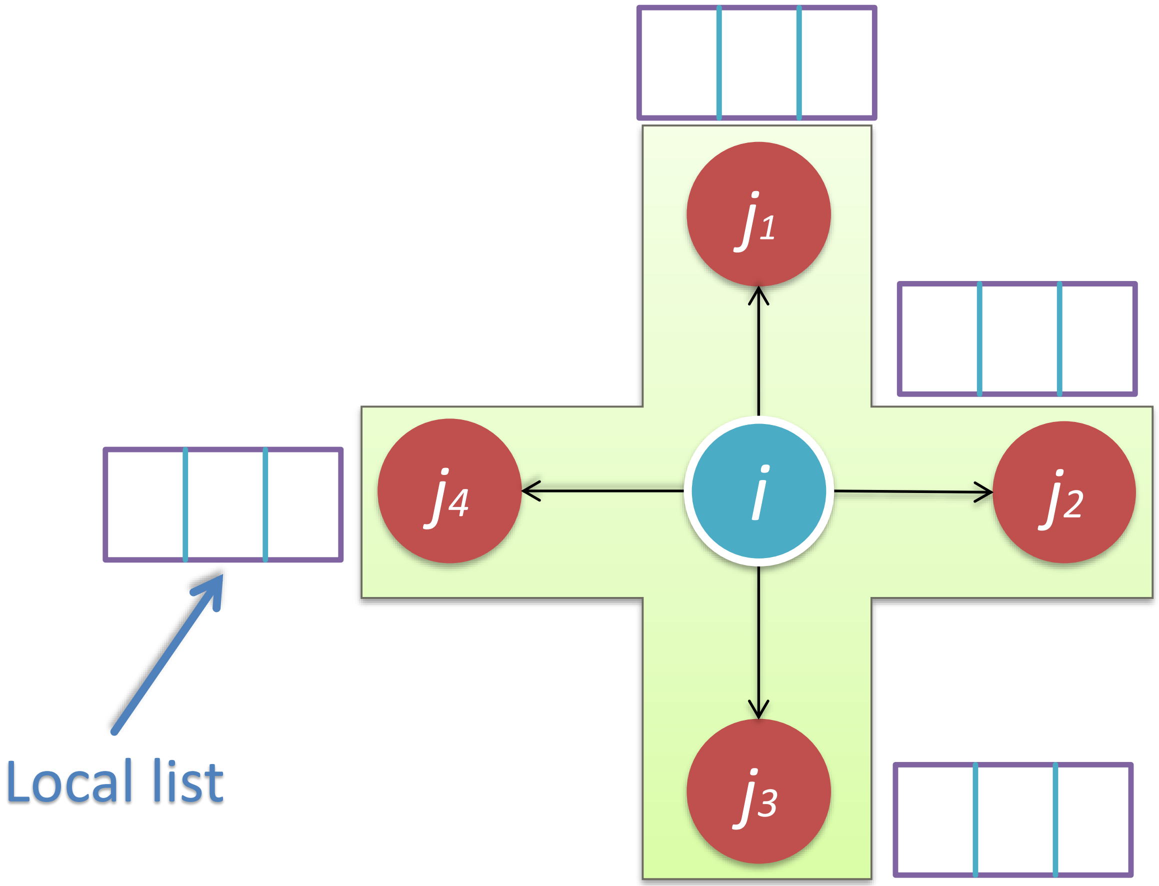
- In general, there can be n *posting* and m *starting* processes
- In this example there is one *posting* and 4 *starting* processes



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

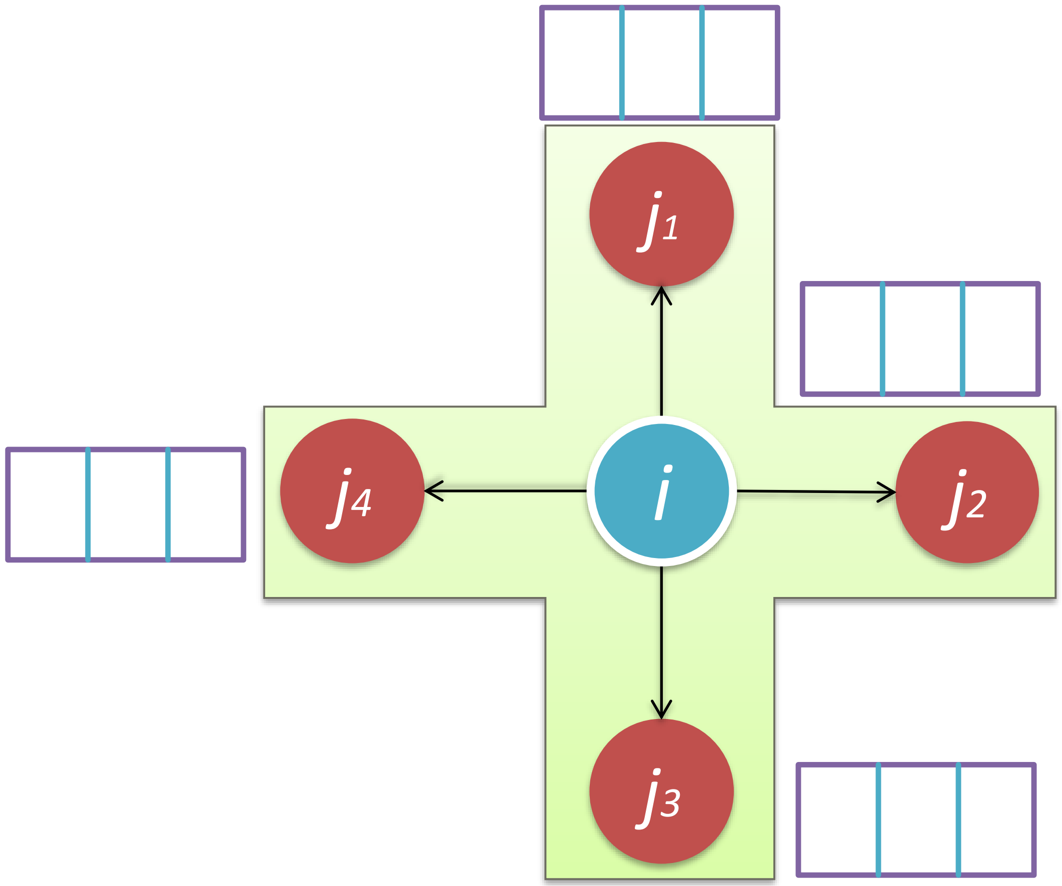
- Each starting process has a local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

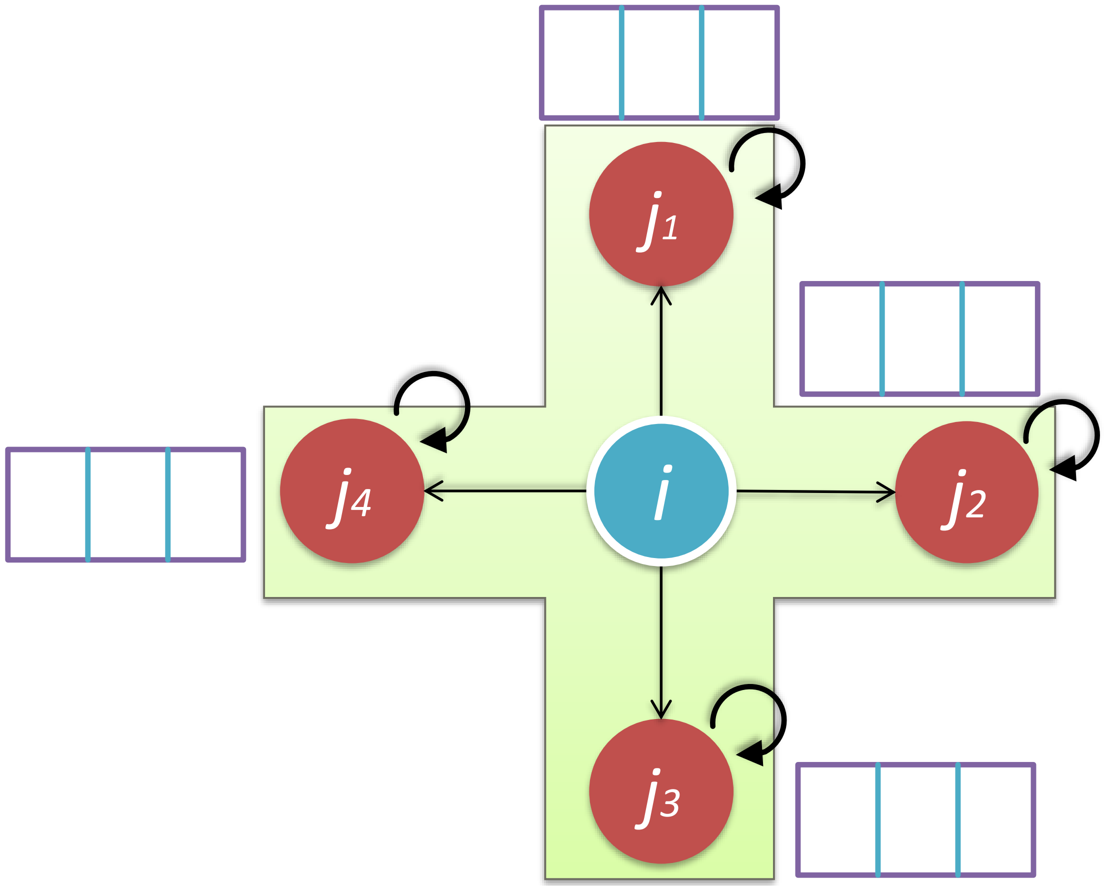
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

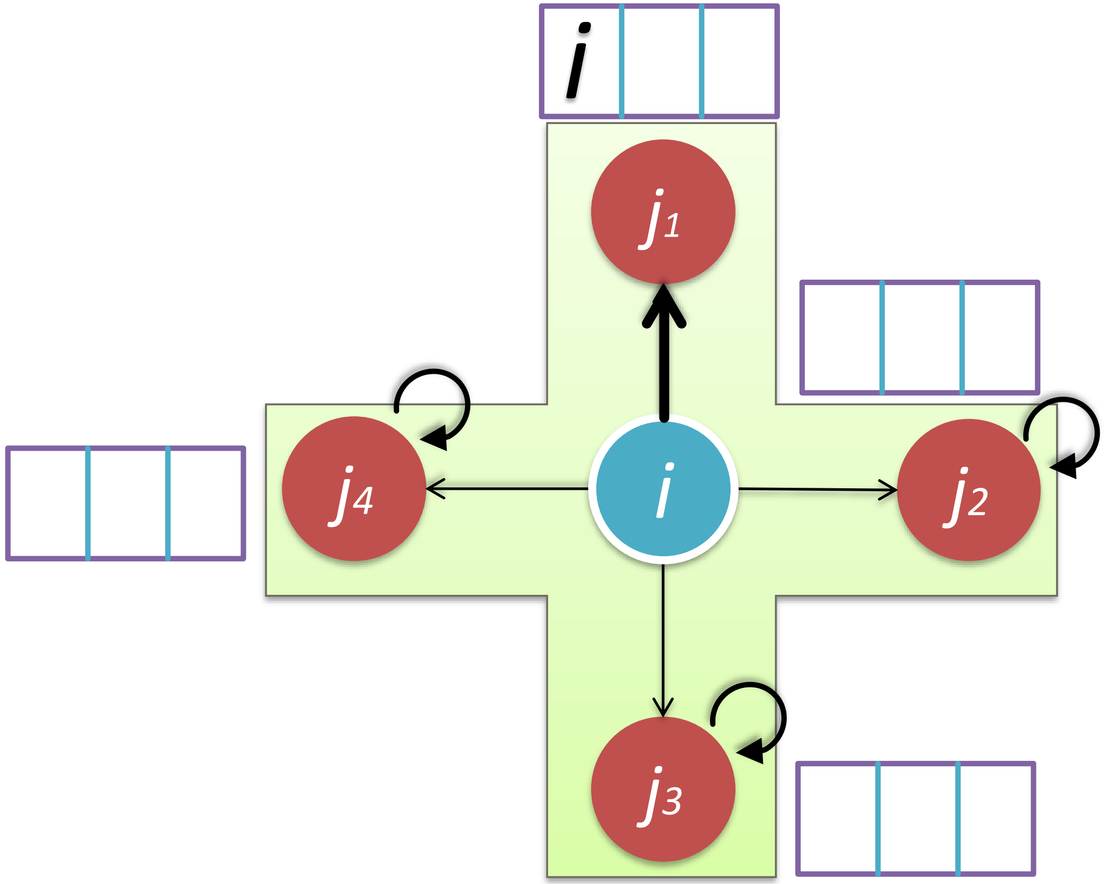
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

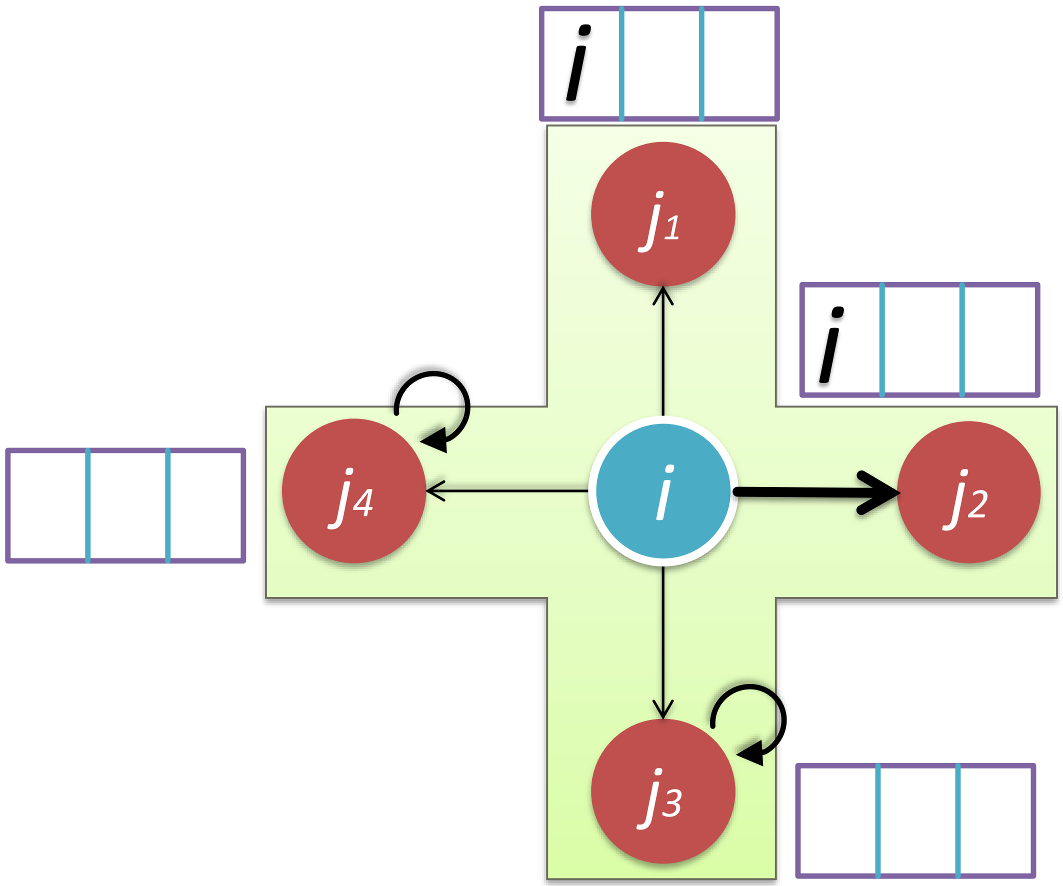
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

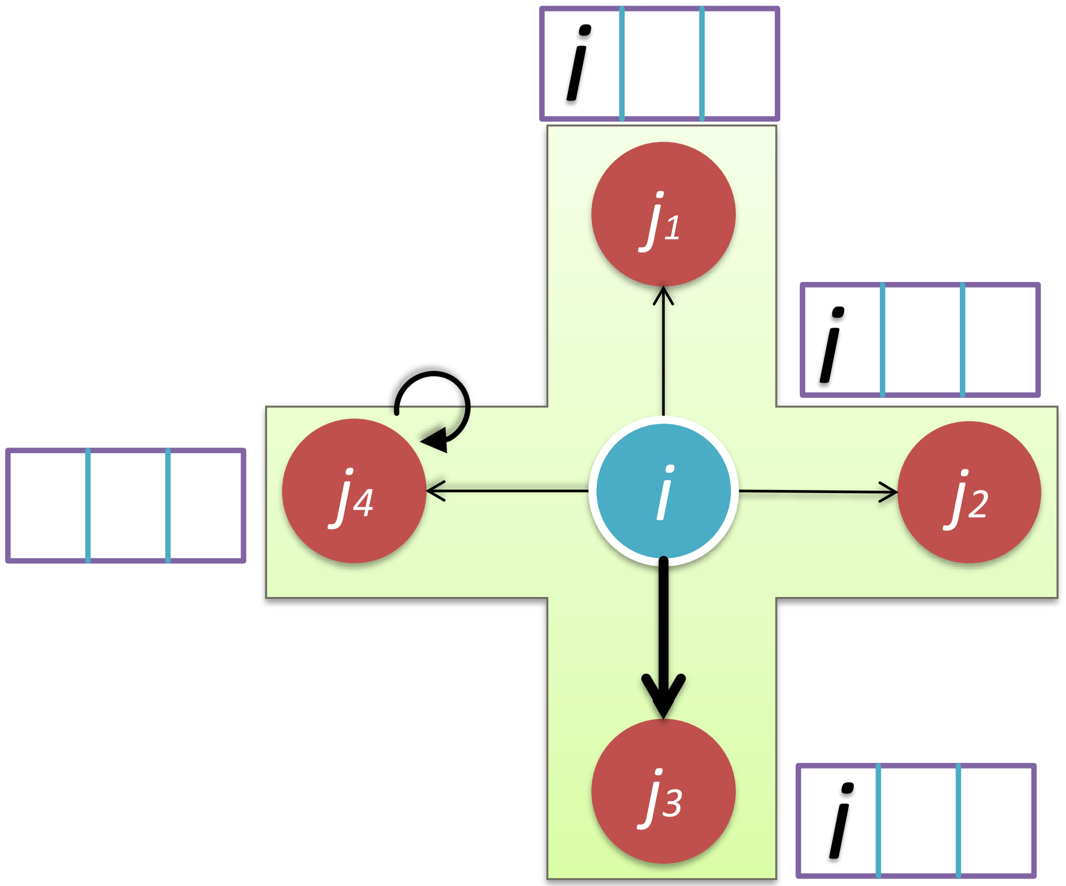
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

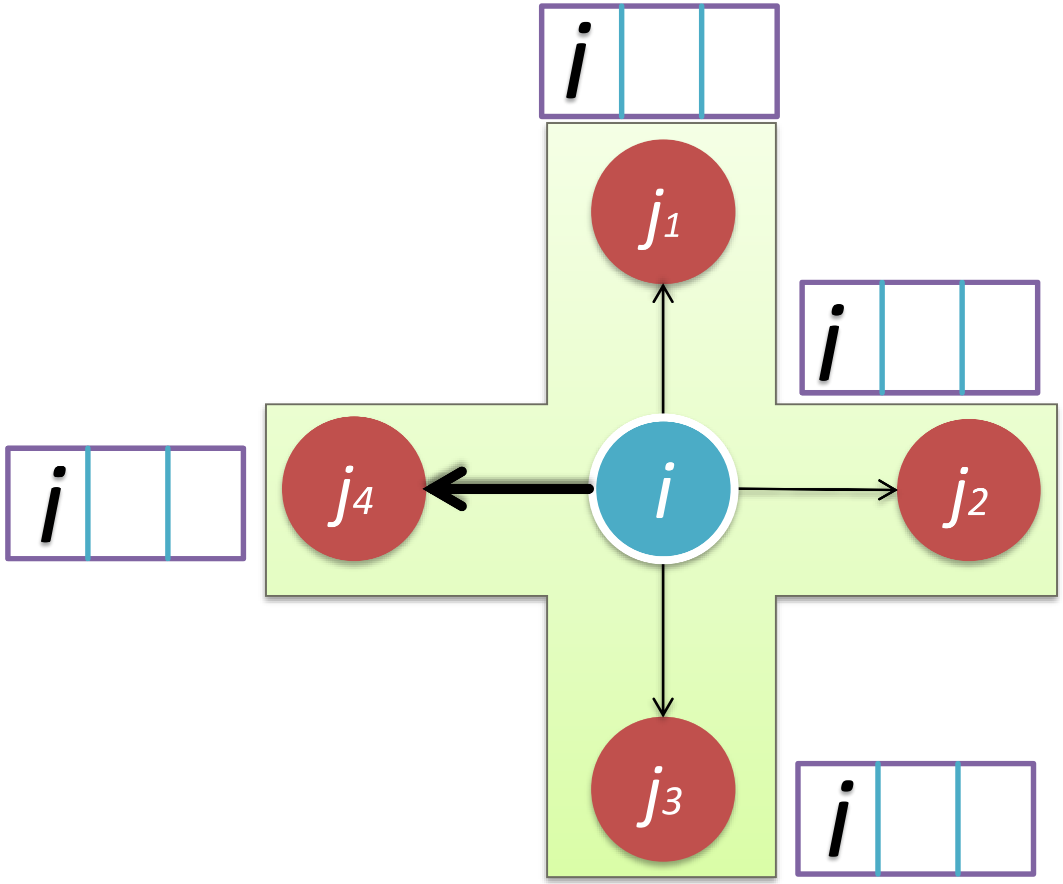
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

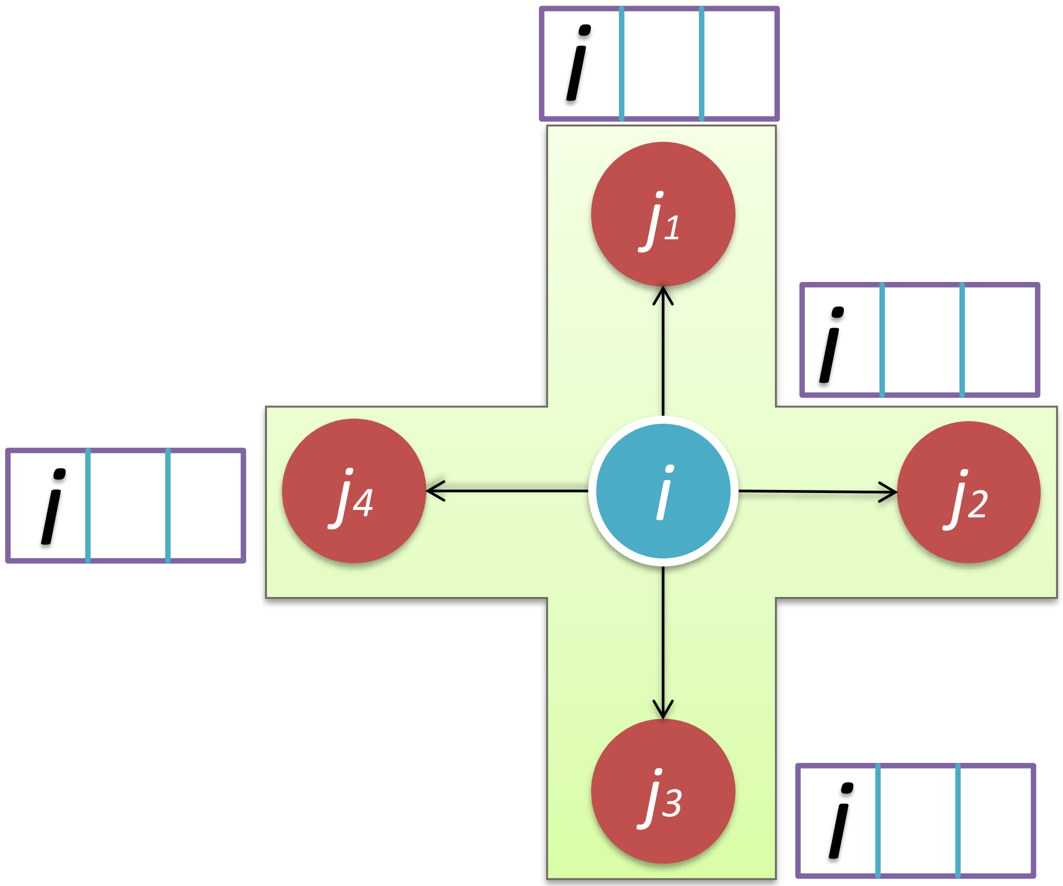
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Post/Start Matching

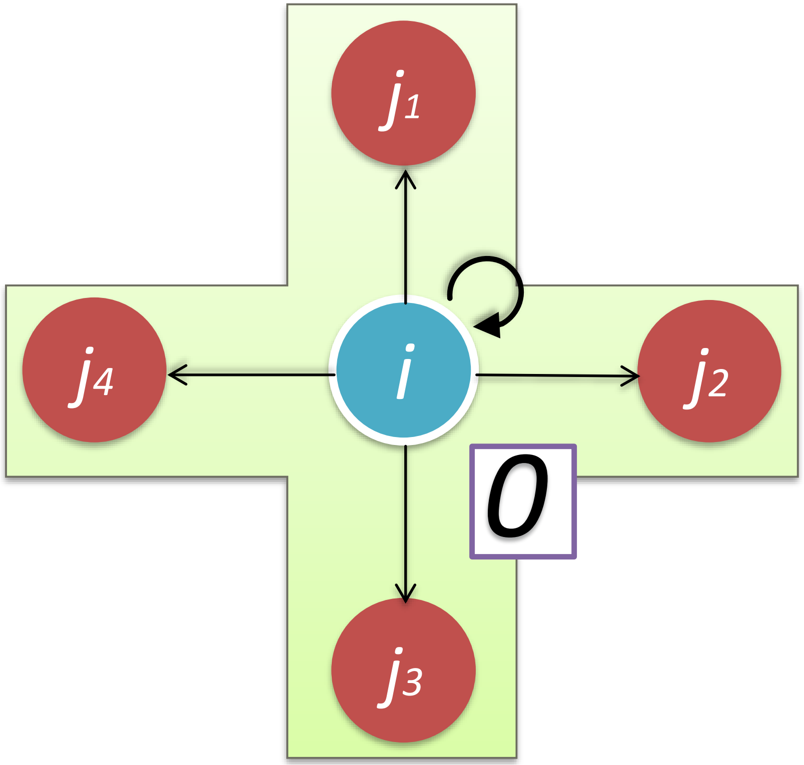
- *Posting* process *i* adds its rank *i* to a list at each *starting* process $j_1, \ldots, j_4$
- Each *starting* process *j* waits until the rank of the *posting* process *i* is present in its local list



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Complete/Wait Matching

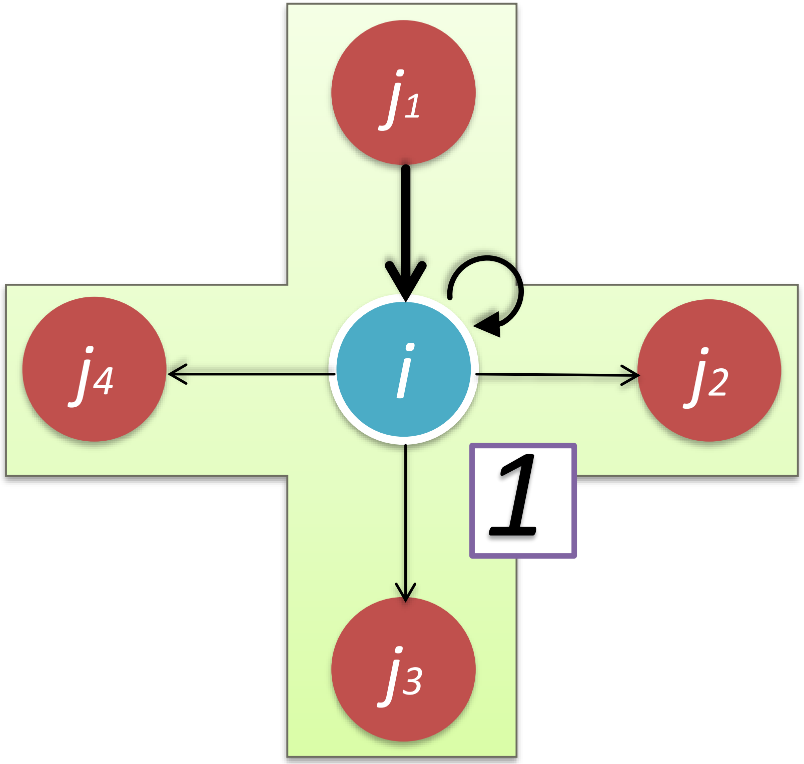
- Each *completing* process increments a counter stored at the *posting* process



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Complete/Wait Matching

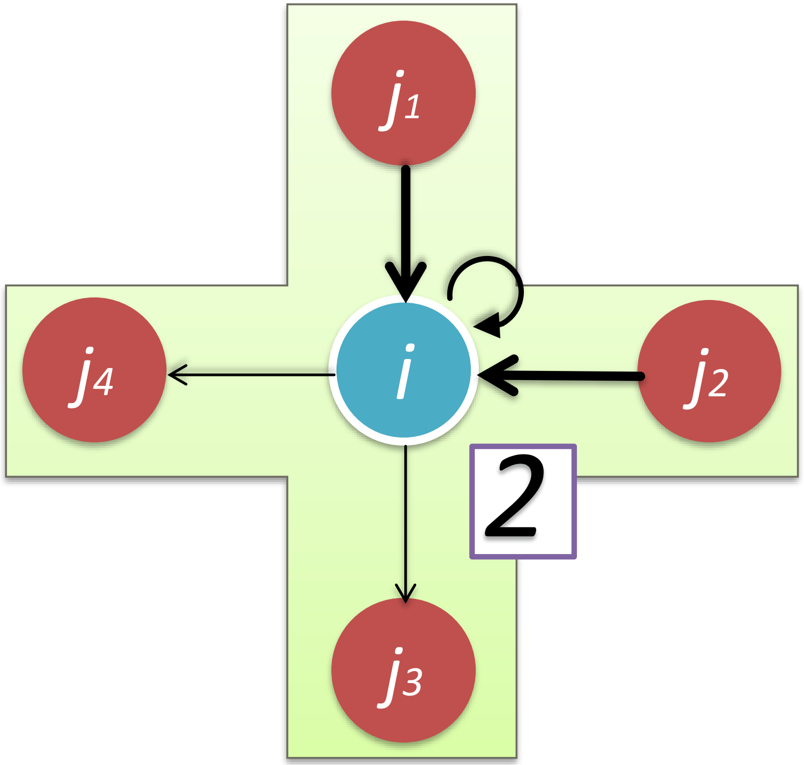
- Each *completing* process increments a counter stored at the *posting* process



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Complete/Wait Matching

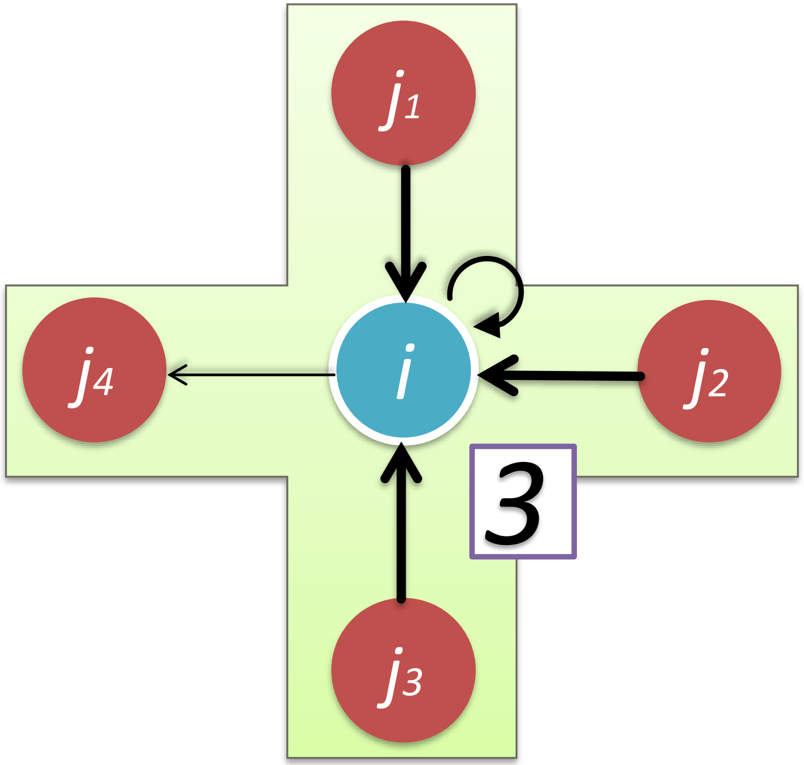
- Each *completing* process increments a counter stored at the *posting* process



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Complete/Wait Matching

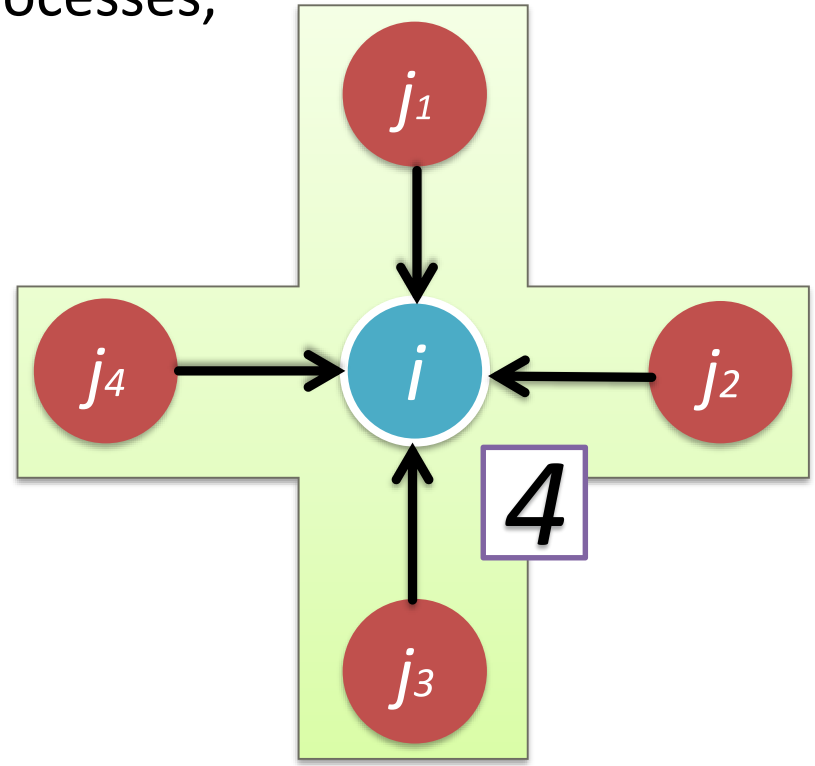
- Each *completing* process increments a counter stored at the *posting* process



Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Scalable Complete/Wait Matching

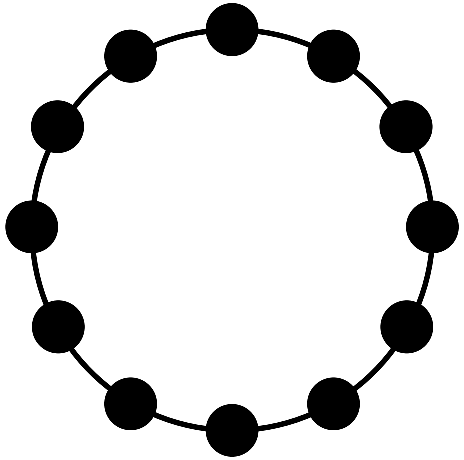
- Each *completing* process increments a counter stored at the *posting* process
- When the counter is equal to the number of *starting* processes, the *posting* process returns from wait

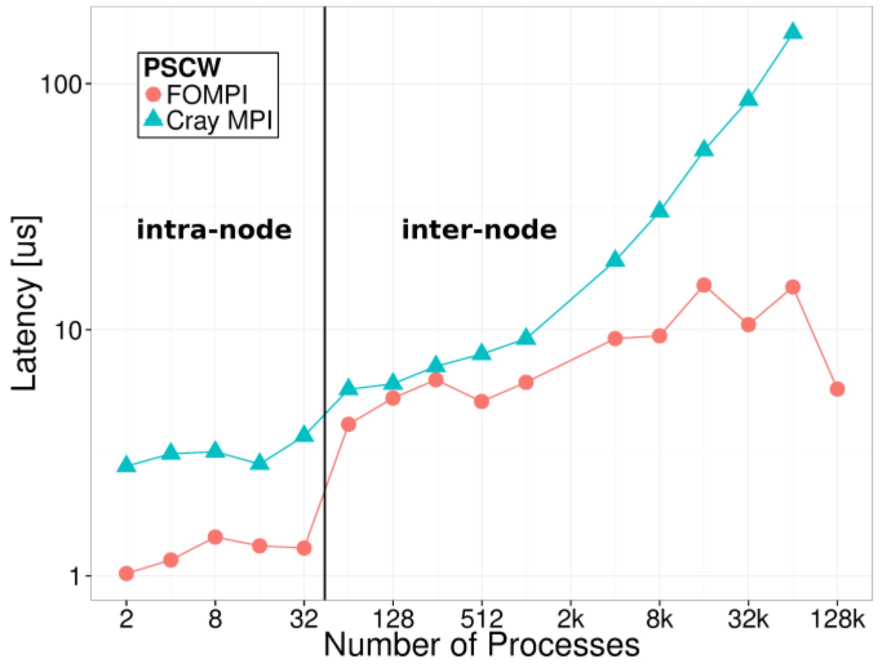


Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

PSCW Performance

Time bound
$\mathcal{P}_{start} = \mathcal{P}_{wait} = \mathcal{O}(1)$ $\mathcal{P}_{post} = \mathcal{P}_{complete} = \mathcal{O}(\log p)$
Memory bound
$\mathcal{O}(\log p)$ (for scalable programs)

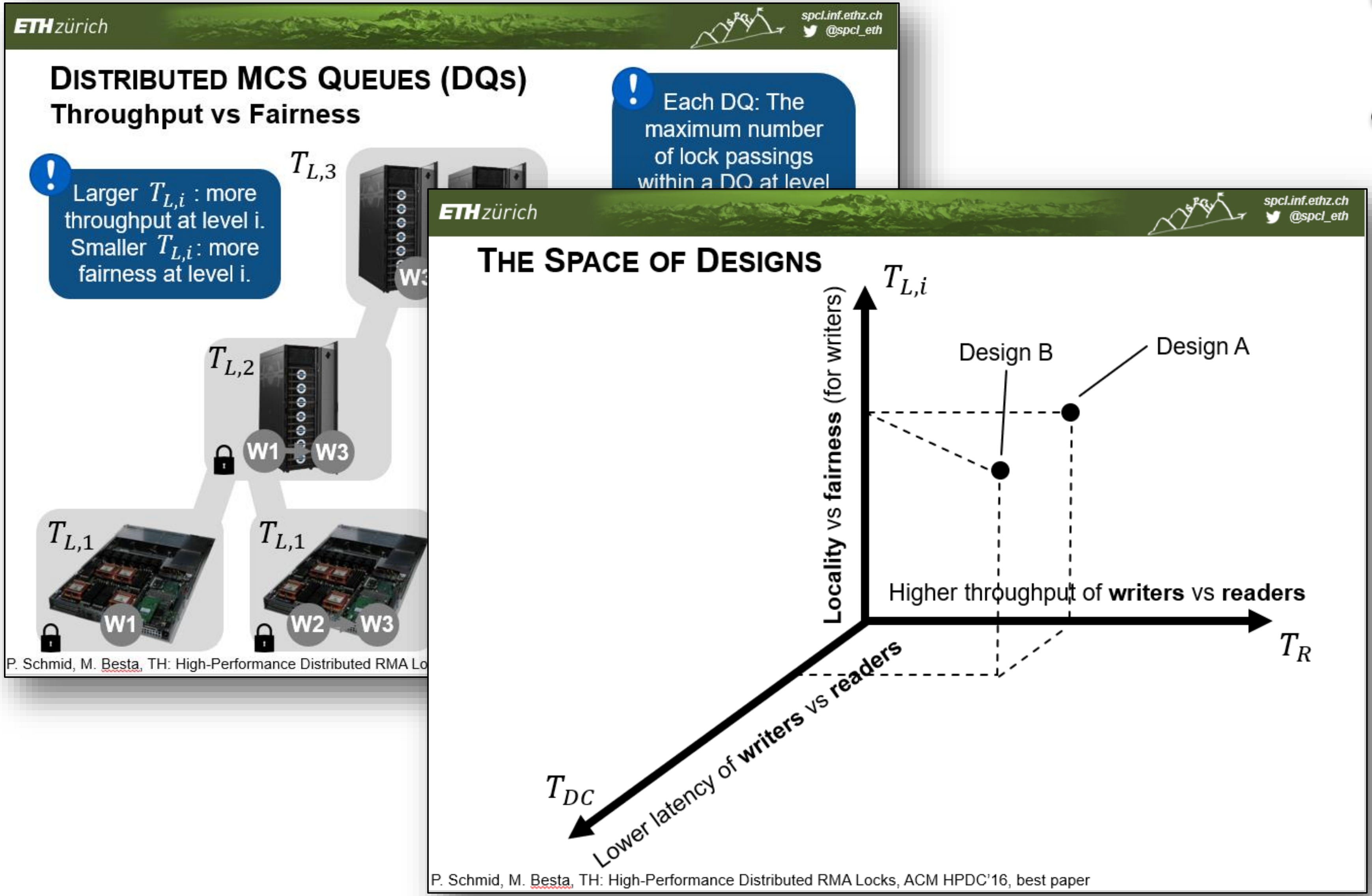




Ring Topology

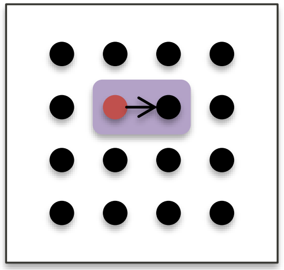
Gerstenberger et al.: „Enabling Highly Scalable Remote Memory Access Programming with MPI-3 One Sided", CACM, Oct. 2018

Scalable Lock Synchronization

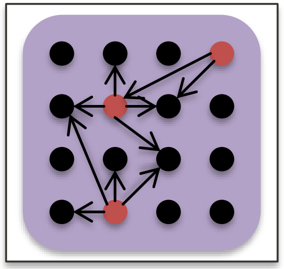


Active process

Passive process



Lock/Unlock (shared/exclusive)



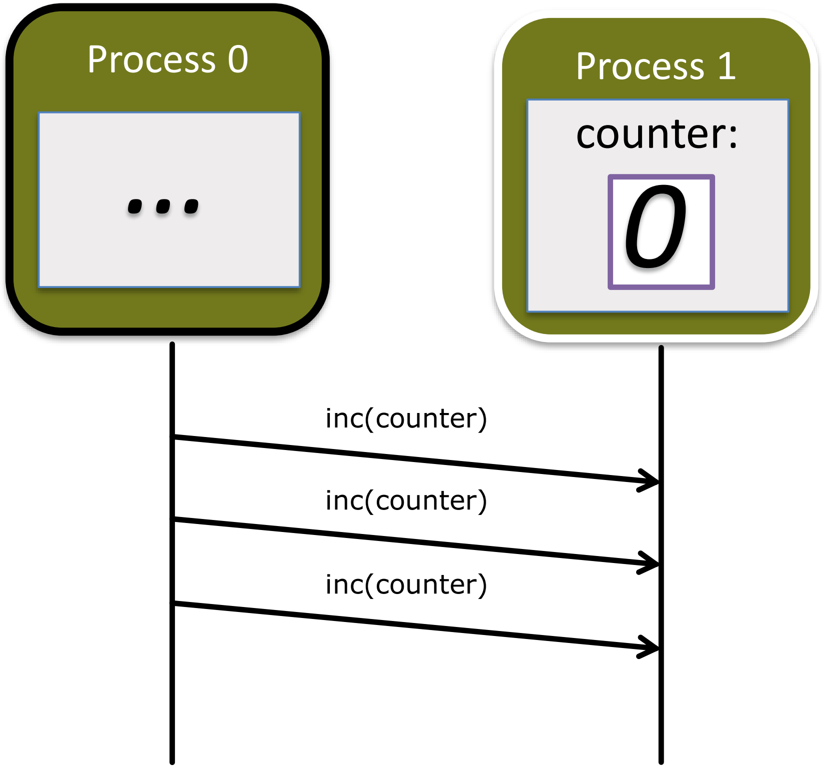
Lock All (always shared)

P. Schmid, M. Besta, TH: High-Performance Distributed RMA Locks, ACM HPDC'16, best paper

Flush Synchronization

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

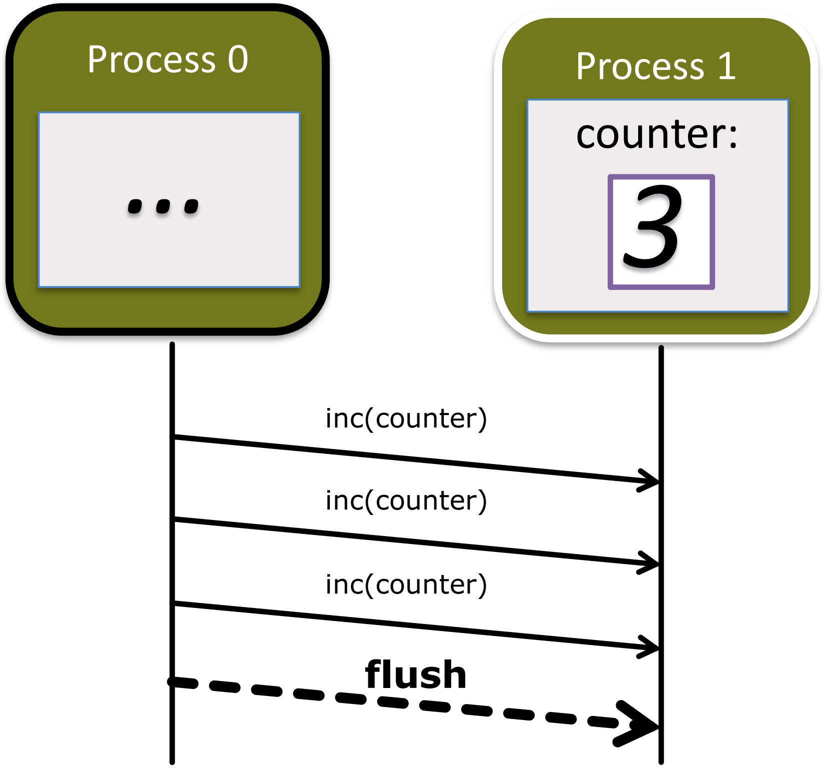
- Guarantees remote completion
- Issues a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path



Flush Synchronization

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

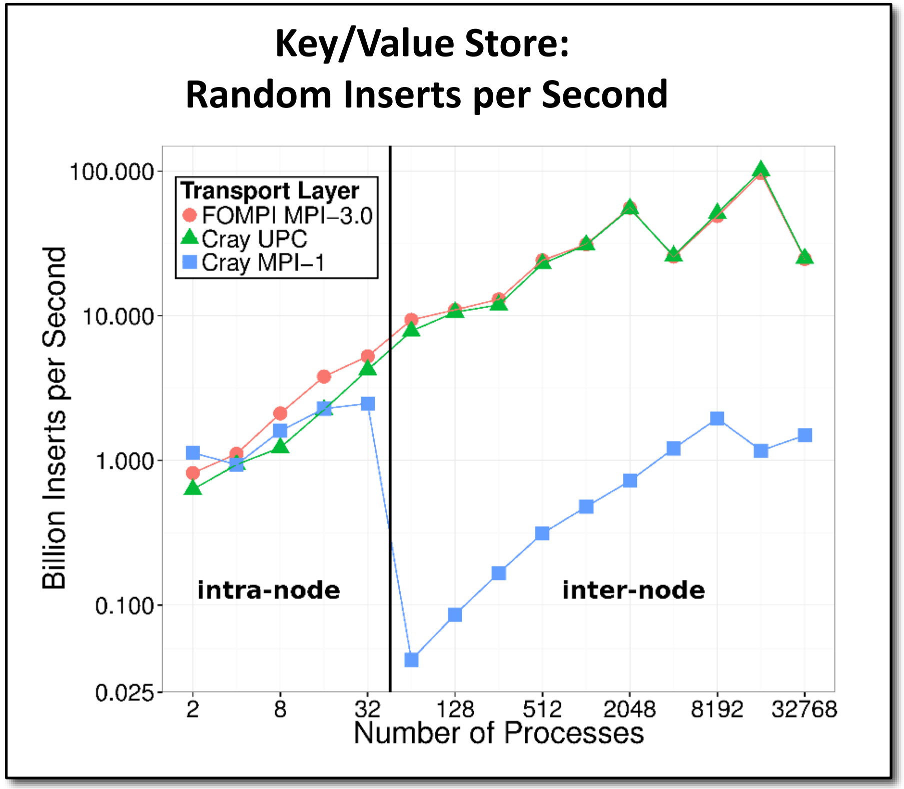
- Guarantees remote completion
- Issues a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path

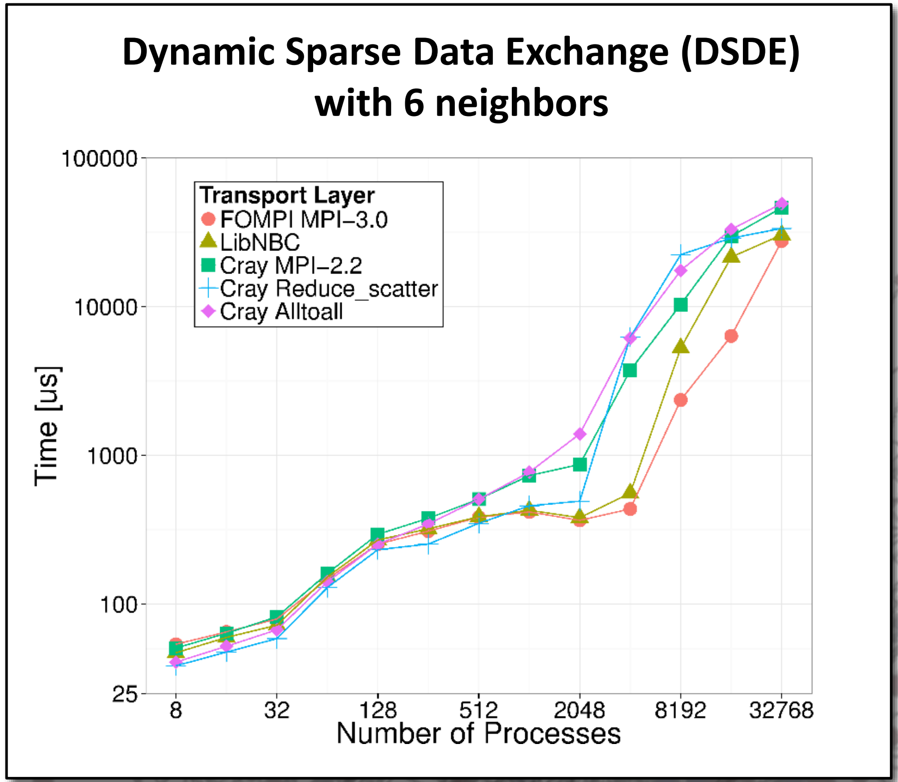


Performance

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

Performance: Motif Applications

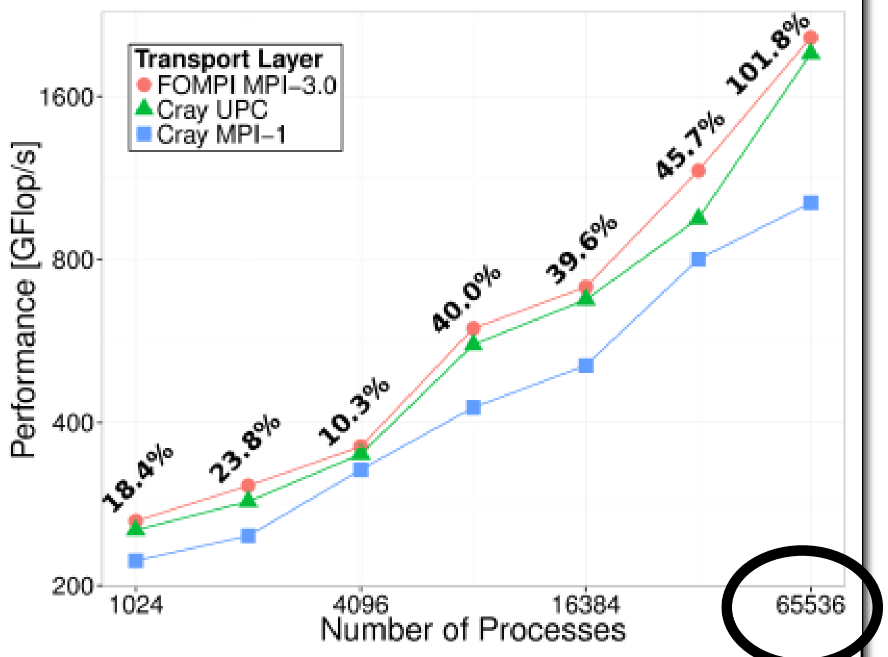




Performance: Applications

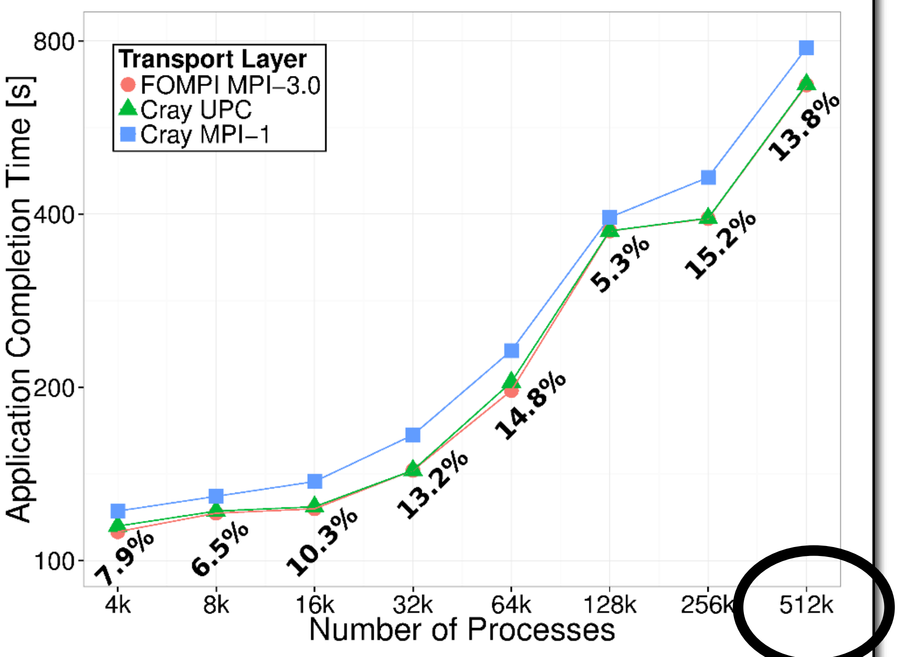
Annotations represent performance gain of foMPI over Cray MPI-1.

NAS 3D FFT [1] Performance



scale
to 65k procs

MILC [2] Application Execution Time



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

The Development of High-Performance Networking Interfaces

Scalable Coherent Interface Myrinet GM+MX Virtual Interface Architecture OFED libfabric

Ethernet+TCP/IP Fast Messages Quadrics QsNet Cray Gemini IB Verbs Portals 4

Quadrics
A Finmeccanica Company

INFINIBAND TRADE ASSOCIATION

OPENFABRICS ALLIANCE

portals

What next?

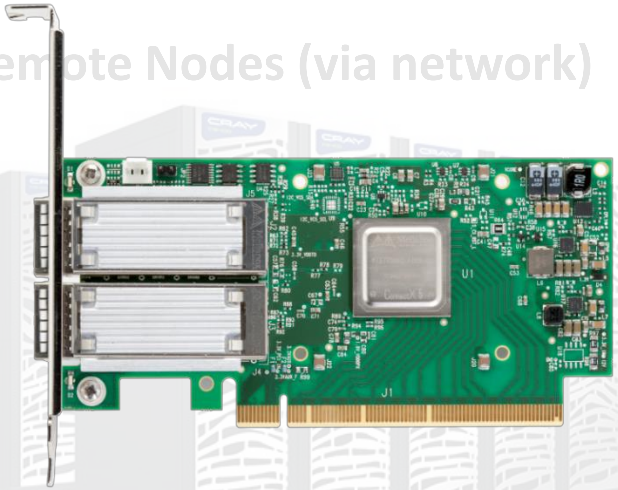
1980 1990 2000 2010 2020

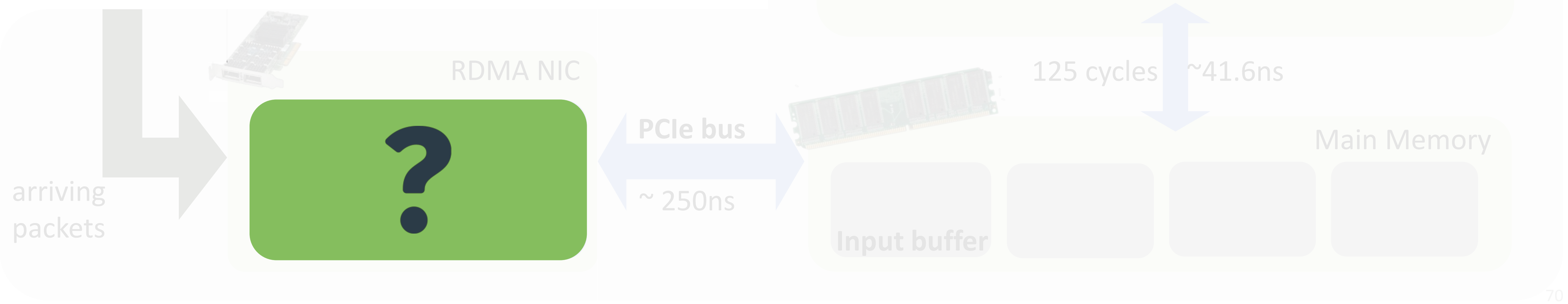
sockets (active) message based protocol offload remote direct memory access (RDMA)

coherent memory access OS bypass zero copy triggered operations

Fundamental development: CPU core speed cannot keep up with the growing network bandwidths! Already today, cannot process packets at line rate!

Data Processing in modern RDMA networks





Mellanox Connect-X5: 1 packet/5ns

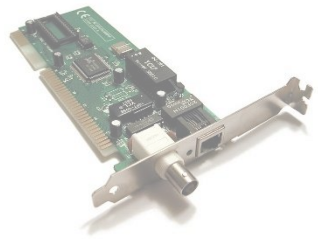
Tomorrow (400G): 1 packet/1.2ns

The future of High-Performance Networking Interfaces

sPIN
Streaming Processing In the Network

Scalable Coherent Interface | Myrinet GM+MX | Virtual Interface Architecture | OFED | libfabric

Ethernet+TCP/IP | Fast Messages | Quadrics QsNet | Cray Gemini | IB Verbs | Portals 4



1980 — 1990 — 2000 — 2010 — 2020

sockets | (active) message based | protocol offload | remote direct memory access (RDMA)

coherent memory access | OS bypass | zero copy | triggered operations

fully programmable NIC acceleration

Established Principles for Compute Acceleration

Specialization | Programmability | Libraries

Ease-of-use | Portability | Efficiency

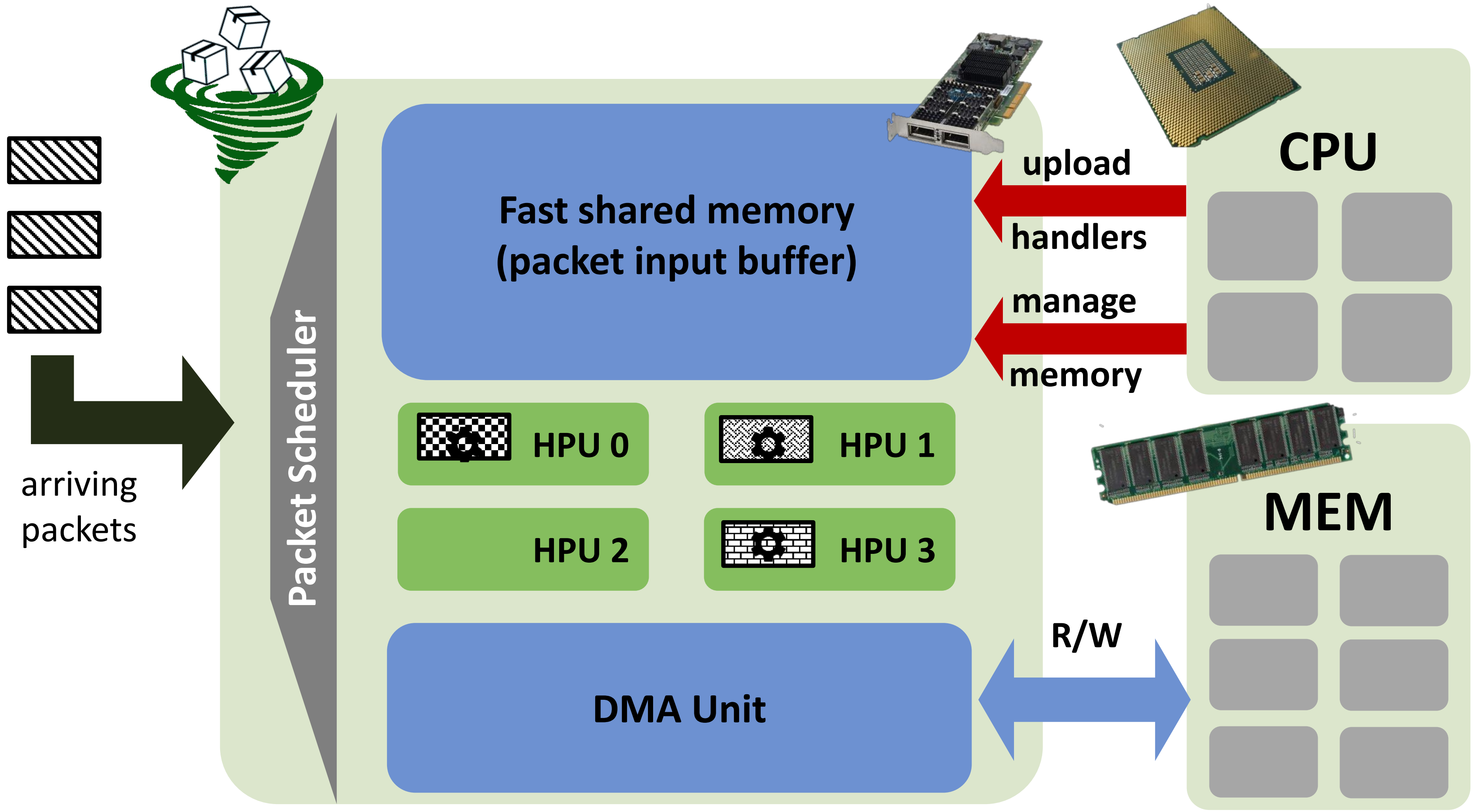
NVIDIA CUDA | OpenCL | OpenMP 4.0

TOP500 SUPERCOMPUTER SITES — June 2017

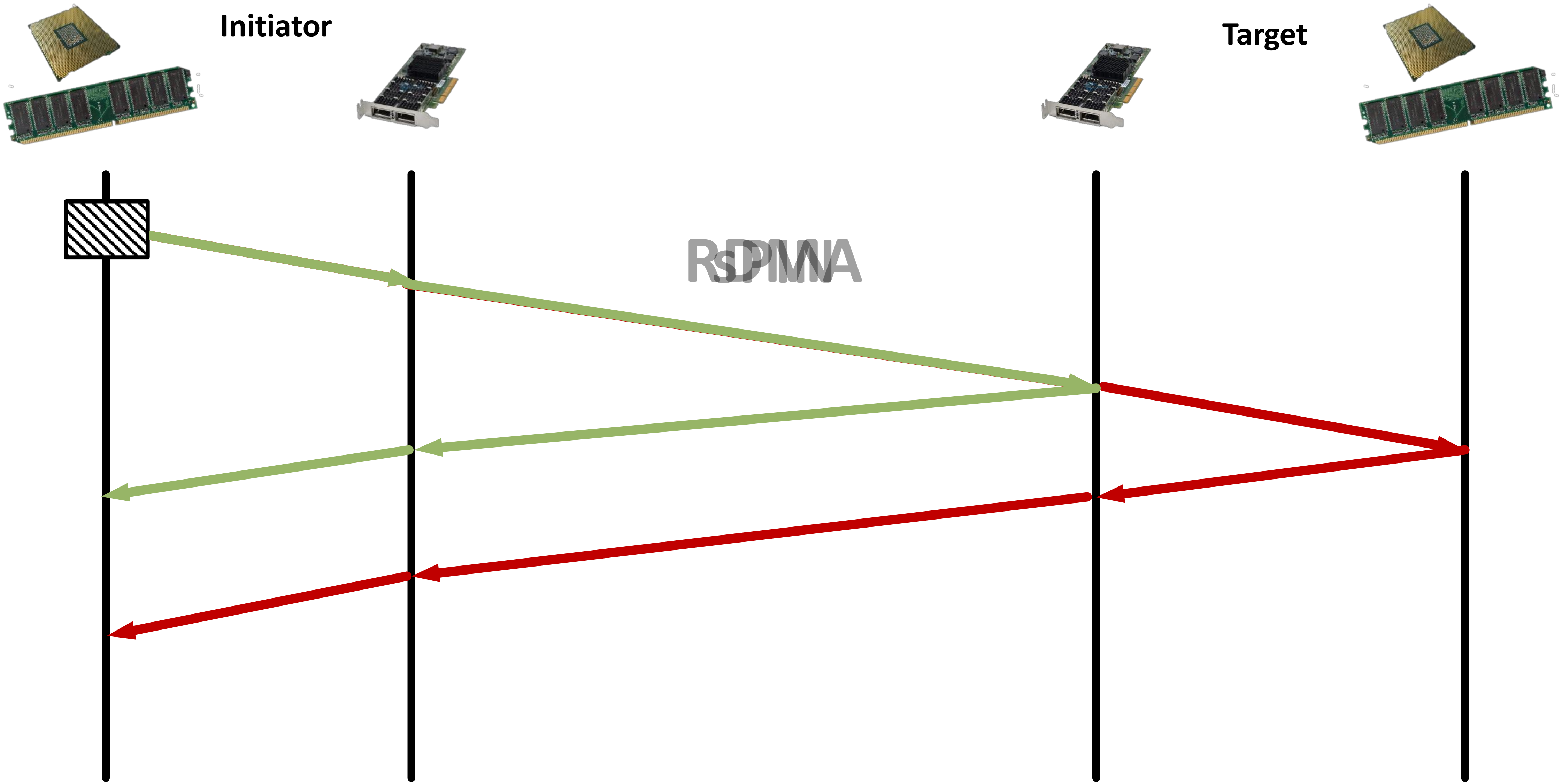
95 / top-100 systems use RDMA

>285 / top-500 systems use RDMA

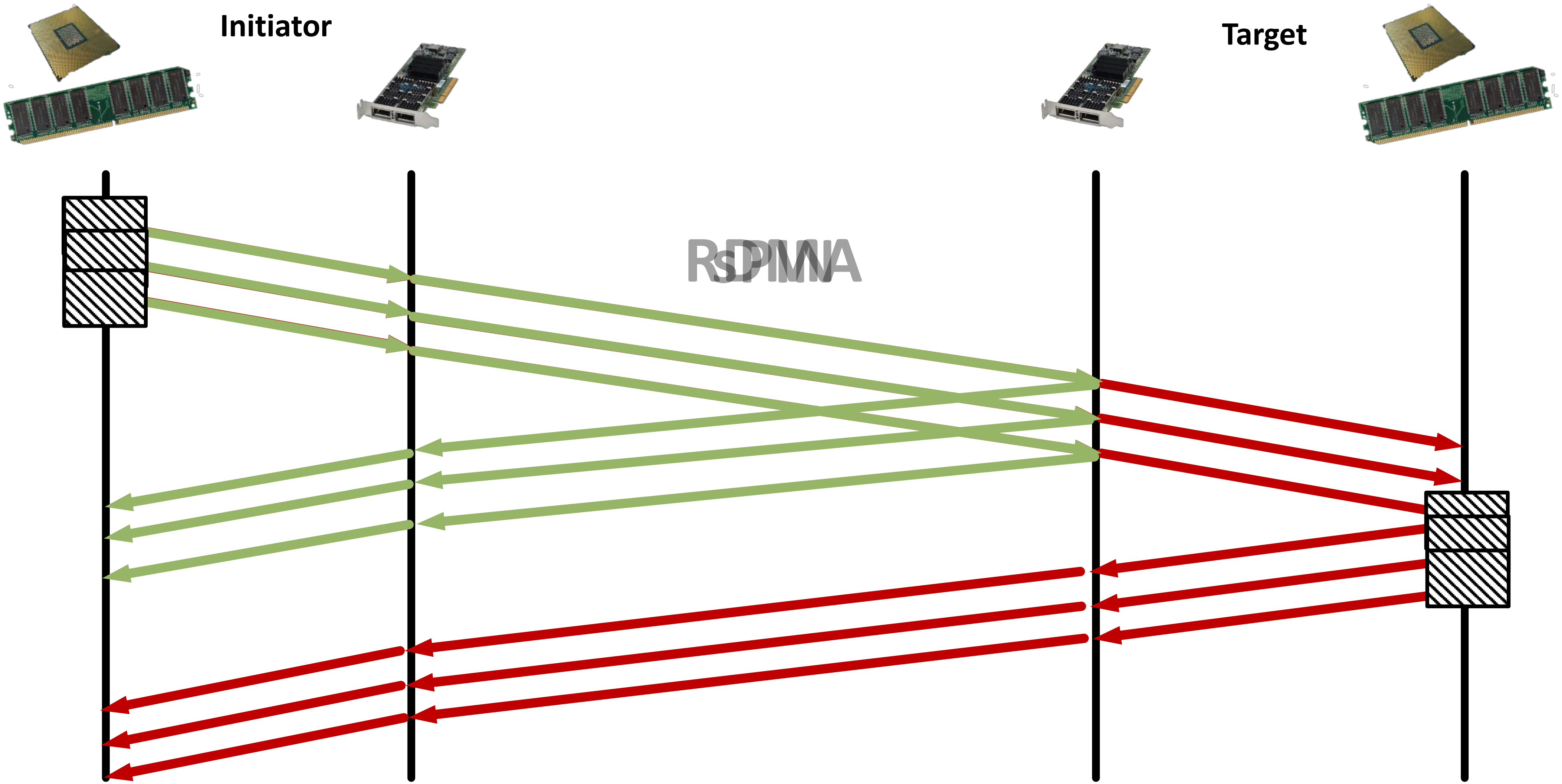
sPIN NIC - Abstract Machine Model



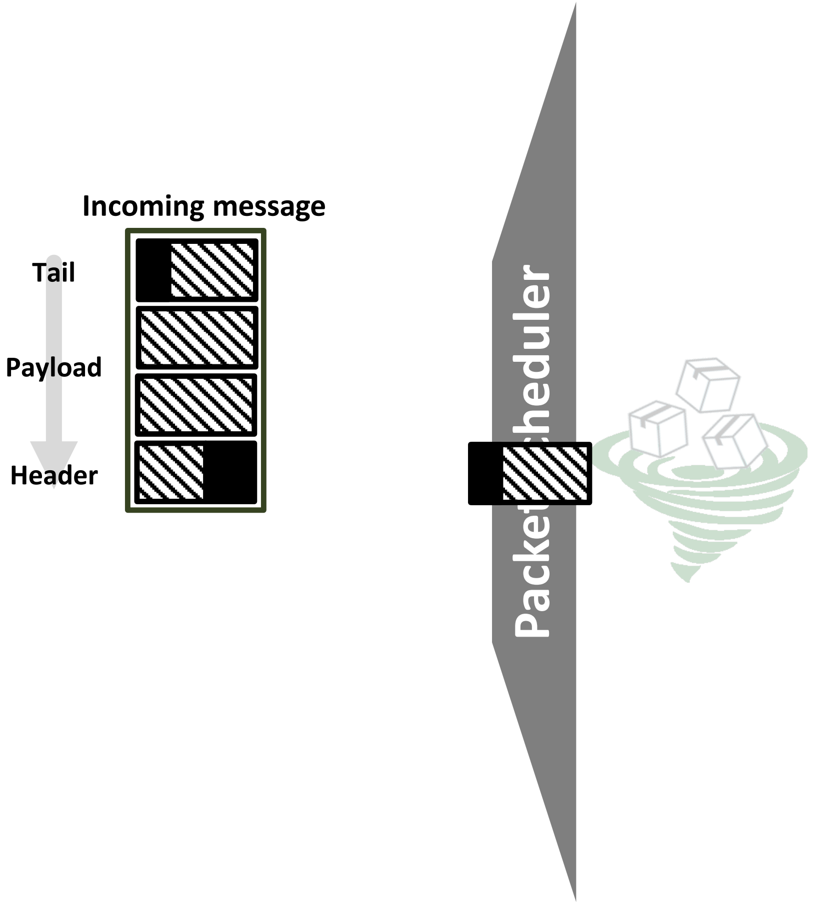
RDMA vs. sPIN in action: Simple Ping Pong



RDMA vs. sPIN in action: Streaming Ping Pong



sPIN – Programming Interface



Header handler

```
__handler int pp_header_handler(const ptl_header_t h, void *state) {
   pingpong_info_t *i = state;
   i->source = h.source_id;
   return PROCESS_DATA; // execute payload handler to put from device
}
```

Payload handler

```
__handler int pp_payload_handler(const ptl_payload_t p, void * state) {
   pingpong_info_t *i = state;
   PtlHandlerPutFromDevice(p.base, p.length, 1, 0, i->source, 10, 0, NULL, 0);
   return SUCCESS;
}
```

Completion handler

```
__handler int pp_completion_handler(int dropped_bytes,
                                    bool flow_control_triggered, void *state) {
   return SUCCESS;
}
```

```
connect(peer, /* … */, &pp_header_handler, &pp_payload_handler, &pp_completion_handler);
```

Possible sPIN implementations

- **sPIN is a programming abstraction, similar to CUDA or OpenCL combined with OFED or Portals 4**
  - It enables a large variety of NIC implementations!
  - For example, massively multithreaded HPUs
    Including warp-like scheduling strategies
- **Main goal: sPIN must not obstruct line-rate**
  - Programmer must limit processing time per packet
    Little's Law: 500 instructions per handler, 2.5 GHz, IPC=1, 1 Tb/s → 25 kiB memory
  - Relies on fast shared memory (processing in packet buffers)
    Scratchpad or registers
  - Quick (single-cycle) handler invocation on packet arrival
    Pre-initialized memory & context
- **Can be implemented in most RDMA NICs with a firmware update**
  - Or in software in programmable (Smart) NICs

at 400G, process more than 833 million messages/s

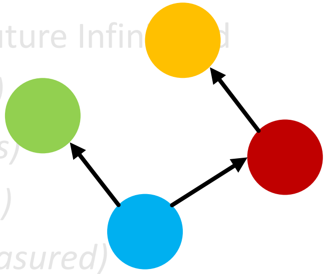
Mellanox Technologies
Innova Flex (Kintex FPGA)

Microsoft
Catapult (Virtex FPGA)

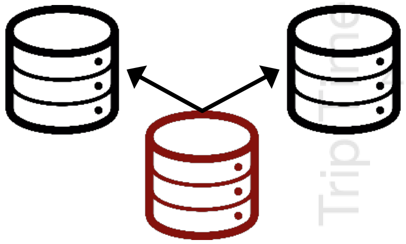
Broadcom
BCM58800 SoC (Full Linux)

Simulating a sPIN NIC – Ping Pong

- LogGOPSim v2 [1]: combine LogGOPSim (packet-level network) with gem5 (cycle accurate CPU simulation)
- Network (LogGOPSim):
  - Supports Portals 4 and MPI
  - Parametrized for future InfiniBand
    - *o=65ns (measured)*
    - *g=6.7ns (150 MM/s)*
    - *G=2.5ps (400 Gib/s)*
    - *Switch L=50ns (measured)*
    - *Wire L=33.4ns (10m cable)*
- NIC HPU
  - 2.5 GHz ARM Cortex A15 OOO
  - ARMv8-A 32 bit ISA
  - Single-cycle access SRAM (no DRAM)
  - Header matching m=30ns, per packet 2ns
    - *In parallel with g!*

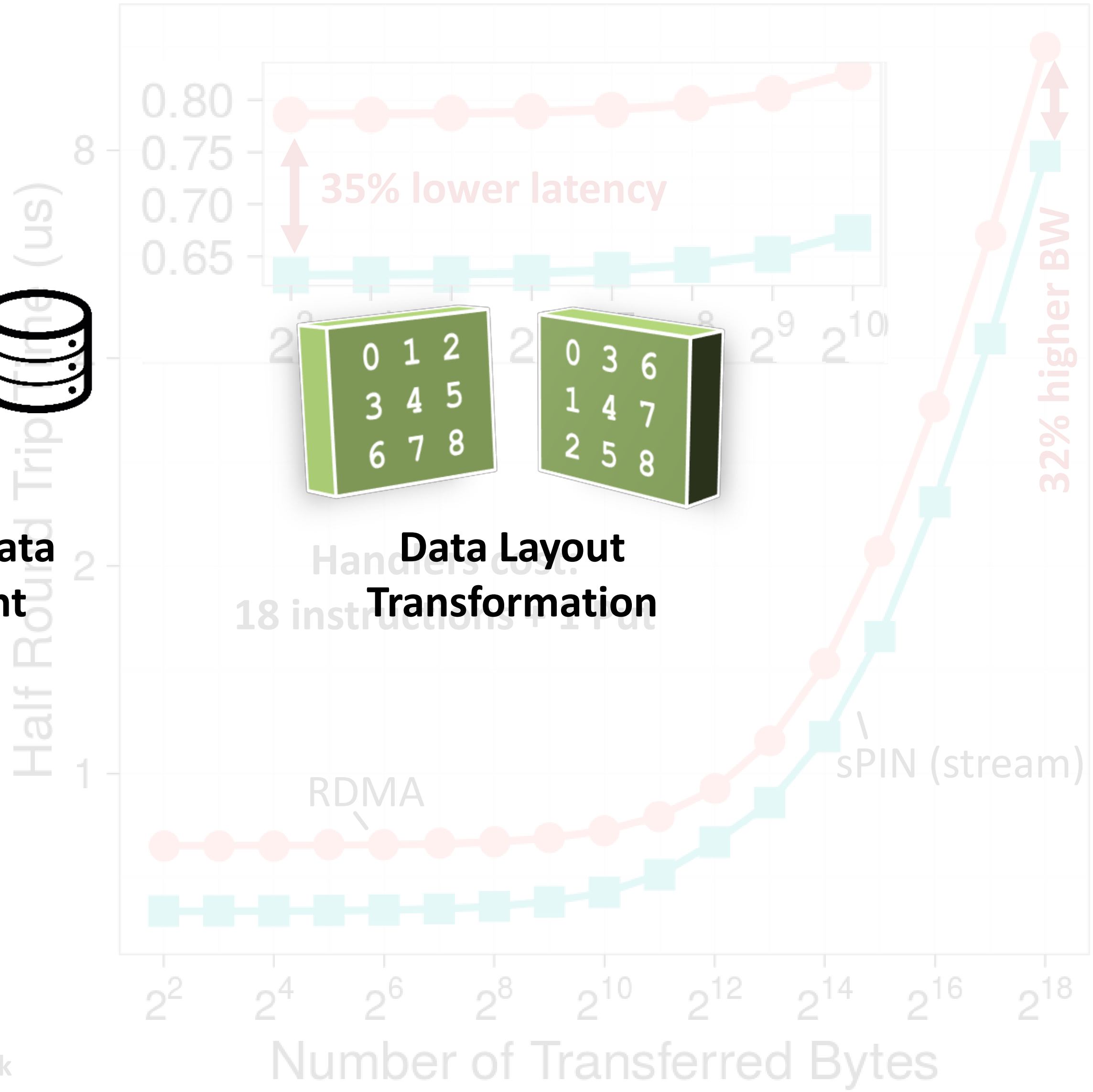


Network Group Communication



Distributed Data Management

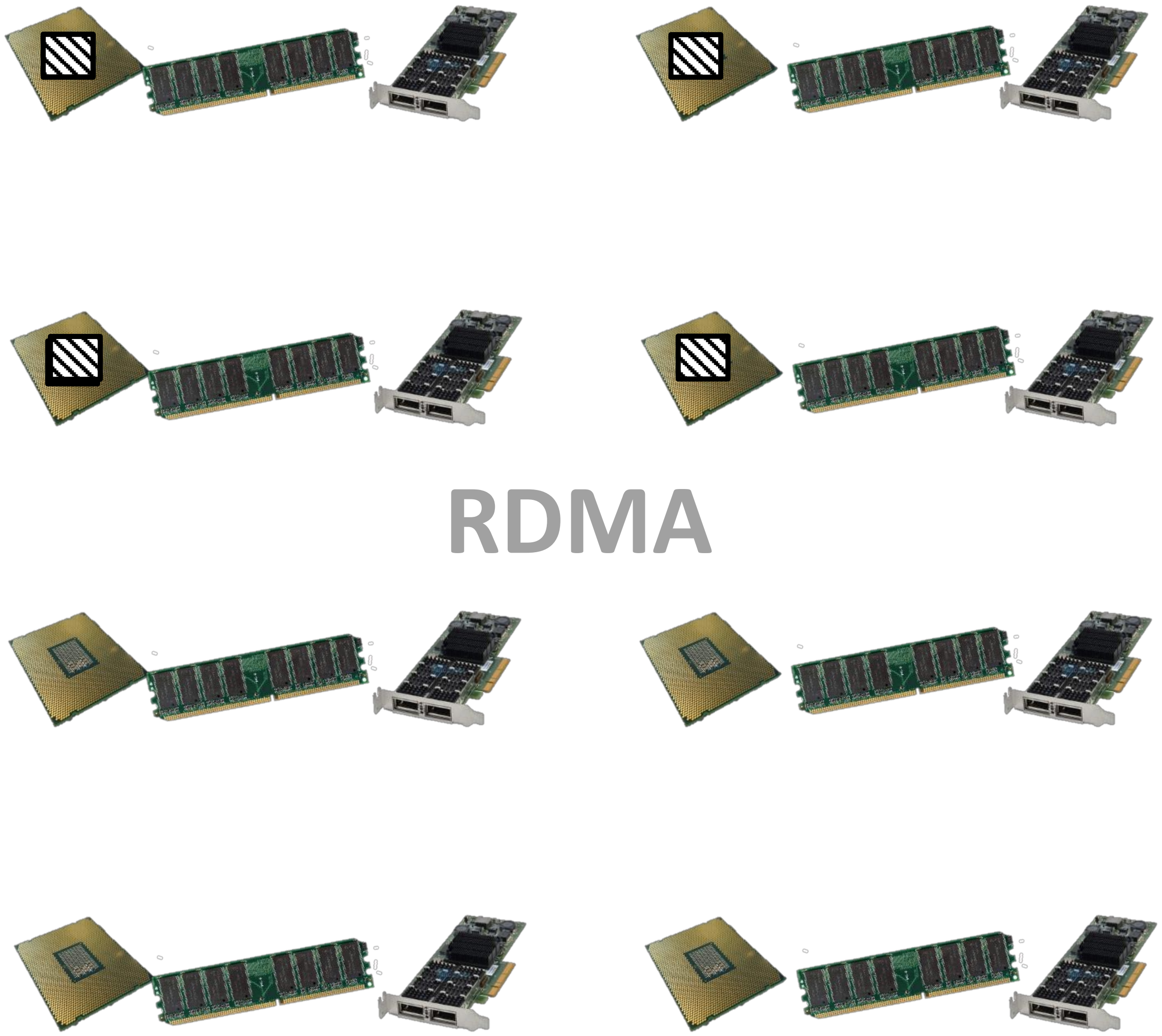
Data Layout Transformation

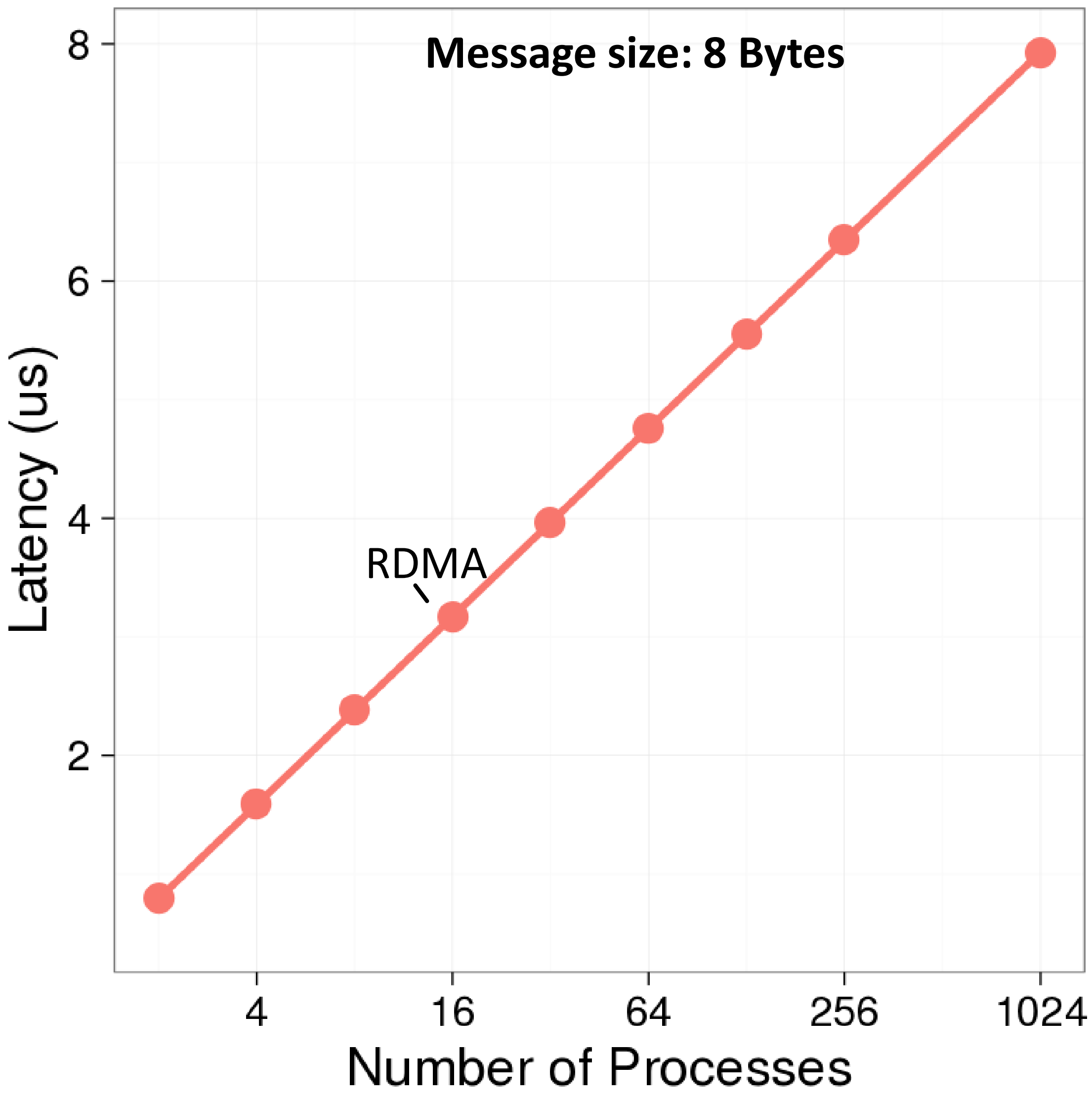


[1] S. Di Girolamo, K. Taranov, T. Schneider, E. Stalder, T. Hoefler, **LogGOPSim+gem5: Simulating Network Offload Engines Over Packet-Switched Networks**. Presented at ExaMPI'17

Use Case 1: Broadcast acceleration

Network Group Communication

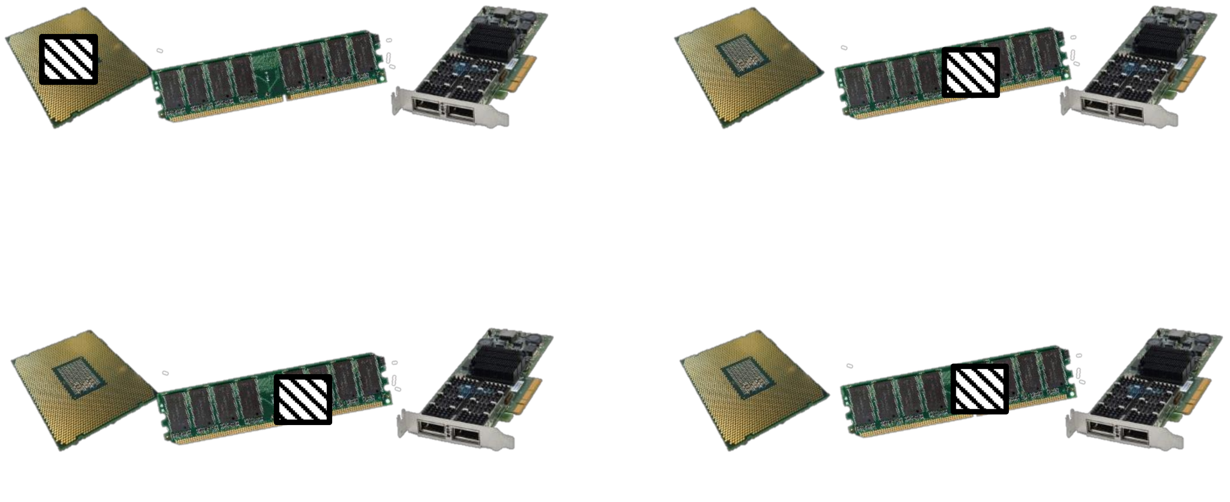




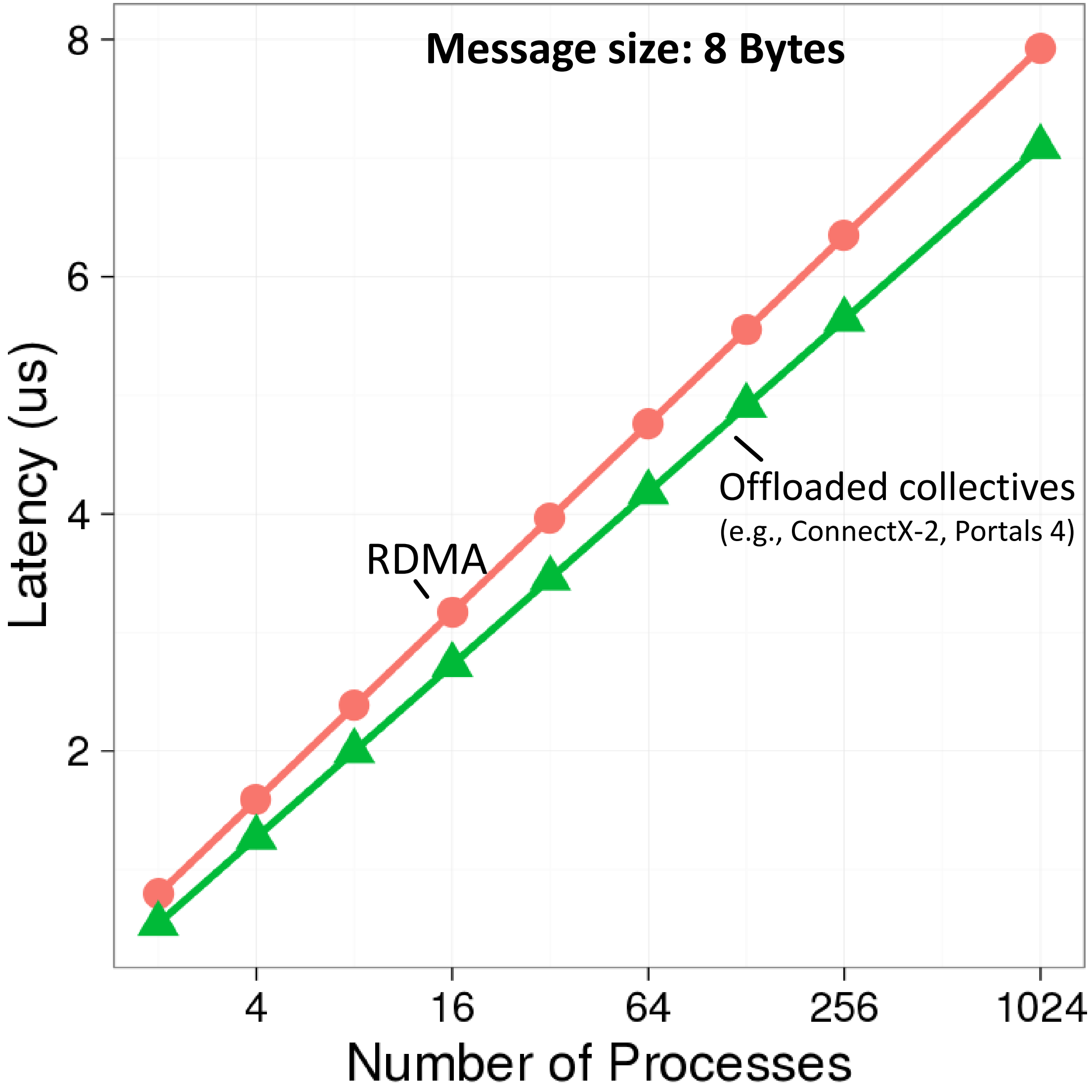
Liu, J., et al., High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming* 2004

Use Case 1: Broadcast acceleration

Network Group Communication



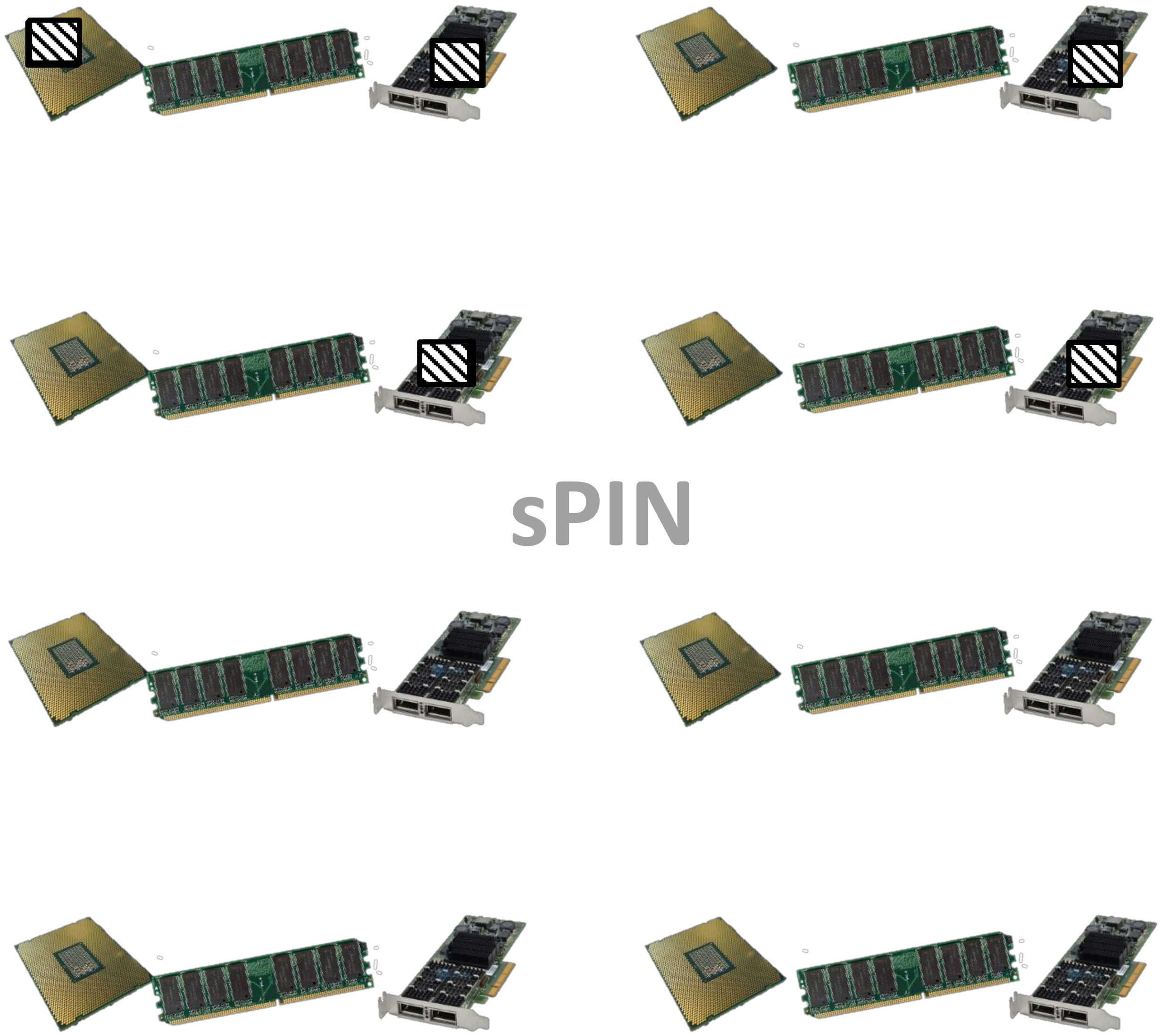
Portals 4



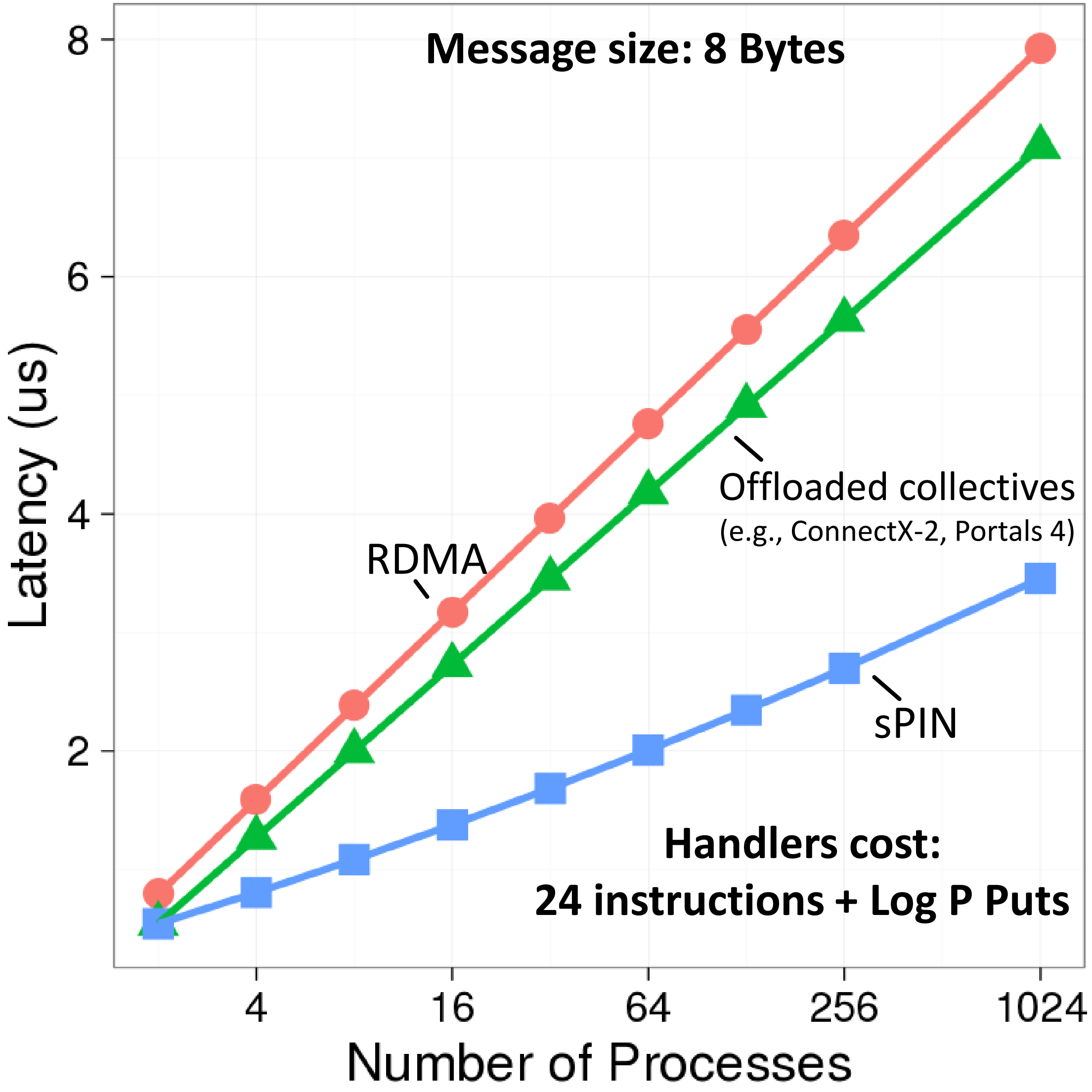
Underwood, K.D., et al., Enabling flexible collective communication offload with triggered operations. *HOTI'11*
Liu, J., et al., High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming* 2004

Use Case 1: Broadcast acceleration

Network Group Communication



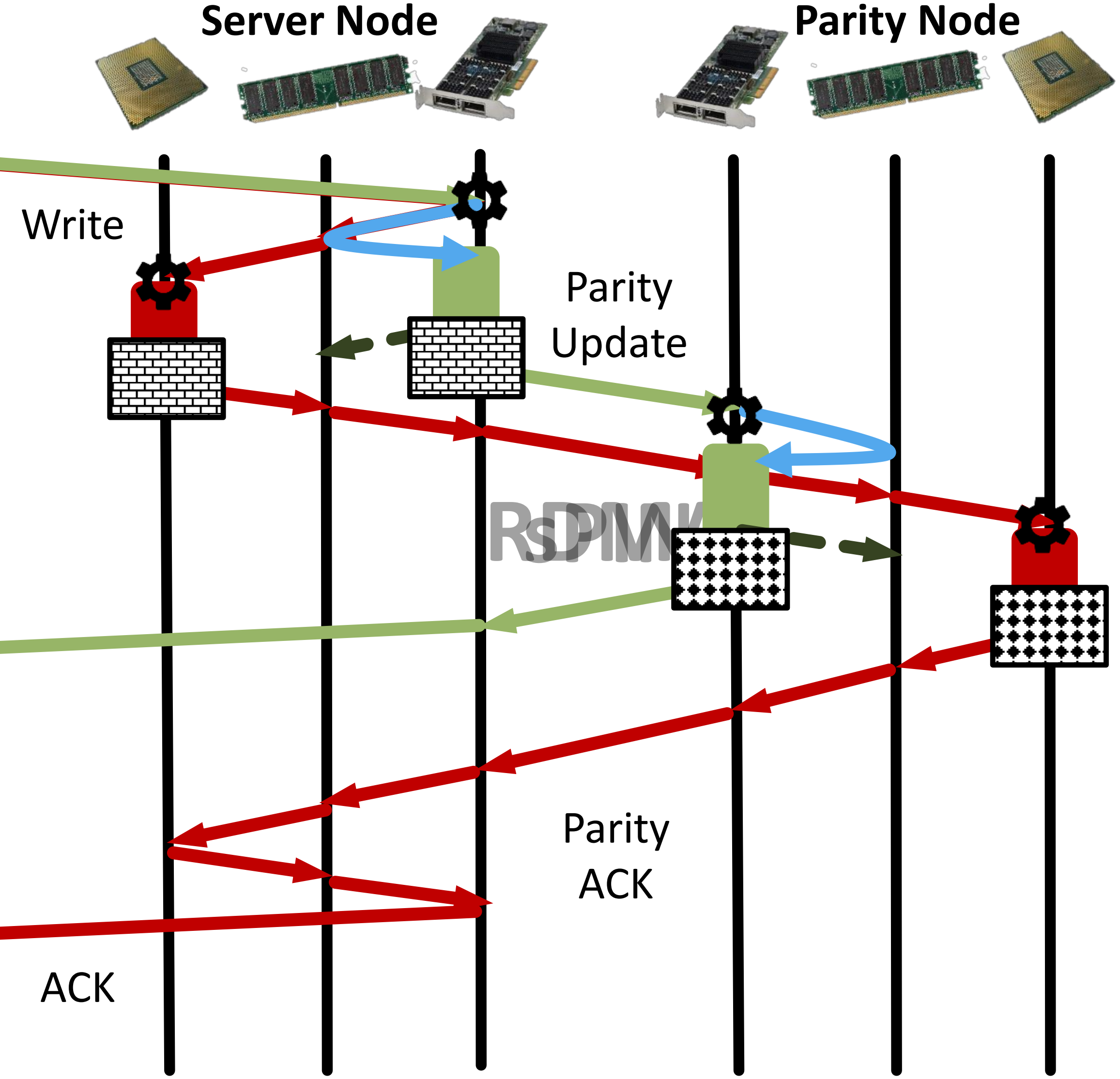
sPIN

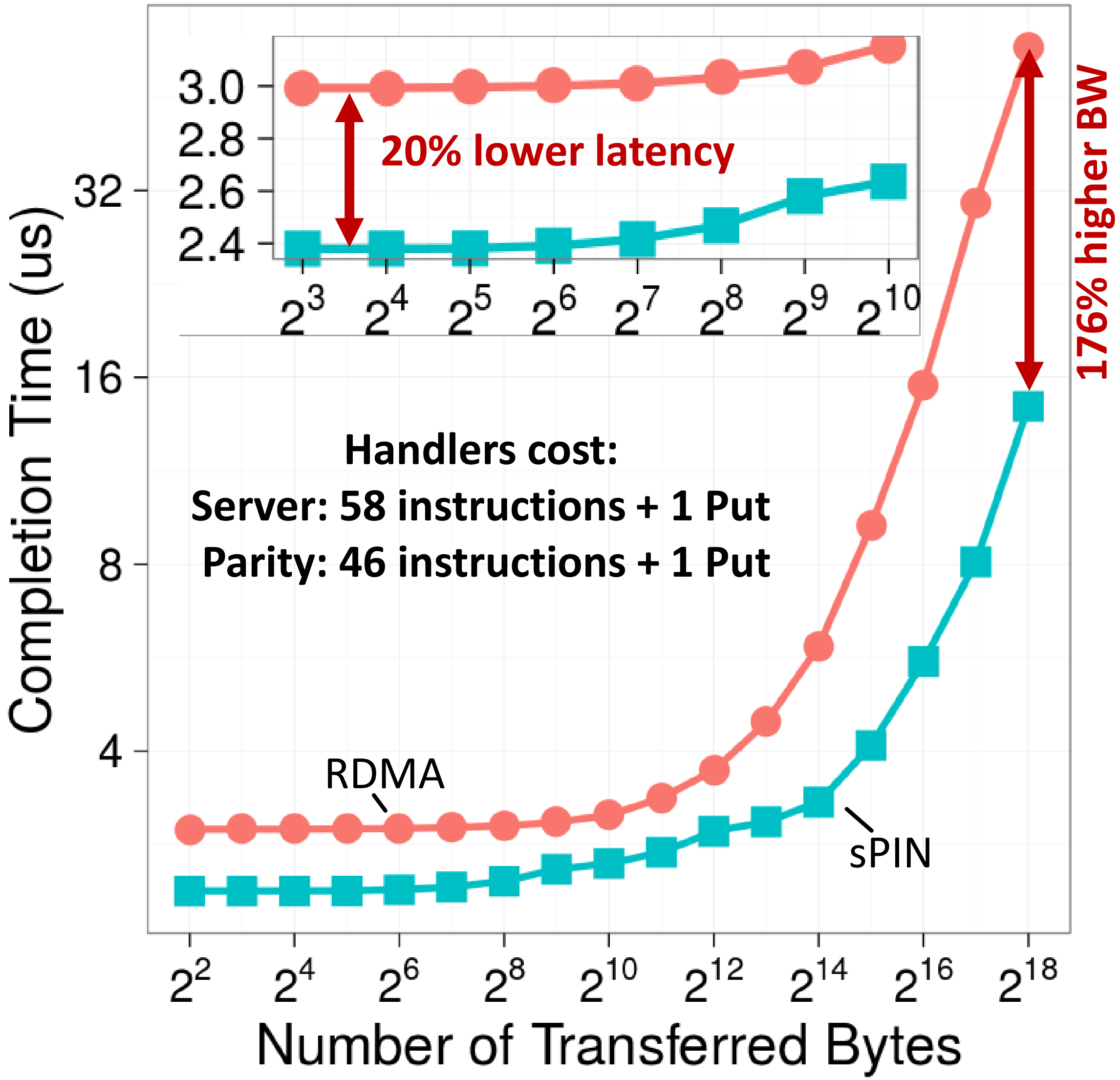


Underwood, K.D., et al., Enabling flexible collective communication offload with triggered operations. *HOTI'11*
Liu, J., et al., High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming* 2004

Use Case 2: RAID acceleration

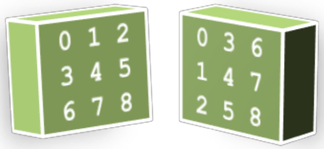
Distributed Data Management

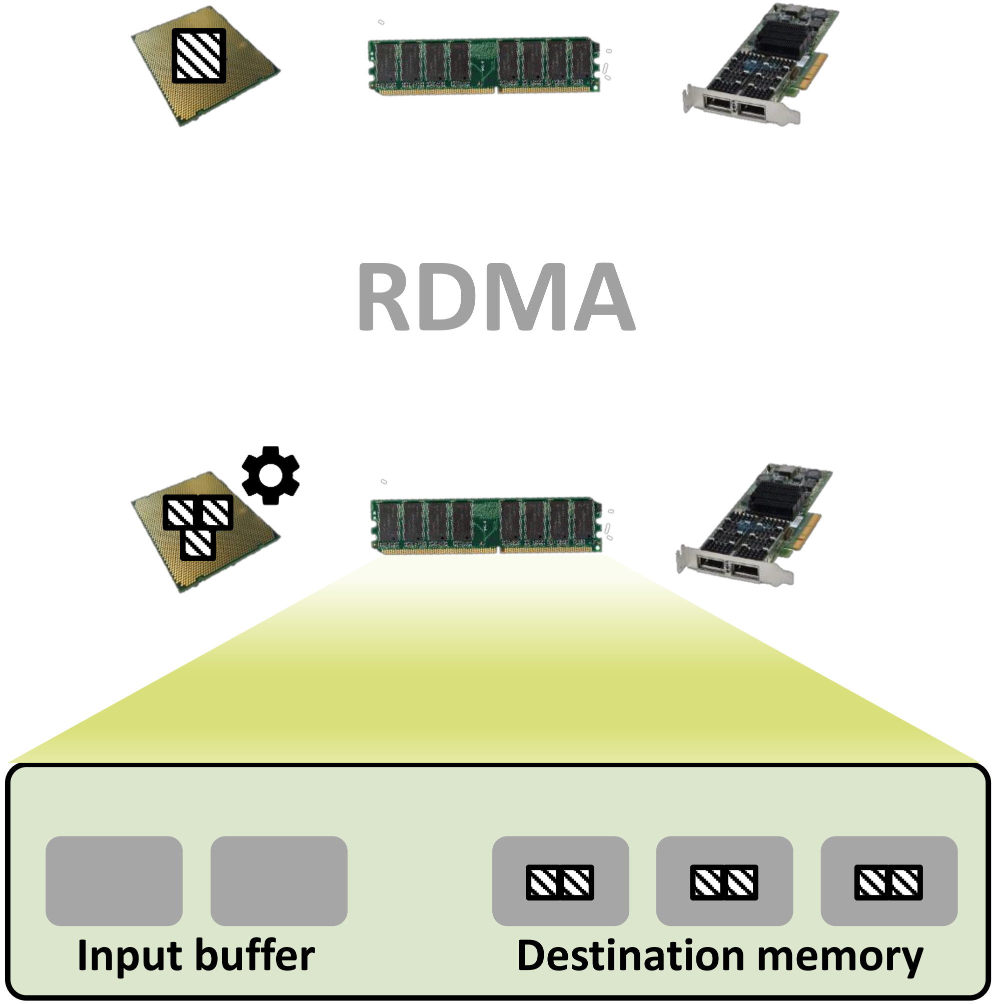


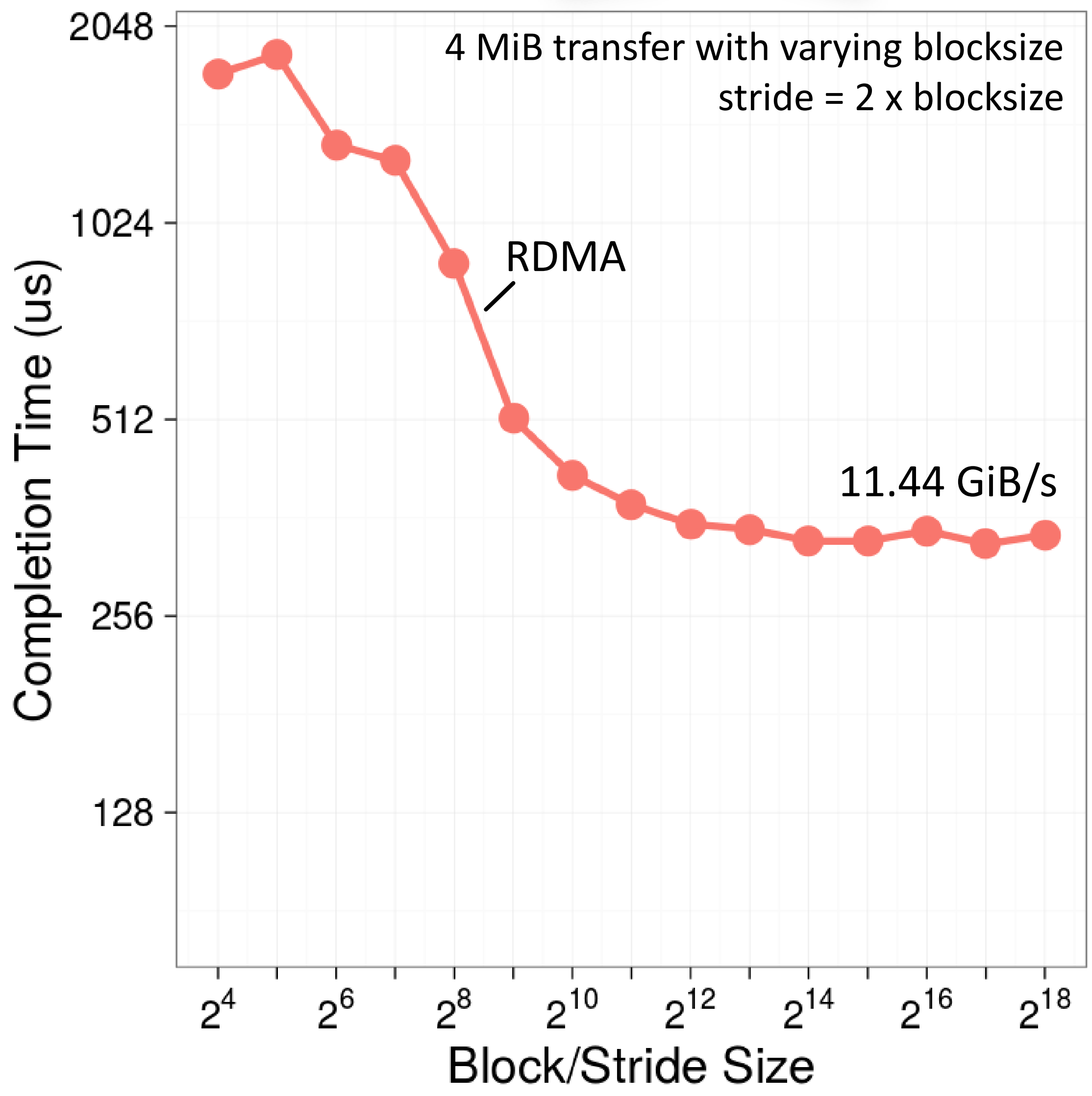


Shankar D. et al., High-performance and Resilient Key-Value Store with Online Erasure Coding for Big Data Workloads. ICDCS'17

Use Case 3: MPI Datatypes acceleration

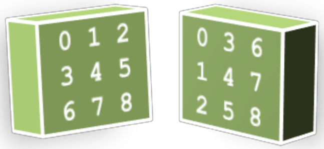




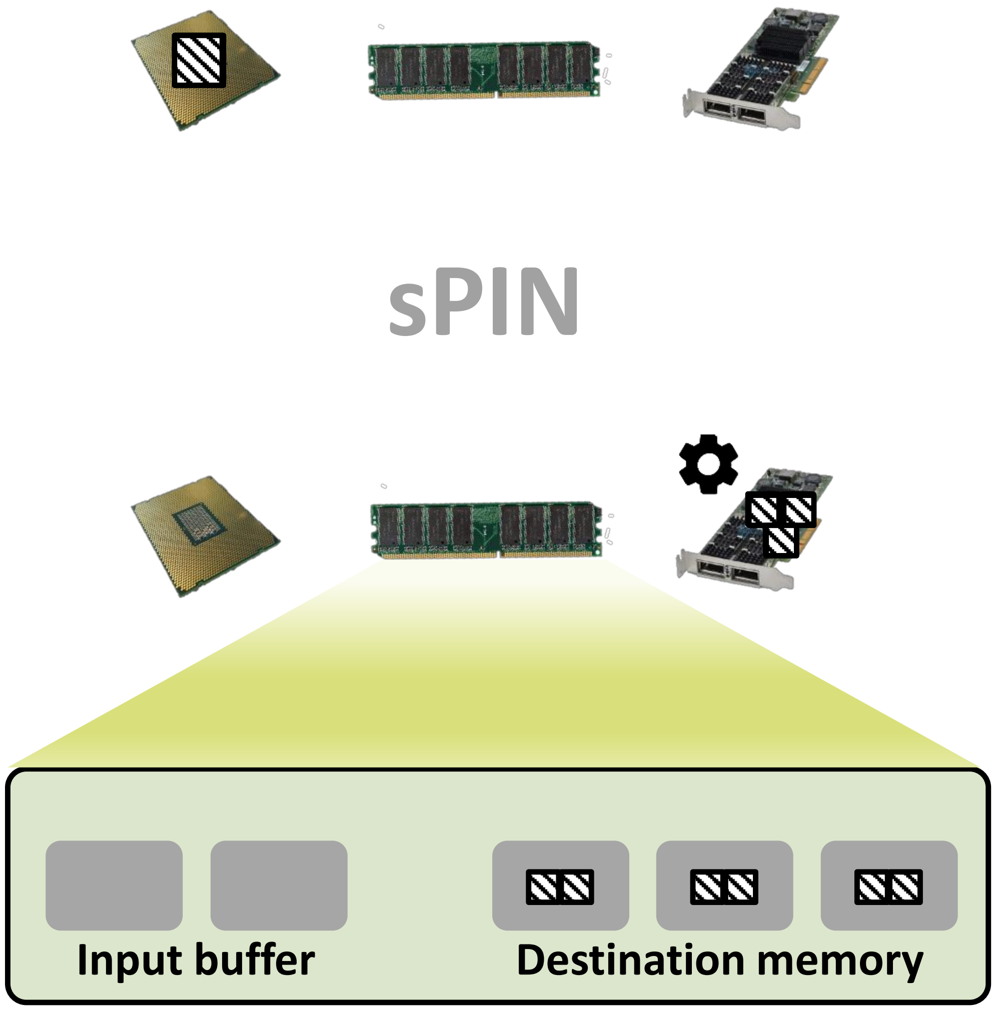


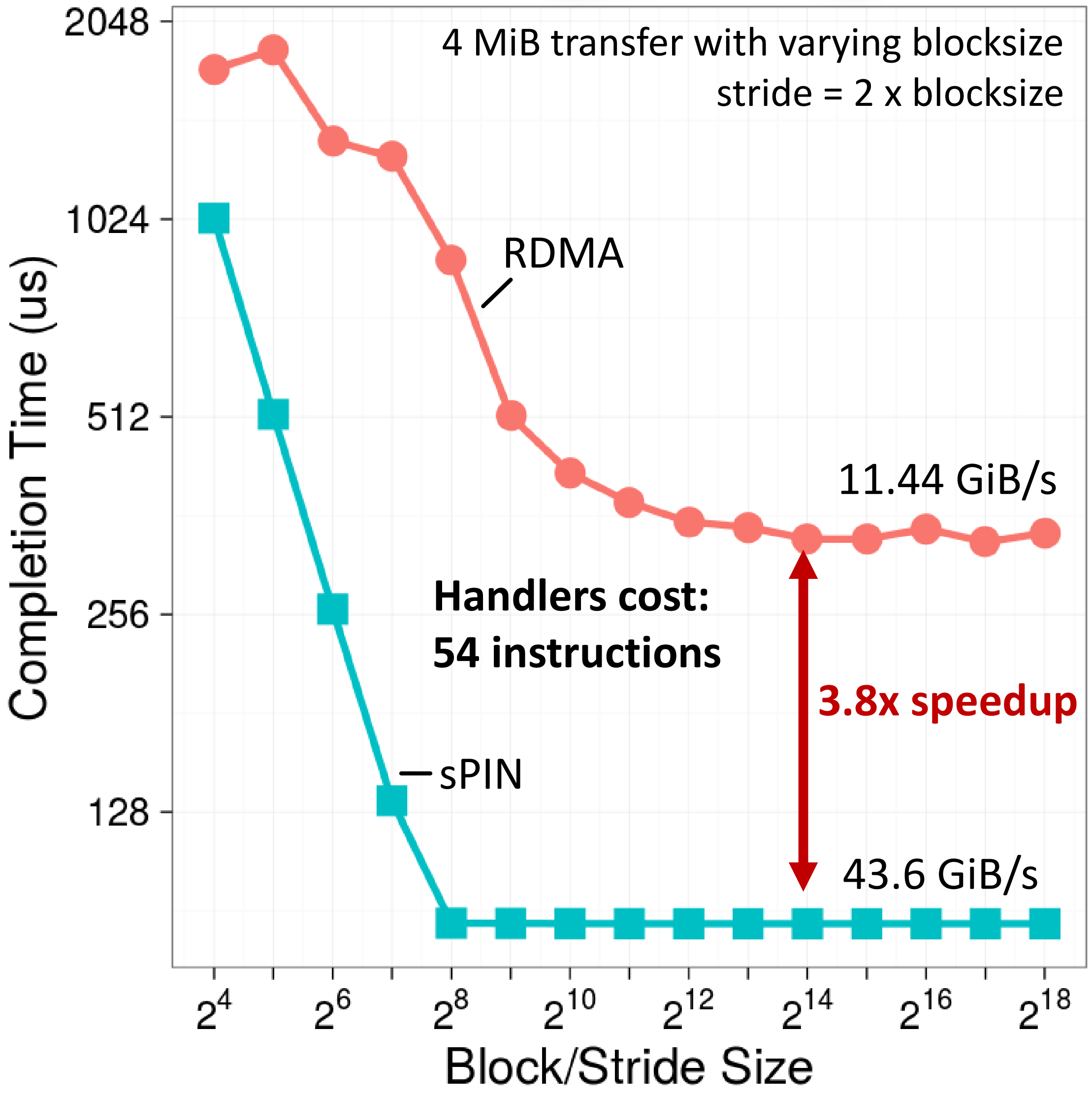
Gropp, W., et al., March. Improving the performance of MPI derived datatypes. *MPIDC'99*

Use Case 3: MPI Datatypes acceleration



Data Layout Transformation





Gropp, W., et al., March. Improving the performance of MPI derived datatypes. *MPIDC'99*

Further results and use-cases

Further results and use-cases

SPCL ETH zürich

Use Case 4: MPI Rendezvous Protocol

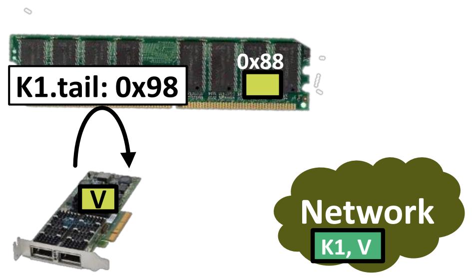
program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

Further results and use-cases

Use Case 4: MPI Rendezvous Protocol

program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

Use Case 5: Distributed KV Store



41% lower latency

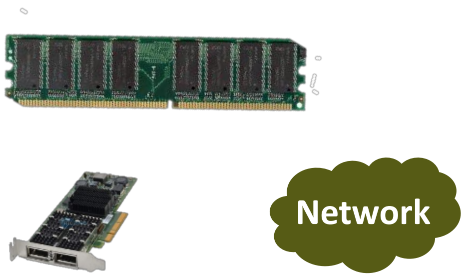
Kalia, A., et al., Using RDMA efficiently for key-value services. In *ACM SIGCOMM Computer Communication Review, 2014*

Further results and use-cases

Use Case 4: MPI Rendezvous Protocol

program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

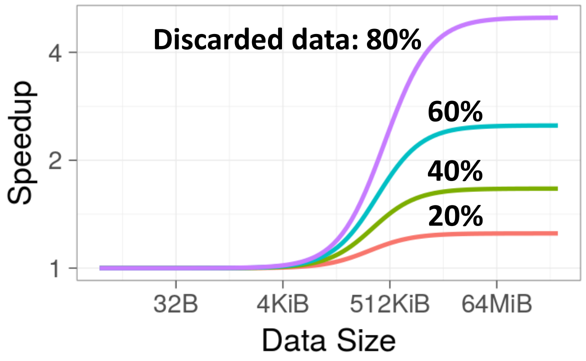
Use Case 5: Distributed KV Store



41% lower latency

Kalia, A., et al., Using RDMA efficiently for key-value services. In *ACM SIGCOMM Computer Communication Review, 2014*

Use Case 6: Conditional Read



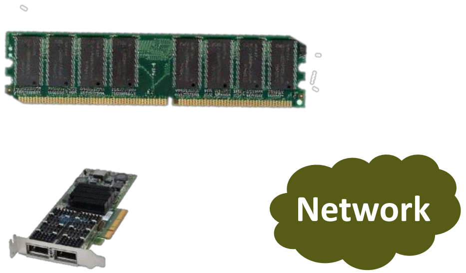
Barthels, C., et al., Designing Databases for Future High-Performance Networks. *IEEE Data Eng. Bulletin, 2017*

Further results and use-cases

Use Case 4: MPI Rendezvous Protocol

program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

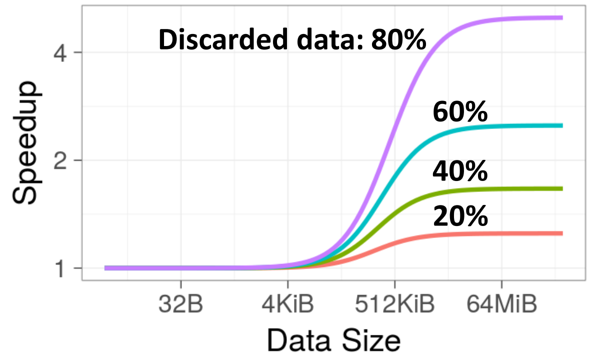
Use Case 5: Distributed KV Store



41% lower latency

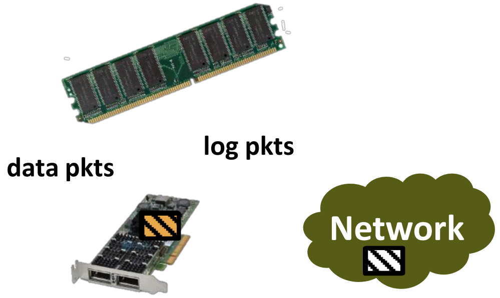
Kalia, A., et al., Using RDMA efficiently for key-value services. In *ACM SIGCOMM Computer Communication Review, 2014*

Use Case 6: Conditional Read



Barthels, C., et al., Designing Databases for Future High-Performance Networks. *IEEE Data Eng. Bulletin, 2017*

Use Case 7: Distributed Transactions



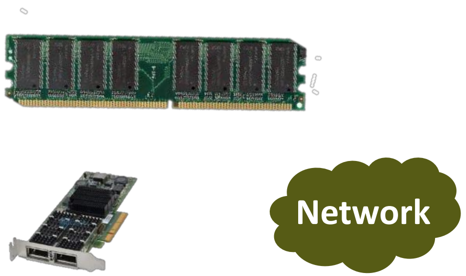
Dragojević, A, et al., No compromises: distributed transactions with consistency, availability, and performance. SOSP’15

Further results and use-cases

Use Case 4: MPI Rendezvous Protocol

program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

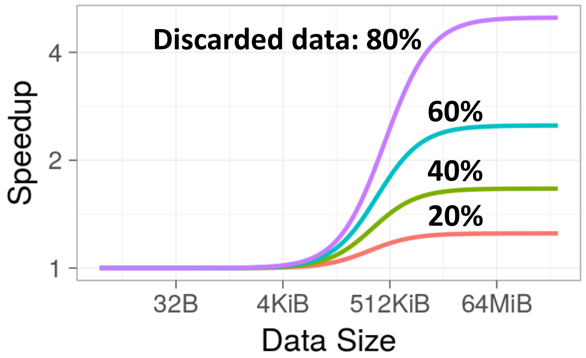
Use Case 5: Distributed KV Store



41% lower latency

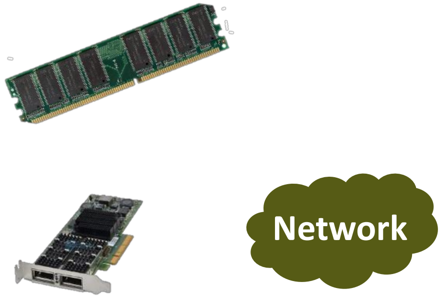
Kalia, A., et al., Using RDMA efficiently for key-value services. In *ACM SIGCOMM Computer Communication Review, 2014*

Use Case 6: Conditional Read



Barthels, C., et al., Designing Databases for Future High-Performance Networks. *IEEE Data Eng. Bulletin, 2017*

Use Case 7: Distributed Transactions



Dragojević, A, et al., No compromises: distributed transactions with consistency, availability, and performance. SOSP'15

Use Case 8: FT Broadcast

Bosilca, G., et al., Failure Detection and Propagation in HPC systems. SC'16

Further results and use-cases

Use Case 4: MPI Rendezvous Protocol

program	p	msgs	ovhd	ovhd	red
MILC	64	5.7M	5.5%	1.9%	65%
POP	64	772M	3.1%	2.4%	22%
coMD	72	5.3M	6.1%	2.4%	60%
coMD	360	28.1M	6.5%	2.8%	58%
Cloverleaf	72	2.7M	5.2%	2.4%	53%
Cloverleaf	360	15.3M	5.6%	3.2%	42%

Use Case 5: Distributed KV Store

41% lower latency

Kalia, A., et al., Usin... services. In ACM SIGCOMM Com...

Use Case 6: Conditional Read

Discarded data: 80%, 60%, 40%, 20% — Speedup vs. Data Size (32B, 4KiB, 512KiB, 64MiB)

Barthels, C., et al., Designing Databases for Future High-Performance Networks. *IEEE Data Eng. Bulletin, 2017*

Use Case 7: Distributed Transactions

Network

Dragojević, A, et al., No compromises: distributed transactions with consistency, availability, and performance. SOSP'15

Bosilca, G., et al., Failure Detection and Propagation in HPC systems. SC'16

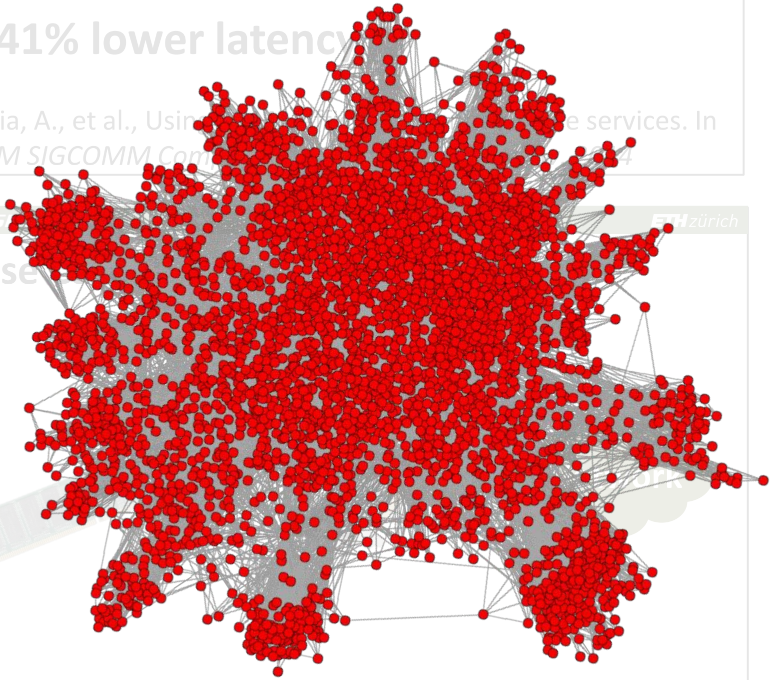
Use Case 9: Distributed Consensus

Consensus

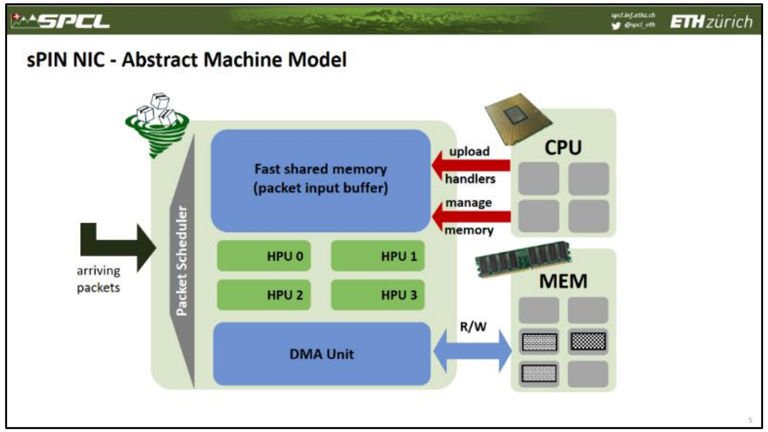
István, Z., et al., Consensus in a Box: Inexpensive Coordination in Hardware. NSDI'16

The Next 700 sPIN use-cases

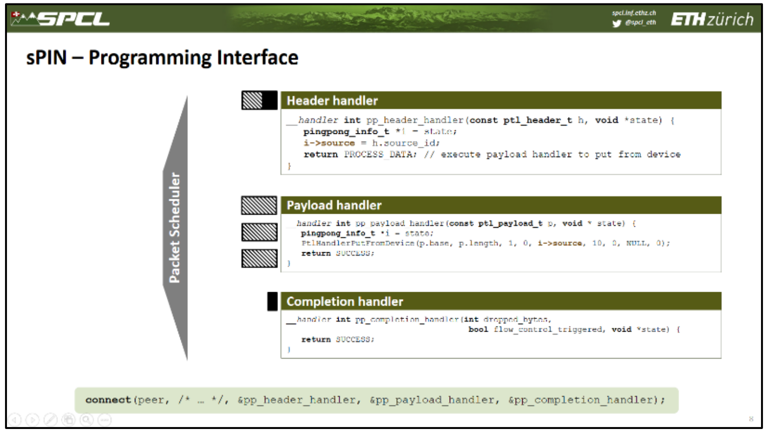
… just think about sPIN graph kernels ….



sPIN *Streaming Processing in the Network* for Network Acceleration

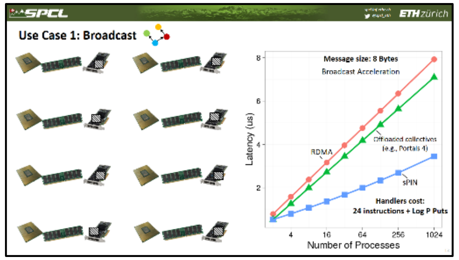


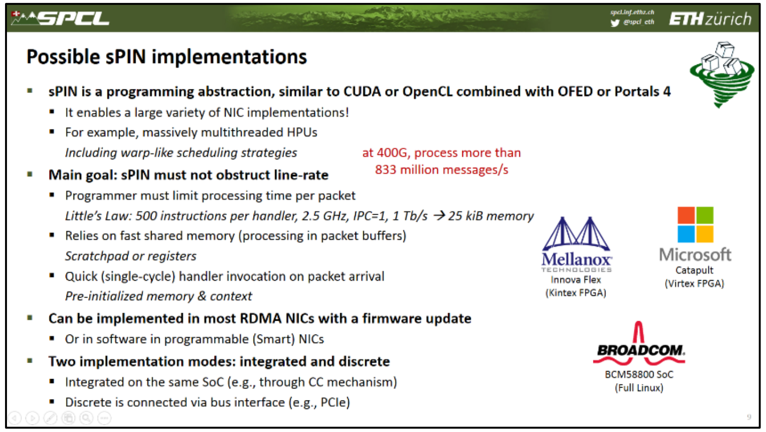
3,800+ reads in
less than 4 hours

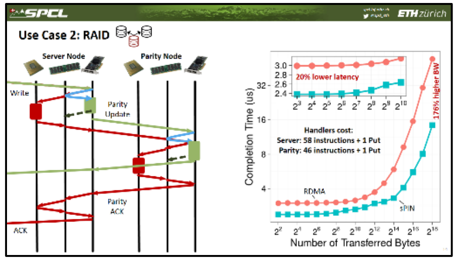


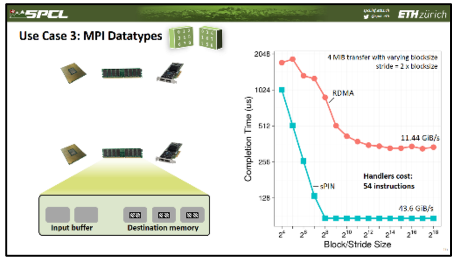
sPIN

beyond RDMA









Full specification: https://arxiv.org/abs/1709.05483

Try it out: https://spcl.inf.ethz.ch/Research/Parallel_Programming/sPIN/

MPI – some new research challenges!

- **More complex network interfaces**
  - sPIN – just discussed in detail
  - How to use it in real applications?

EuroMPI'19
September 11-13 2019
Zurich, Switzerland
https://eurompi19.inf.ethz.ch
Submit papers by April 15th!

- **How to implement sPIN**
  - Collaboration with Luca Benini, RISC-V SoC
  - Full FPGA/simulation environment available as open-source

- **New application use-cases – "Deep Learning is HPC [1]"**
  - Different communication requirements and opportunities for optimization
    *Mainly **sparsity** [2] and **asynchronity**!*
  - Need to be a bit careful though when entering a new field

T. Hoefler: "Twelve ways to fool the masses when reporting performance of deep learning workloads"
(my humorous guide to floptimize deep learning)

[1]: Ben-Nun, Hoefler: "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv:1802.09941
[2]: Renggli et al.: "SparCML: High-Performance Sparse Communication for Machine Learning", arXiv:1802.08021


EuroMPI'19
September 11-13 2019
Zurich, Switzerland
https://eurompi19.inf.ethz.ch
Important dates:
Submission server opens: January 14th, 2019
Full paper submission: April 15th, 2019 (AOE)
Notification: July 1st, 2019
Camera-ready: August 5th, 2019