TORSTEN HOEFLER

Active RDMA - new tricks for an old dog

with M. Besta, R. Belli, S. di Girolamo @ SPCL

presented at Salishan Meeting, Gleneden Beach, OR, USA, April 2016

Systems @ ETH zürich

TORSTEN HOEFLER

Active RDMA - new tricks for an old dog

with M. Besta, R. Belli, S. di Girolamo @ SPCL

presented at Salishan Meeting, Salishan, OR, USA, April 2016

Alternative (better) title: Beyond RDMA

Remote Operations

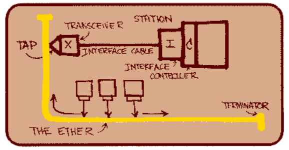
- put, get, atomics

Remote Matching

- partial control at target

Lossy Networks
Ethernet

1980's



Lossless Networks
RDMA

2000's

Full Device Programs
Offload

2020's

Remote Operations

- put, get, atomics

Mellanox TECHNOLOGIES

intel

Remote Matching

- partial control at target

portals

Remote Synchronization

[IPDPS'15]

- Extend RMA semantics
- Fully one-sided (in HW)
- Synchronization

Remote Transactions

[HPDC'15]

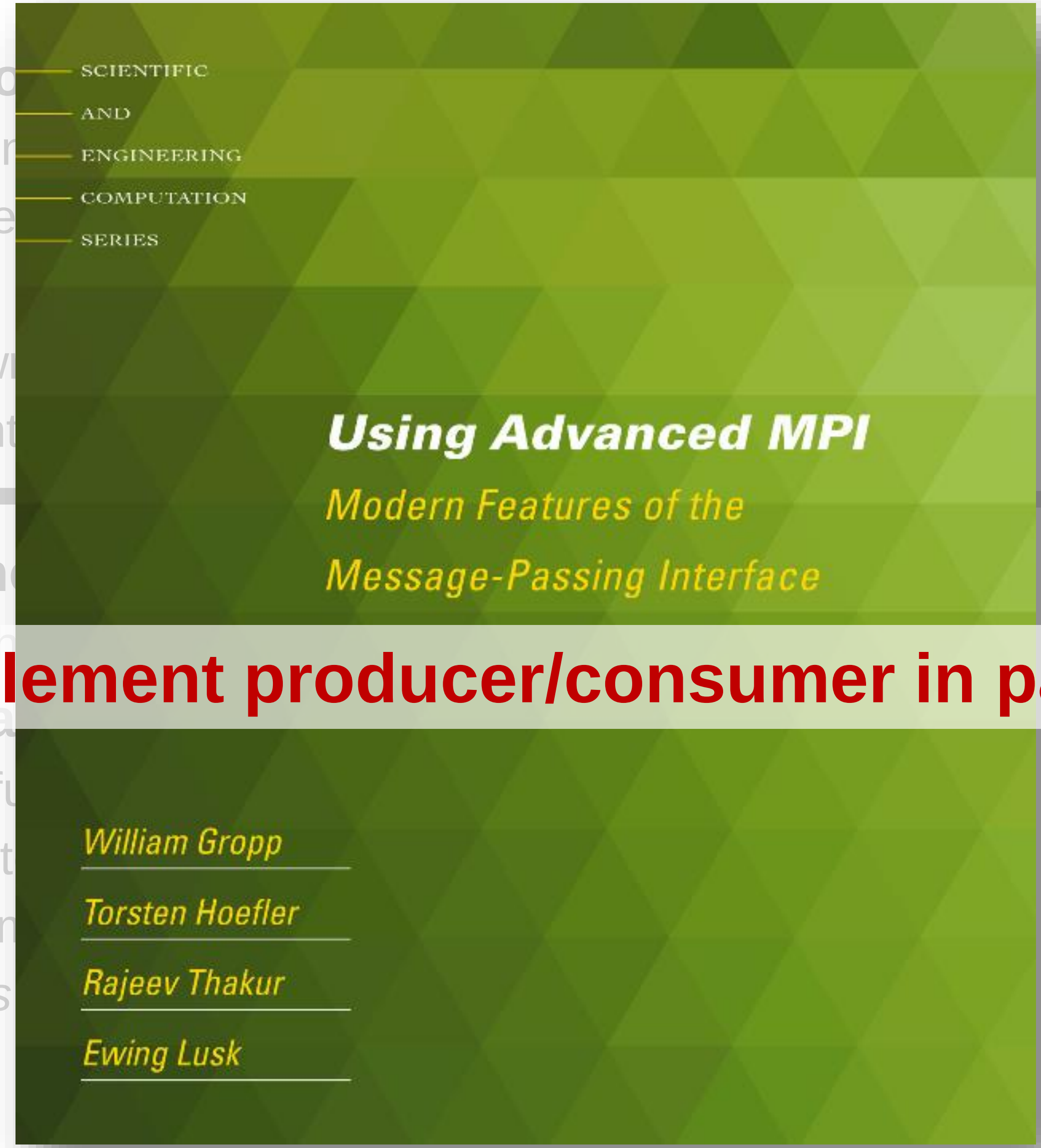
- Similar to HTM
- Extend across nodes
- Think active messages?

Remote Invocation

[ICS'15]

- Utilizes IOMMUs
- Control transfer
- Active memory

IN CASE YOU WANT TO LEARN MORE ABOUT RMA



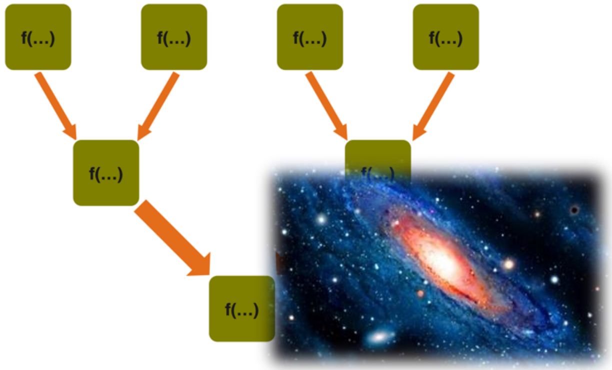
How to implement producer/consumer in passive mode?

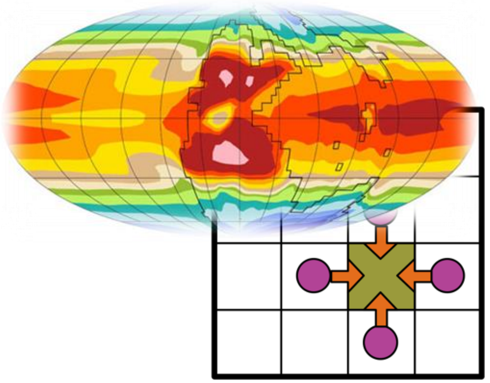
RDMA

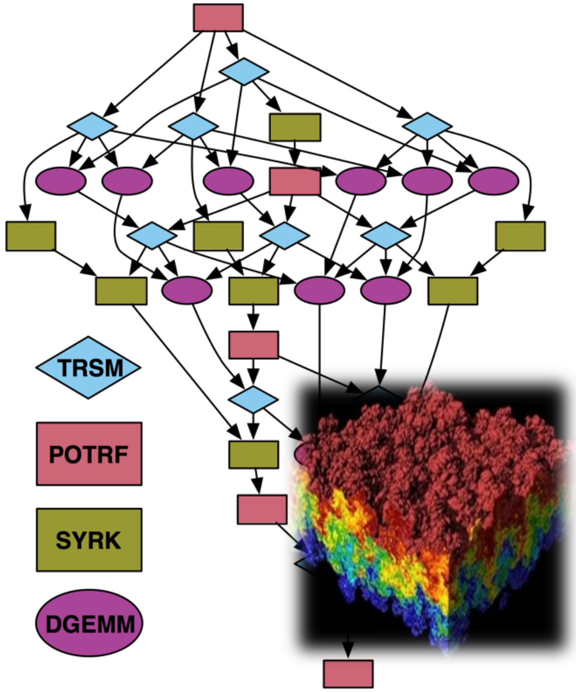
- PGAS and RMA are pro
  - PGAS as language exte
  - RMA as library (integrate
- Offer abstraction for
  - Data placement, read, w
  - Target has very little cont
- RDMA is a hardware m
  - Often accessible through
- Specific to a (set of) ha
  - Offers varying levels of fu
  - Most common: read, writ
  - Address-space managem
  - *Common denominator is*

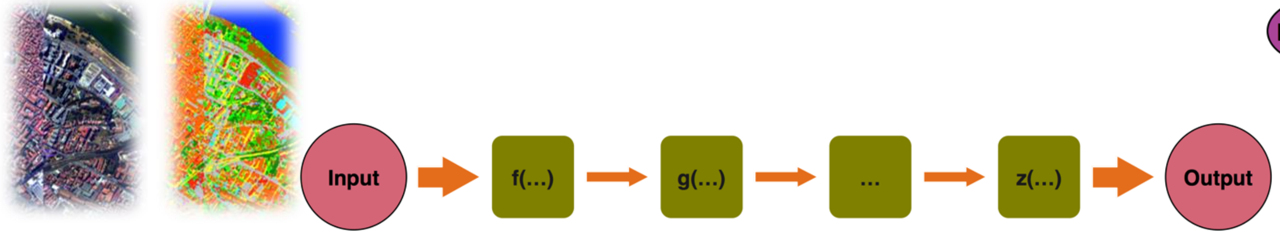
Producer-Consumer Relations

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







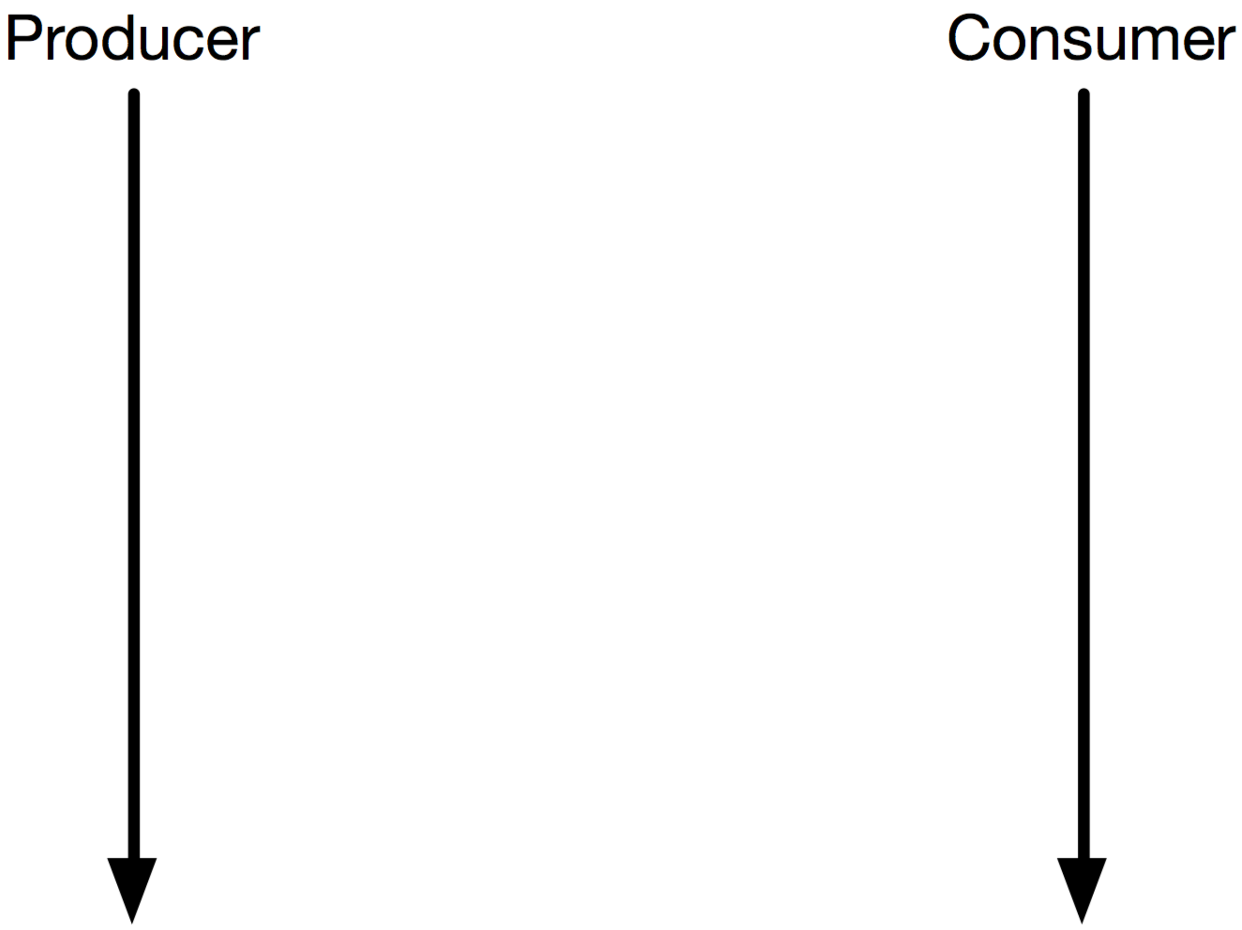


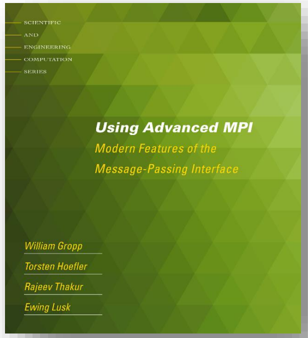
- **Perfectly supported by MPI-1 Message Passing**
  - But how does this actually work over RDMA?

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

Remote Synchronization

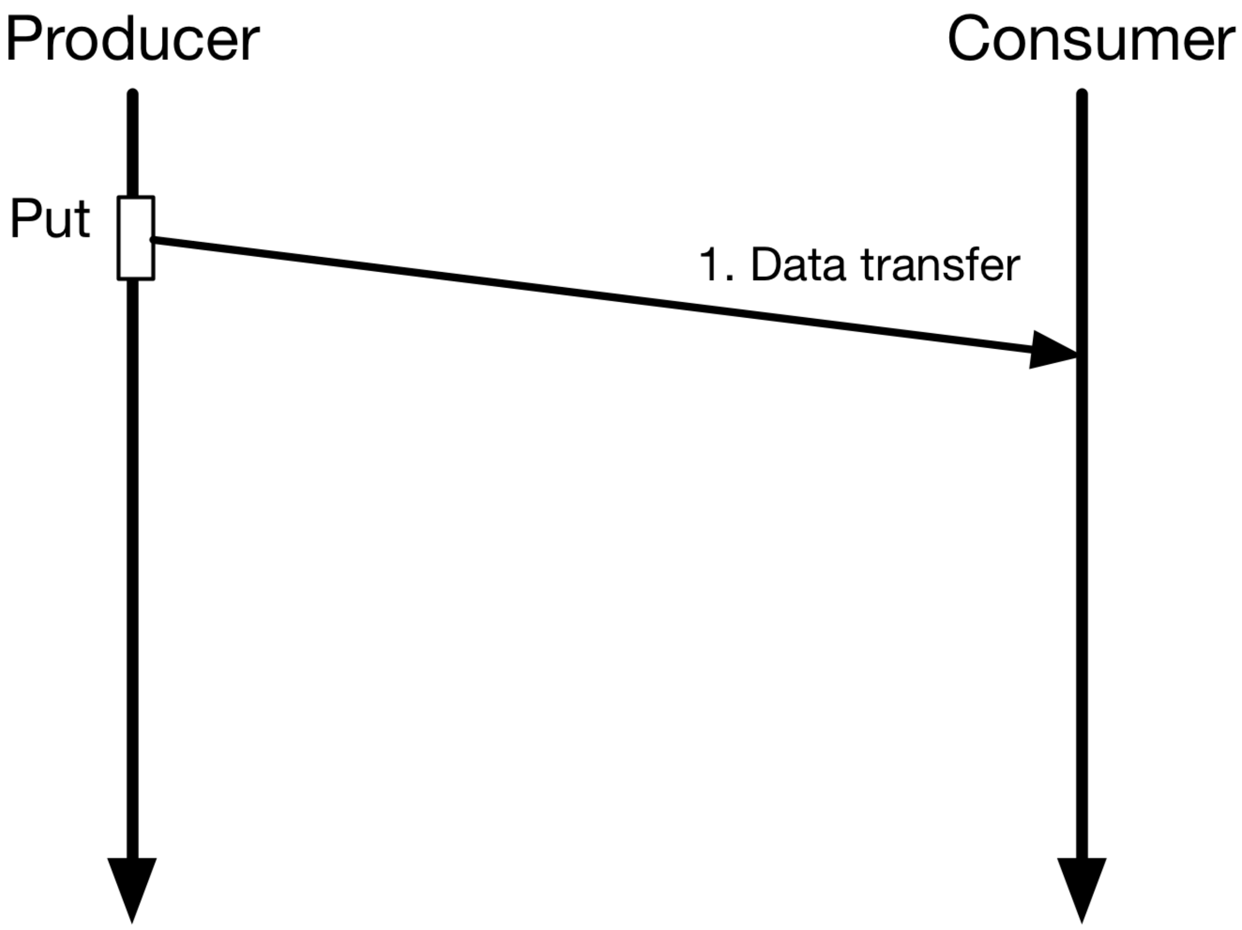
ONE SIDED – PUT + SYNCHRONIZATION

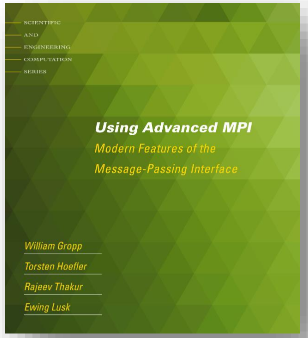




Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

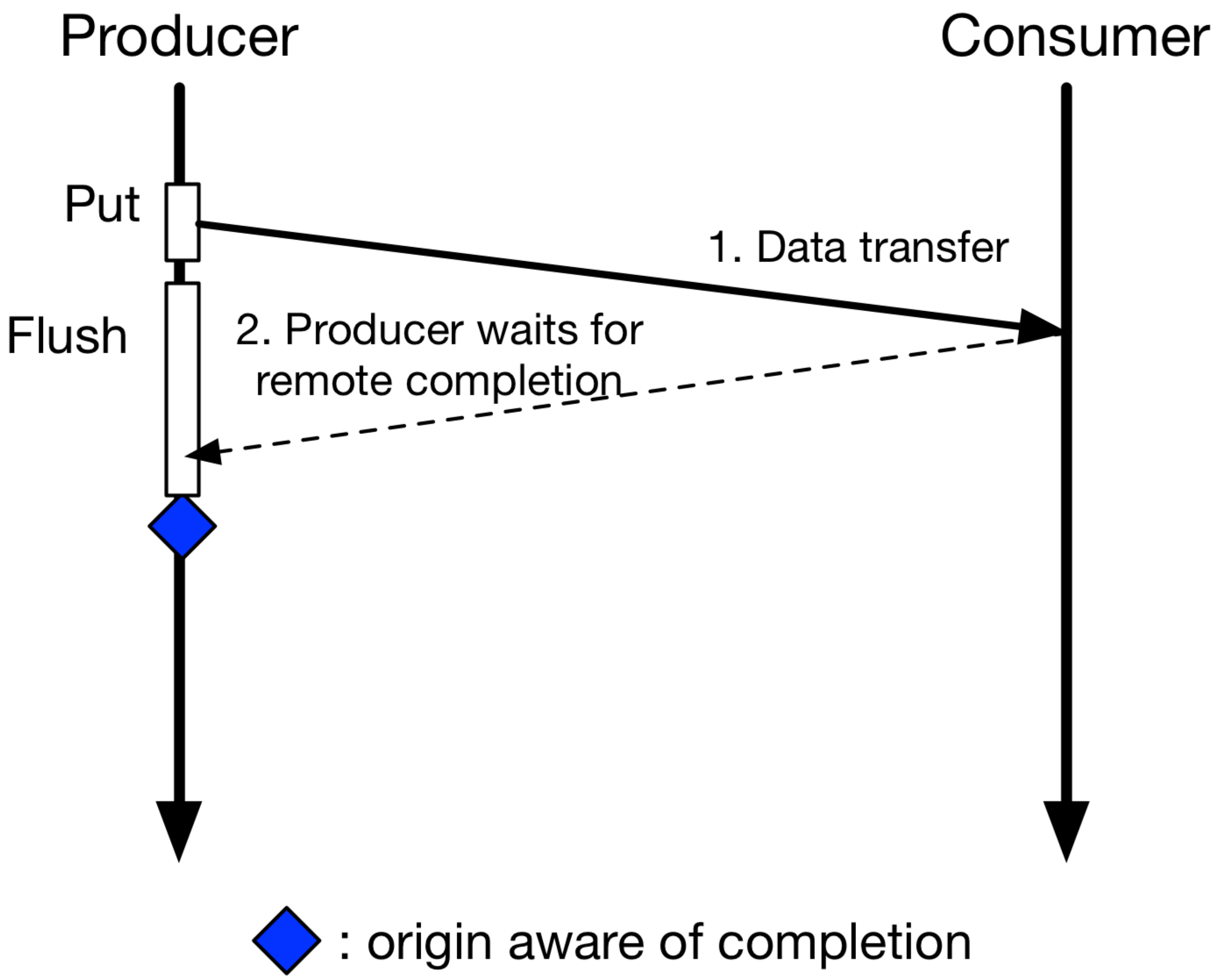
ONE SIDED – PUT + SYNCHRONIZATION

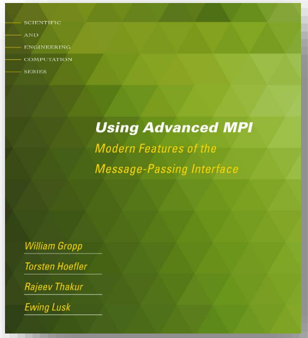




Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

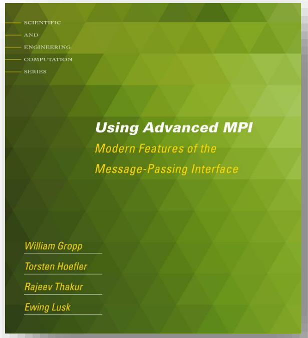
One Sided – Put + Synchronization

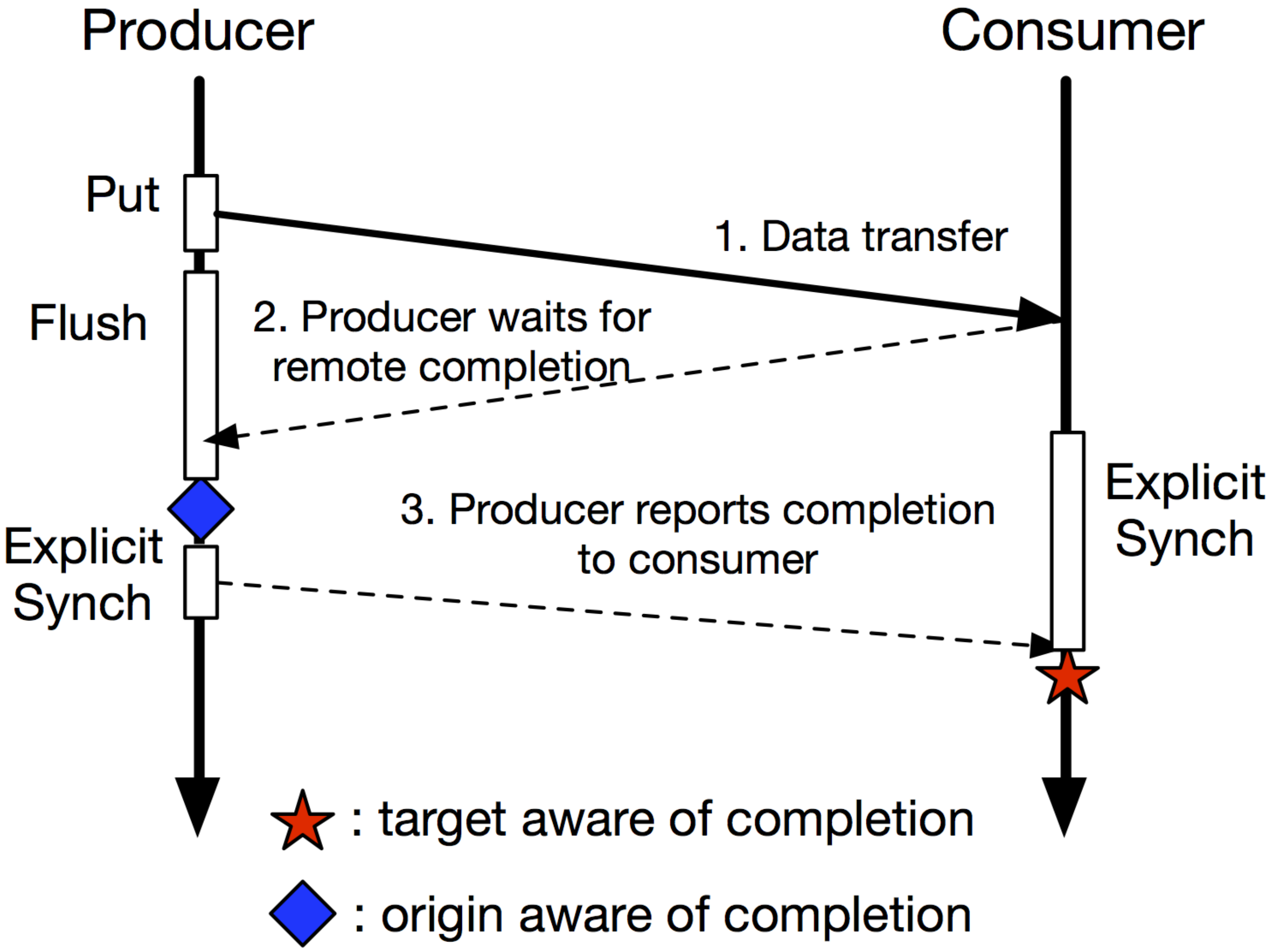




Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

ONE SIDED – PUT + SYNCHRONIZATION

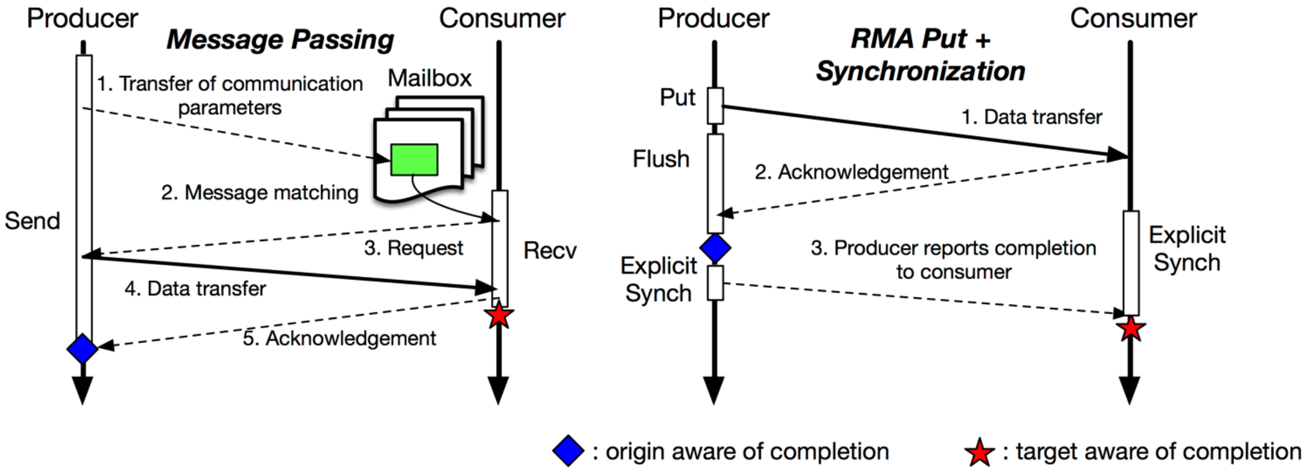




Critical path: 3 latencies

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

COMPARING APPROACHES



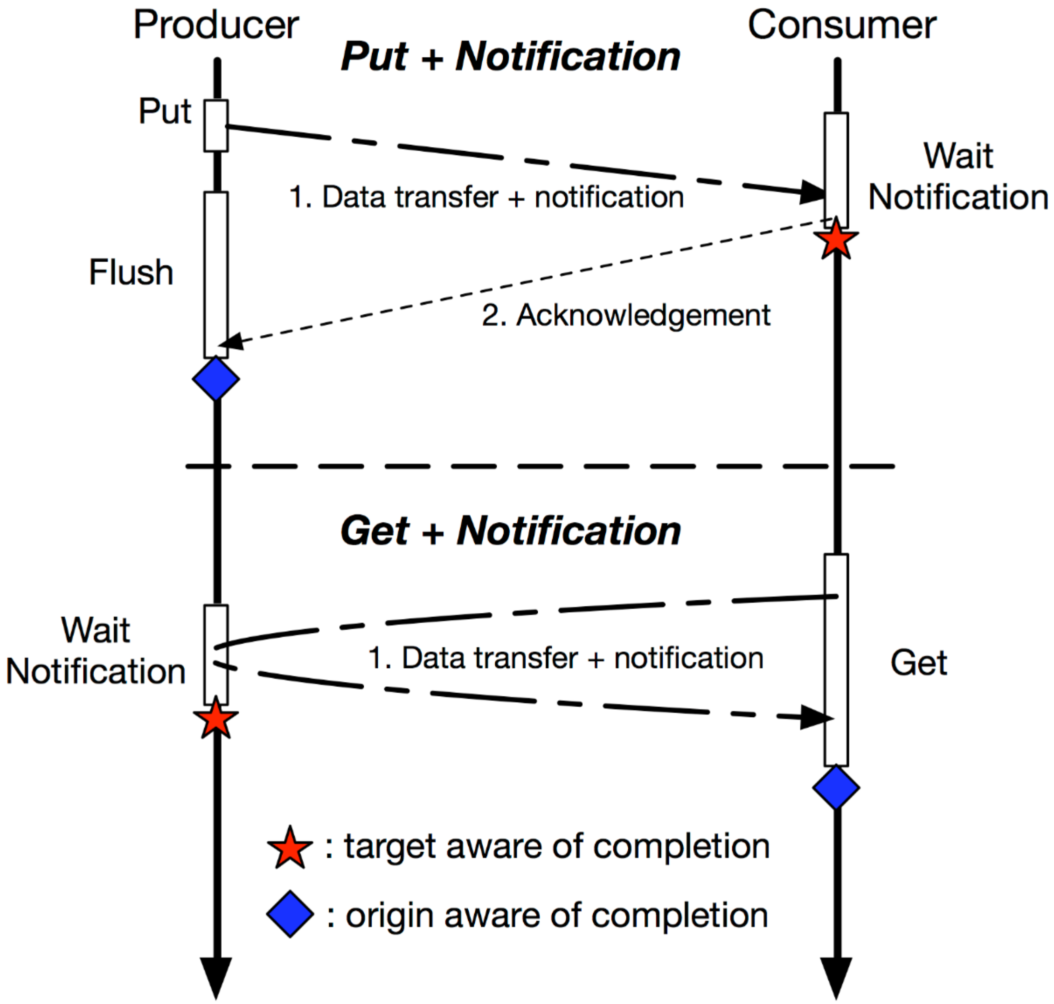
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

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

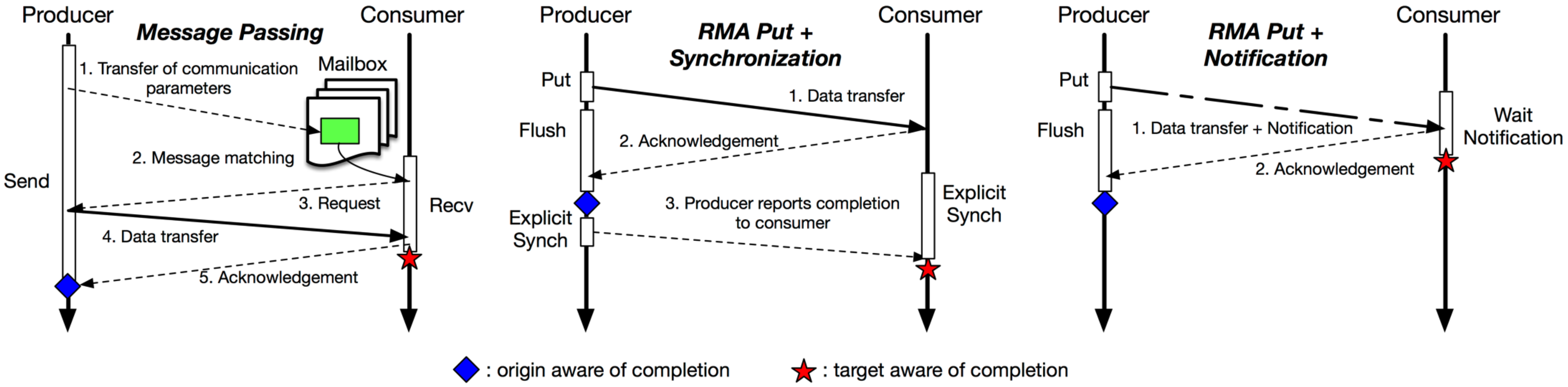
Idea: RMA Notifications

- **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

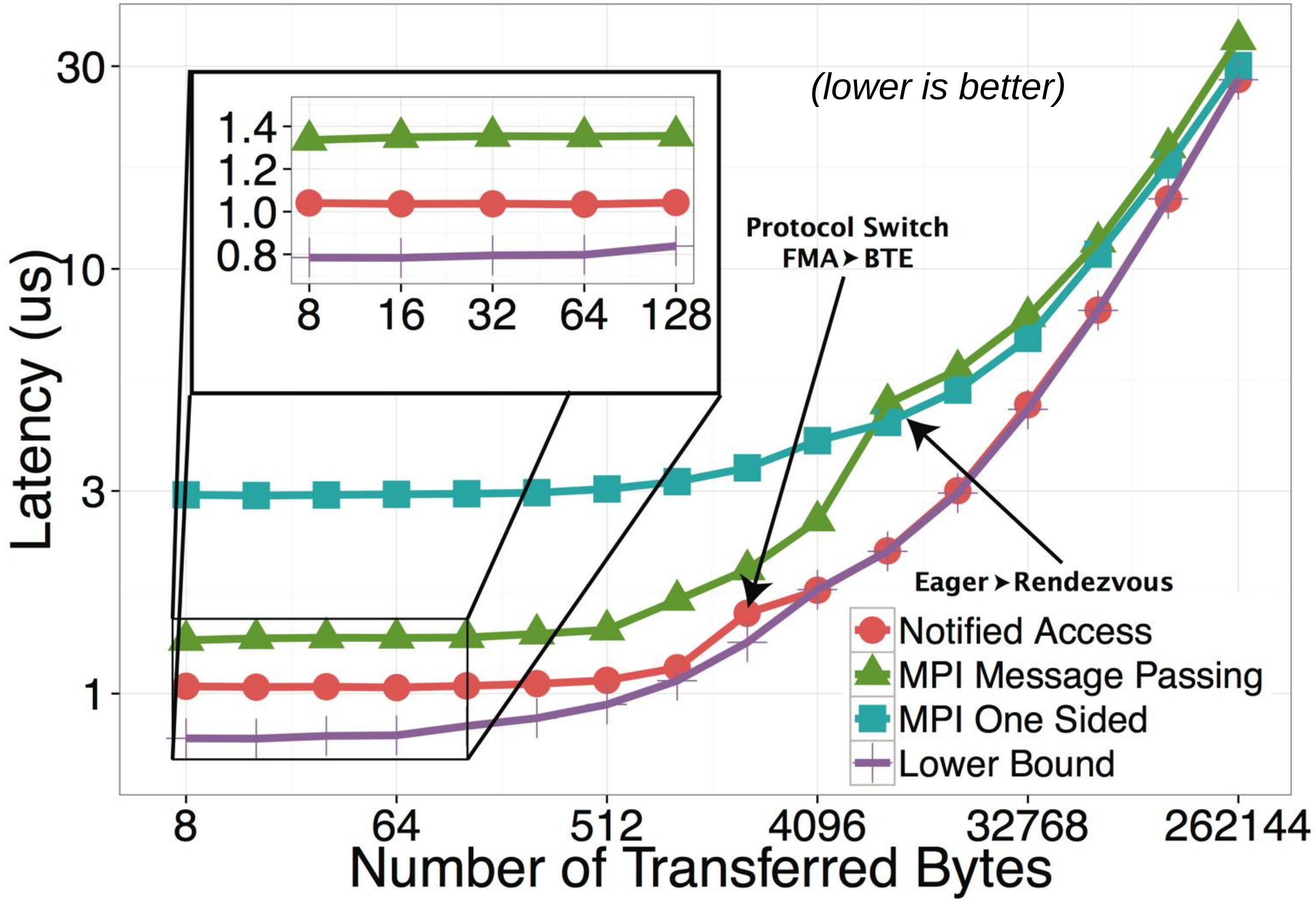
One Sided
3 latencies

Notified Access
1 latency

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

PING PONG PERFORMANCE (INTER-NODE)

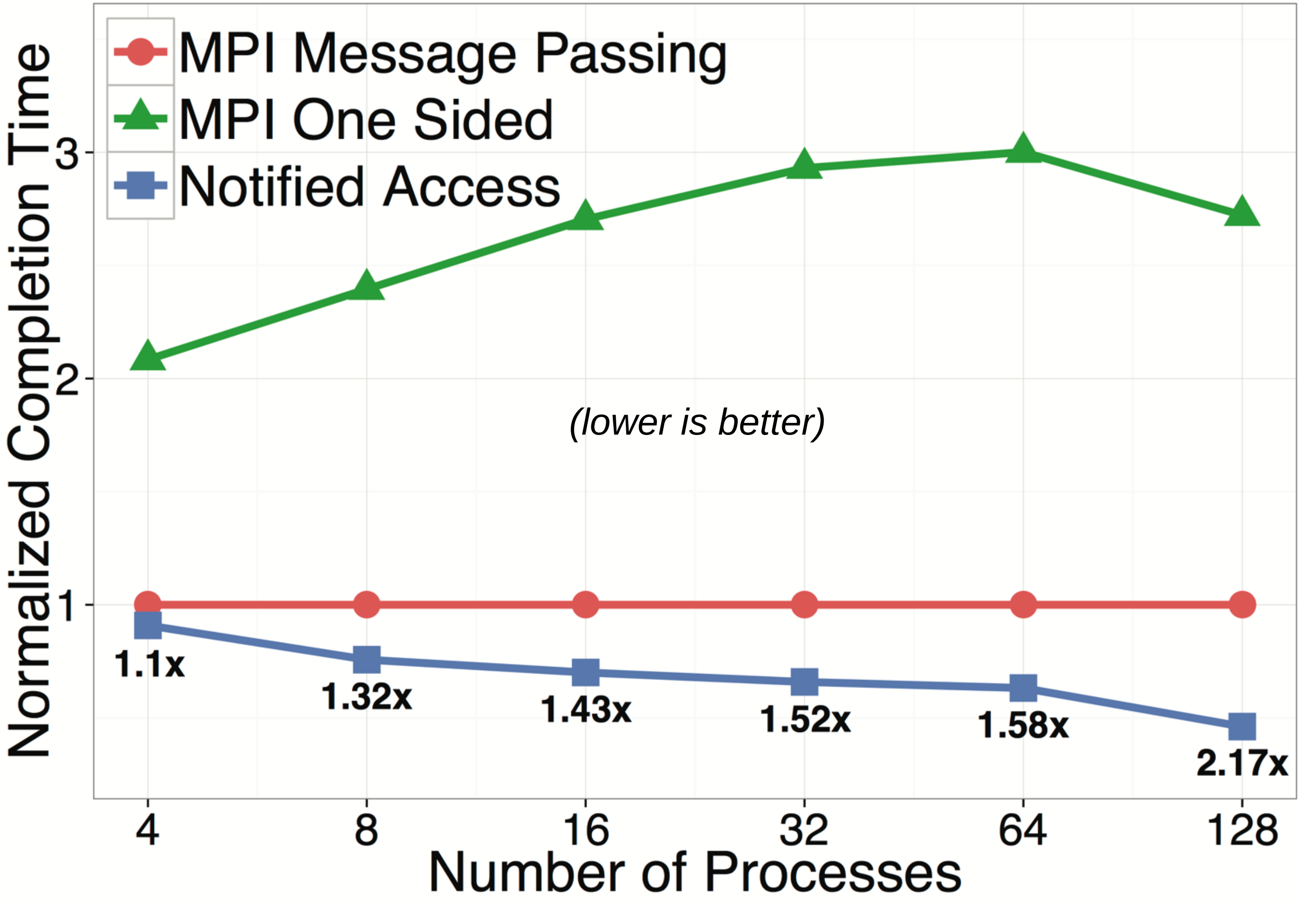
- 1000 repetitions, each timed separately, RDTSC timer
- 95% confidence interval always within 1% of median

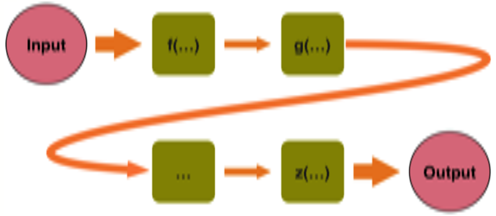


Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Pipeline – One-to-One Synchronization

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

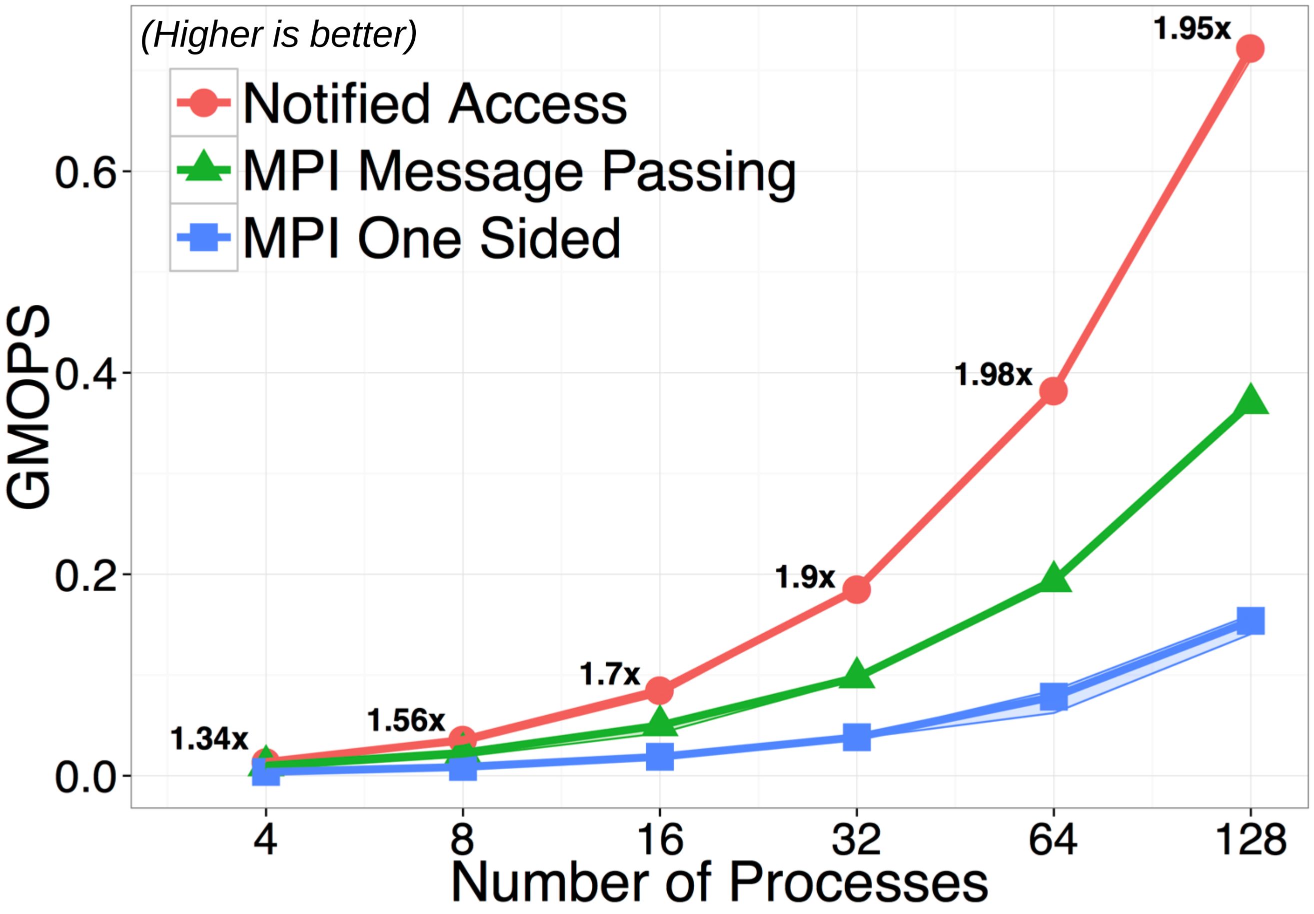


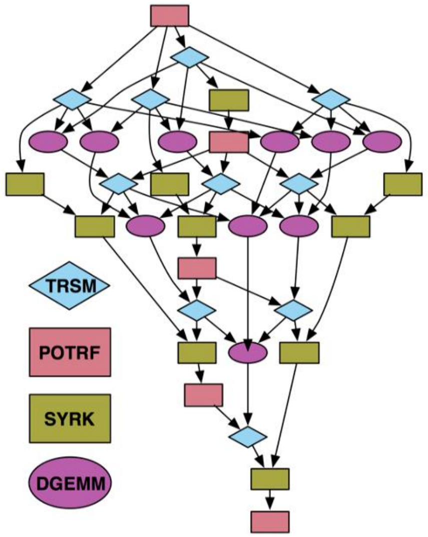


[1] https://github.com/intelesg/PRK2

Cholesky – Many-to-Many Synchronization

- **1000 repetitions, each timed separately, RDTSC timer**
- **95% confidence interval always within 10% of median**



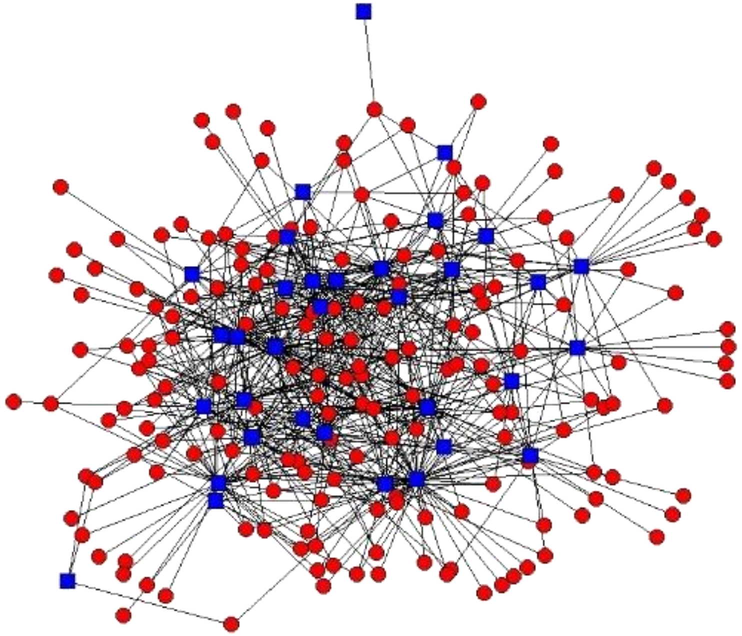


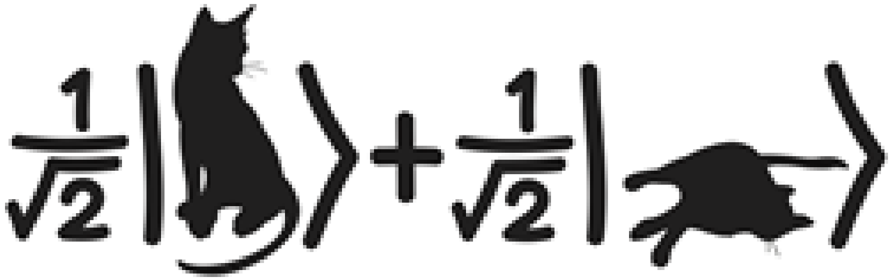
[1]: J. Kurzak, H. Ltaief, J. Dongarra, R. Badia: "Scheduling dense linear algebra operations on multicore processors", CCPE 2010

(Remote) Transactions

LARGE-SCALE IRREGULAR GRAPH PROCESSING

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





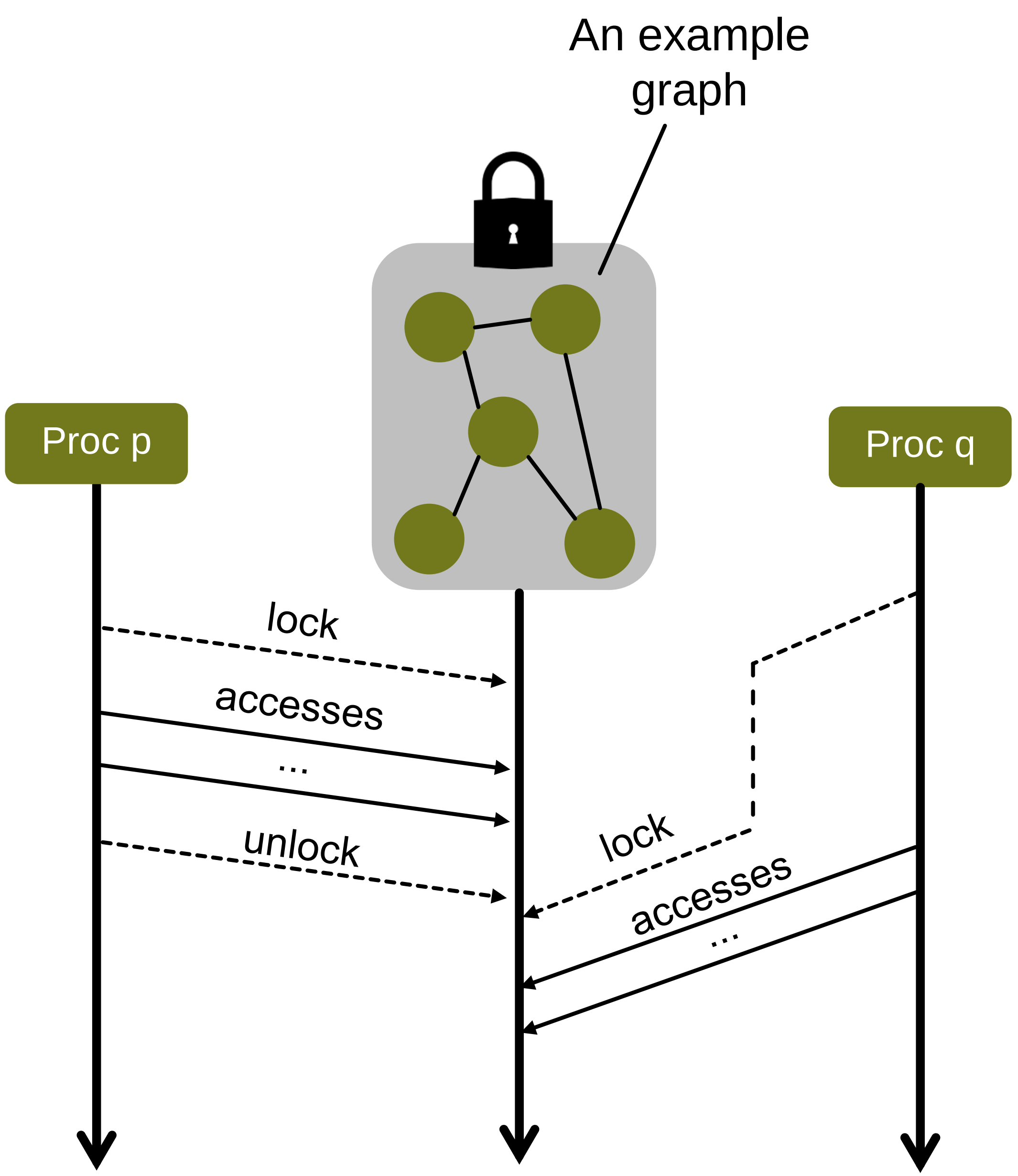
[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Letters, 2007

SYNCHRONIZATION MECHANISMS
COARSE LOCKS

Simple protocols

Serialization

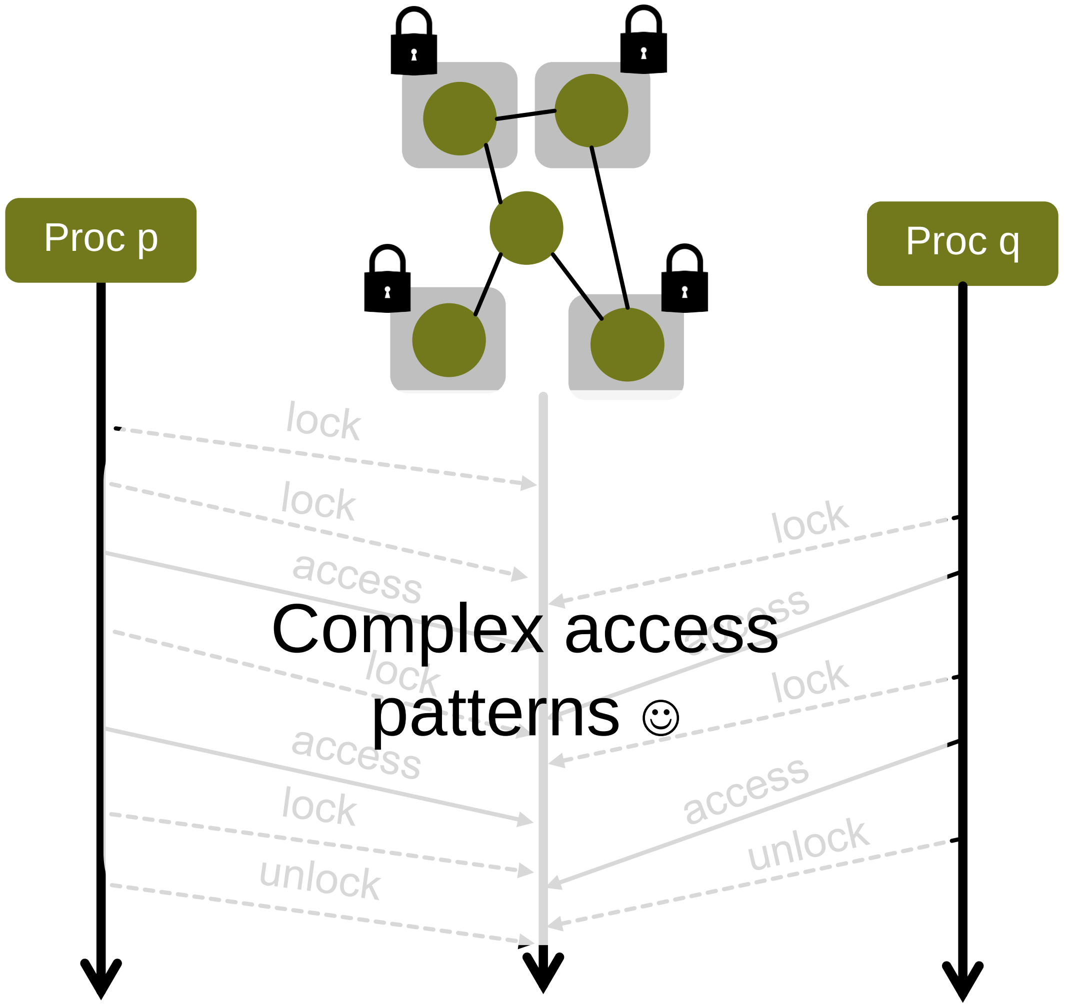
Detrimental performance



M. Kulkarni et al., Optimistic Parallelism Benefits from Data Partitioning, ASPLOS'08

Synchronization Mechanisms
Fine locks

- Higher performance possible
- Complex protocols
- Risk of deadlocks



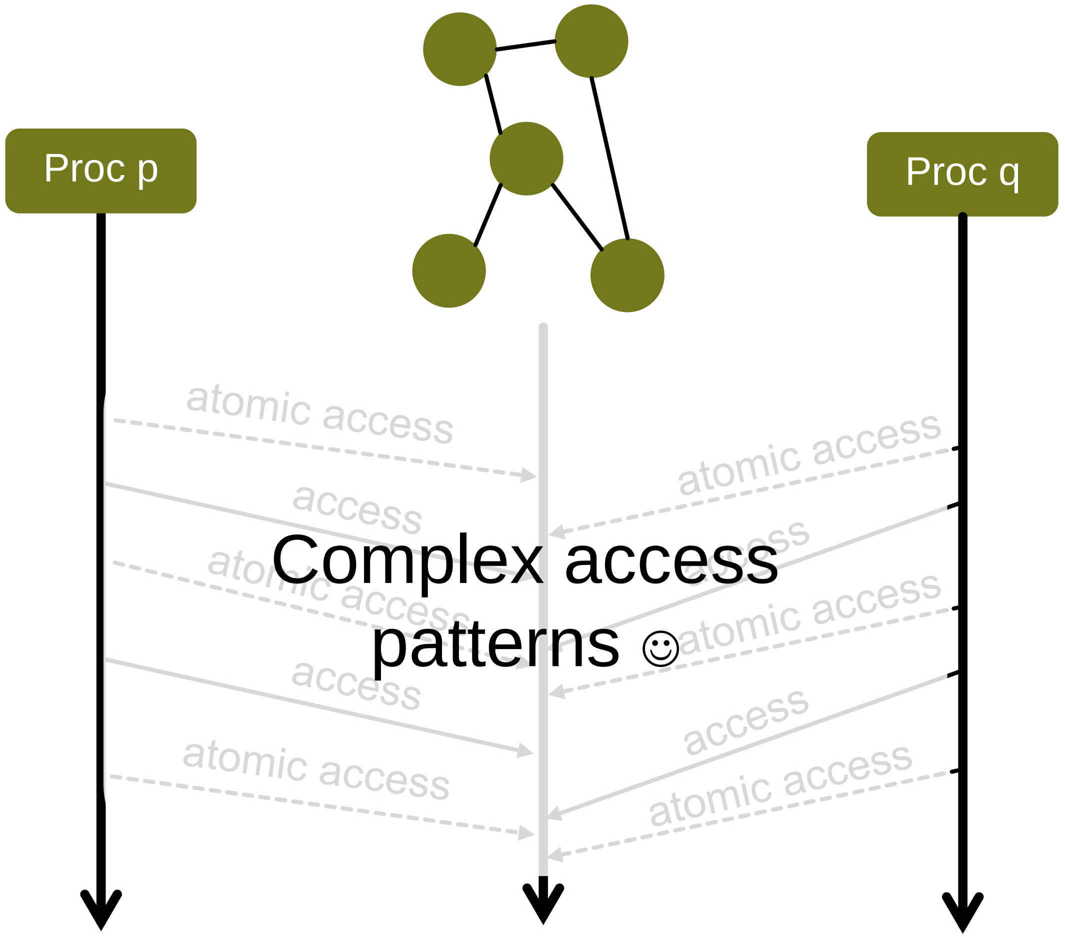
J. Yan et al., Exploiting fine-grained parallelism in graph traversal algorithms via lock virtualization on multi-core architecture, Journ. of Supercomp.

Synchronization Mechanisms
Atomic Operations

High performance (may be challenging to get)

Complex protocols

Subtle issues (ABA, ...)



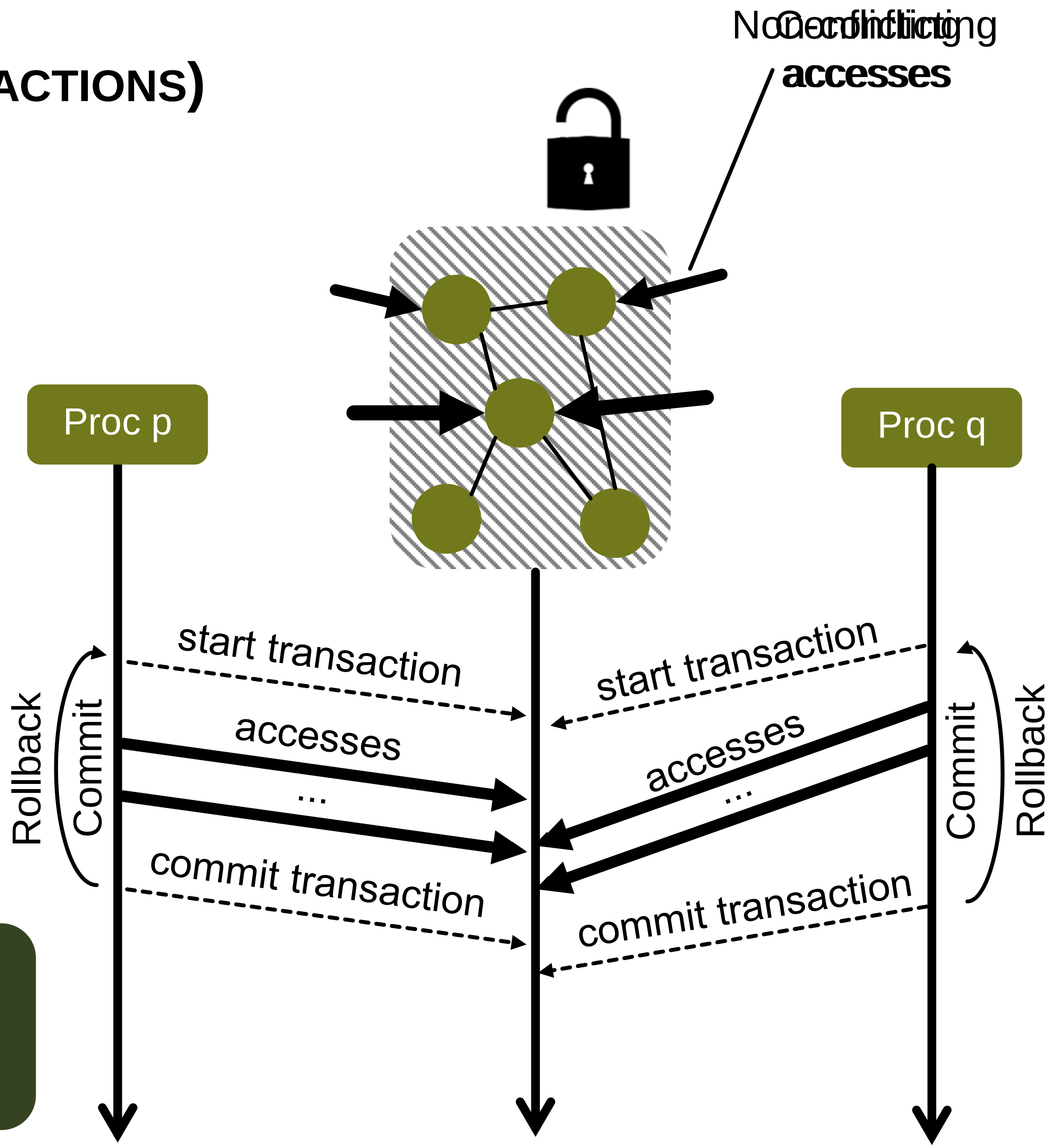
V. Agarwal et al., Scalable Graph Exploration on Multicore Processors, IEEE/ACM Supercomputing 2010 (SC10)

Synchronization Mechanisms
Transactional Memory (cf. DB transactions)

Conflicts solved with rollbacks and/or serialization.

Software overheads

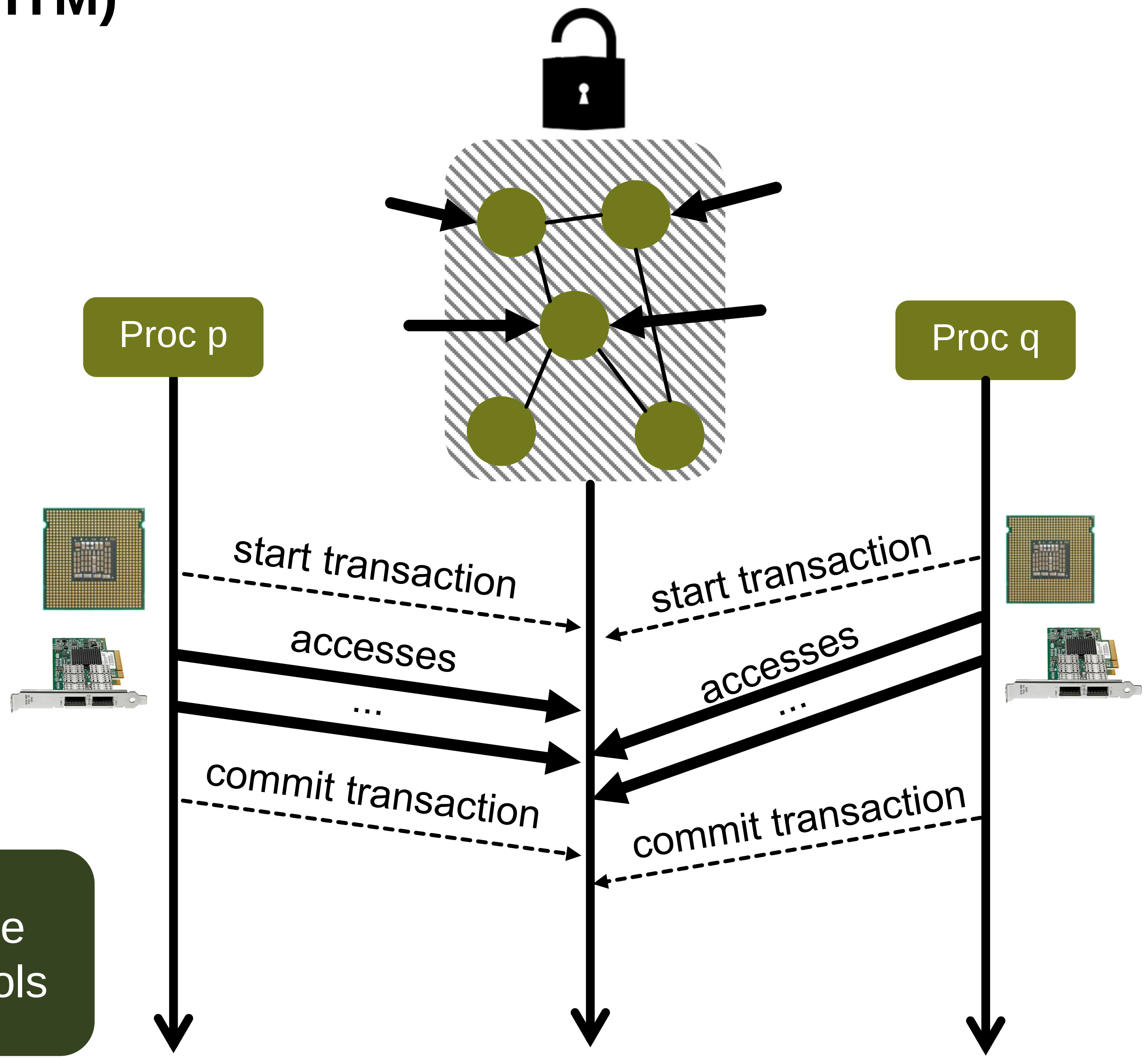
Simple protocols



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

Synchronization Mechanisms
Hardware transactional memory (HTM)

Conflicts solved with rollbacks and/or HW serialization.



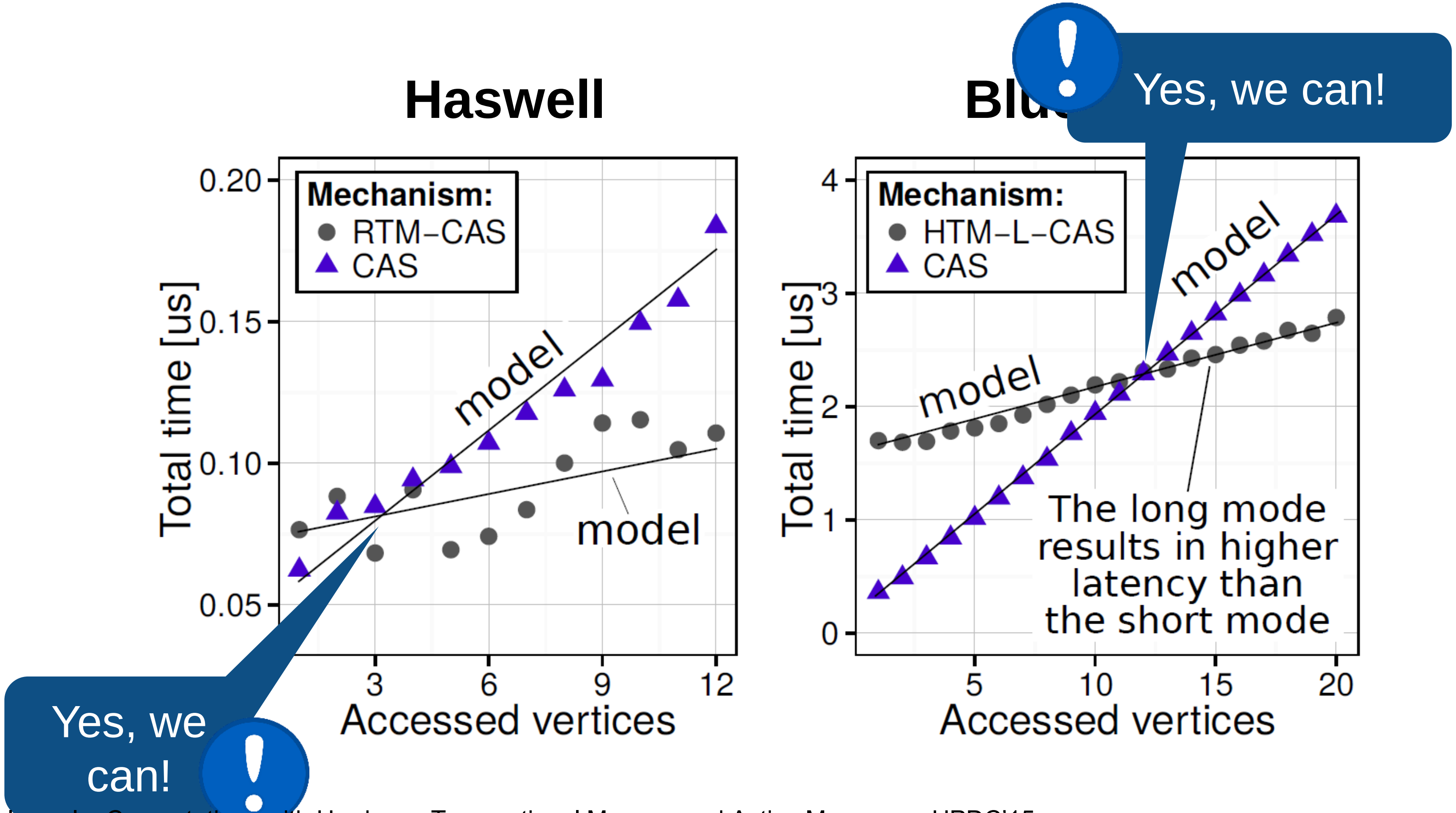
High performance? For graphs?

Simple protocols

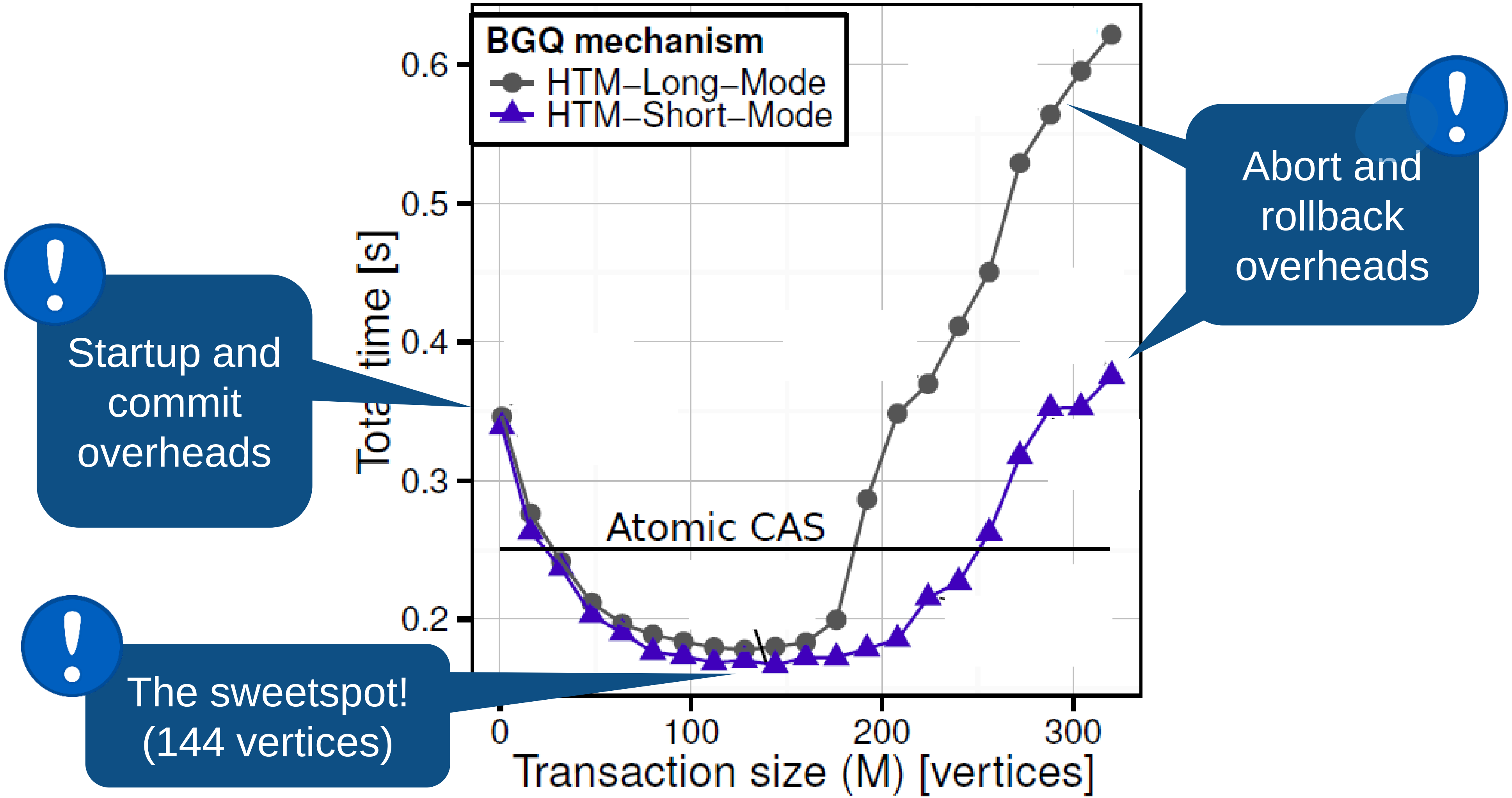
Besta, Hoefler: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, HPDC'15

Performance Model
Atomics vs transactions

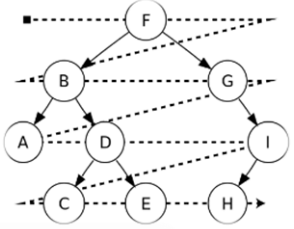
- Can we amortize HTM startup/commit overheads with larger transaction sizes?



MULTI-VERTEX TRANSACTIONS IN A BFS (GRAPH 500)
MARKING VERTICES AS VISITED



REAL-GRAPH PERFORMANCE

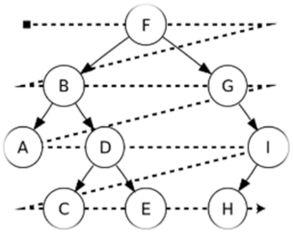


	Input graph properties				BG/Q analysis			Haswell analysis					
Type	ID	Name	$\|V\|$	$\|E\|$	S over g500 ($M = 24$)	M	S over g500	S over g500 ($M = 2$)	S over Galois ($M = 2$)	M	S over g500	S over Galois	S over HAMA
Comm. networks (CNs)	cWT	wiki-Talk	2.4M	5M	2.82	48	3.35	0.91	1.22	6	0.96	1.28	344
	cEU	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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	[illegible]	0.74	$> 10^4$
	sLJ	com-lj	4M	34M	[illegible]	12	[illegible]	[illegible]	[illegible]	[illegible]	1.04	1.04	603
	sYT	com-youtube	1.1M	2.9M	1.07	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	[illegible]	925k	[illegible]	8	1.62	1.04	1.64	2	1.04	1.64	1426
Purchase network (PNs)	pAM	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	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

☺ No, you don't have to read it.

☺ Here: just a summary.

REAL-GRAPH PERFORMANCE



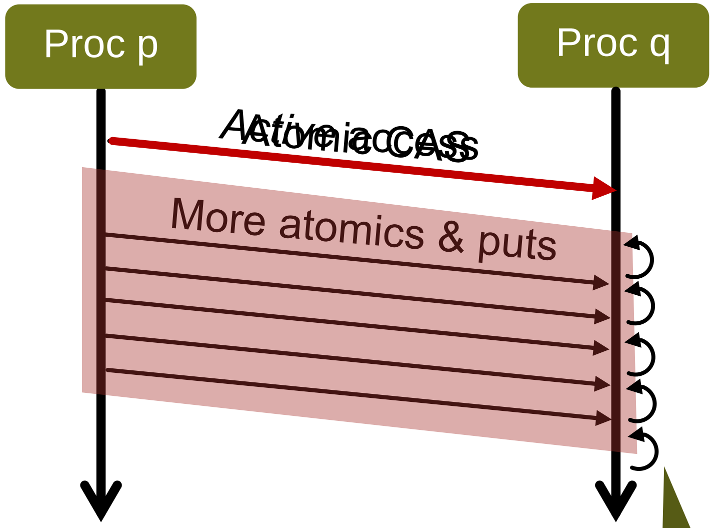
Average overall speedup (geomean) over
Graph 500: 1.07,
Galois [1]: 1.40, HAMA: ~1000

1.85x on average, up to 4.3x

[1]: Satish et al.: Navigating the Maze of Graph Analytics Frameworks Using Massive Graph Datasets, SIGMOD'14

Remote Invocation

IMAGINE A SIMPLE DISTRIBUTED HASH-TABLE



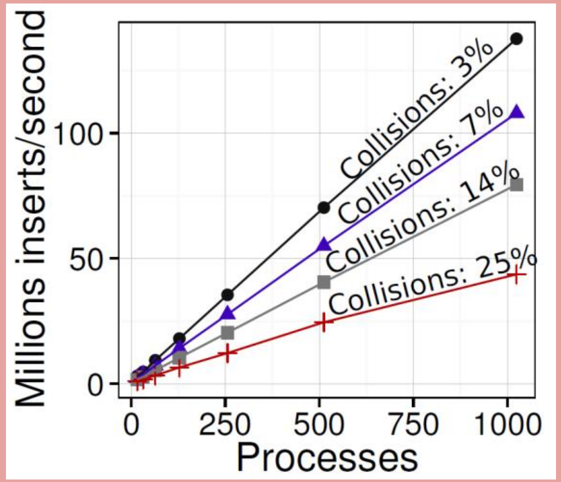
Local execution; triggered by an active *access*. In RMA?

No collision:

➔ 1 remote atomic
➔ Up to 5x speedup over MP [1]

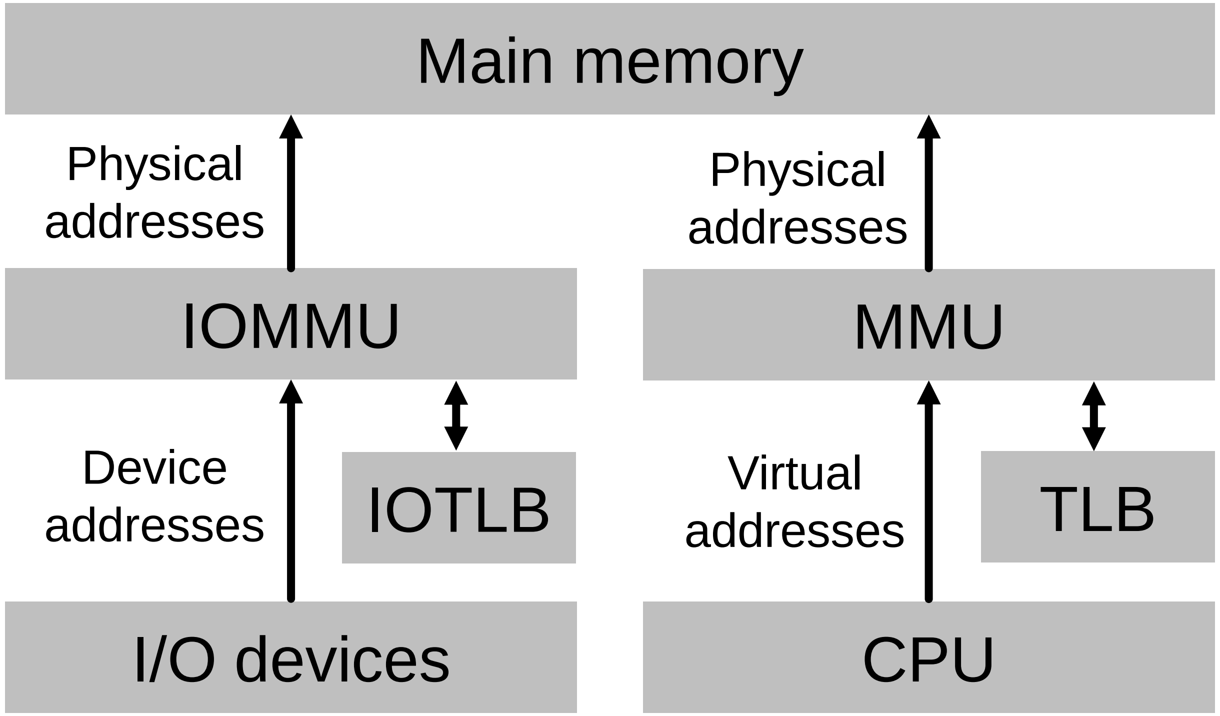
A collision:

➔ 4 remote atomics + 2 remote puts
➔ Significant performance drops



[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided, SC13

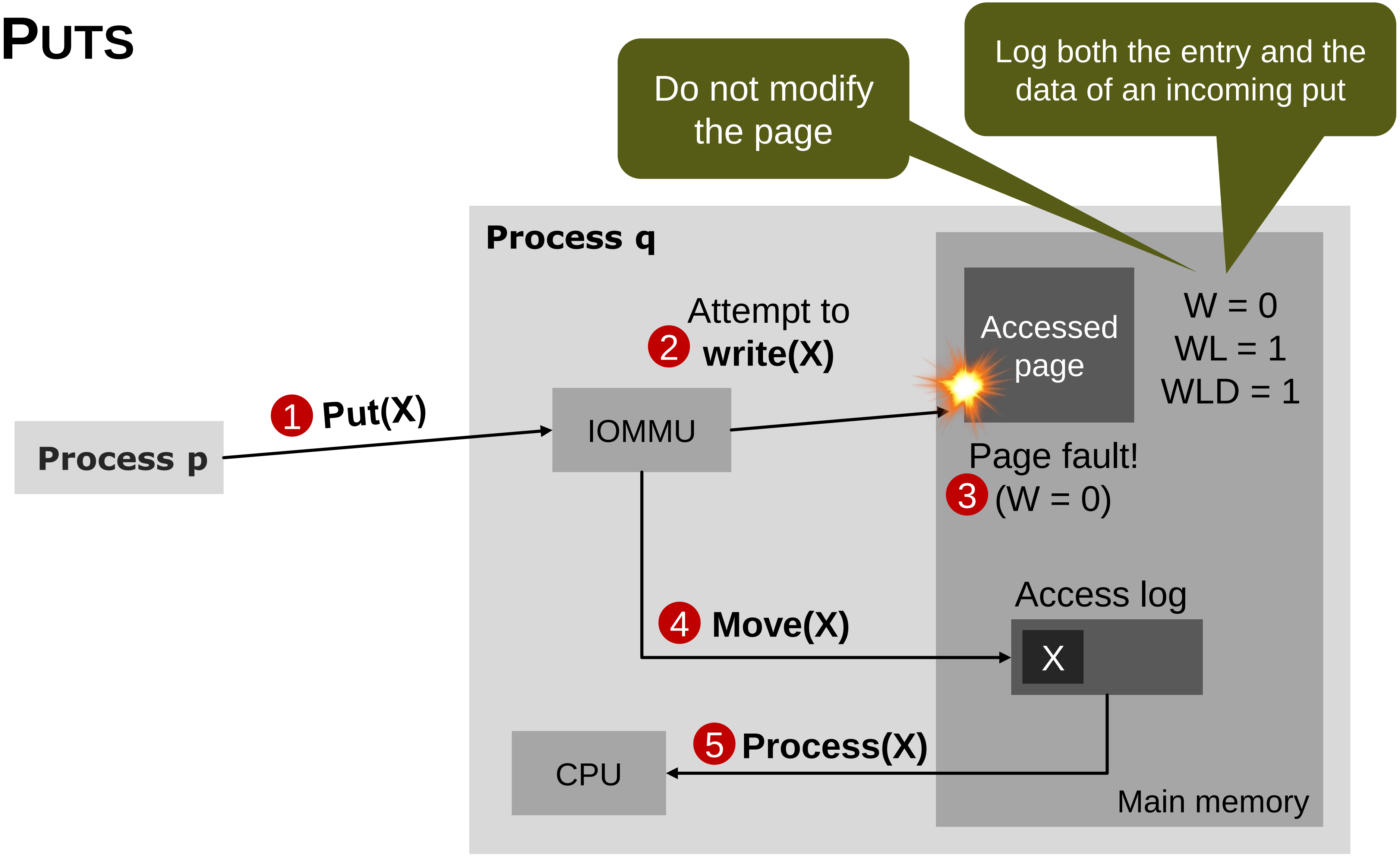
Use Input/Output Memory Management Units



AMD
IBM
ARM
Sun microsystems
SOLARFLARE
intel
+ PCI EXPRESS

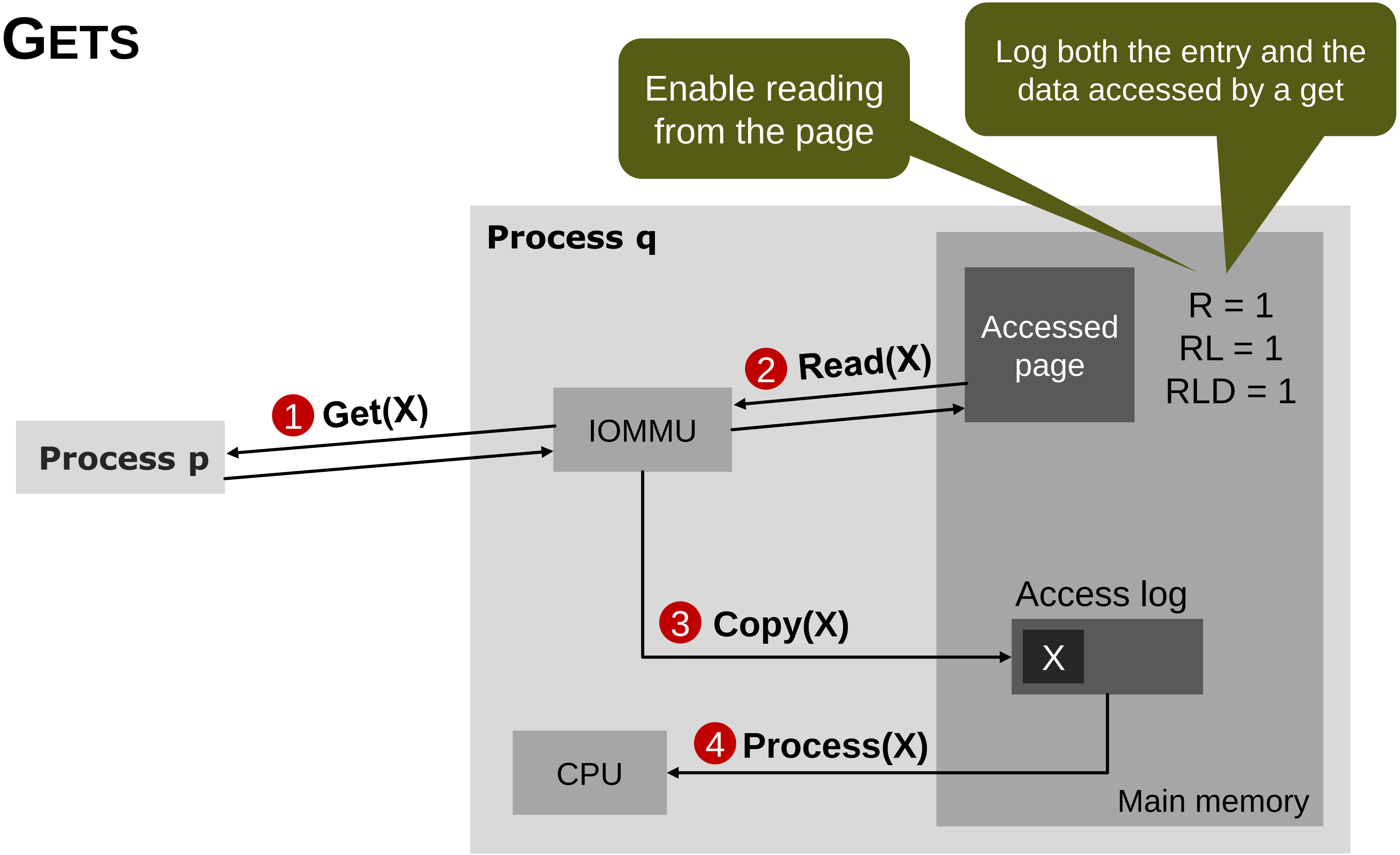
M. Besta and T. Hoefler, Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations, ICS'15

Active Puts



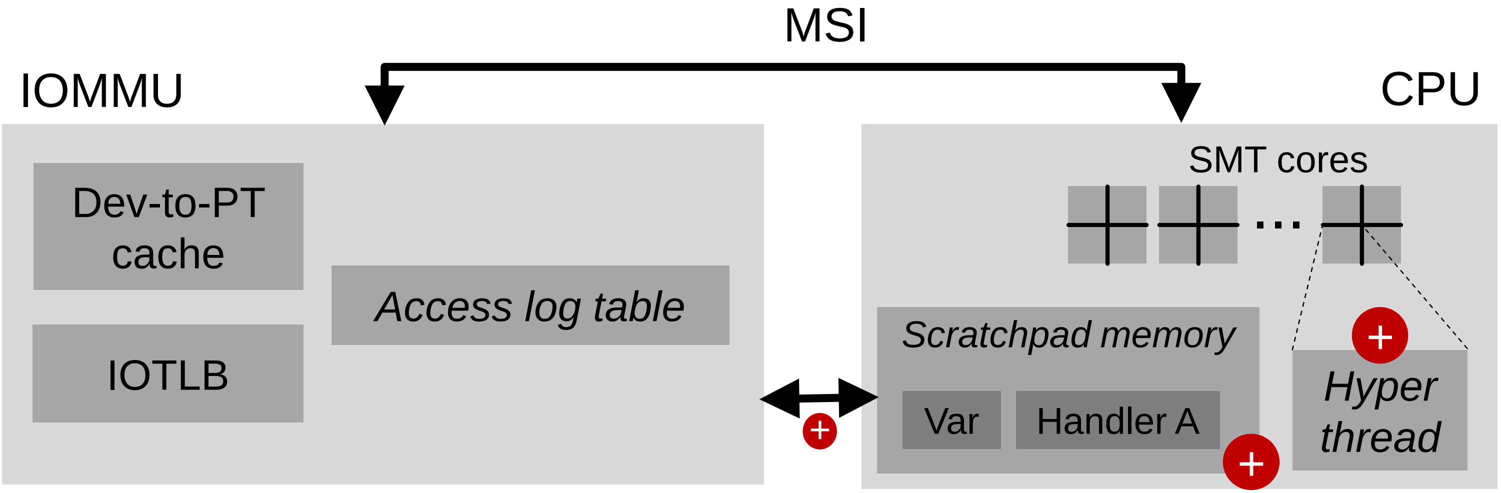
M. Besta and T. Hoefler, Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations, ICS'15

Active Gets



M. Besta and T. Hoefler, Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations, ICS'15

INTERACTIONS WITH THE CPU

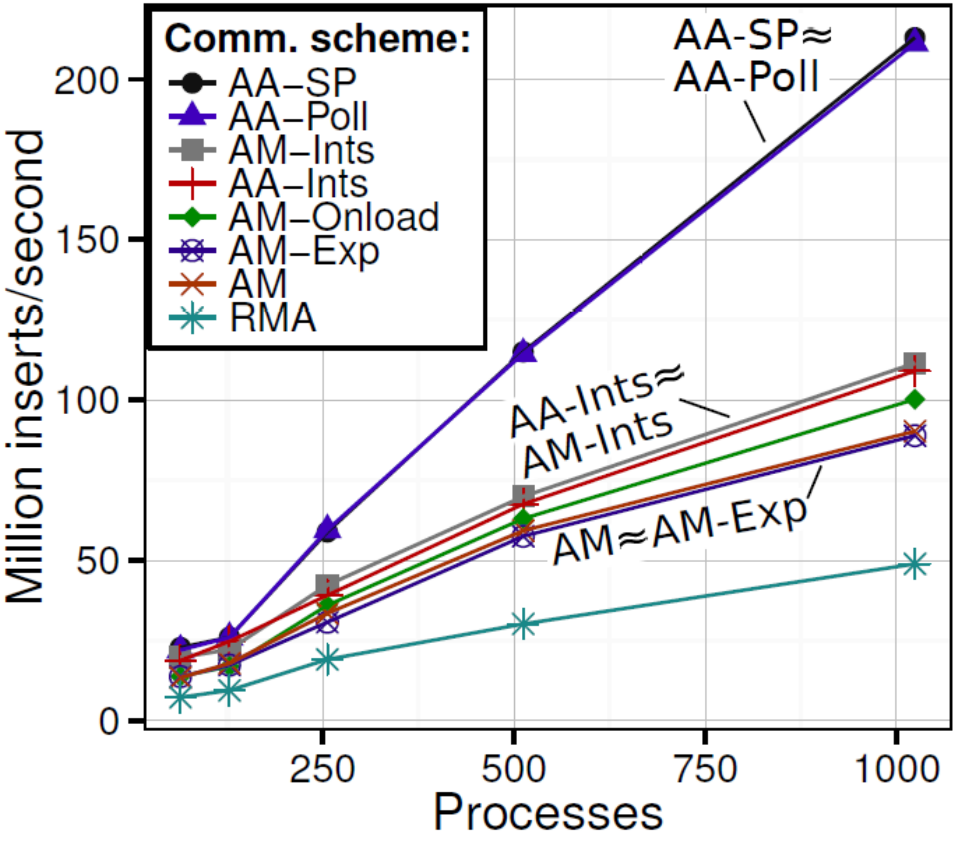


- Interrupts
- Polling
- Direct notifications via scratchpads

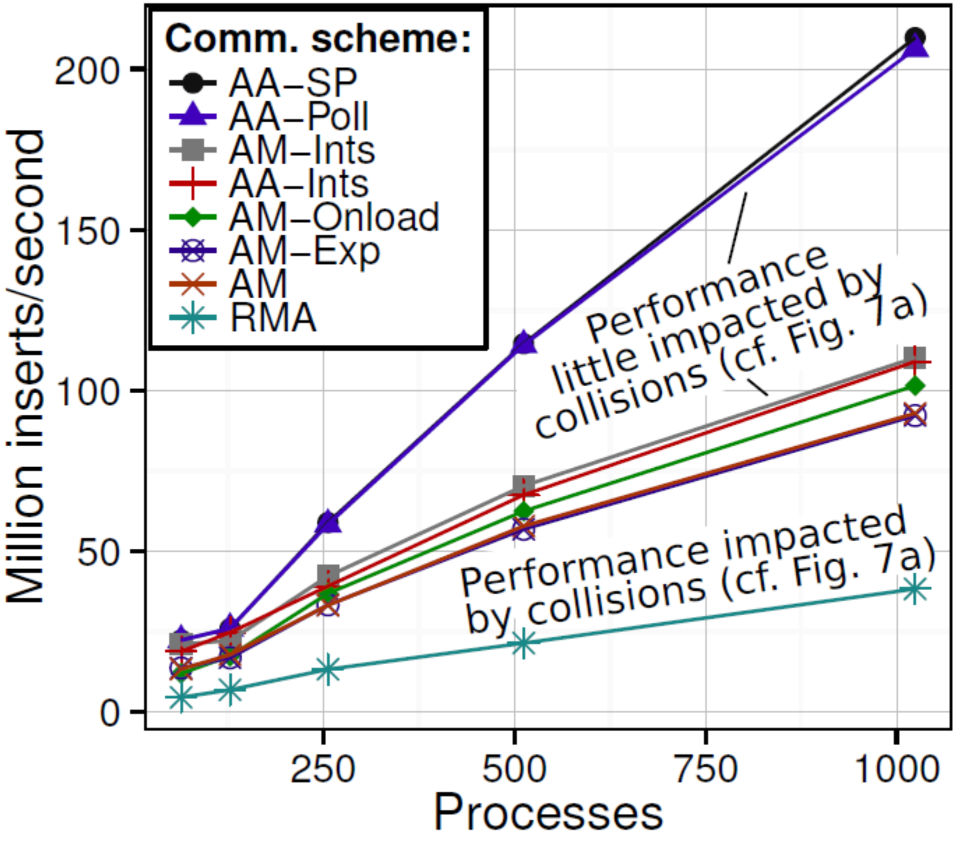
Performance: Large-Scale Codes

Distributed Hashtable

Collisions: 5%



Collisions: 25%

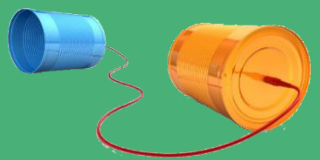


M. Besta and T. Hoefler, Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations, ICS'15

Towards a Network Instruction Set Architecture (NISA)

An example for offloading

Communications
(non-blocking)



Computations

Dependencies

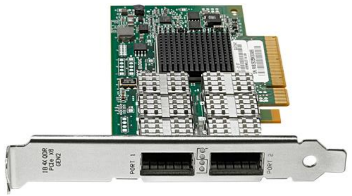
recv → comp → send

EXPRESS

```
L0: recv a from P1;
L1: b = compute f(buff, a);
L2: send b to P1;
L0 and CPU-> L1
L1 -> L2
```

CPU

OFFLOAD



Offload Engine

portals

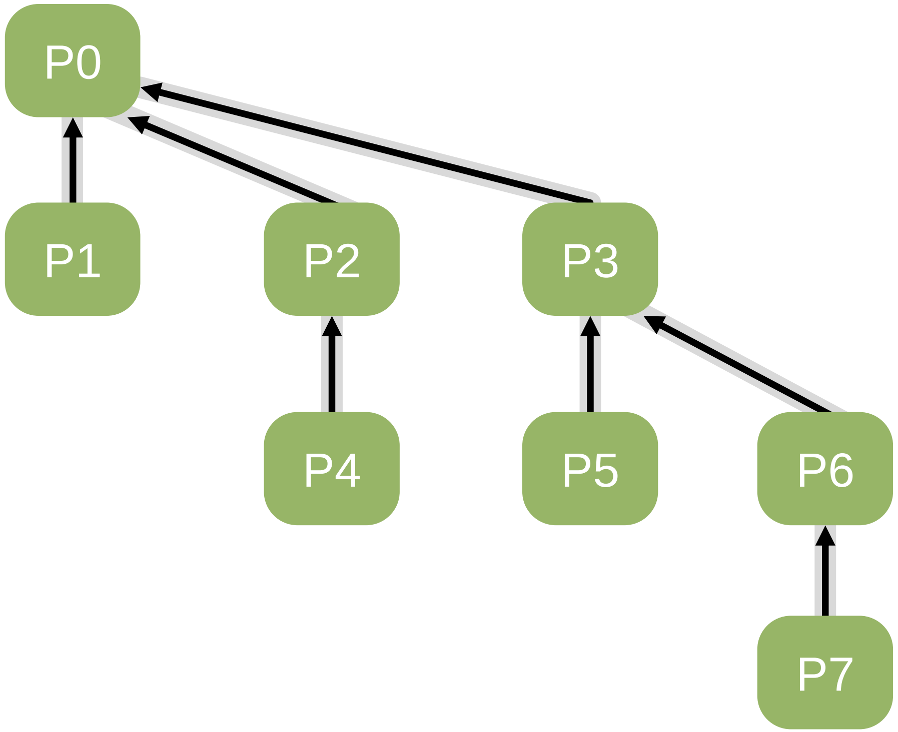
OPENFABRICS ALLIANCE

S. di Girolamo, P. Jolivet, K. D. Underwood, T. Hoefler: Exploiting Offload Enabled Network Interfaces, HOTI'

Fully Offloaded Collectives

Collective communication: A communication that involves a group of processes

Non-blocking collective: Once initiated the operation may progress independently of any computation or other communication at participating processes



S. di Girolamo, P. Jolivet, K. D. Underwood, T. Hoefler: Exploiting Offload Enabled Network Interfaces, HOTI'

Fully Offloaded Collectives

Collective communication: A communication that involves a group of processes

Non-blocking collective: Once initiated the operation may progress independently of any computation or other communication at participating processes

P0

Fully Offloading:

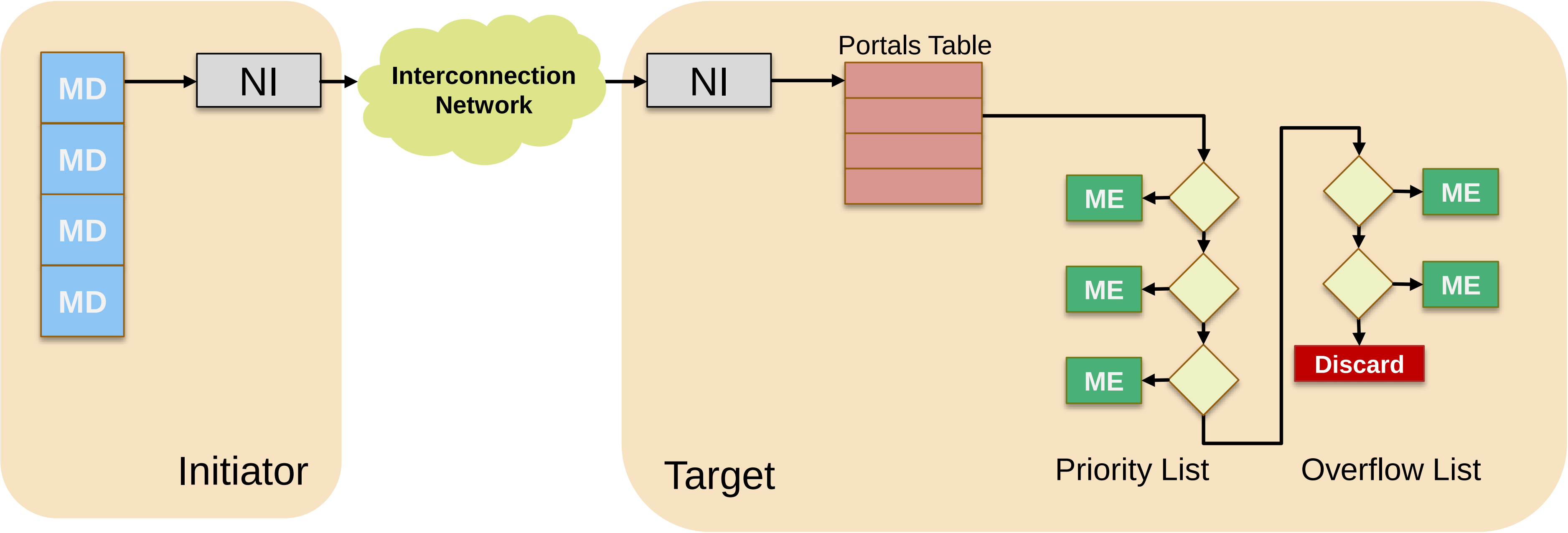
1. *No synchronization* is required in order to start the collective operation
2. Once a collective operation is started, *no further CPU intervention* is required in order to progress or complete it.

P7

S. di Girolamo, P. Jolivet, K. D. Underwood, T. Hoefler: Exploiting Offload Enabled Network Interfaces, HOTI'

A Case Study: Portals 4

- **Based on the one-sided communication model**
- **Matching/Non-Matching semantics can be adopted**



[2] The Portal 4.0.2 Network Programming Interface

A Case Study: Portals 4

Communication primitives

- Put/Get operations are natively supported by Portals 4
- One-sided + matching semantic

Atomic operations

- Operands are the data specified by the MD at the initiator and by the ME at the target
- Available operators: *min, max, sum, prod, swap, and, or, …*

Counters

- Associated with MDs or MEs
- Count specific events (e.g., operation completion)

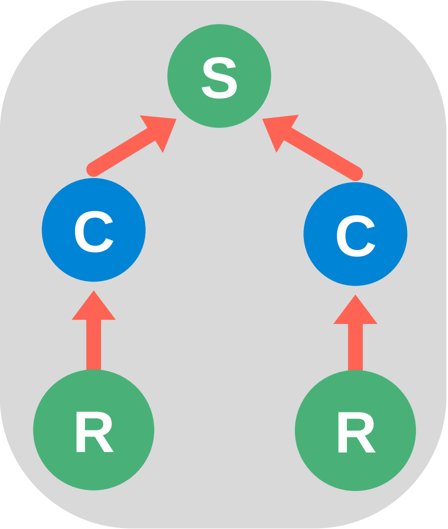
Triggered operations

- Put/Get/Atomic associated with a counter
- Executed when the associated counter reaches the specified threshold

FFlib: An Example

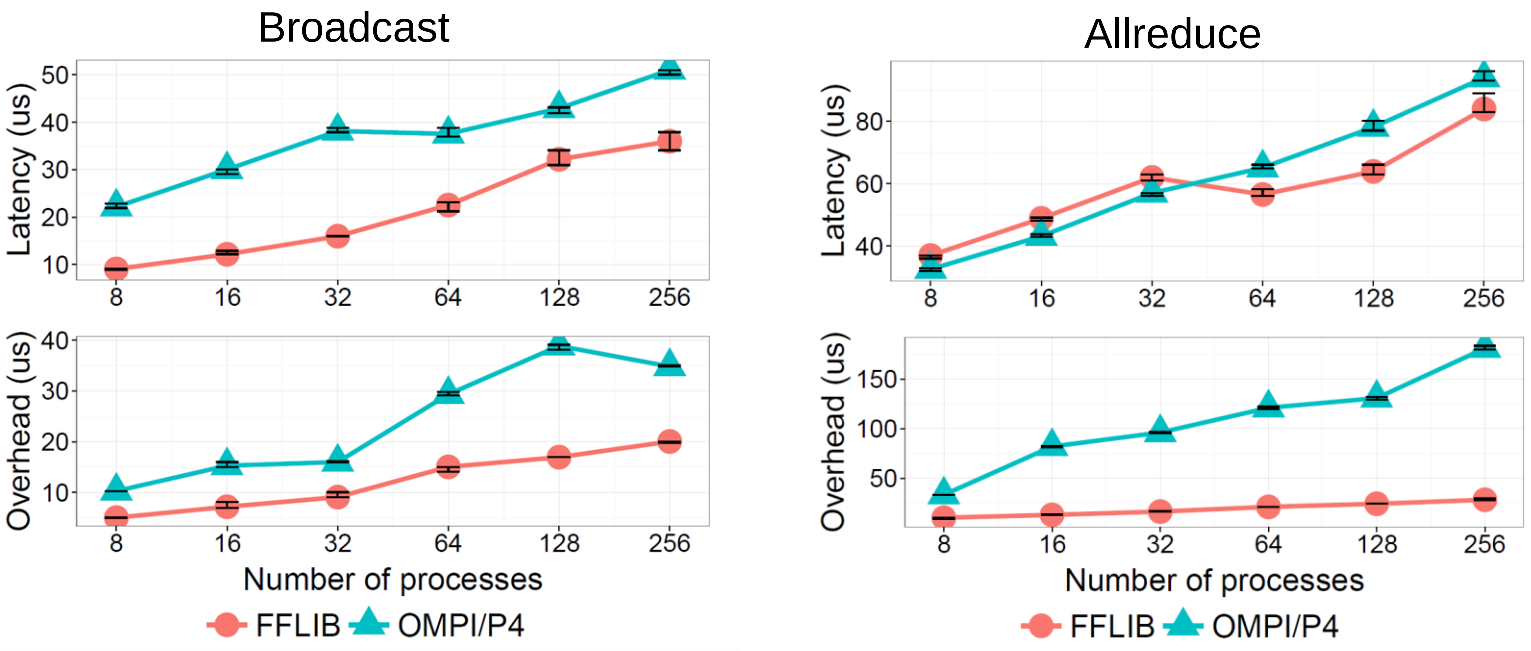
Proof of concept library implemented on top of Portals 4

```
ff_schedule_h sched = ff_schedule_create(…);

ff_op_h r1 = ff_op_create_recv(tmp + blocksize, blocksize, child1, tag);
ff_op_h r2 = ff_op_create_recv(tmp + 2*blocksize, blocksize, child2, tag);

ff_op_h c1 = ff_op_create_computation(rbuff, blocksize, tmp + blocksize, blocksize, operator, datatype, tag)
ff_op_h c2 = ff_op_create_computation(rbuff, blocksize, tmp + 2*blocksize, blocksize, operator, datatype, tag)

ff_op_h s = ff_op_create_send(rbuff, blocksize, parent, tag)

ff_op_hb(r1, c1)
ff_op_hb(r2, c2)
ff_op_hb(c1, s)
ff_op_hb(c2, s)

ff_schedule_add(sched, r1)
ff_schedule_add(sched, r2)
ff_schedule_add(sched, c1)
ff_schedule_add(sched, c2)
ff_schedule_add(sched, s)
```



S. di Girolamo, P. Jolivet, K. D. Underwood, T. Hoefler: Exploiting Offload Enabled Network Interfaces, HOTI’

Experimental Results: Latency/Overhead



Target machine: Curie

- 5,040 nodes
- 2 eight-core Intel Sandy Bridge processors
- Full fat-tree Infiniband QDR

OMPI/P4: Open MPI 1.8.4 + Portals 4 RL

FFLIB: proof of concept library

More about FFLIB at : http://spcl.inf.ethz.ch/Research/Parallel_Programming/FFlib/

Active RDMA – what could it be?

Remote Synchronization

[IPDPS'15]

- Extend RMA semantics
- Fully one-sided (in HW)
- Synchronization

Remote Transactions

[HPDC'15]

- Similar to HTM
- Extend across nodes
- Think active messages?

Network Instruction Set Architecture (NISA)

Remote Invocation

[ICS'15]

- Utilizes IOMMUs
- Control transfer
- Active memory

NISA: Process the data while it moves!