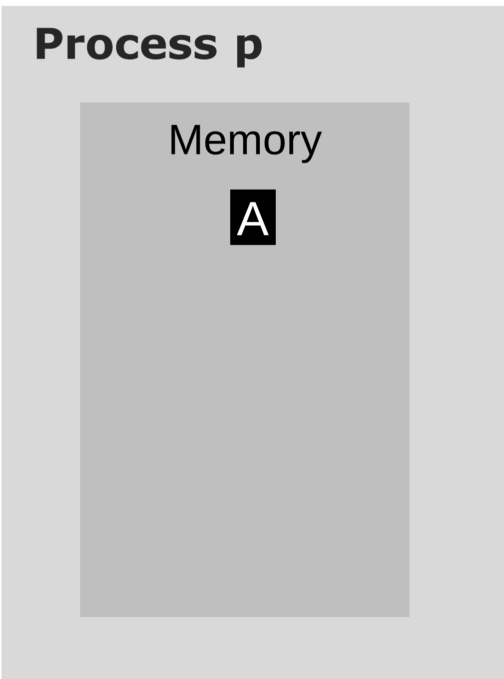
Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations

MACIEJ BESTA, TORSTEN HOEFLER

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Remote Memory Access (RMA) Programming



REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Remote Memory Access (RMA) Programming

Process p

Memory

A

Process q

Memory

B

Cray
BlueWaters

Remote Memory Access (RMA) Programming

Process p

Memory

A

Process q

Memory

B

Cray BlueWaters

Remote Memory Access (RMA) Programming

Process p

Memory

A

Process q

Memory

B

Cray BlueWaters

Remote Memory Access (RMA) Programming

Process p

Memory

A

Process q

Memory

B

Cray
BlueWaters

Remote Memory Access (RMA) Programming

Process p

Memory

A

A put

Process q

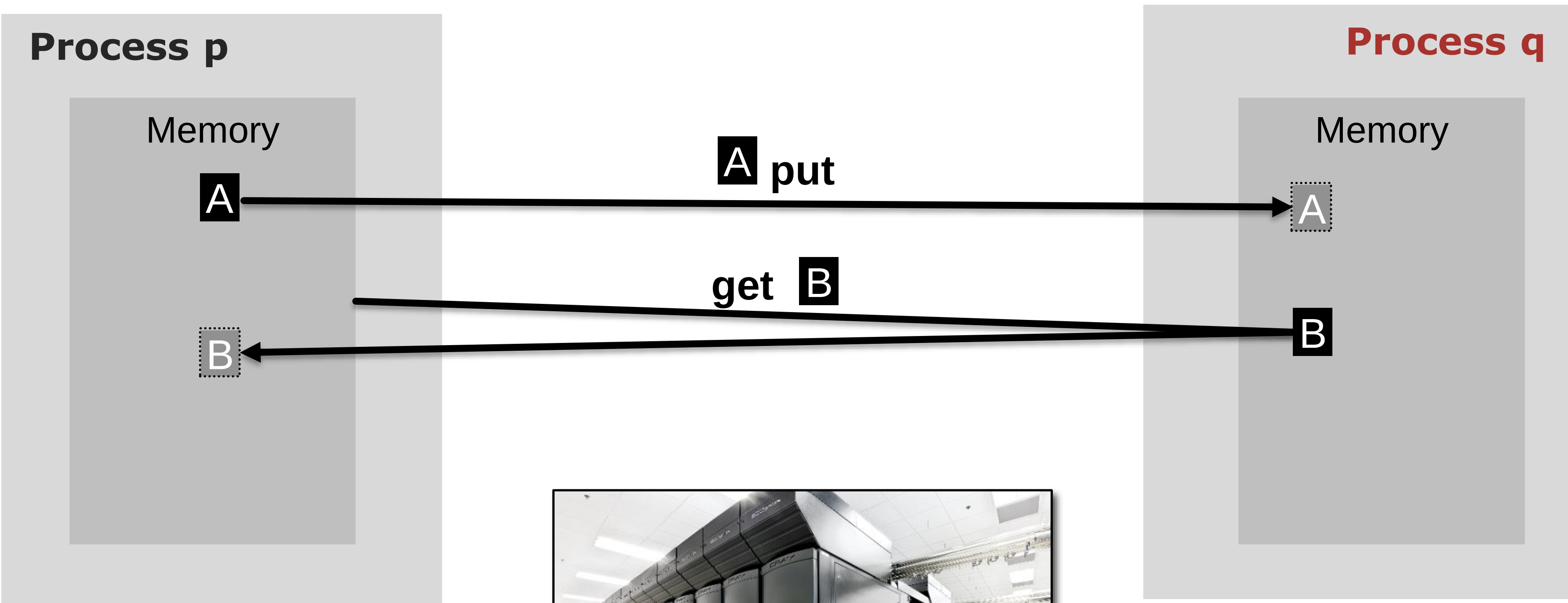
Memory

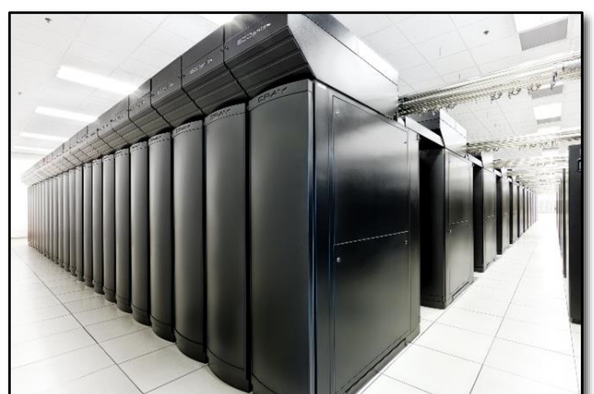
A

B

Cray BlueWaters

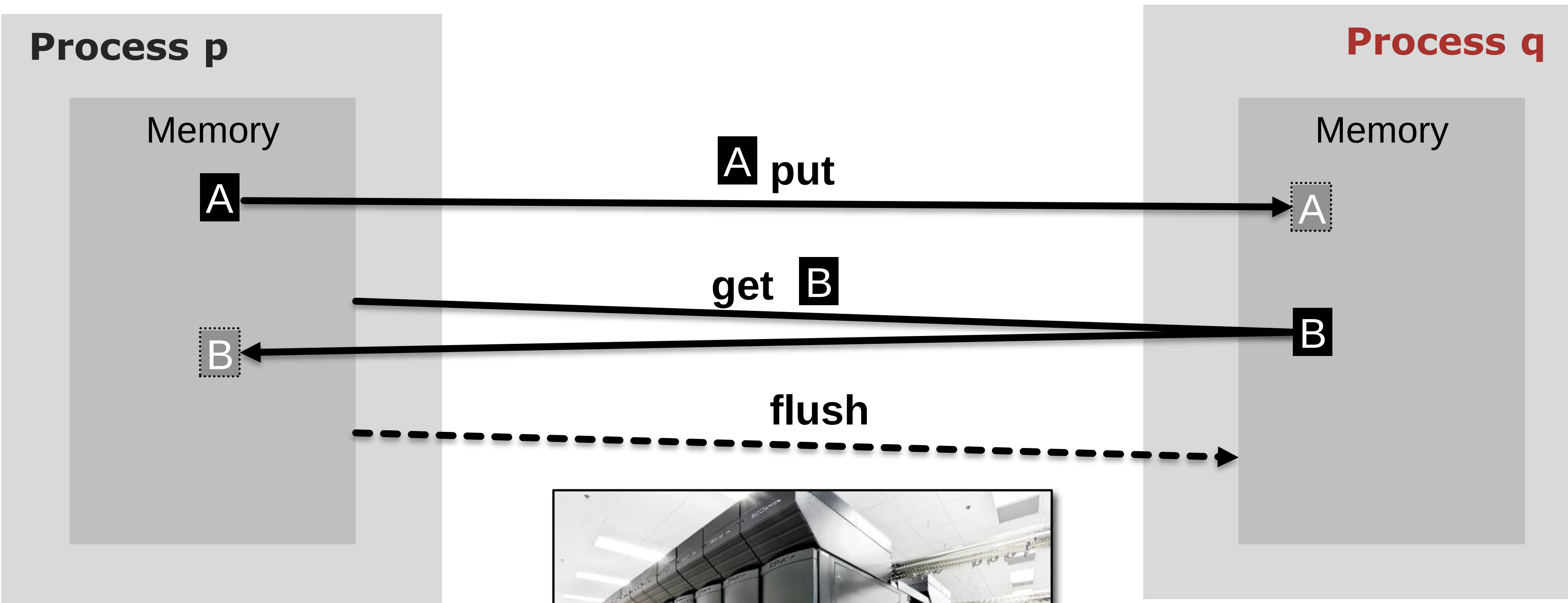
Remote Memory Access (RMA) Programming

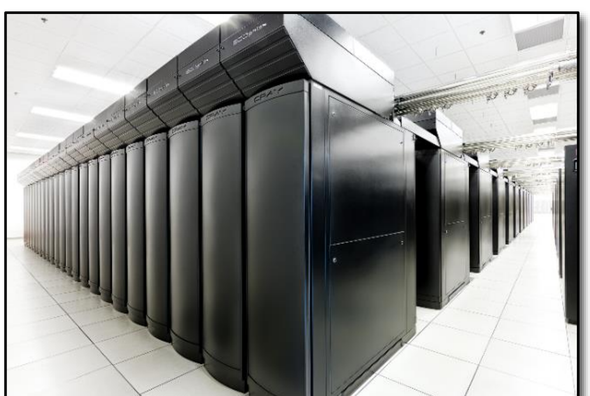




Cray
BlueWaters

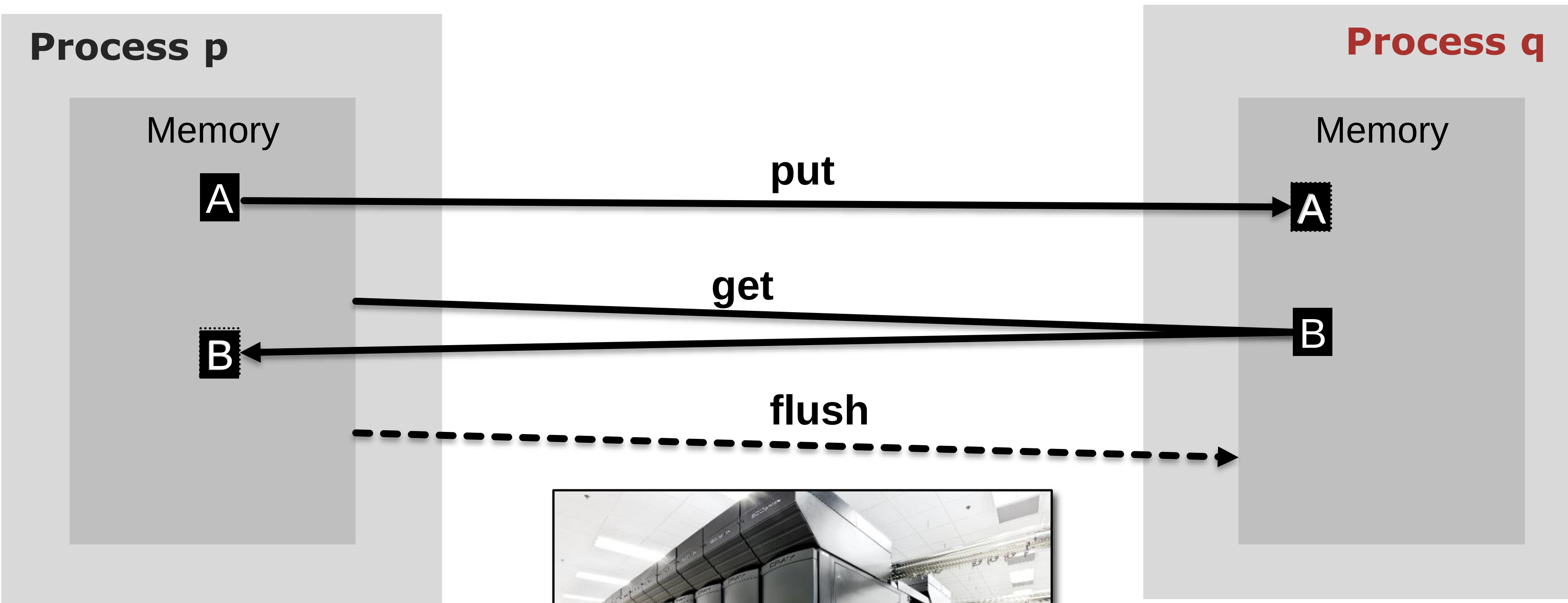
Remote Memory Access (RMA) Programming





Cray BlueWaters

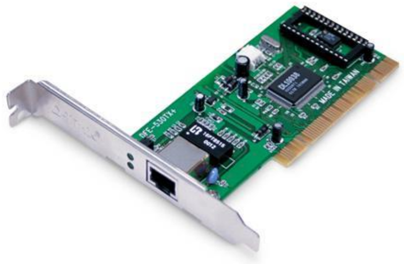
Remote Memory Access (RMA) Programming



Cray BlueWaters

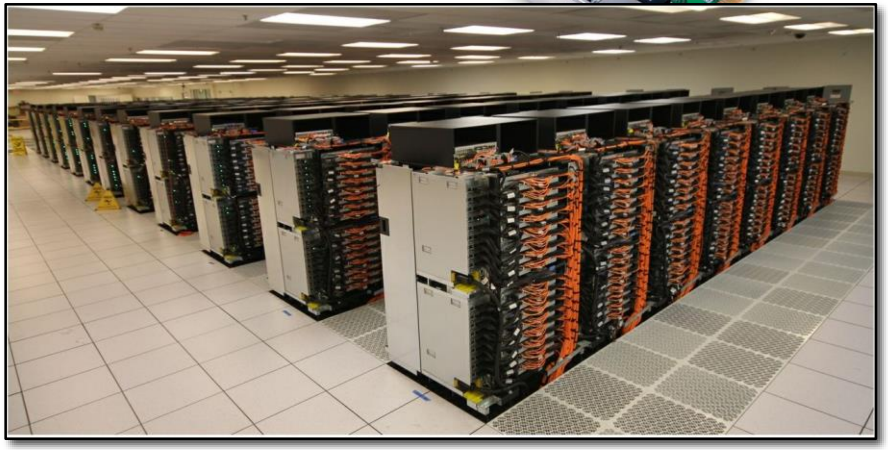
Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks (RDMA)



Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks (RDMA)

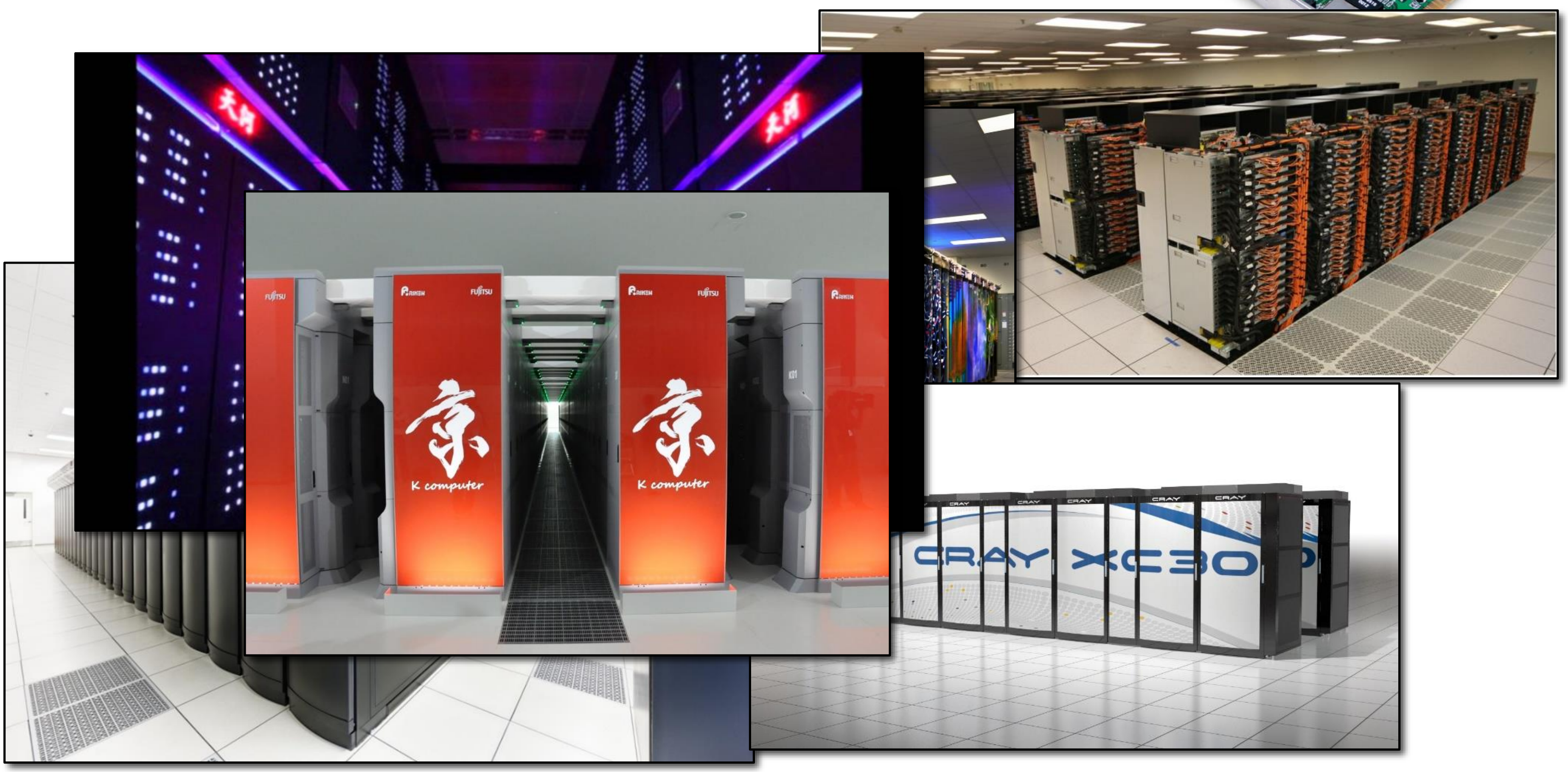


Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks (RDMA)

Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks (RDMA)



Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks (RDMA)

Remote Memory Access Programming

- Supported by many HPC libraries and languages

Remote Memory Access Programming

- Supported by many HPC libraries and languages

Remote Memory Access Programming

- Supported by many HPC libraries and languages

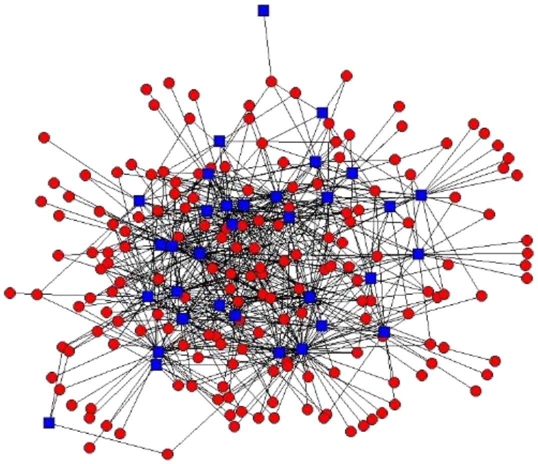
Remote Memory Access Programming

- Enables significant speedups over message passing in many types of applications, e.g.:

[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

Remote Memory Access Programming

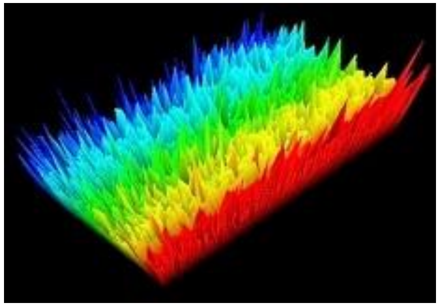
- Enables significant speedups over message passing in many types of applications, e.g.:
  - Speedup of ~1.5 for communication patterns in irregular workloads

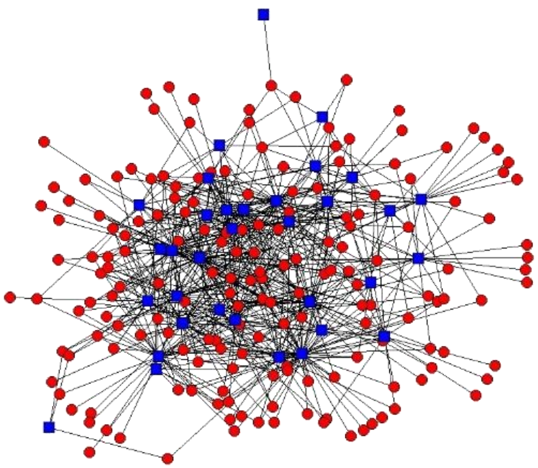


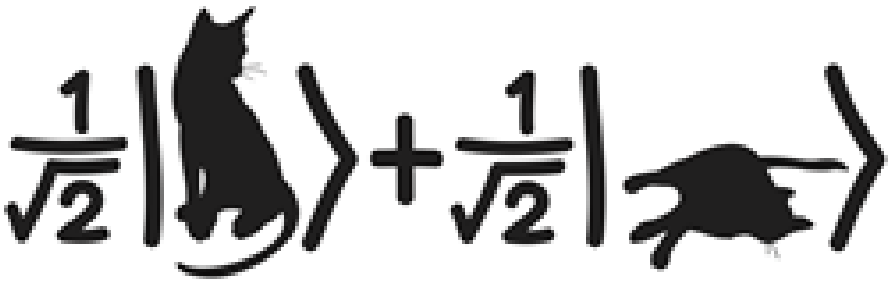
[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

Remote Memory Access Programming

- Enables significant speedups over message passing in many types of applications, e.g.:
  - Speedup of ~1.5 for communication patterns in irregular workloads
  - Speedup of ~1.4-2 in physics computations



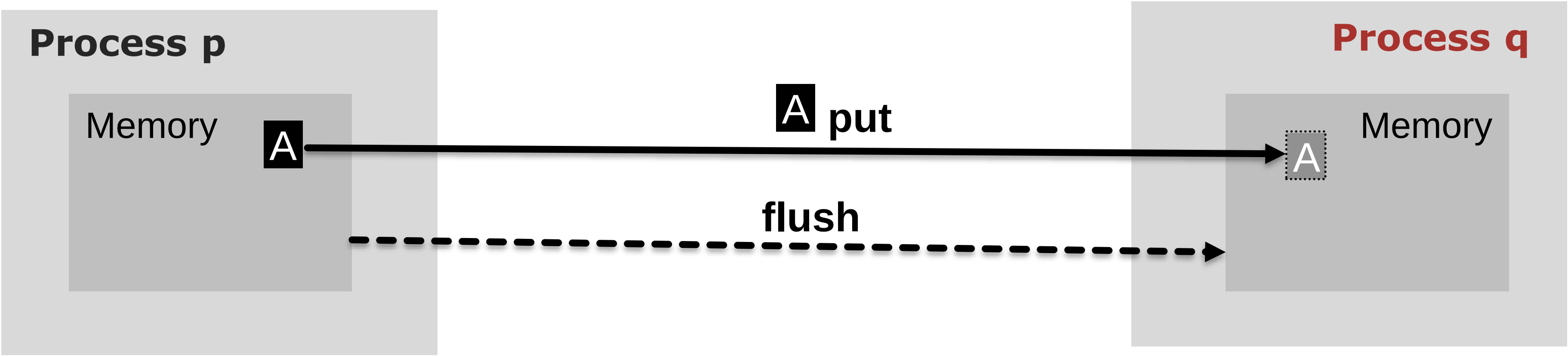




[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

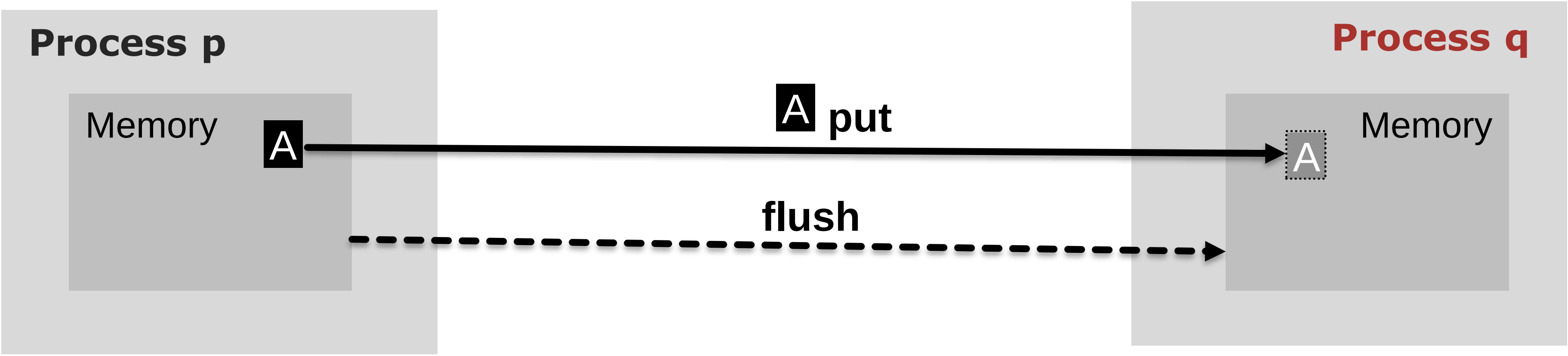
RMA vs. Message Passing

RMA:



RMA vs. Message Passing

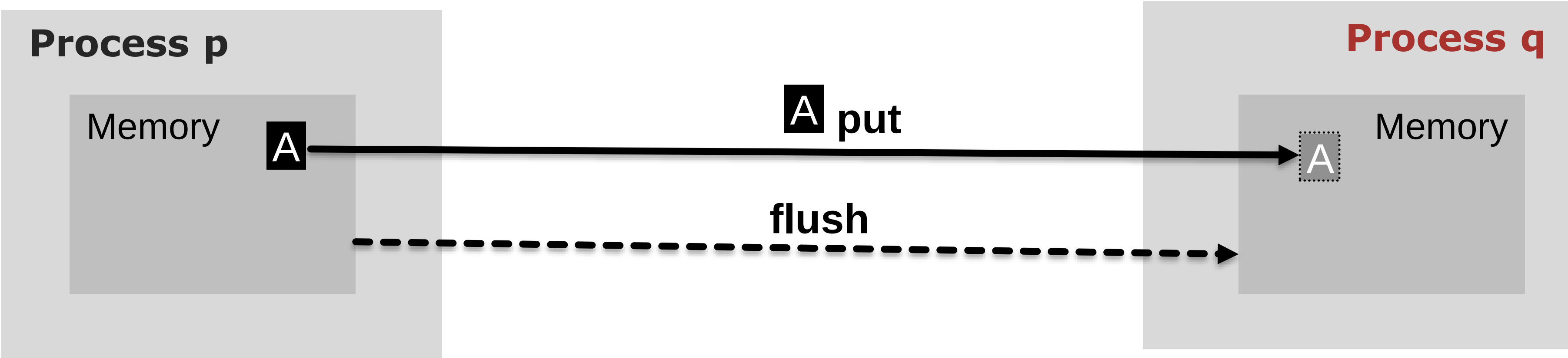
RMA:



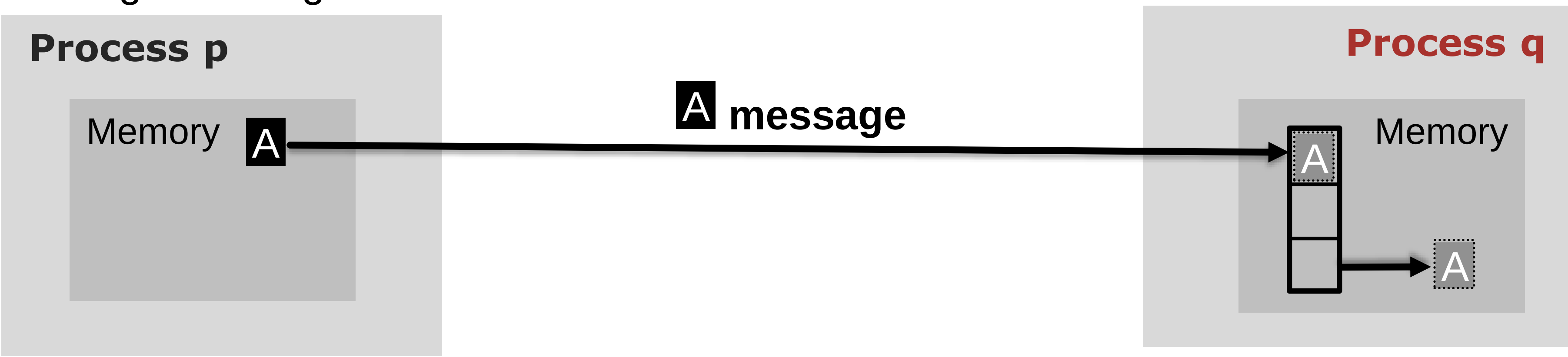
Message Passing:

RMA vs. Message Passing

RMA:



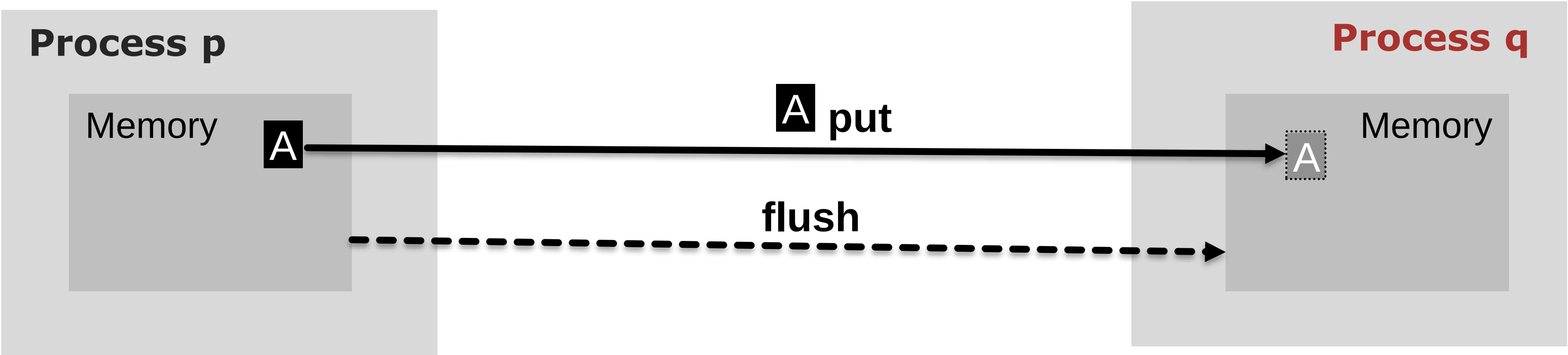
Message Passing:



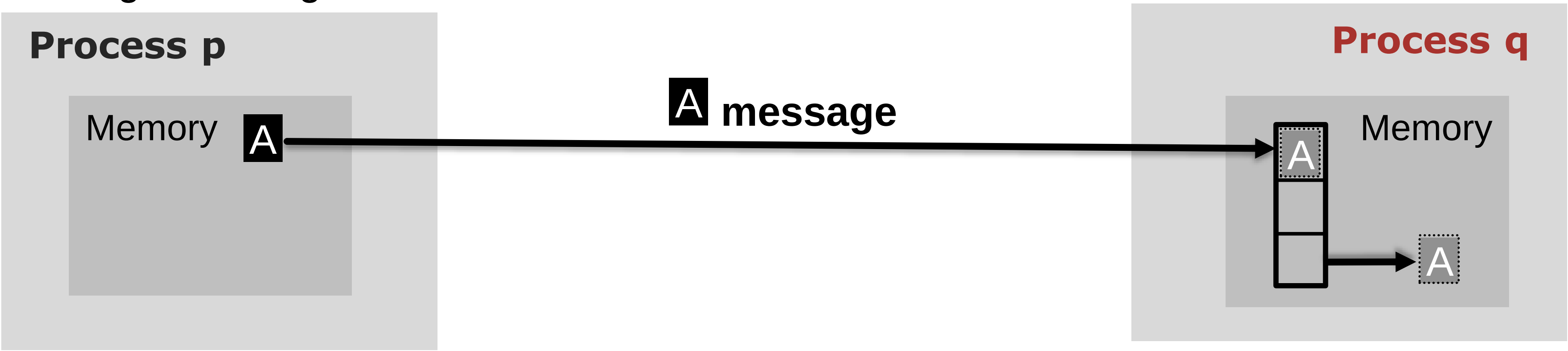
RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:

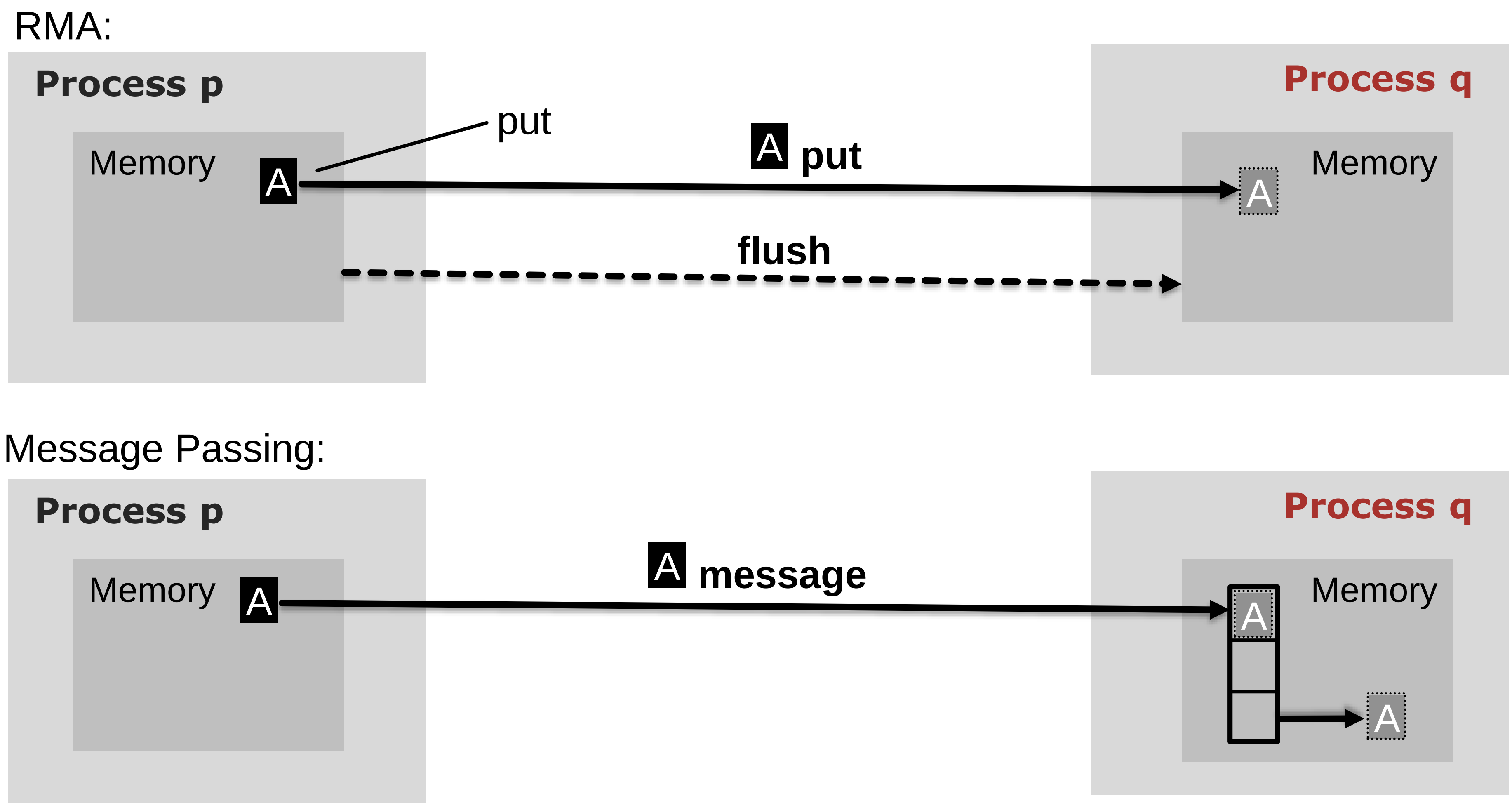


Message Passing:



RMA vs. Message Passing

- Communication in RMA is one-sided

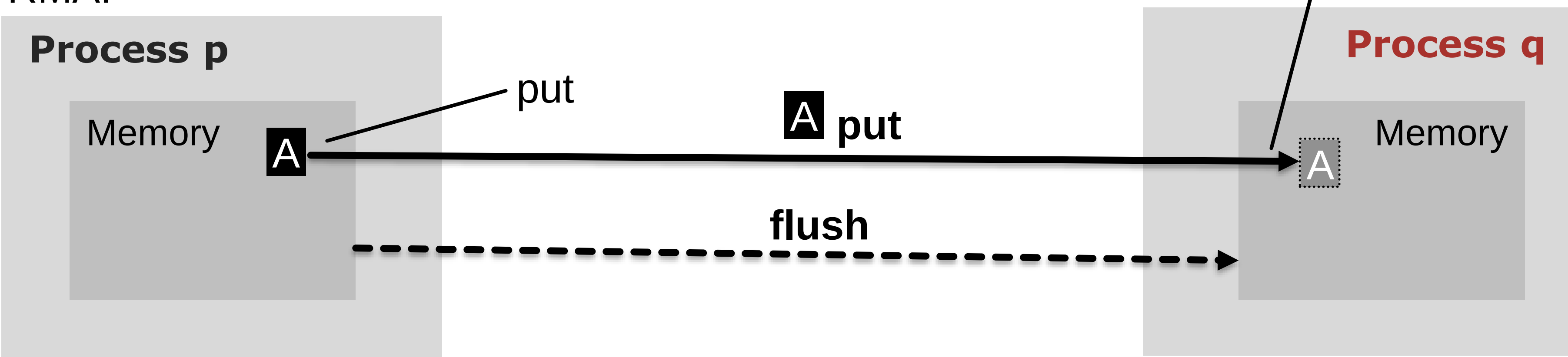


RMA vs. Message Passing

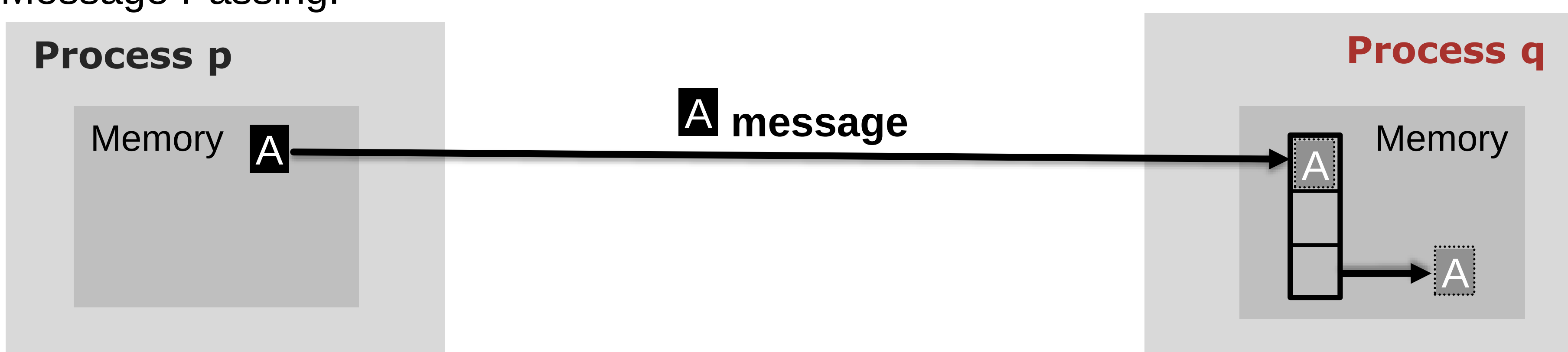
- Communication in RMA is one-sided

no active participation, direct access to memory

RMA:

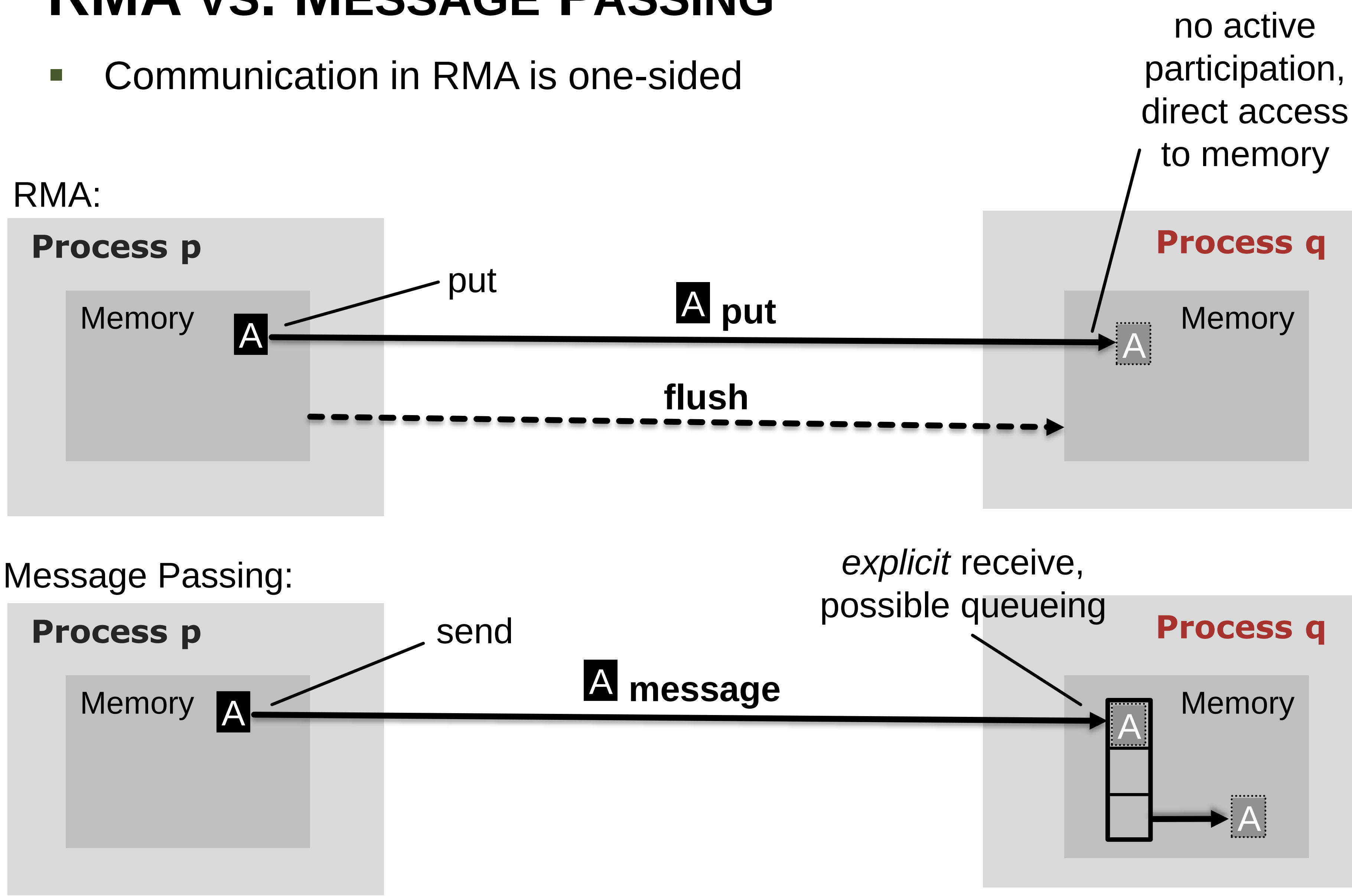


Message Passing:



RMA vs. Message Passing

- Communication in RMA is one-sided



REMOTE MEMORY ACCESS PROGRAMMING

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

Remote Memory Access Programming

- Is it ideal?

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

Remote Memory Access Programming

- Is it ideal? 

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

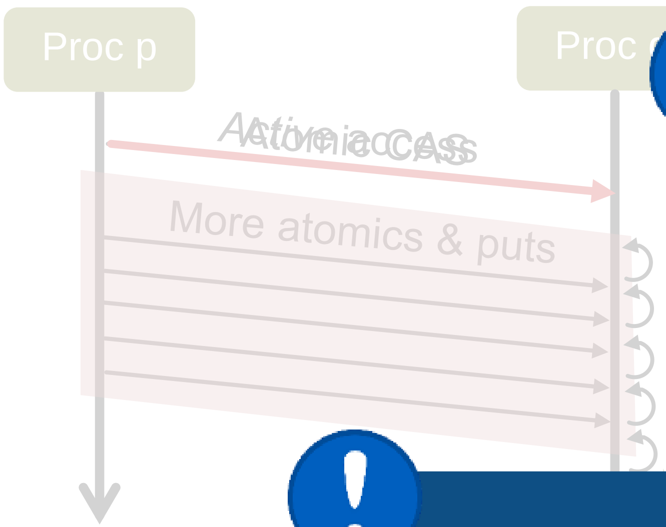
REMOTE MEMORY ACCESS PROGRAMMING

- Is i
- Co
distributed hashtable...

How to enable it?

No hash collision:

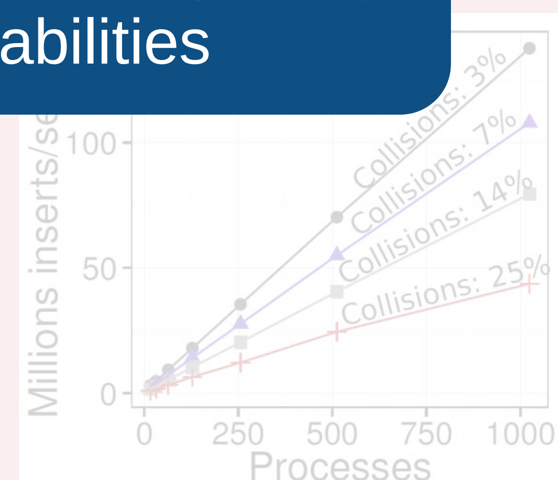
→ 1 remote atomic
→ Up to 5x speedup over MP [1]



A hash collision:

…emote puts
…drops

Use and extend I/O MMUs and their paging capabilities

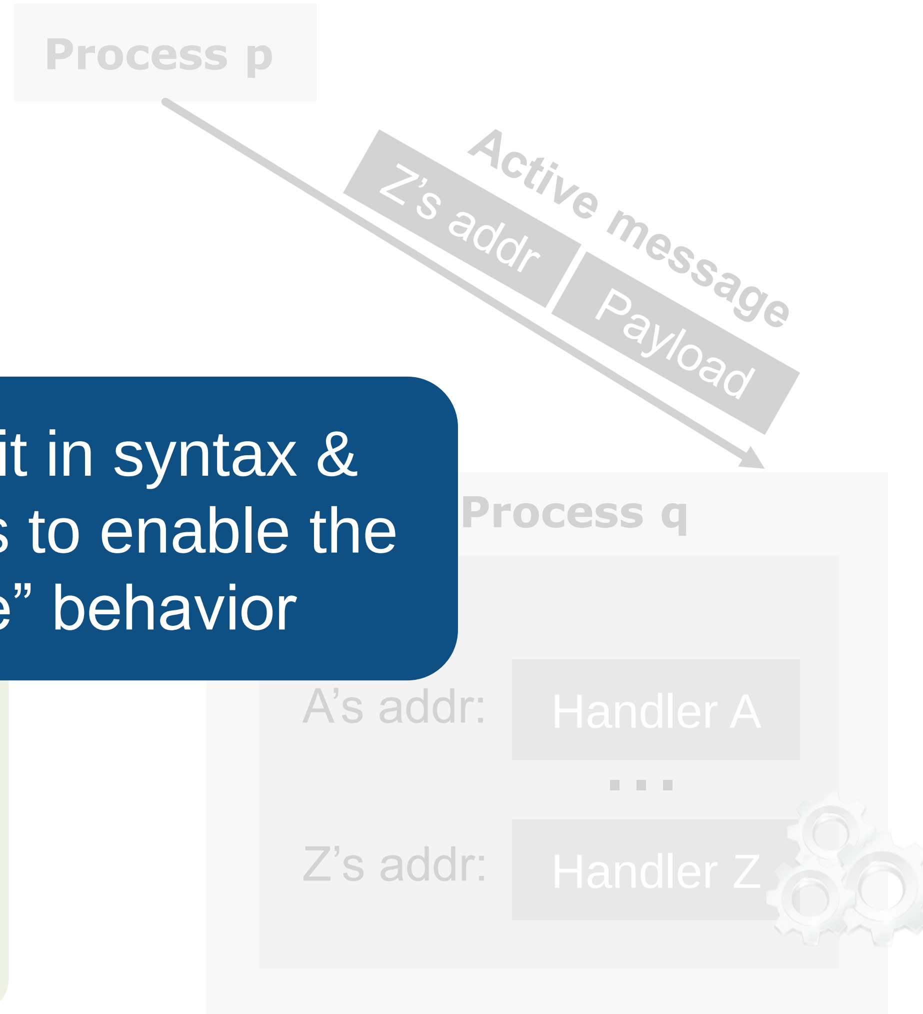


Local executi…
active access: in RMA?

Use “active” semantics

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

Use semantics from Active Messages (AM) [1]



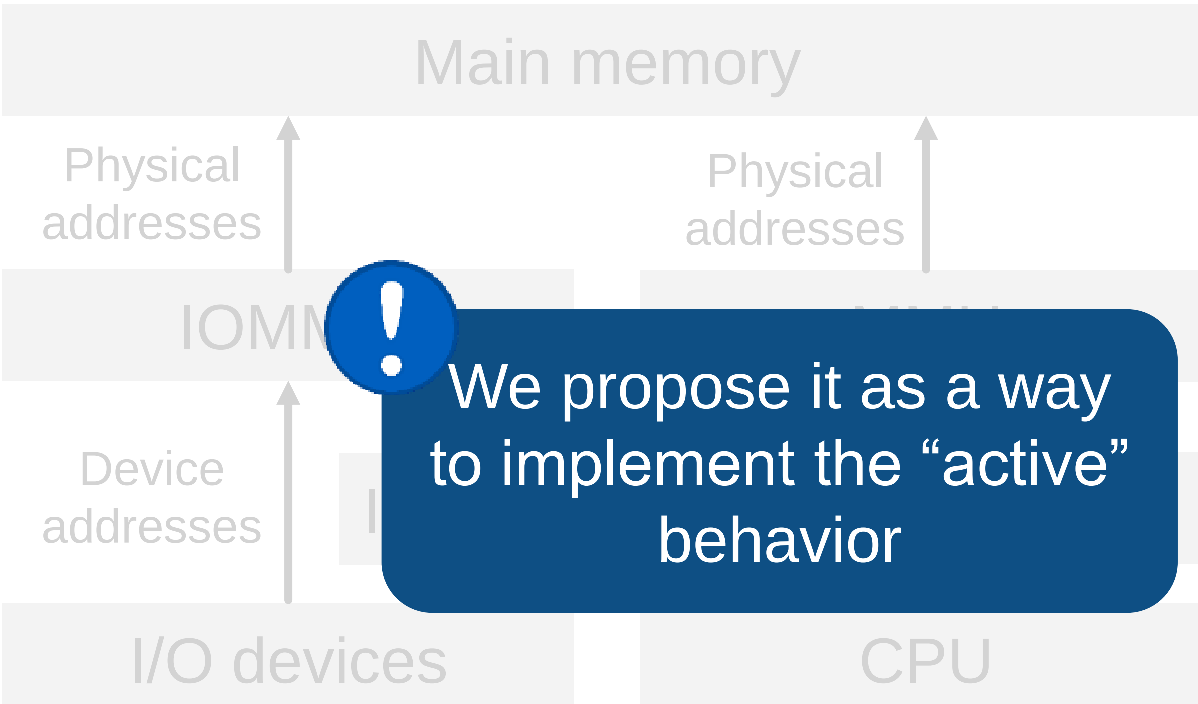
We use it in syntax & semantics to enable the "active" behavior

We need *active* puts/gets:

- Invoke a handler upon accessing a given page
- Preserve one-sided RMA behavior

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.
[2] J. J. Willcock et al. AM++: A generalized active message framework. PACT'10.
[3] D. Bonachea, GASNet Specification, v1.1. Berkeley Technical Report. 2002.

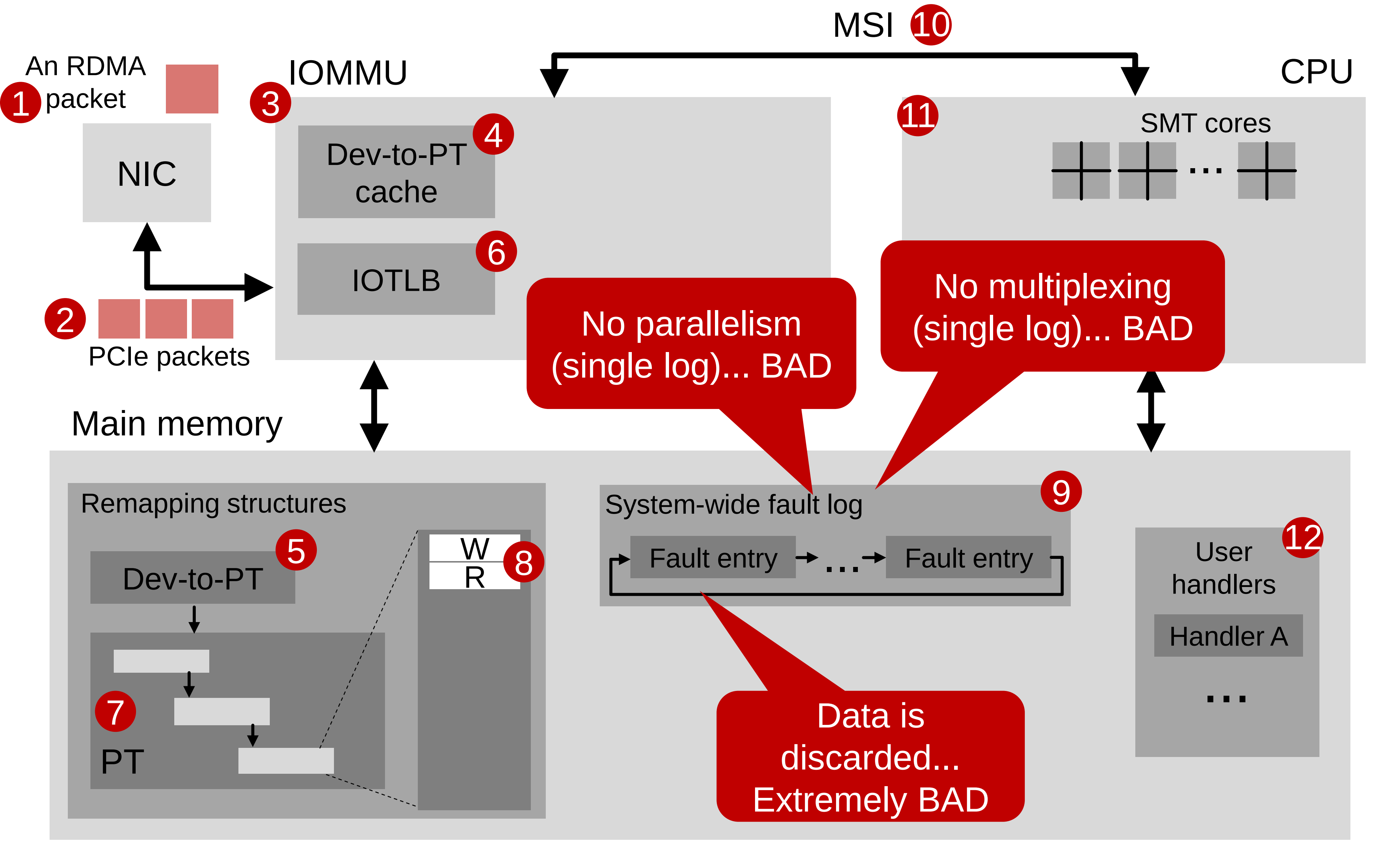
Use Input/Output Memory Management Units



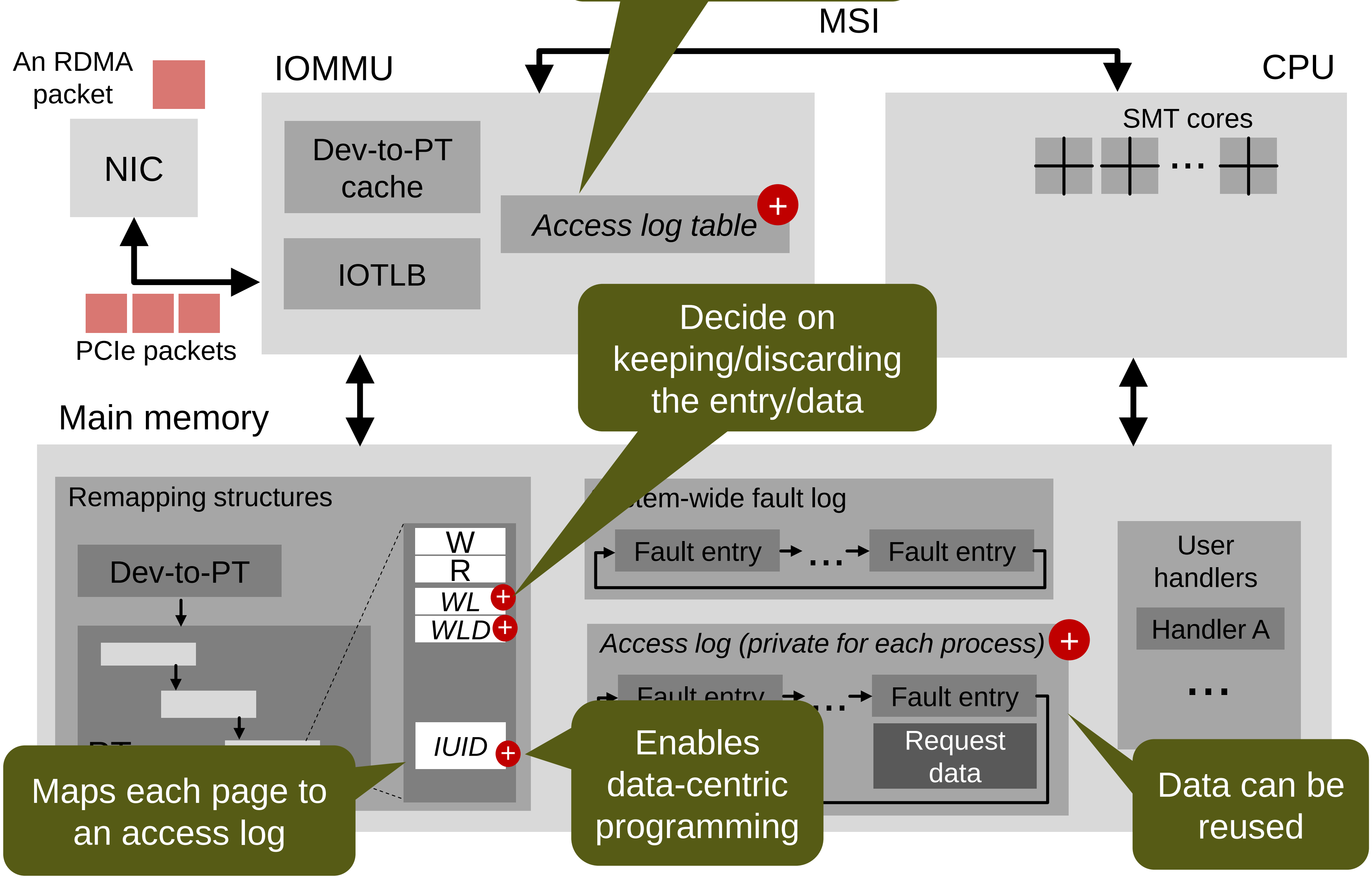
We propose it as a way to implement the "active" behavior

IOMMUs AND RMA

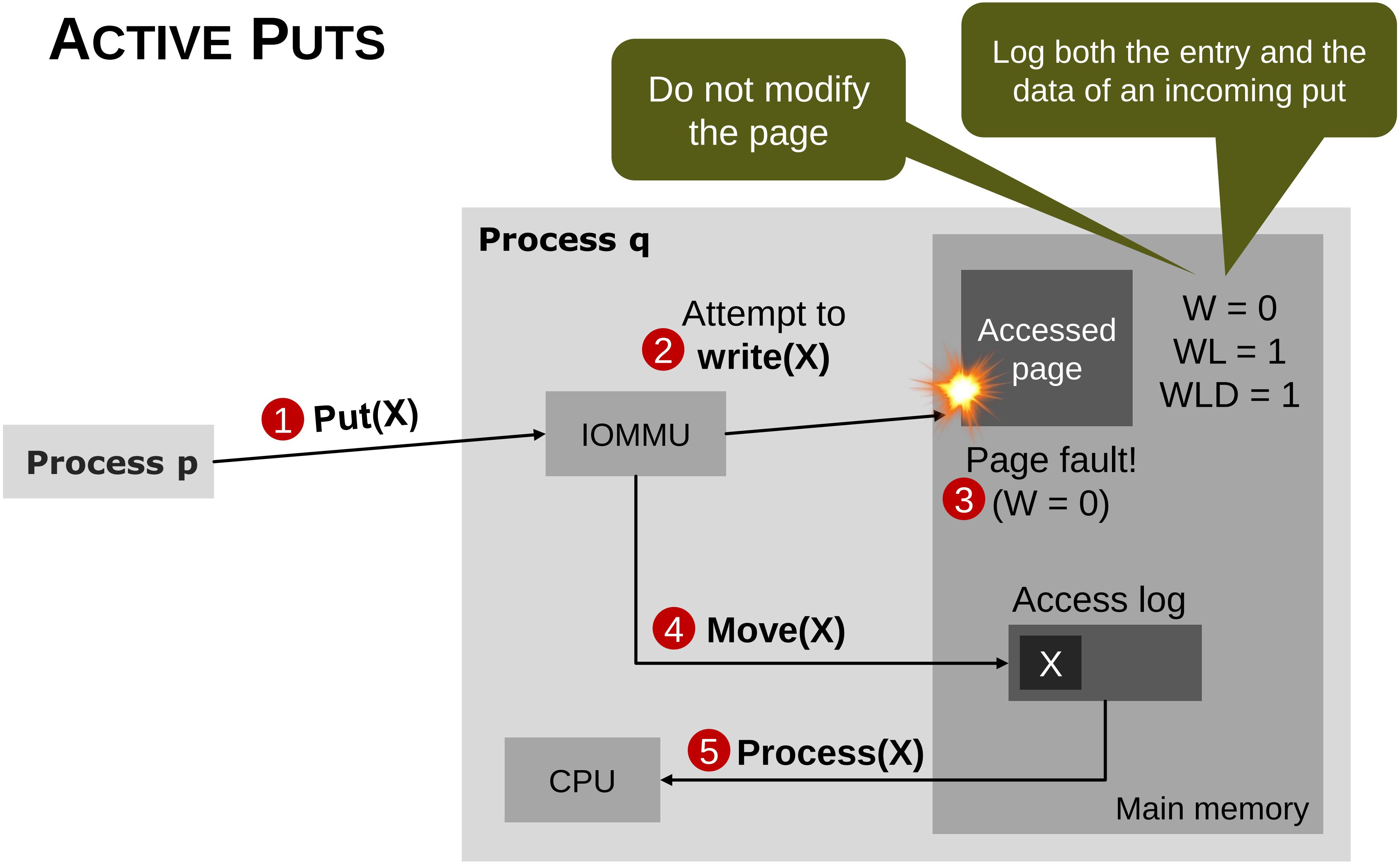
We could use it somehow. But…



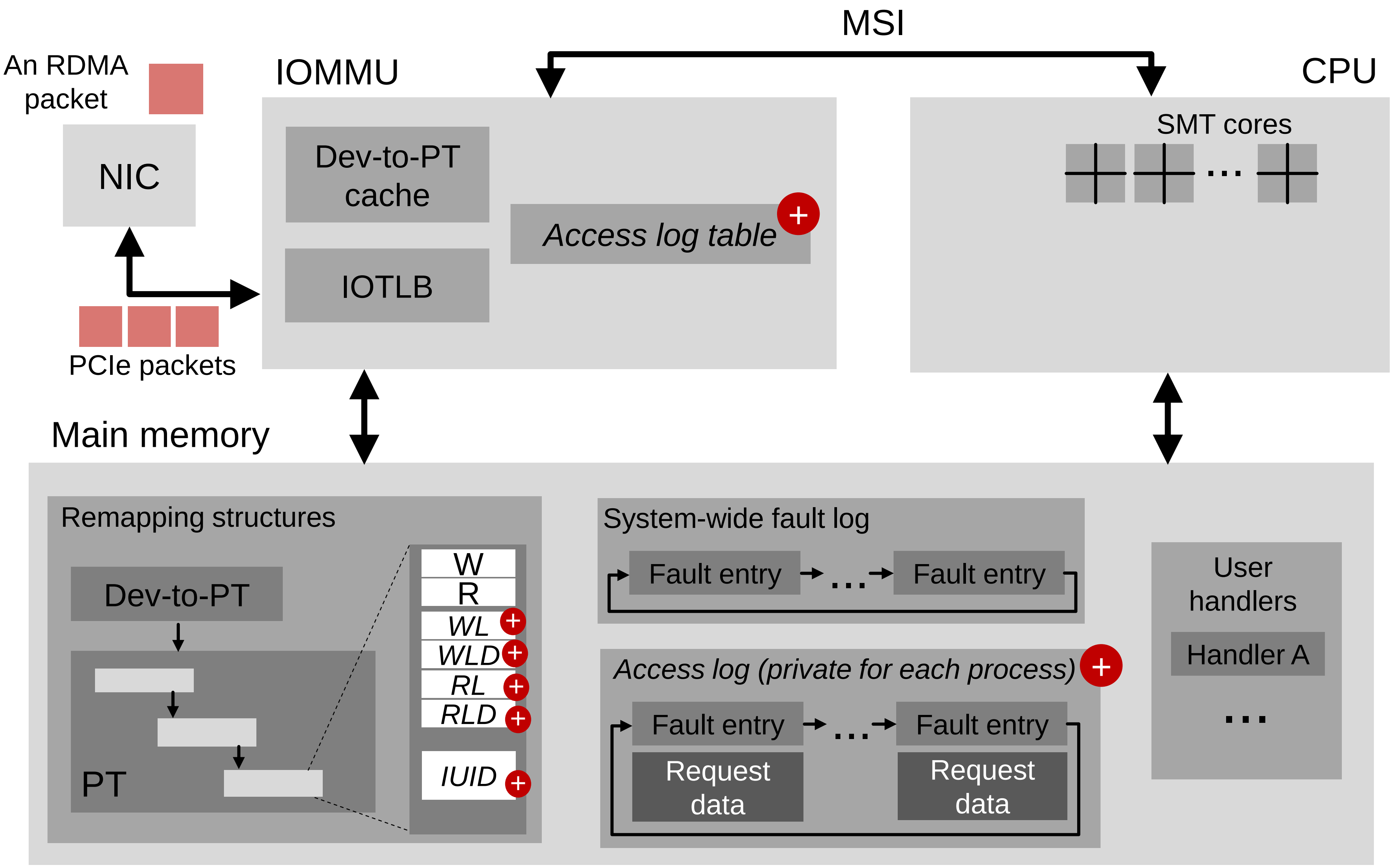
Active Puts


Stores addresses of each access log
MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
Access log table
SMT cores
PCIe packets
Decide on keeping/discarding the entry/data
Main memory
Remapping structures
Dev-to-PT
W
R
WL
WLD
IUID
System-wide fault log
Fault entry
Fault entry
User handlers
Handler A
Access log (private for each process)
Fault entry
Fault entry
Request data
Maps each page to an access log
Enables data-centric programming
Data can be reused

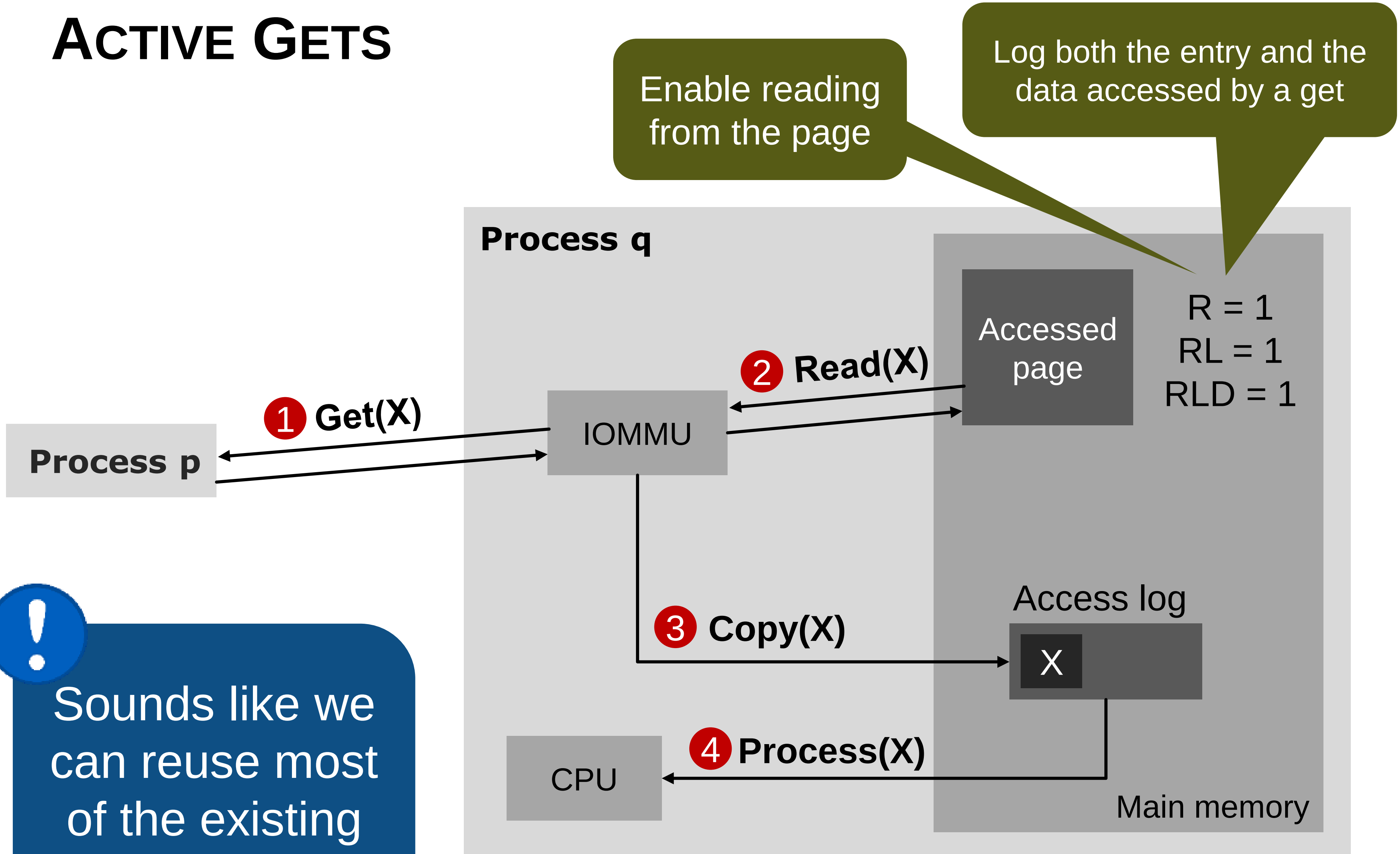
Active Puts


Do not modify the page
Log both the entry and the data of an incoming put
Process q
Attempt to
2 write(X)
1 Put(X)
Process p
IOMMU
Accessed page
W = 0
WL = 1
WLD = 1
Page fault!
3 (W = 0)
Access log
4 Move(X)
X
CPU
5 Process(X)
Main memory

Active Gets


MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
Access log table
SMT cores
PCIe packets
Main memory
Remapping structures
Dev-to-PT
PT
W
R
WL
WLD
RL
RLD
IUID
System-wide fault log
Fault entry
Fault entry
Access log (private for each process)
Fault entry
Fault entry
Request data
Request data
User handlers
Handler A

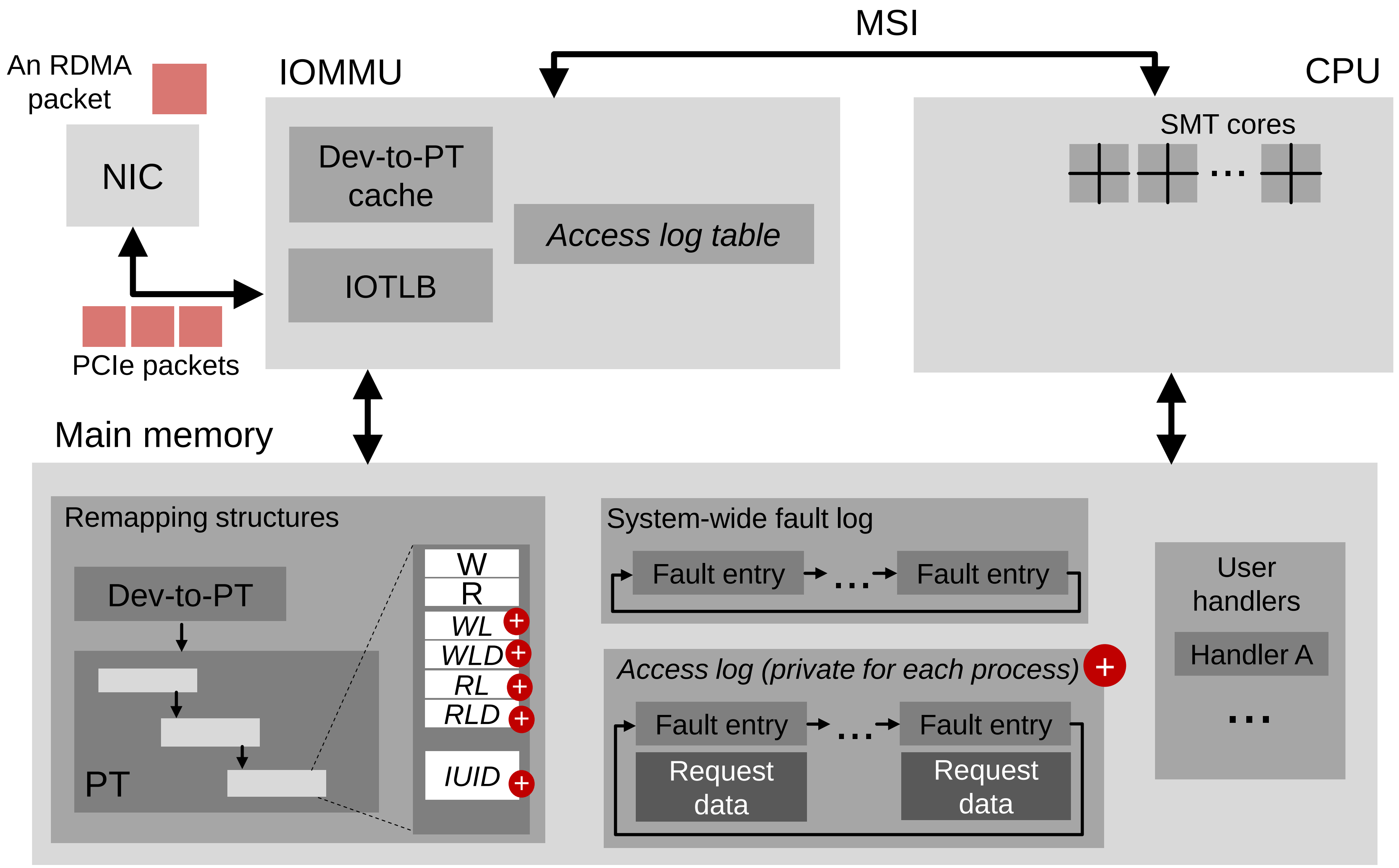
Active Gets



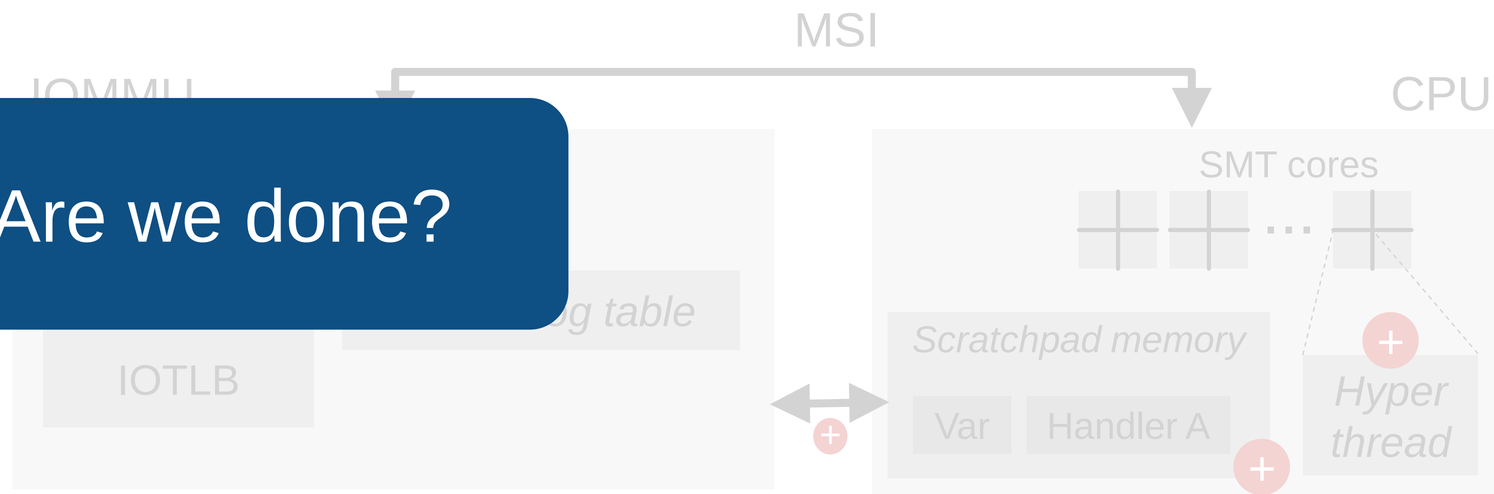
Sounds like we can reuse most of the existing stuff!

intel + PCI EXPRESS

INTERACTIONS WITH THE CPU



Interactions with the CPU

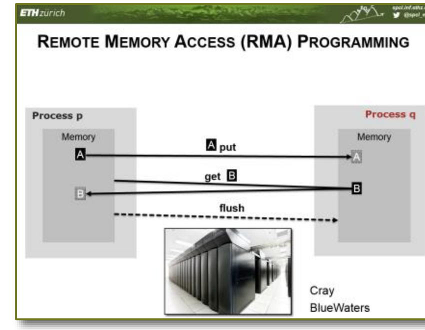


Are we done?

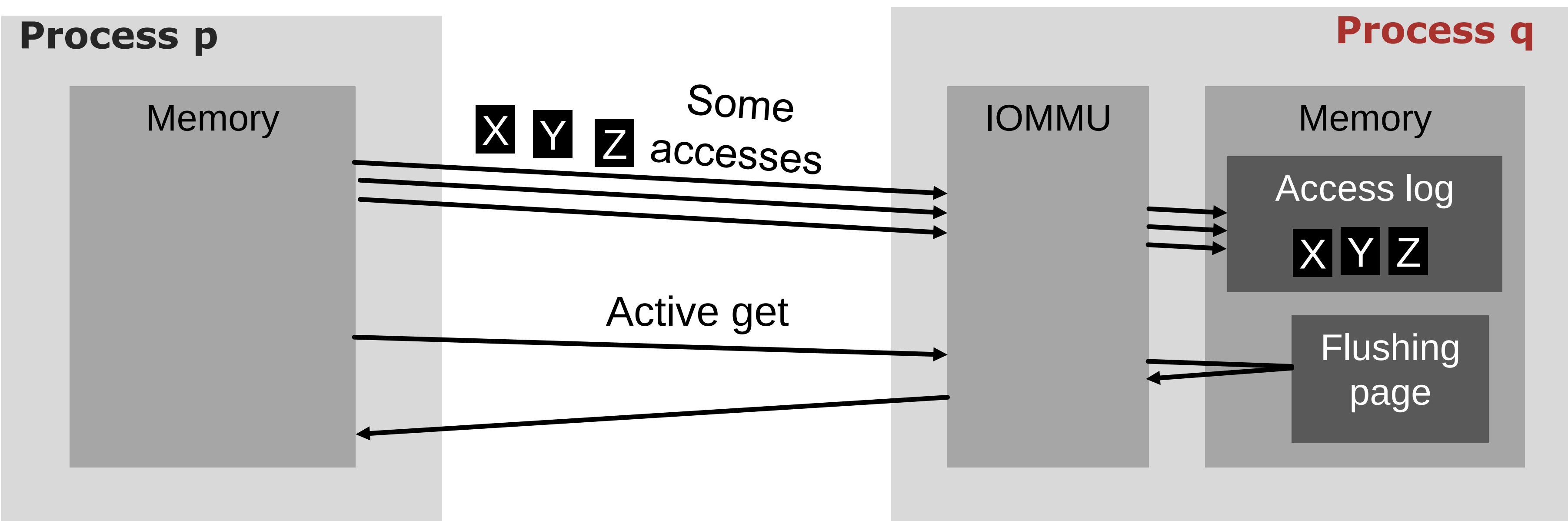
- Interrupts
- Polling
- Direct notifications via sc

Well…

Consistency

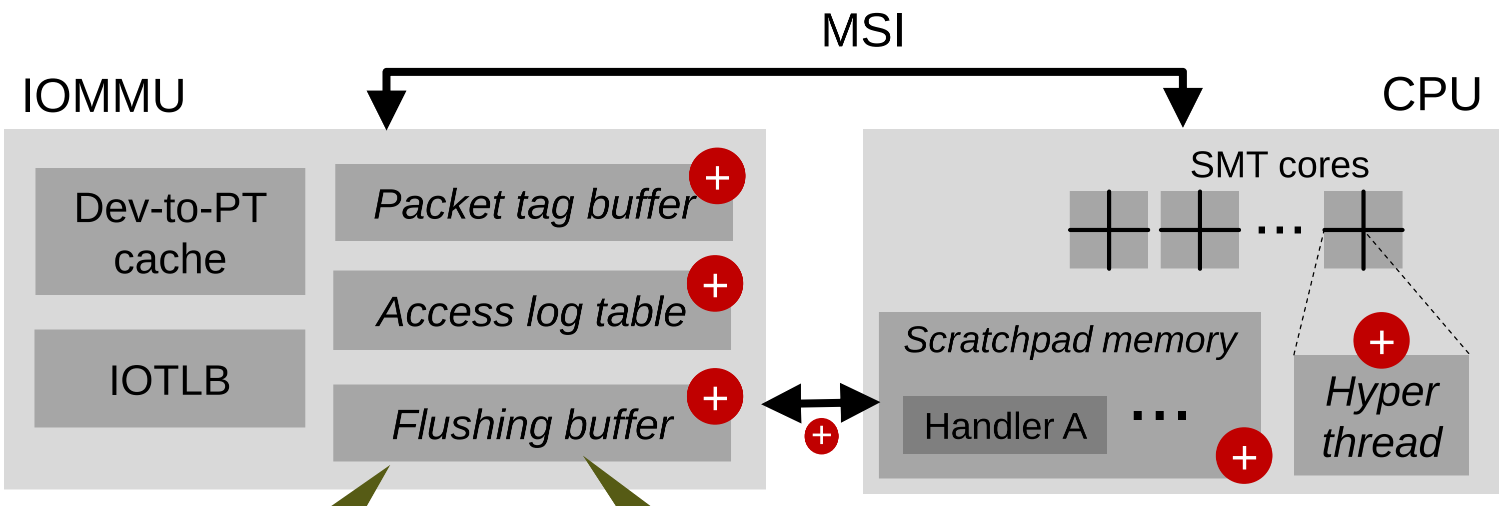


- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

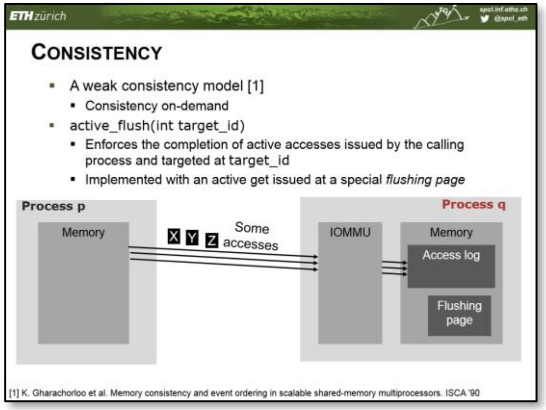
Consistency



Contains the addresses of flushing pages

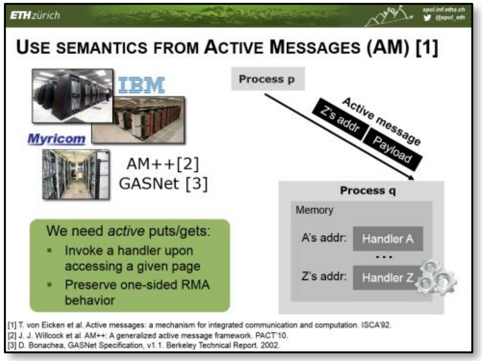
Maps flushing pages to IUIDs and access logs

Let's summarize…

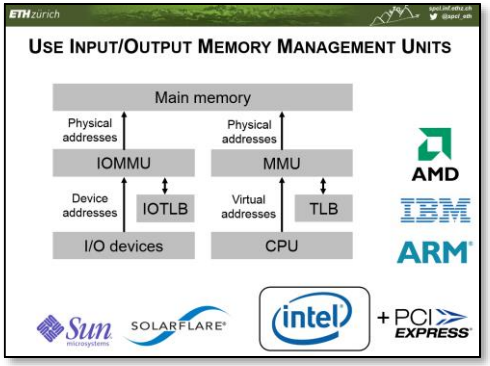


Consistency

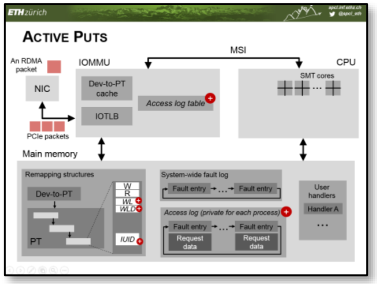
IOMMUs

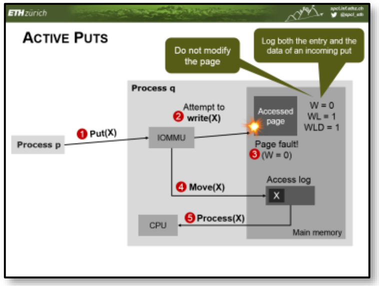


Active Messages



Active Puts/Gets



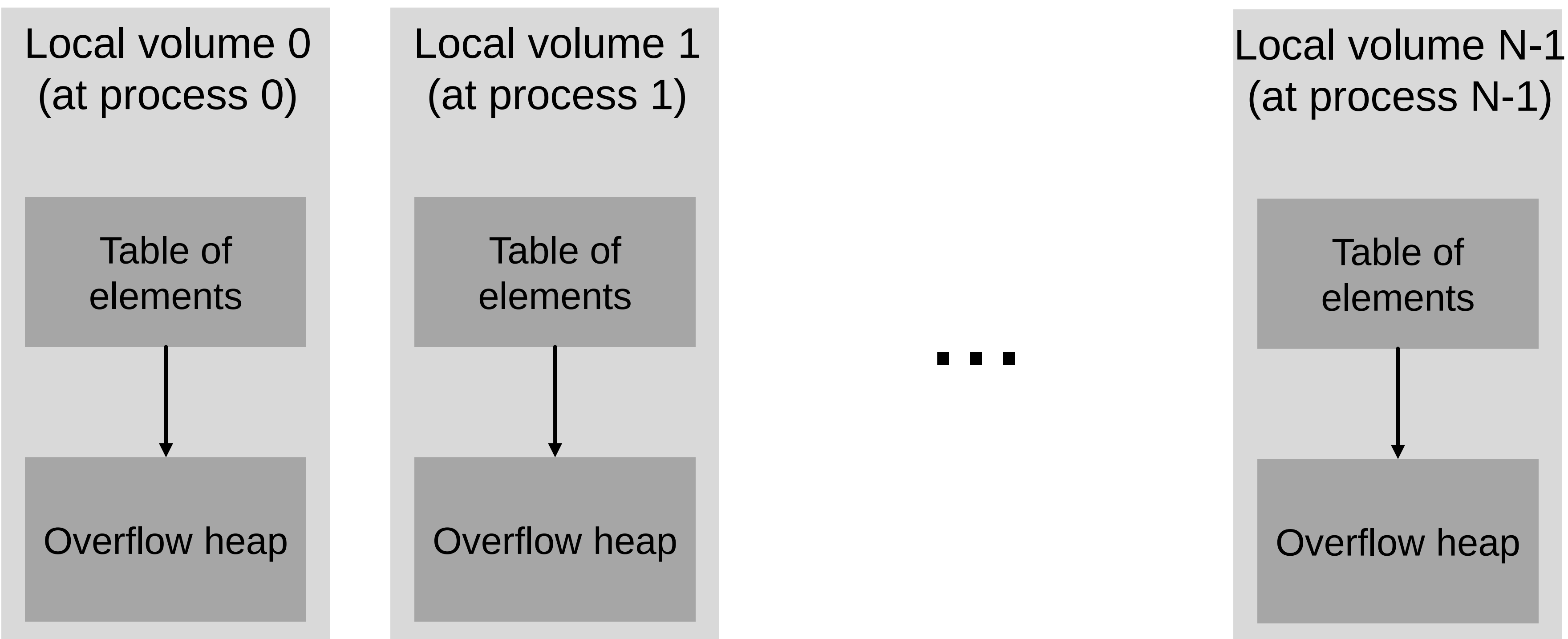


How can we use it?

Active Access Use-Cases

Distributed Hashtable

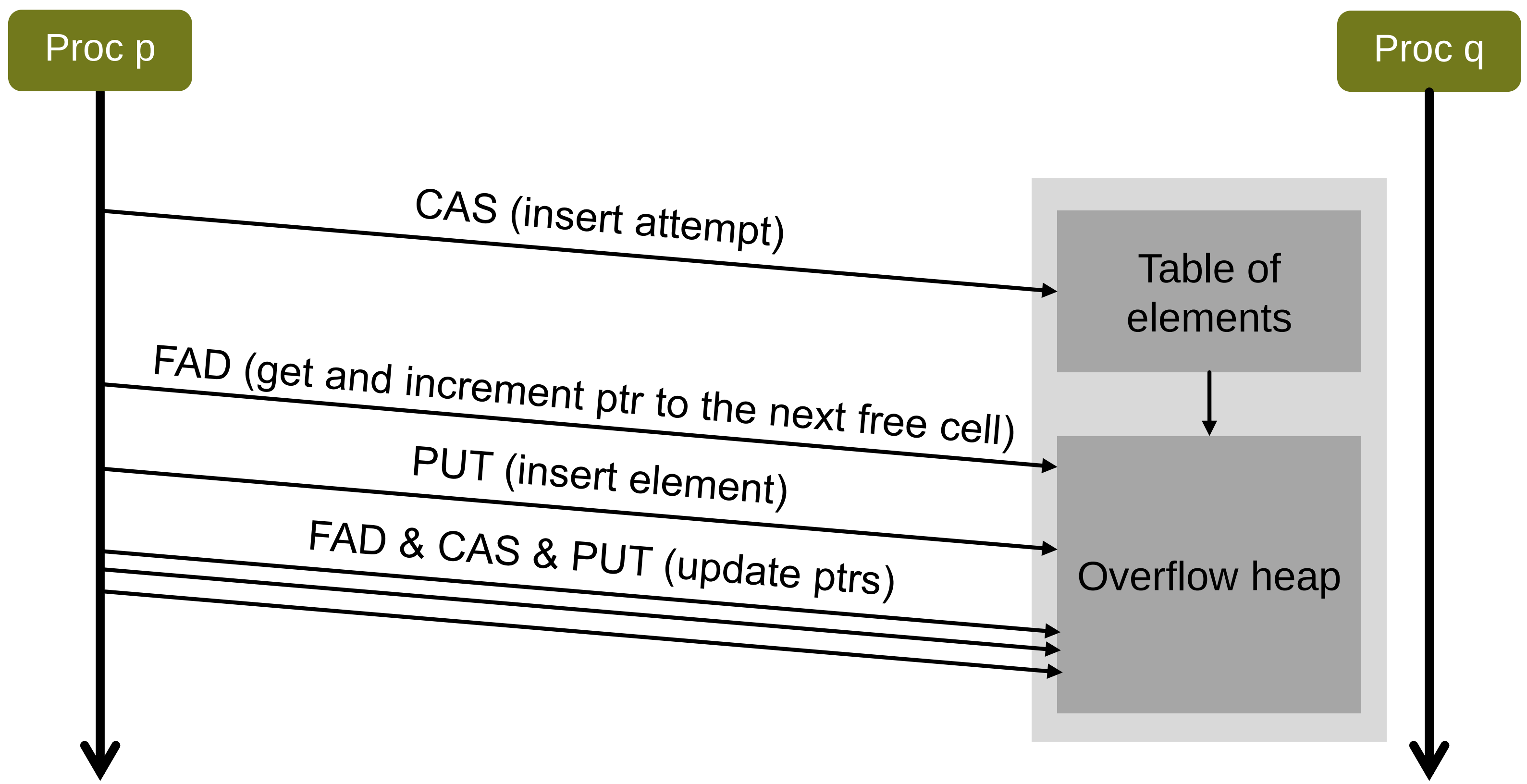
- Used to construct key-value stores (e.g., Memcached [1])



[1] B. Fitzpatrick. Distributed caching with memcached. Linux journal, 2004.

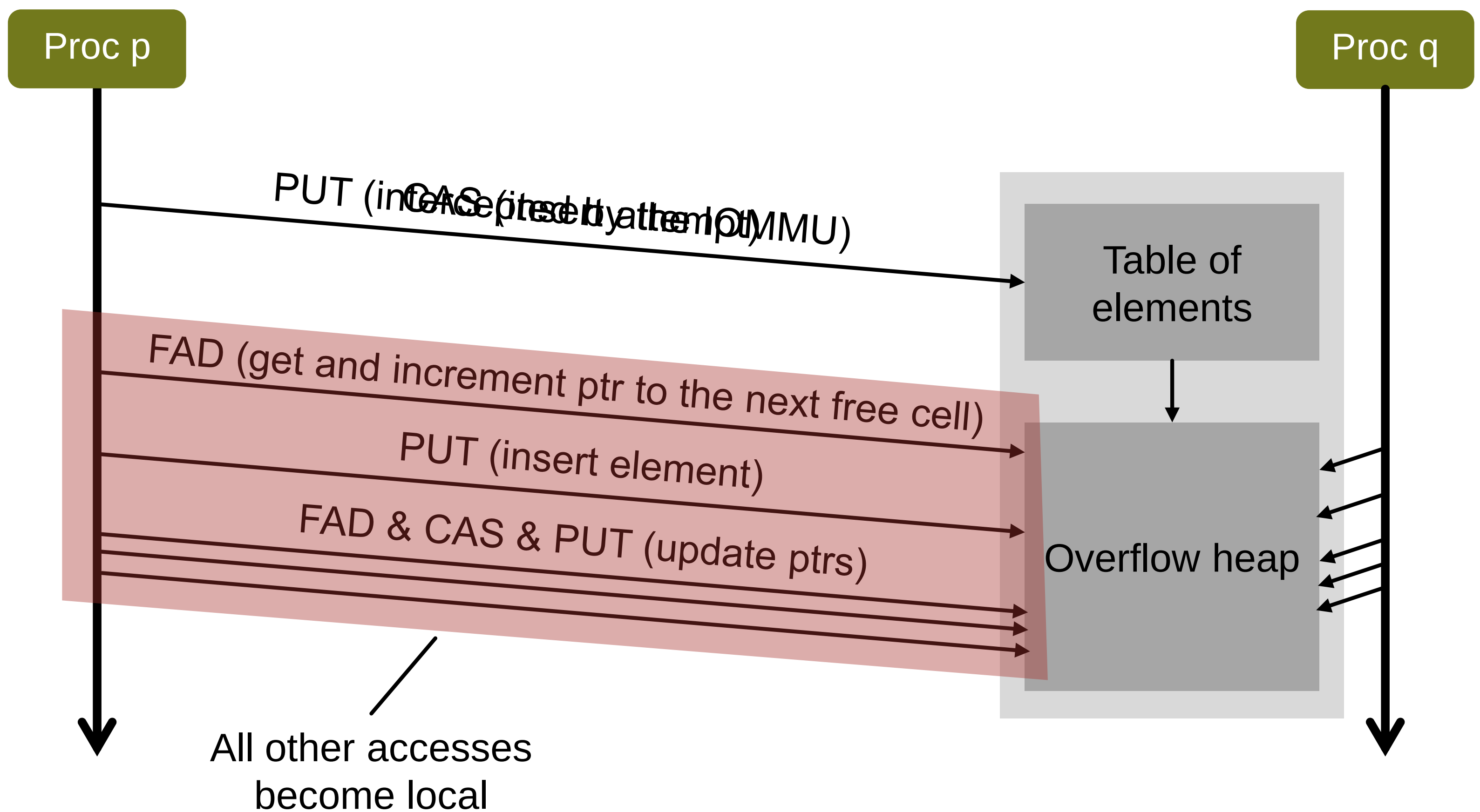
Active Access Use-Cases

Distributed Hashtable: Inserts (RMA)



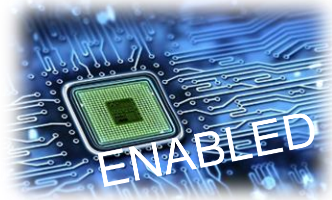
Active Access Use-Cases

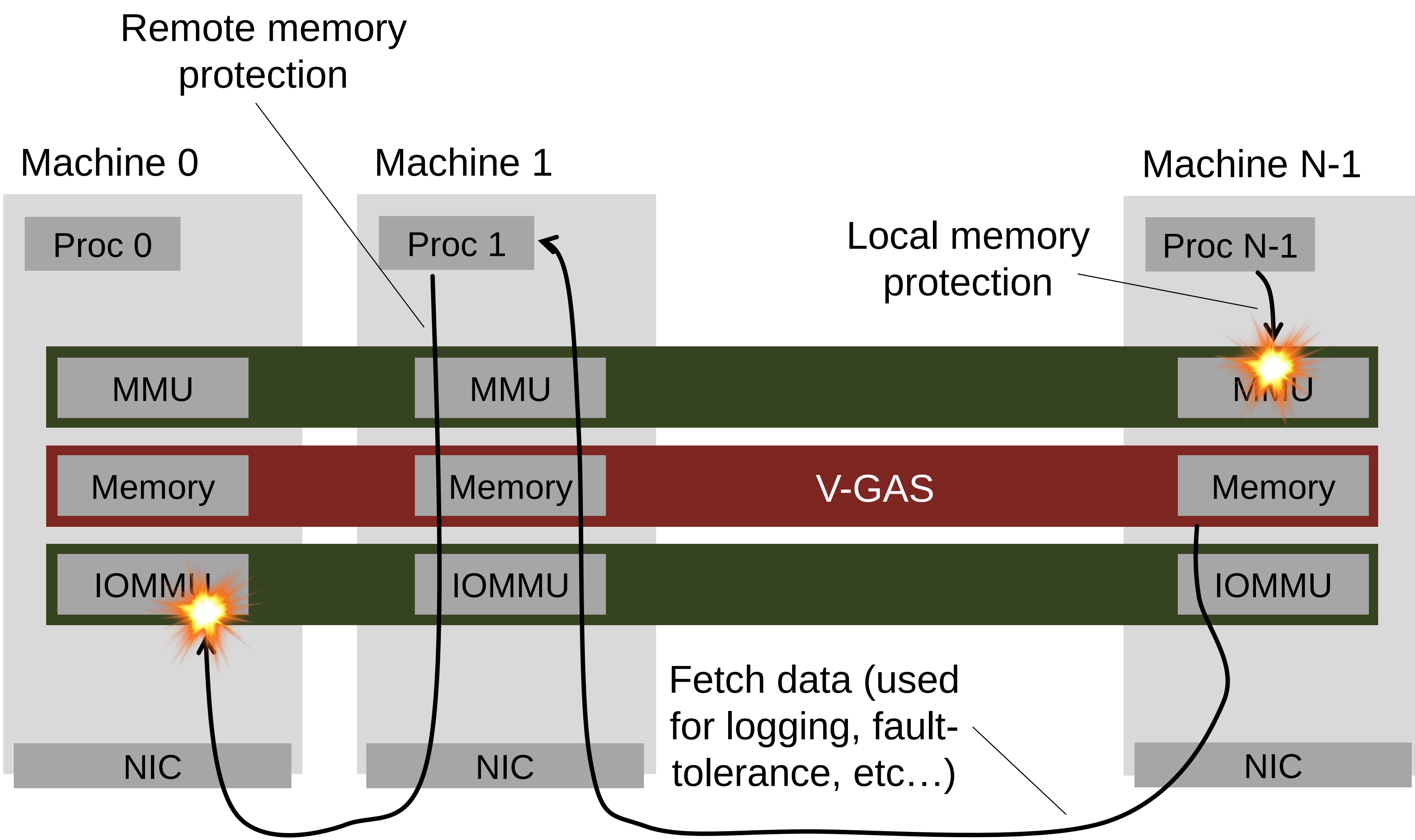
Distributed Hashtable: Inserts (AA)



Active Access Use-Cases

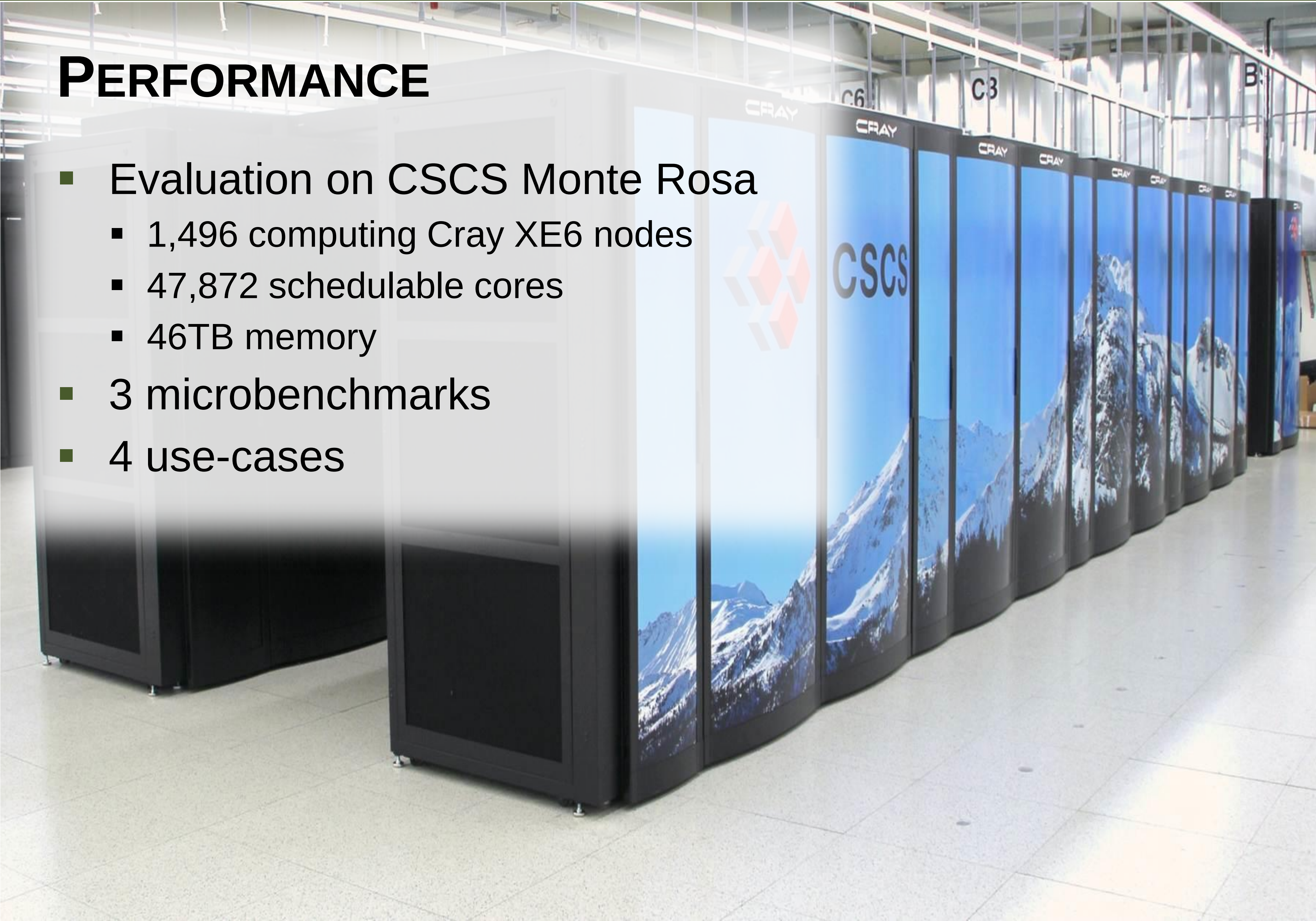
Virtual Global Address Space (V-GAS)





PERFORMANCE

- Evaluation on CSCS Monte Rosa
  - 1,496 computing Cray XE6 nodes
  - 47,872 schedulable cores
  - 46TB memory
- 3 microbenchmarks
- 4 use-cases

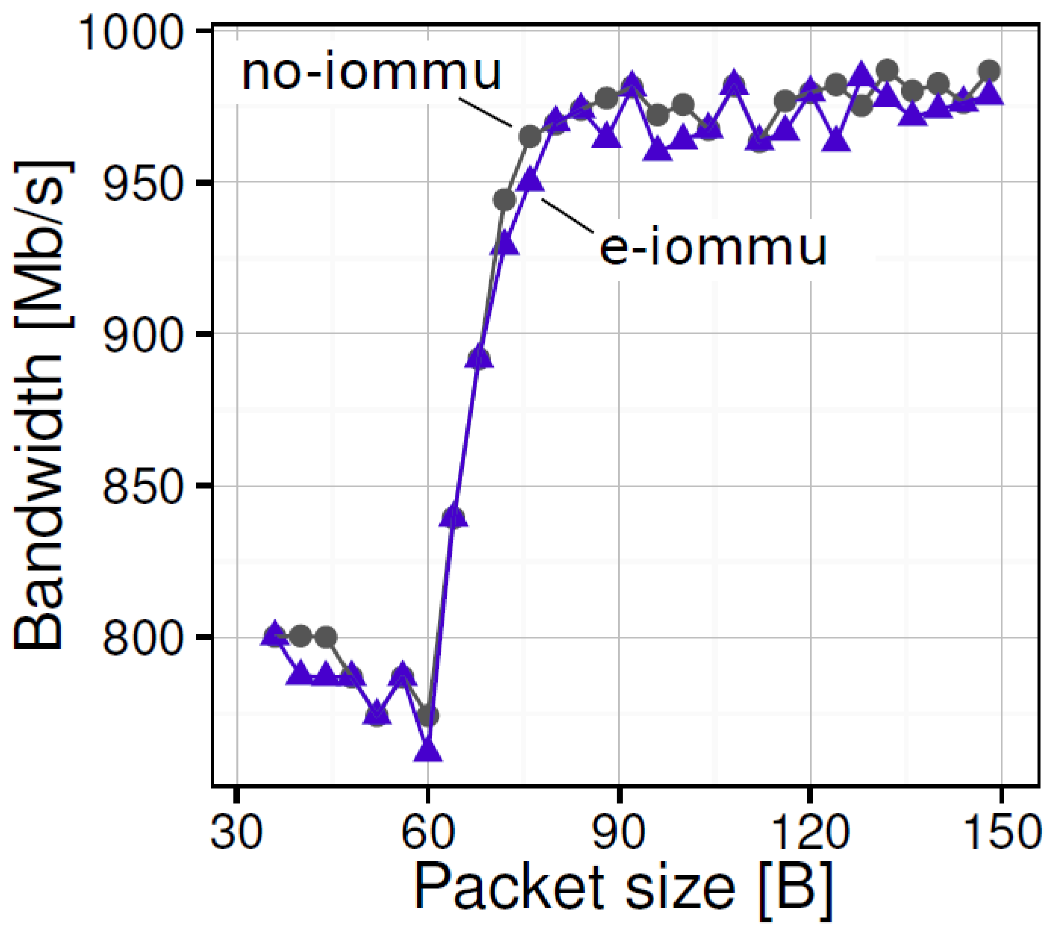


Performance: Microbenchmarks
Raw Data Transfer

- Workload simulated with [1]:

gem5

- Data generated with:
  - PktGen [2]
  - Netmap [3]



[1] N. Binkert et al. The gem5 simulator. SIGARCH Comput. Archit. News. 2011
[2] R. Olsson. PktGen the linux packet generator. Linux Symposium. 2005
[3] L. Rizzo. netmap: A novel framework for fast packet i/o. USENIX Annual Technical Conference. 2012

Performance: Large-Scale Codes

Comparison Targets

Active Access

AA-Poll
AA-Int
AA-SP

RMA

IBM Cell
Cray DMAPP
InfiniBand
Mellanox Technologies RoCE

Active Messages

AM AM-Onload
AM-Exp AM-Ints

IBM DCMF LAPI PAMI

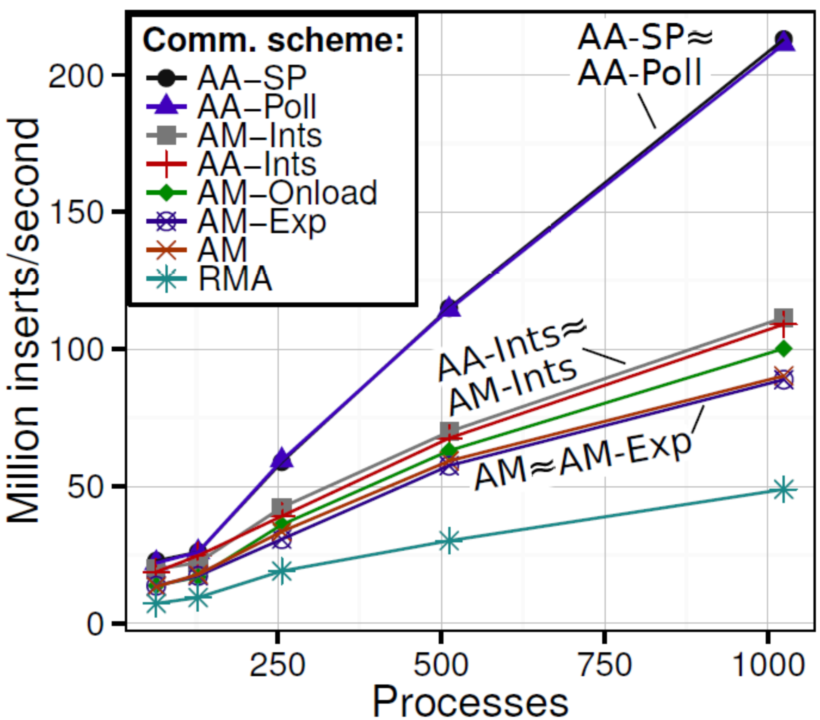
Myricom MX

AM++ GASNet

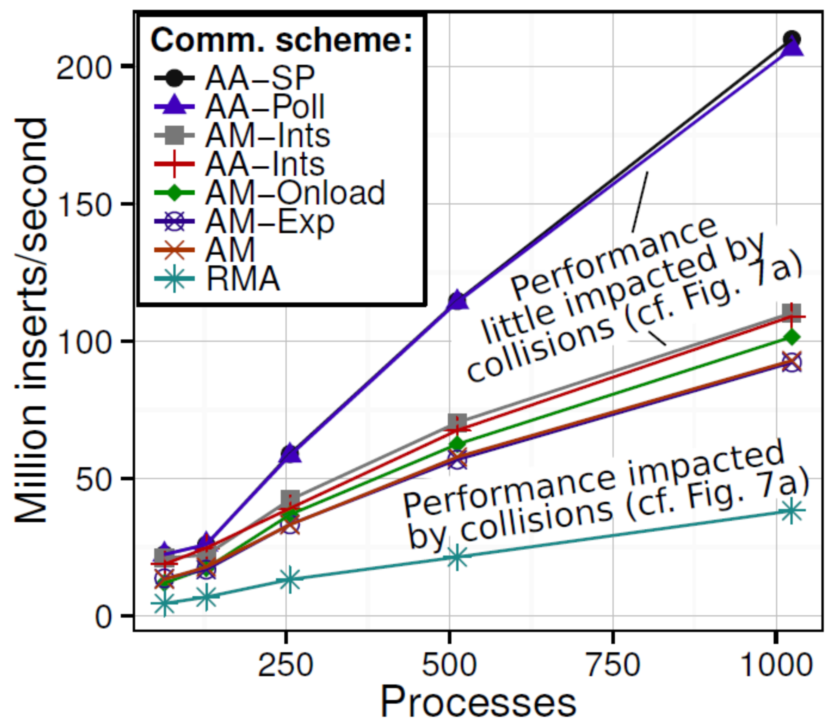
Performance: Large-Scale Codes

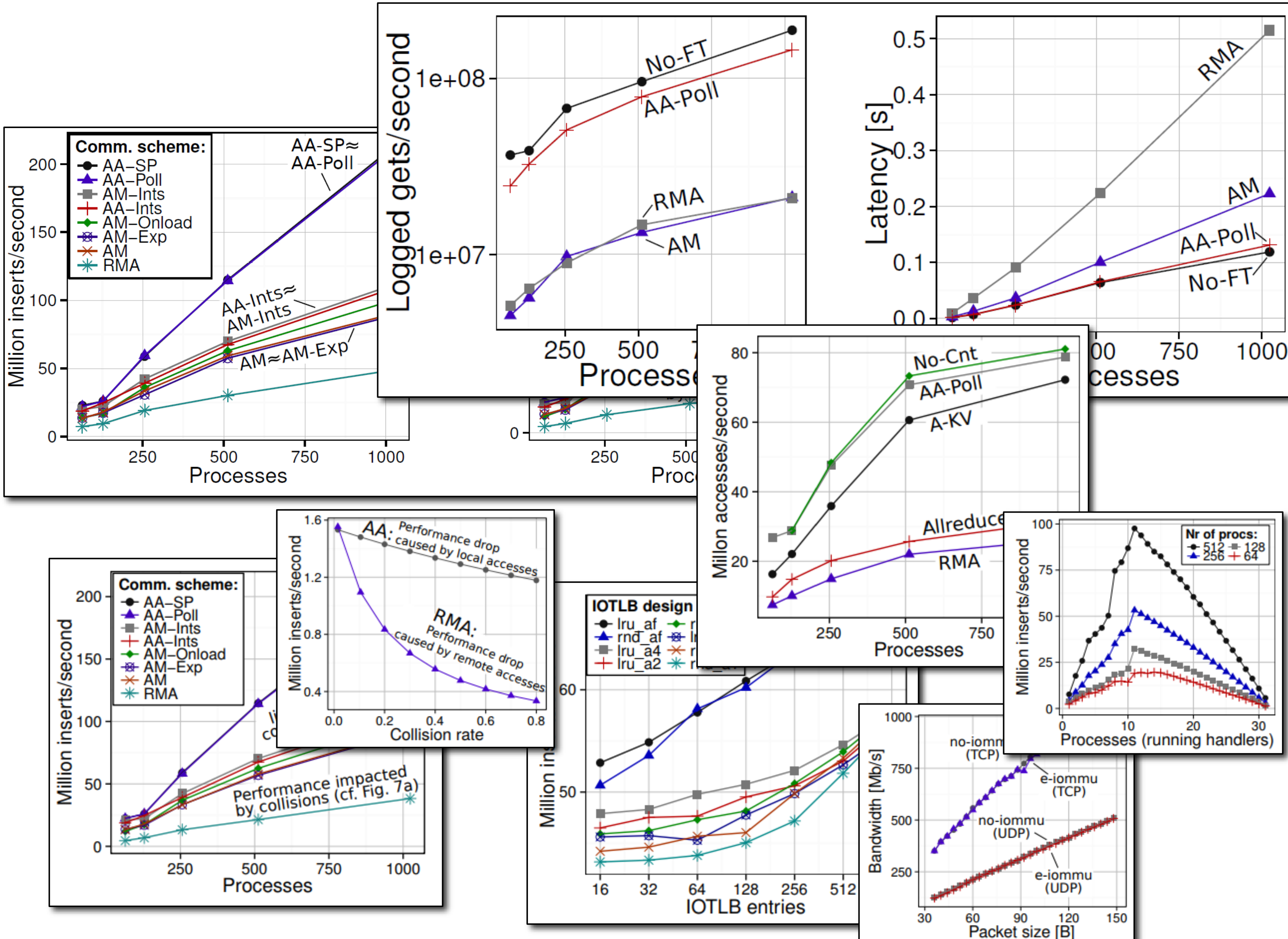
Distributed Hashtable

Collisions: 5%



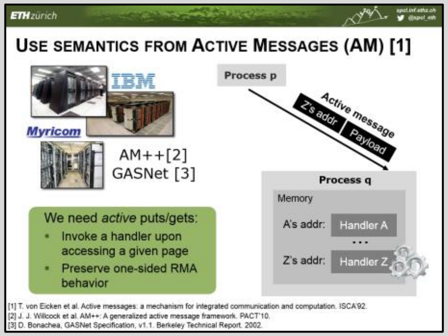
Collisions: 25%




Comm. scheme:
AA–SP
AA–Poll
AM–Ints
AA–Ints
AM–Onload
AM–Exp
AM
RMA
AA-SP≈
AA-Poll
AA-Ints≈
AM-Ints
AM≈AM-Exp
Million inserts/second
Processes
Logged gets/second
No-FT
AA-Poll
RMA
AM
1e+08
1e+07
Latency [s]
Millon accesses/second
No-Cnt
A-KV
Allreduce
Million inserts/second
Collision rate
AA: Performance drop caused by local accesses
RMA: Performance drop caused by remote accesses
Performance impacted by collisions (cf. Fig. 7a)
IOTLB design
lru_af
rnd_af
lru_a4
lru_a2
IOTLB entries
Bandwidth [Mb/s]
Packet size [B]
no-iommu (TCP)
e-iommu (TCP)
no-iommu (UDP)
e-iommu (UDP)
Nr of procs:
512
128
256
64
Processes (running handlers)

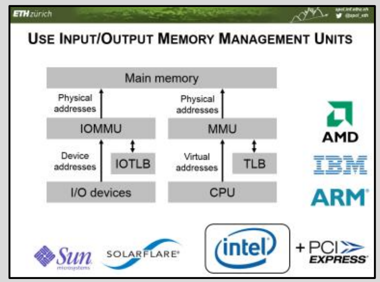
Conclusions

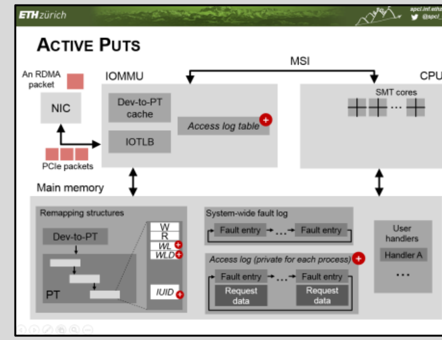
Active Access



Alleviates RMA's problems with AMs while preserving one-sided semantics

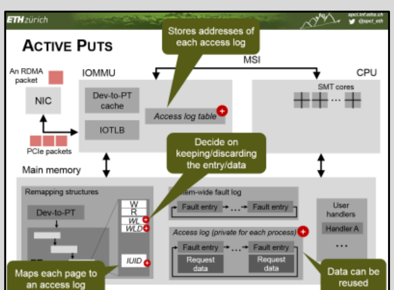
Uses commodity & common IOMMUs



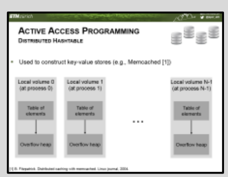


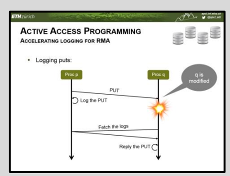
Extends paging capabilities in a distributed environment

Data-centric programming



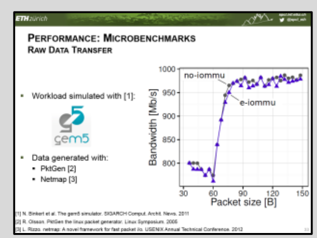
Addresses of pages guide the execution of handlers

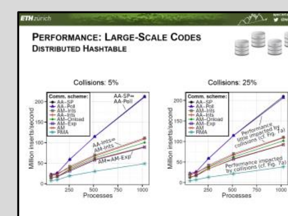




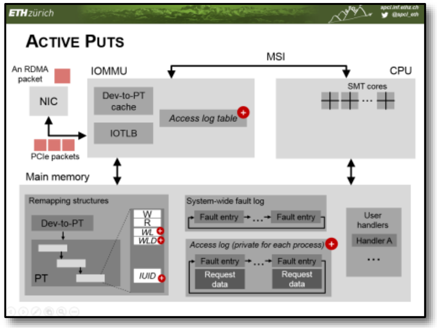
Hashtables, logging schemes, counters, V-GAS, checkpointing...

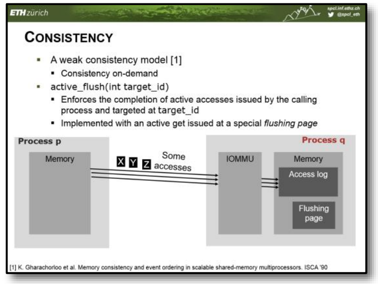
Performance

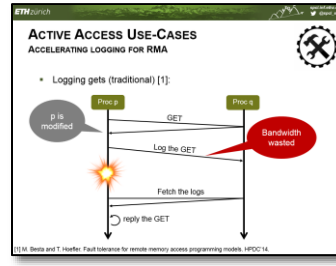


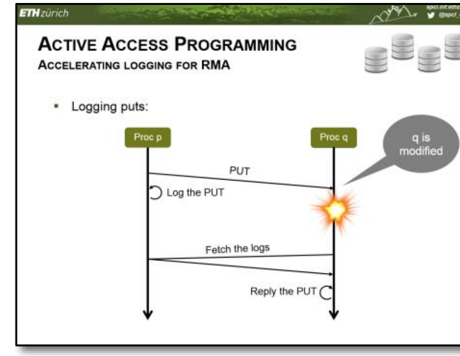


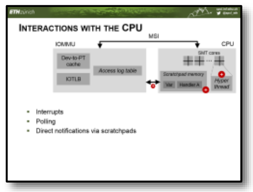
Accelerates various distributed codes

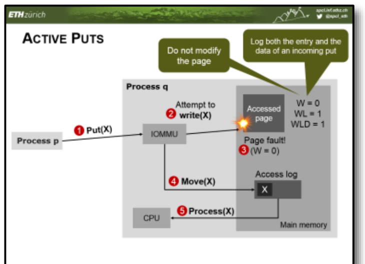


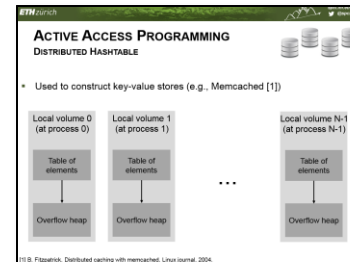




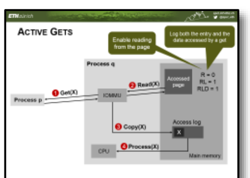


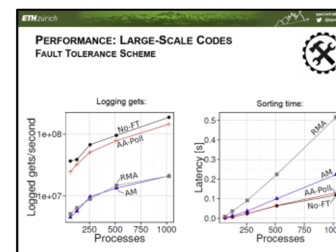


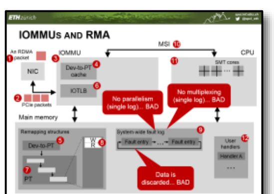


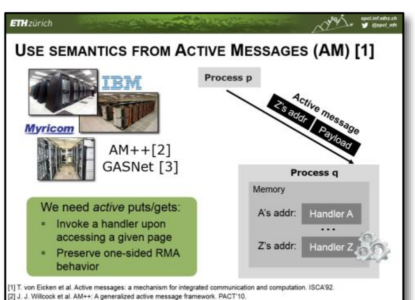


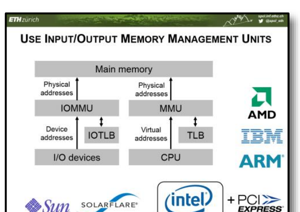
Thank you for your attention

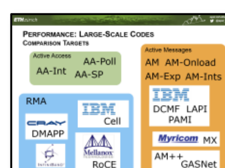


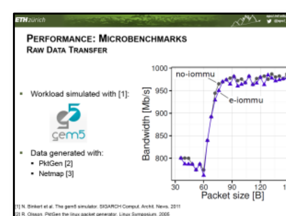


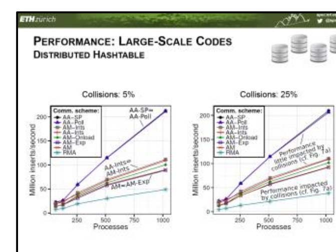












Active Access Use-Cases

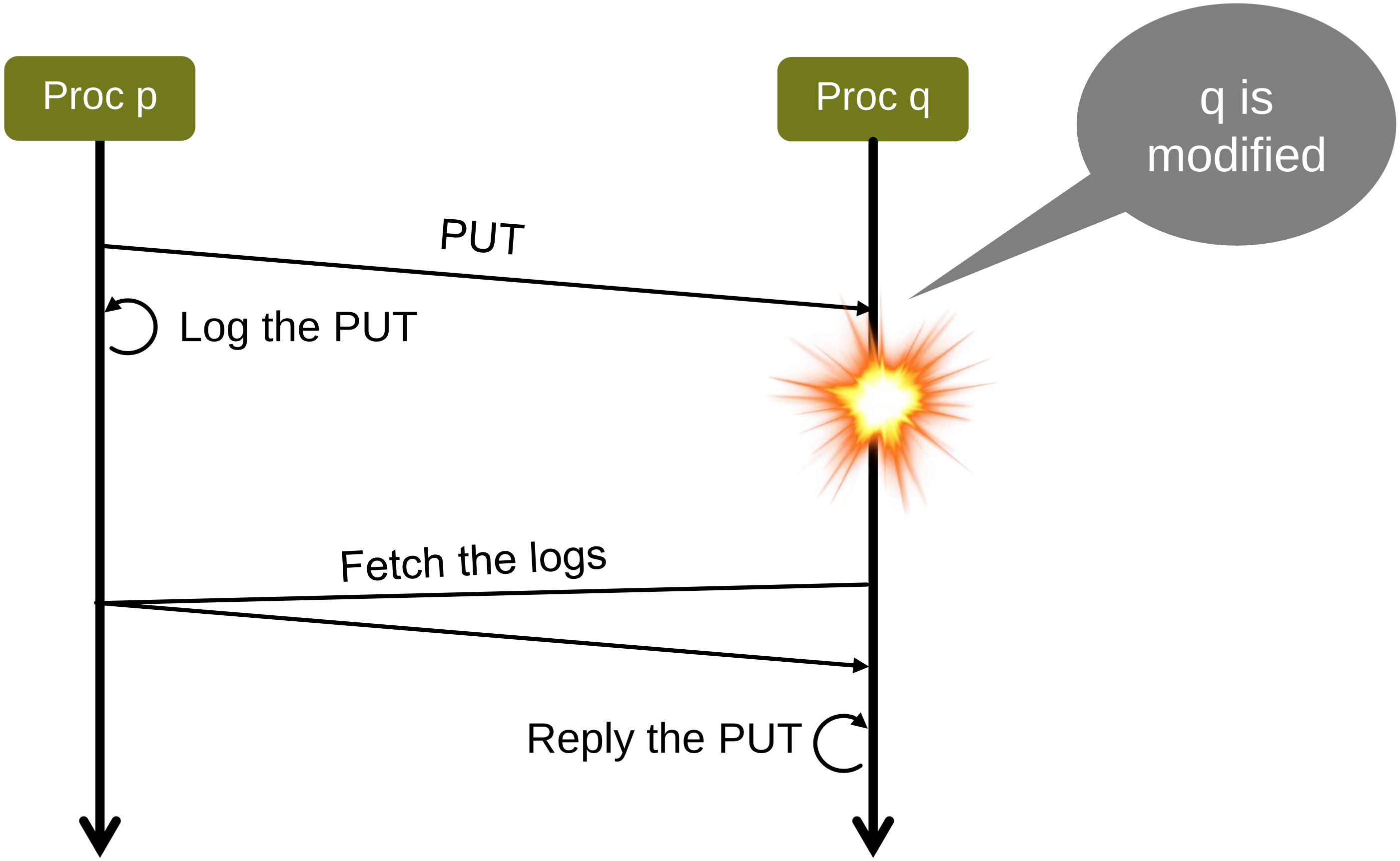
Accelerating logging for RMA

- Logging – a popular mechanism for fault-tolerance.
- Remote communication (puts/gets) is logged.
- Upon a process crash, it is restored and uses the logs to replay its previous actions.
- Logs are stored in volatile memories.

ACTIVE ACCESS USE-CASES

ACCELERATING LOGGING FOR RMA

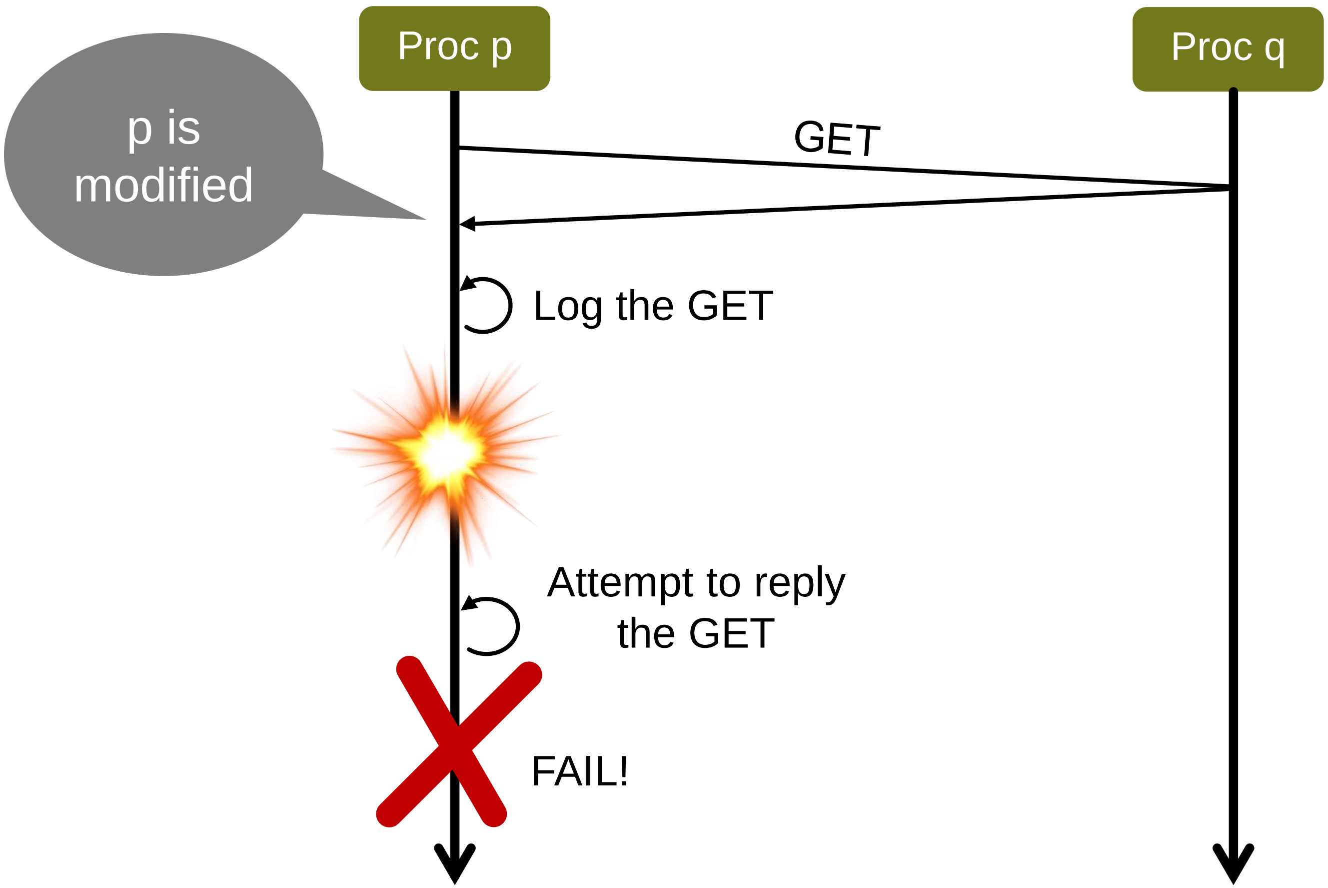
- Logging puts:



Active Access Use-Cases

Accelerating logging for RMA

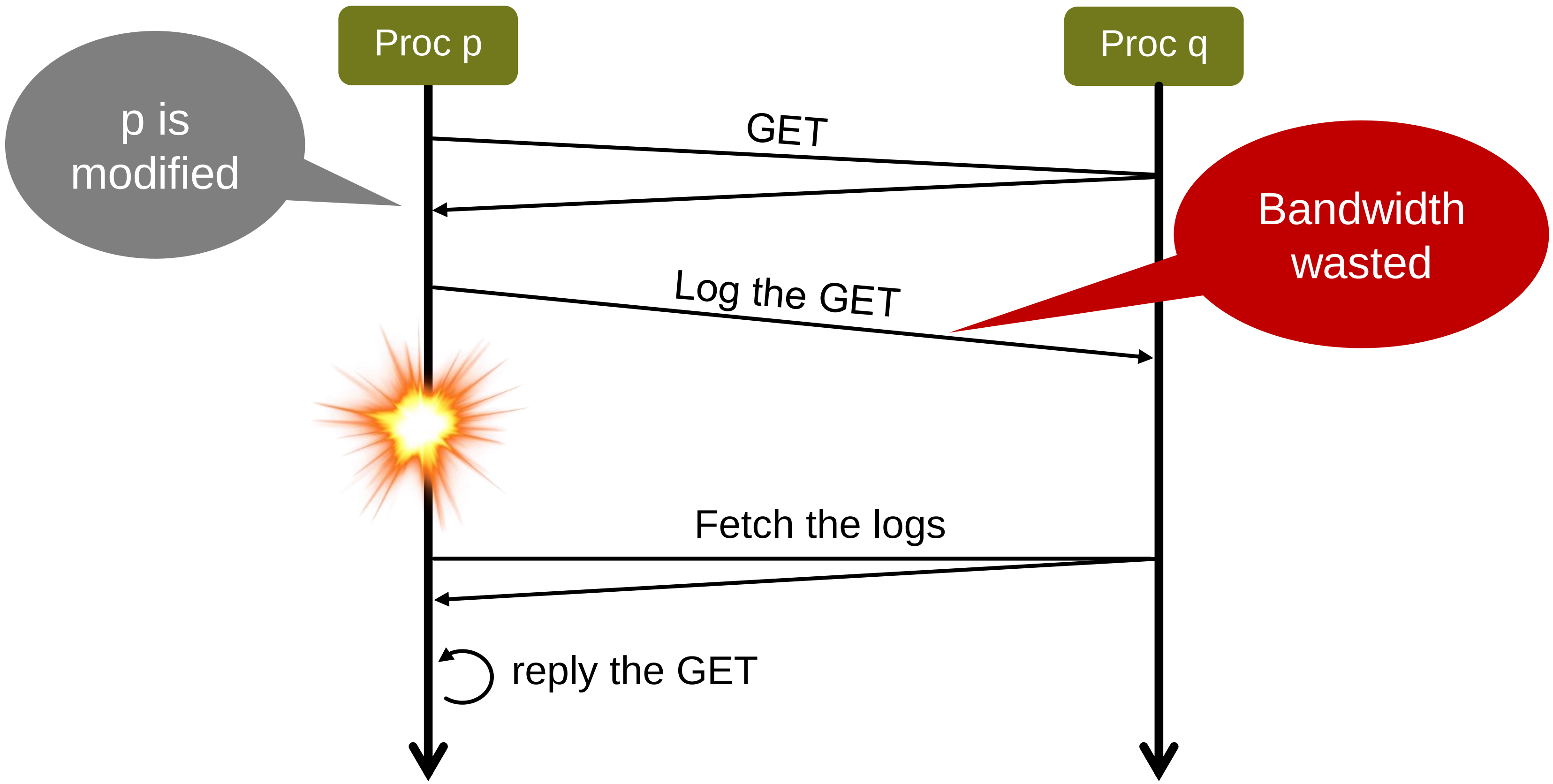
- Logging gets (naive):



Active Access Use-Cases

Accelerating logging for RMA

- Logging gets (traditional) [1]:

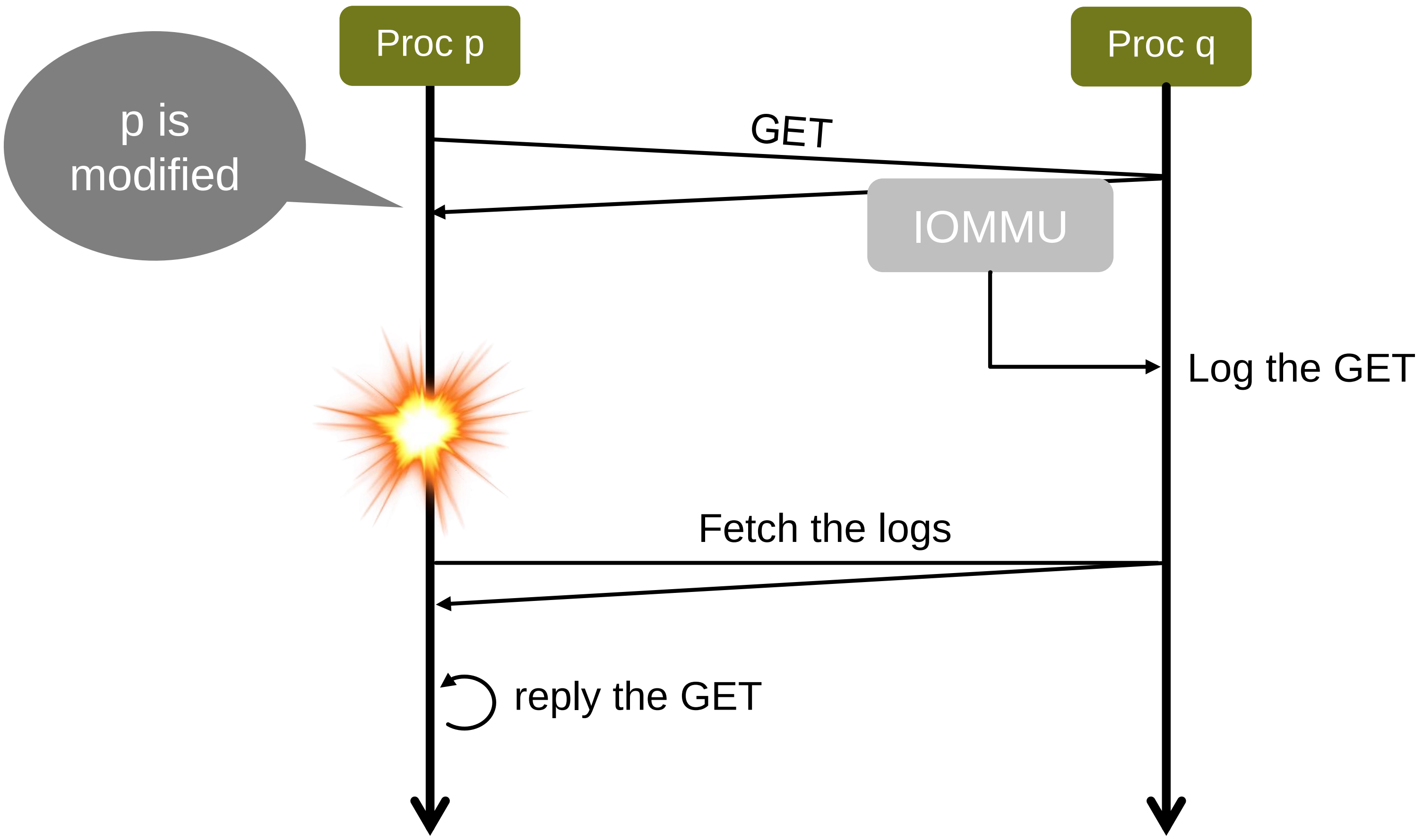


[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

Active Access Use-Cases

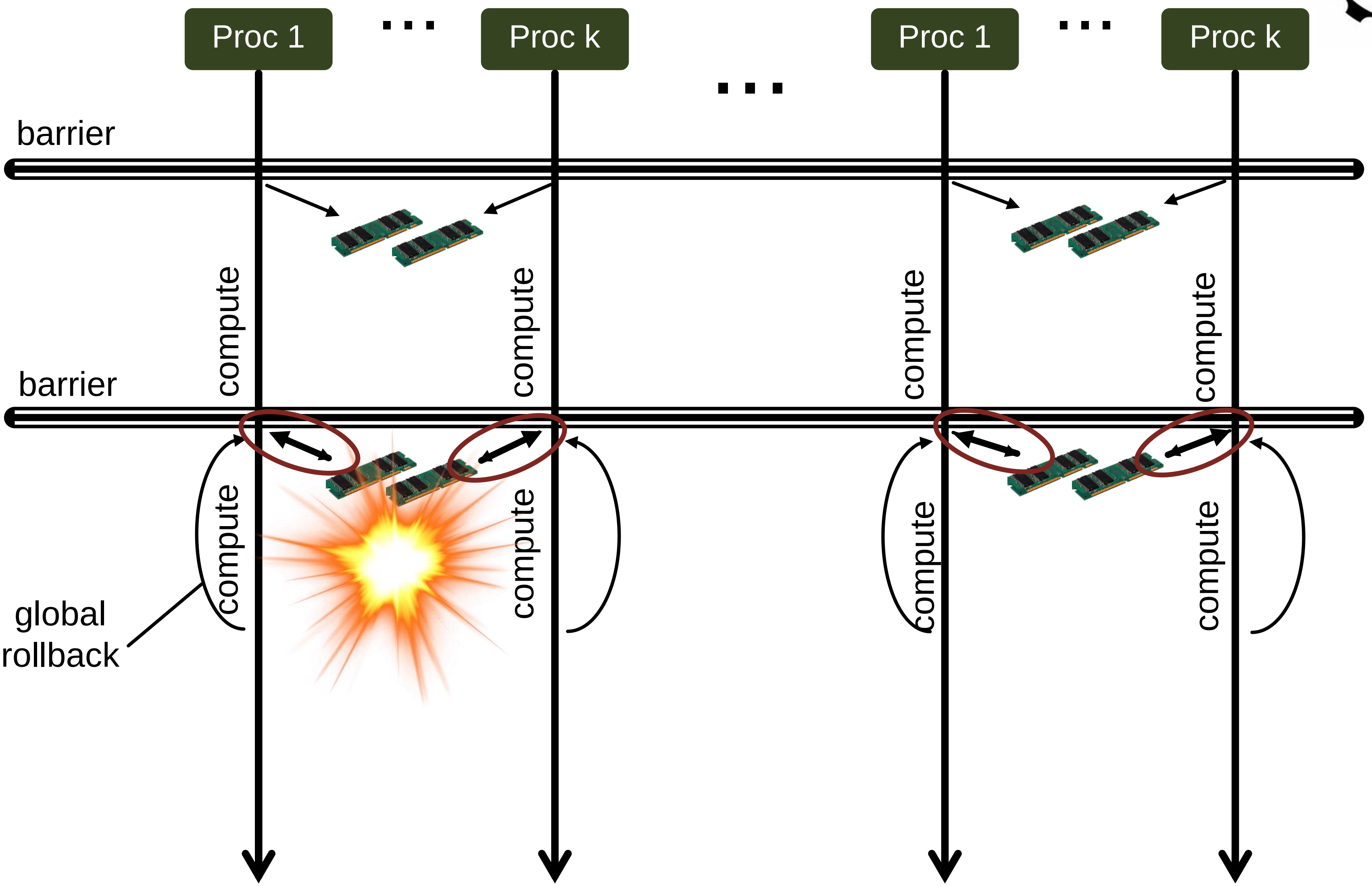
Accelerating logging for RMA

- Logging gets (AA):

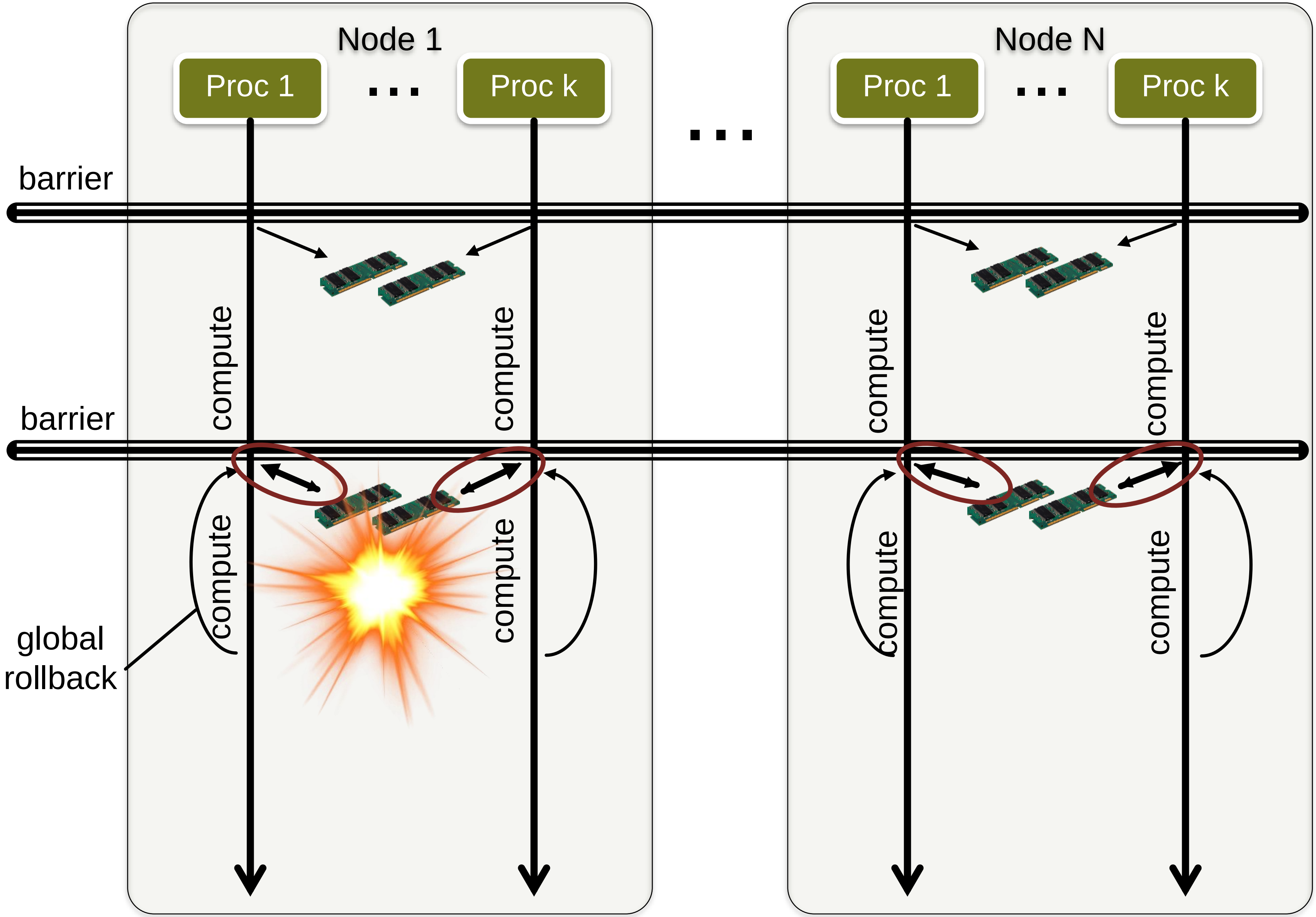


Active Access Use-Cases

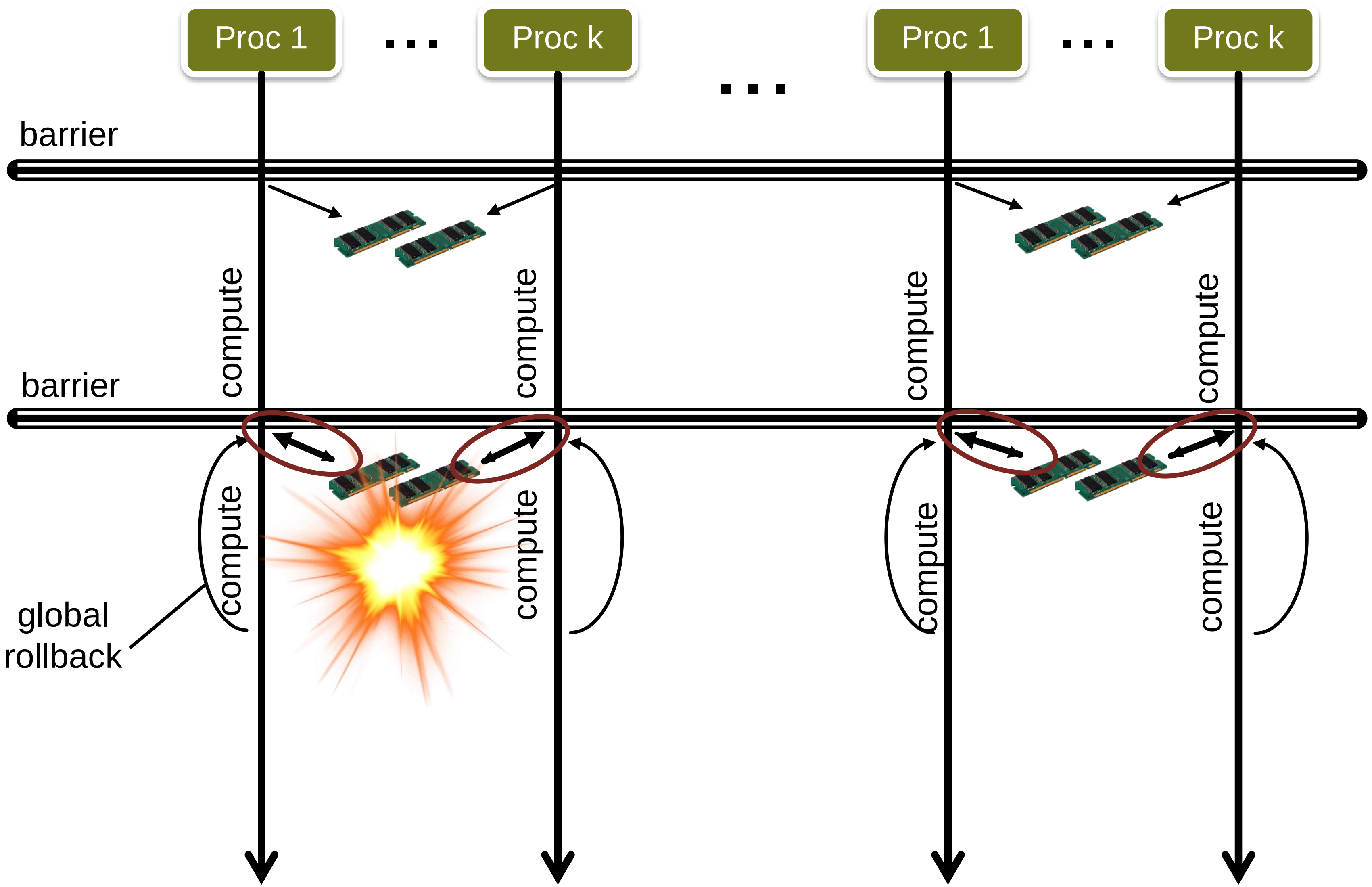
Incremental checkpointing for RMA



COORDINATED CHECKPOINTING (MP)



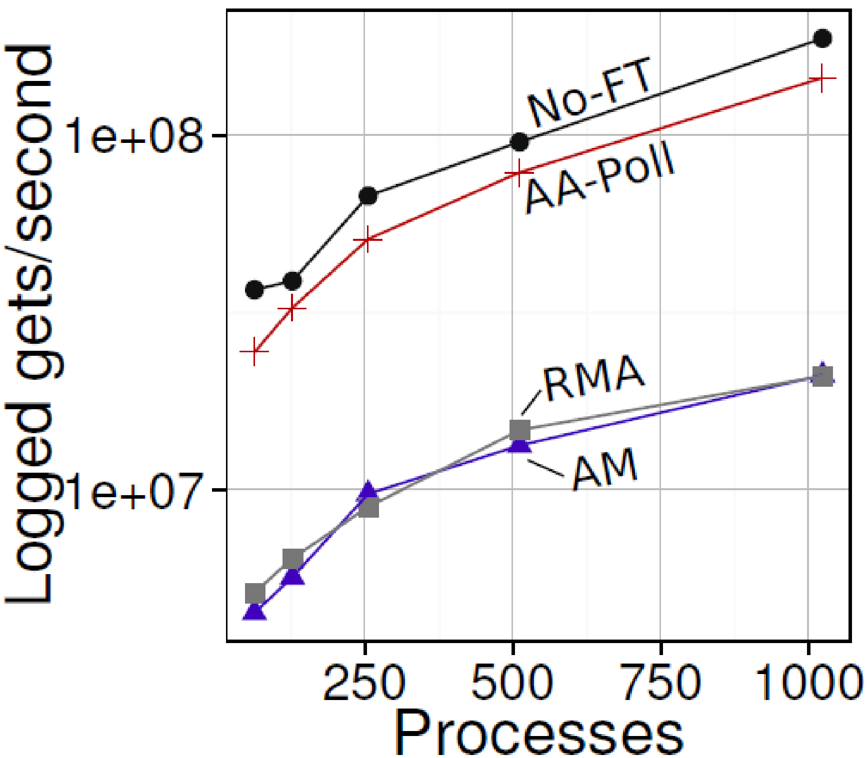
Coordinated Checkpointing (MP)



Performance: Large-Scale Codes

Fault Tolerance Scheme

Logging gets:



Sorting time: