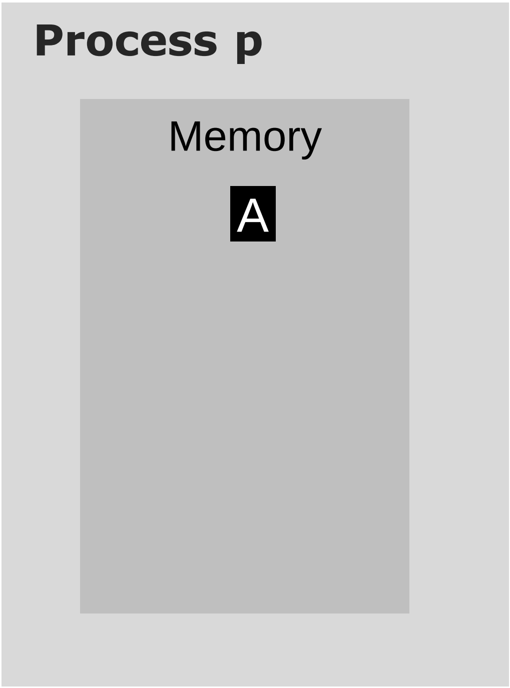
# Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations

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**

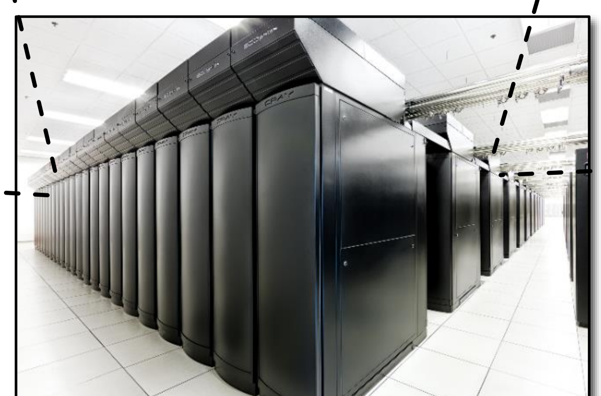
Memory

A

**Process q**

Memory

B



Cray
BlueWaters

# Remote Memory Access (RMA) Programming

Process p

Memory

A

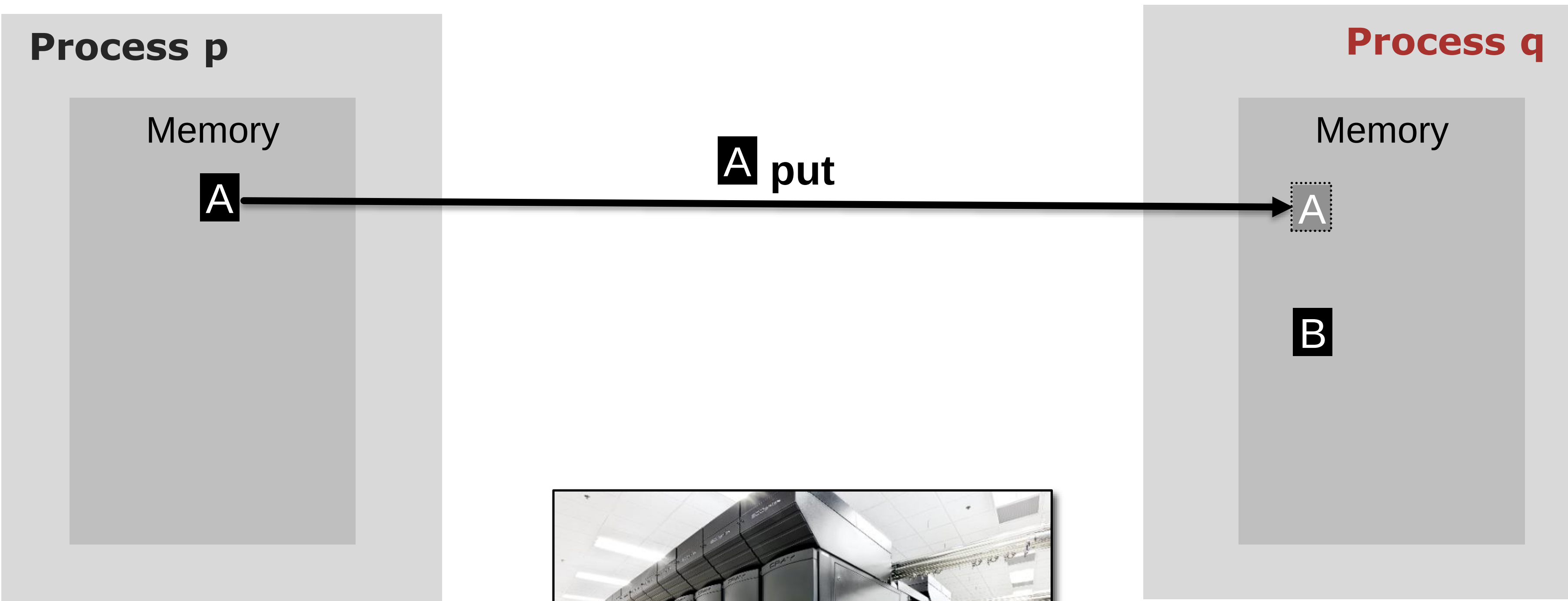
Process q

Memory

B

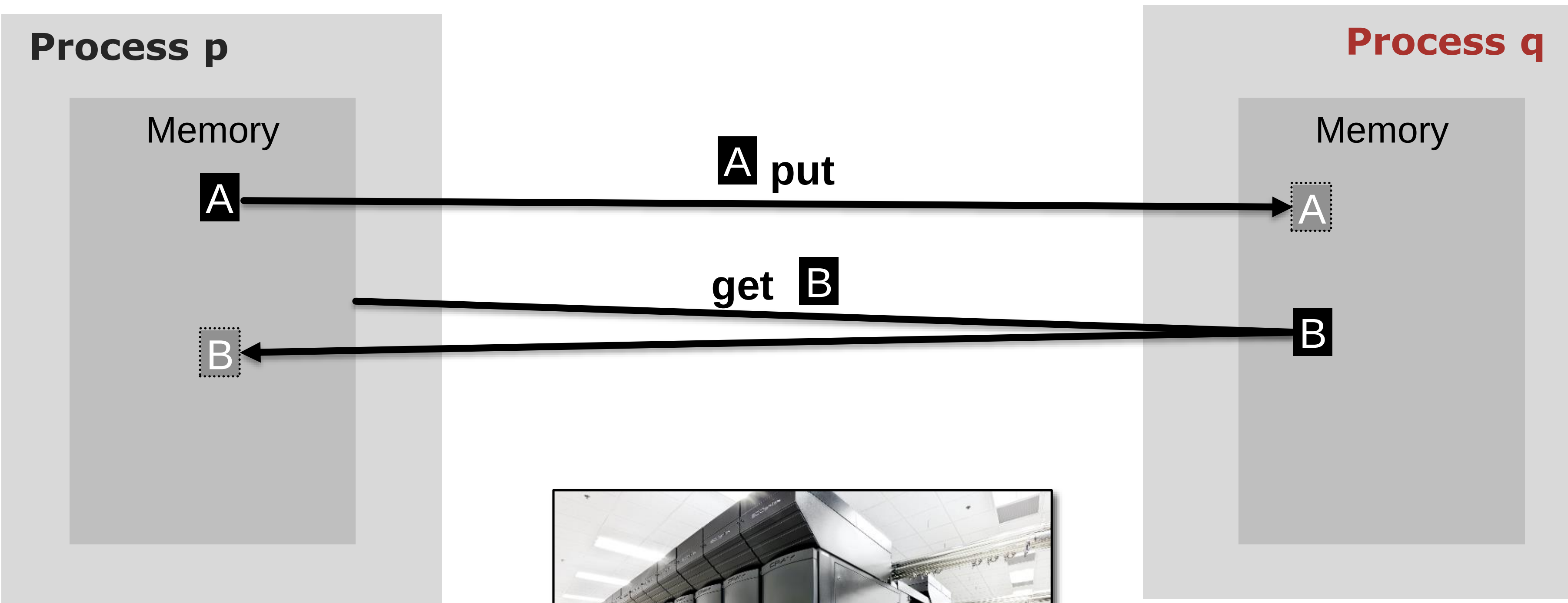
Cray BlueWaters

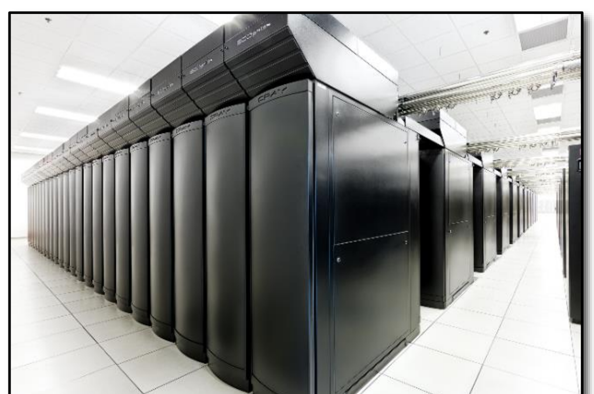
# Remote Memory Access (RMA) Programming




Cray
BlueWaters

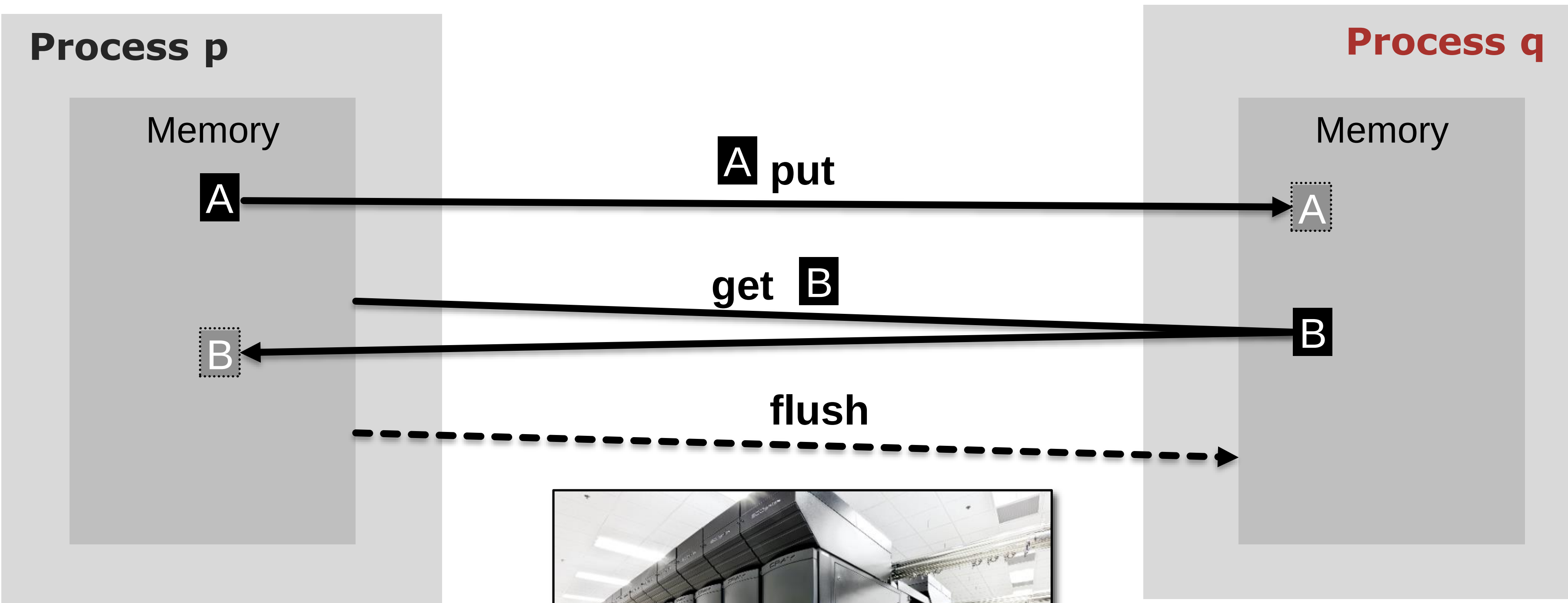
# Remote Memory Access (RMA) Programming

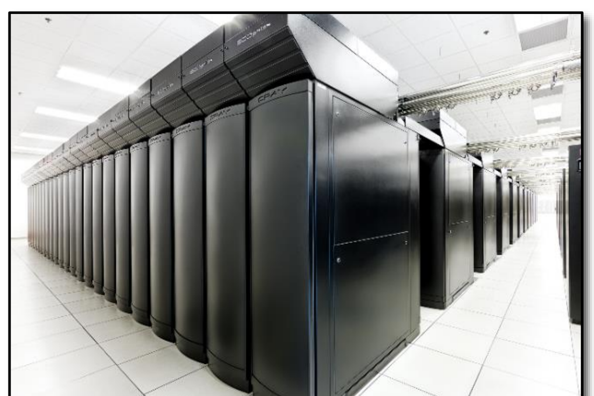


Cray
BlueWaters

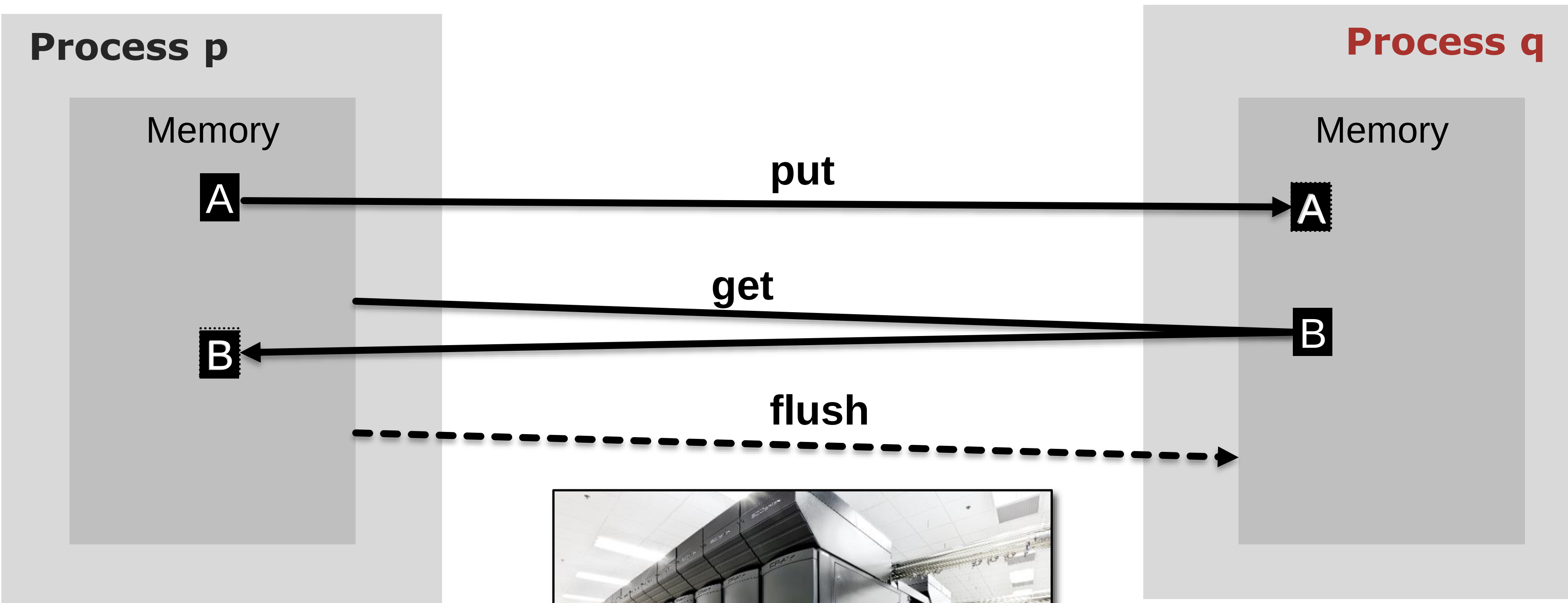
# Remote Memory Access (RMA) Programming



Cray
BlueWaters

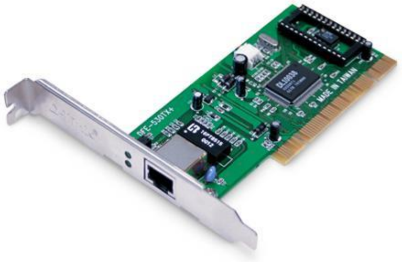
# Remote Memory Access (RMA) Programming




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

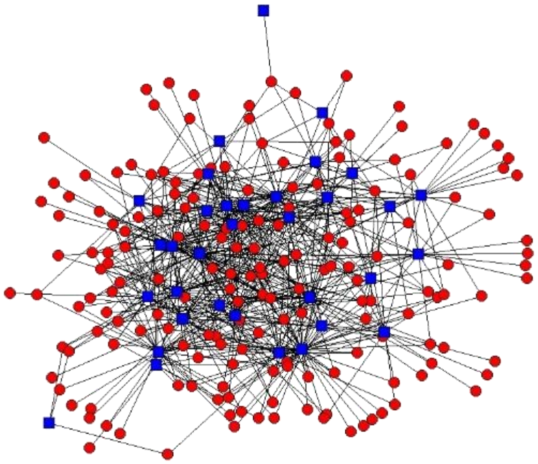
# Remote Memory Access Programming

- 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

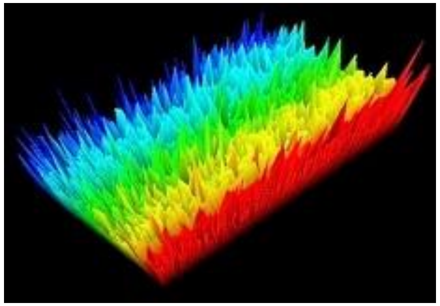
- Enables significant speedups over message passing in many types of applications, e.g.:
  - Speedup of ~1.5 for communication patterns in irregular workloads

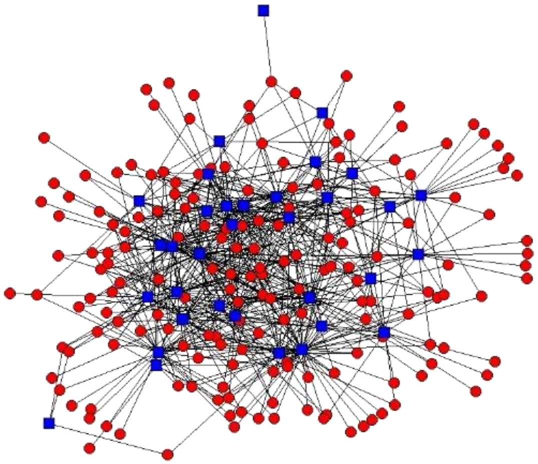


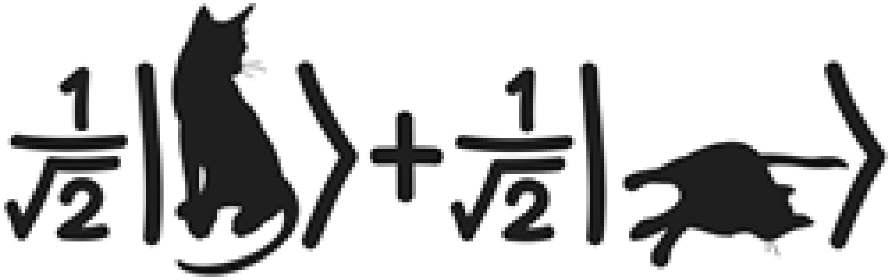
[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

# 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



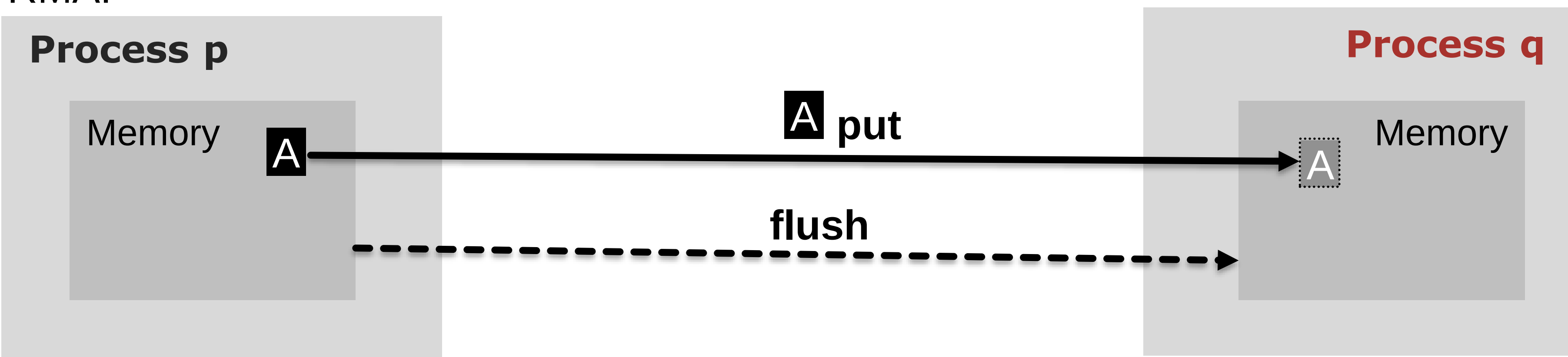




[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

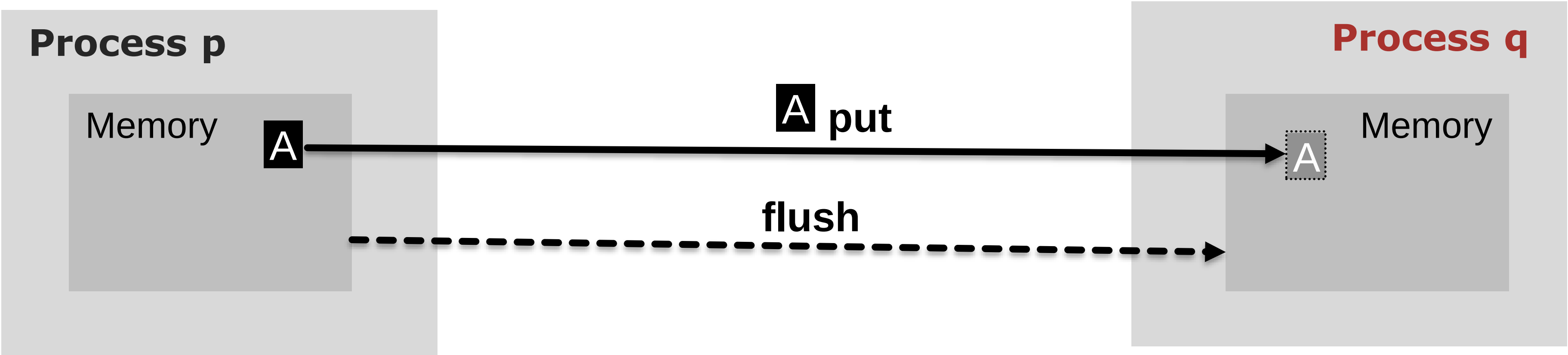
# RMA vs. Message Passing

RMA:



# RMA vs. Message Passing

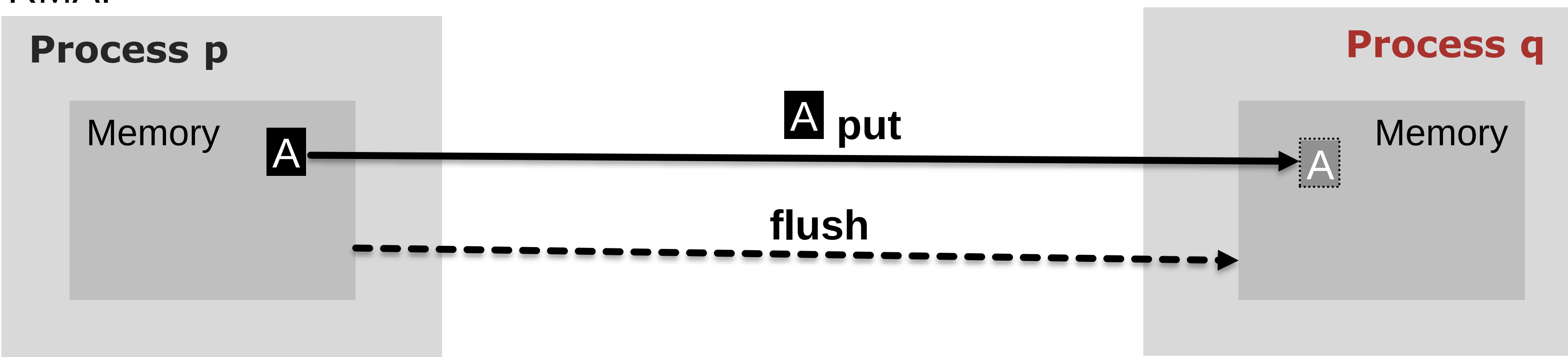
RMA:



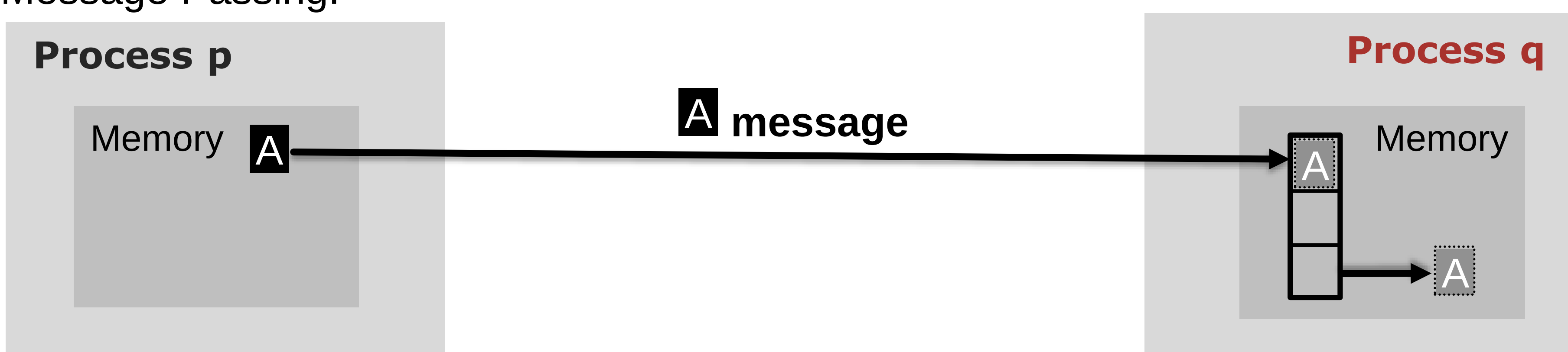

Message Passing:

# RMA vs. Message Passing

RMA:



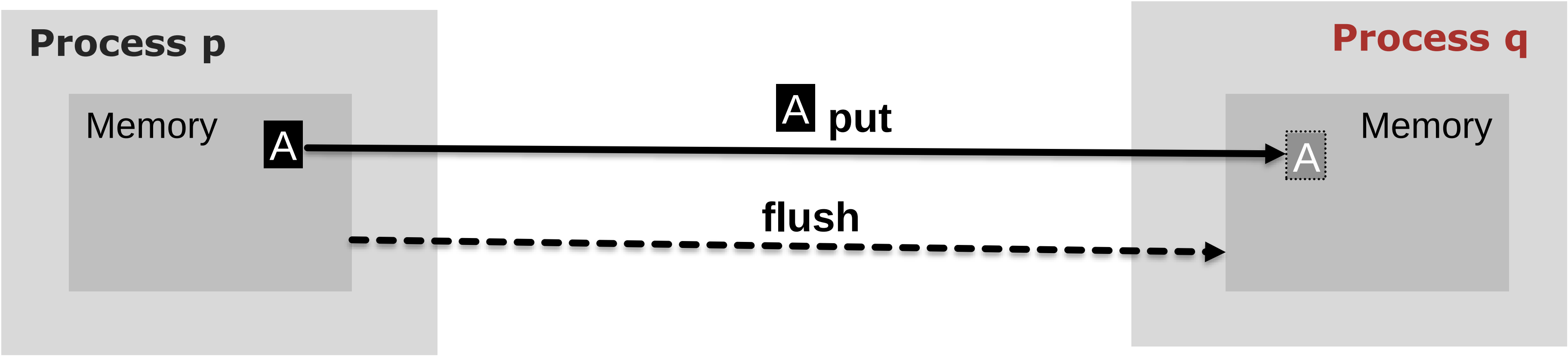

Message Passing:



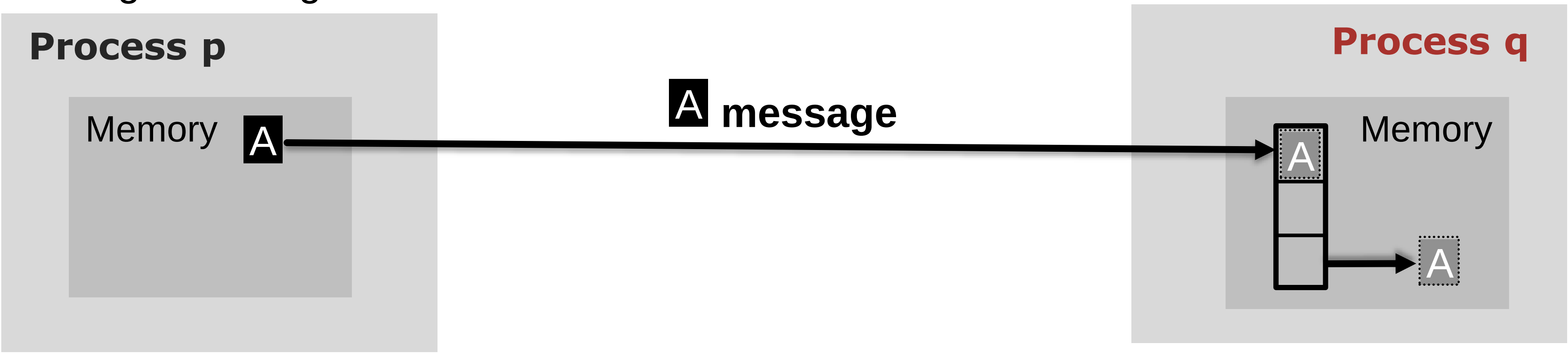
# RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:



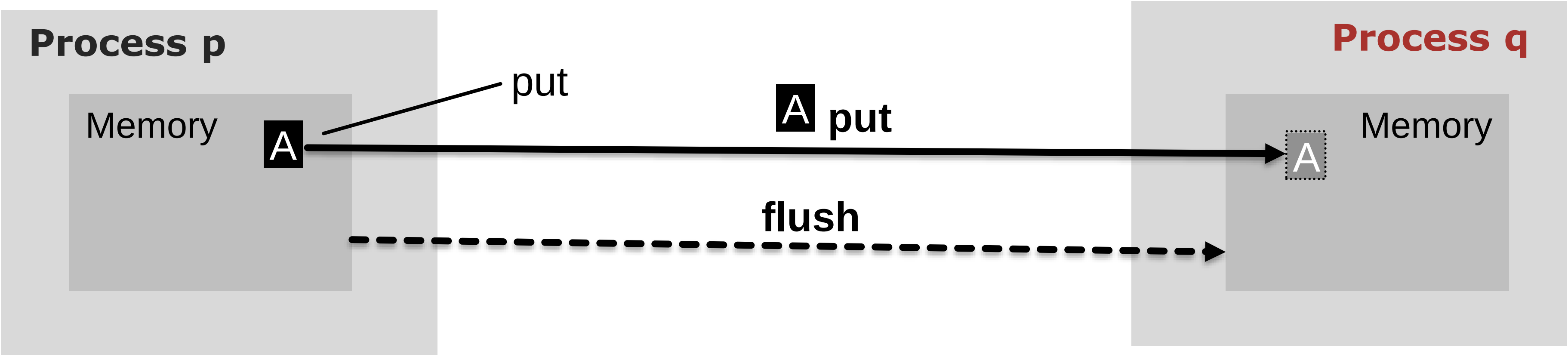

Message Passing:



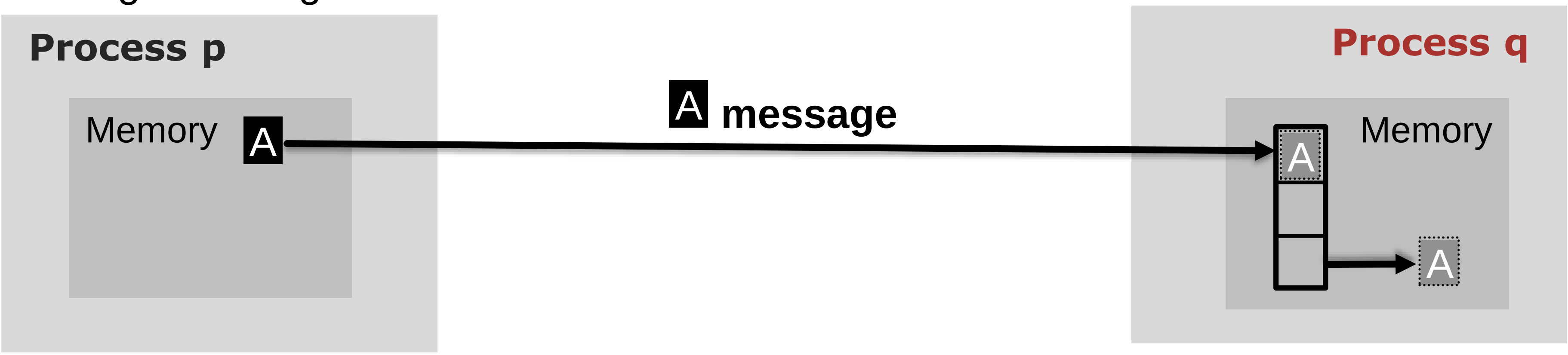
# RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:



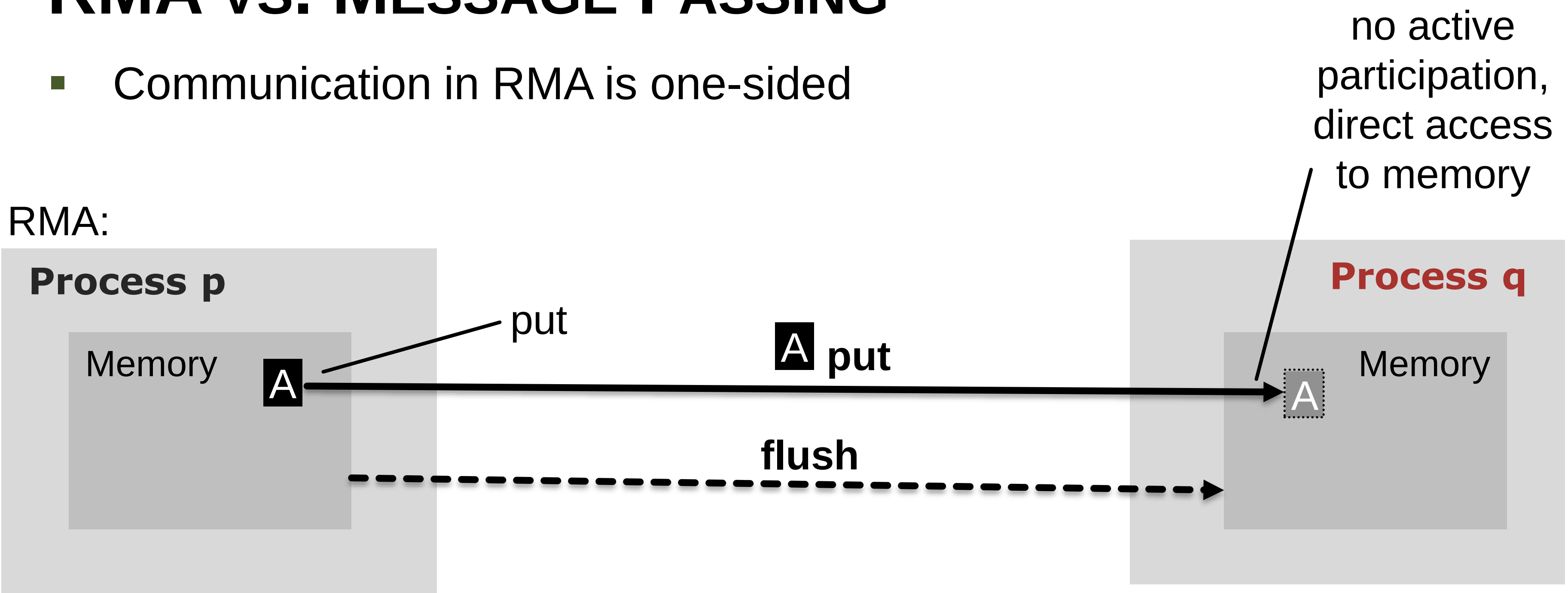

Message Passing:



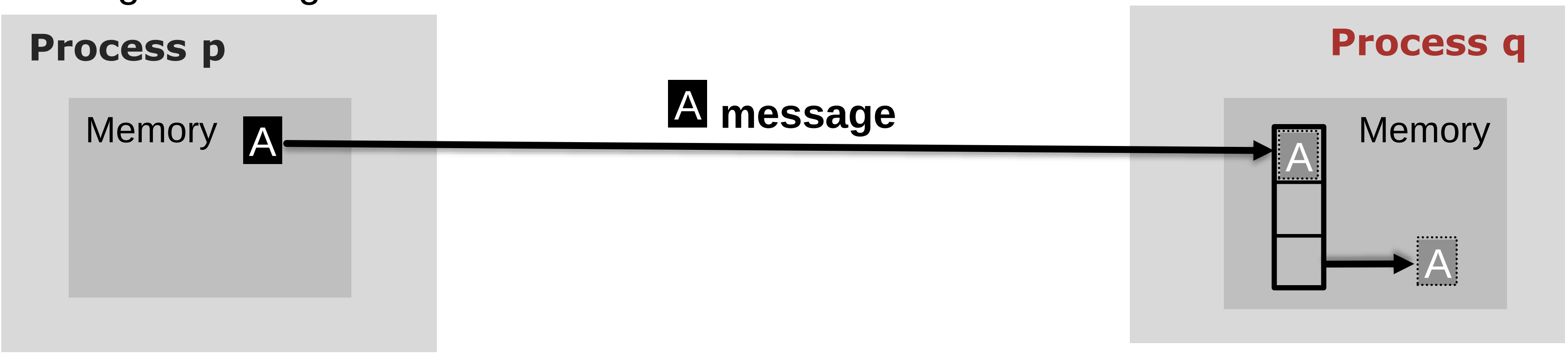
# RMA VS. MESSAGE PASSING

- Communication in RMA is one-sided

RMA:



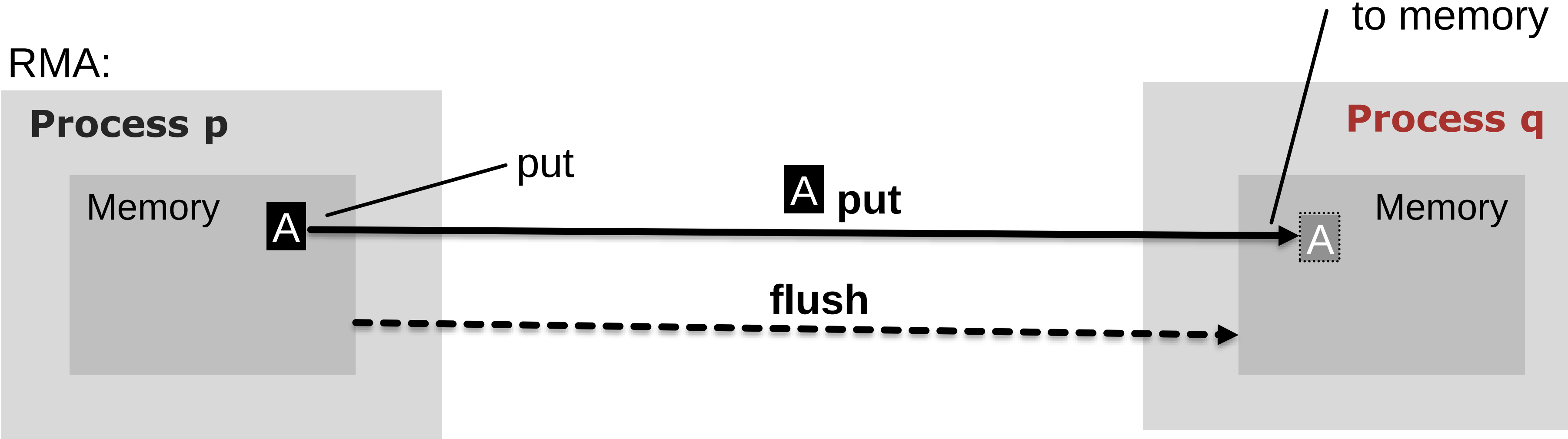

Message Passing:



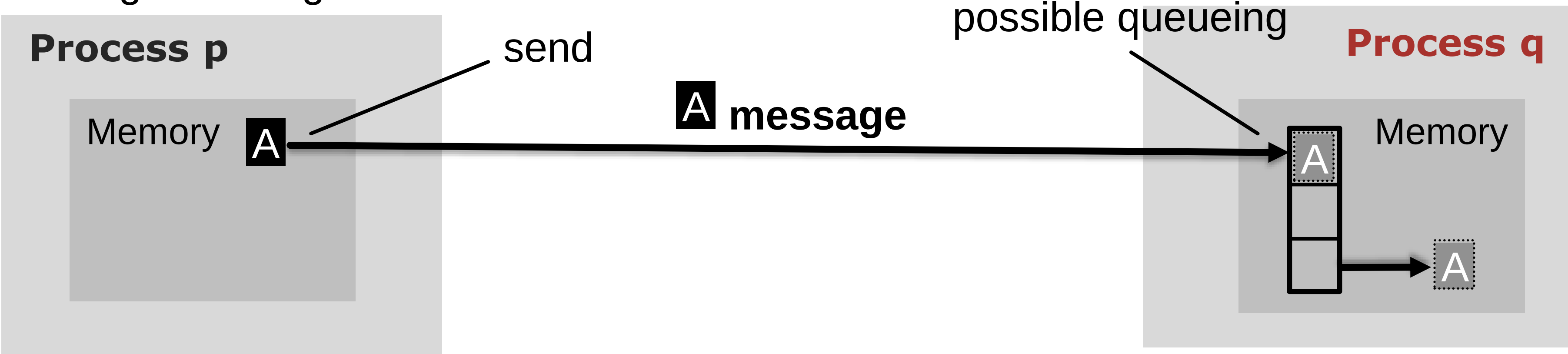
# RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:




Message Passing:



# 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 it ideal? 

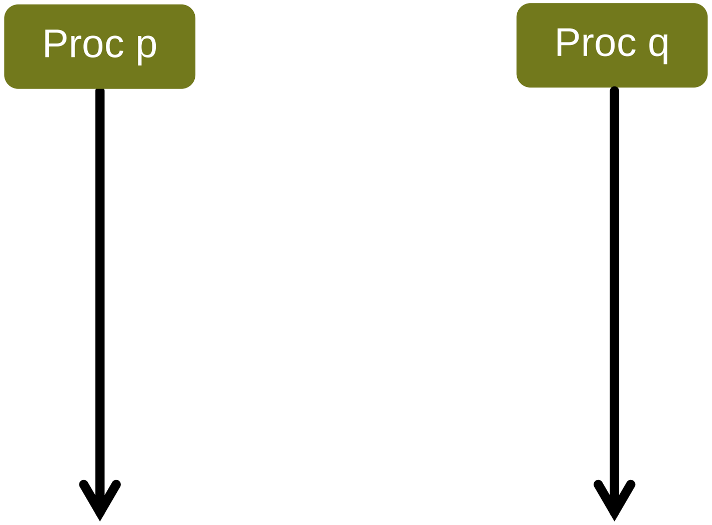
- Consider an insert in a distributed hashtable...

[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? 

- Consider an insert in a distributed hashtable...



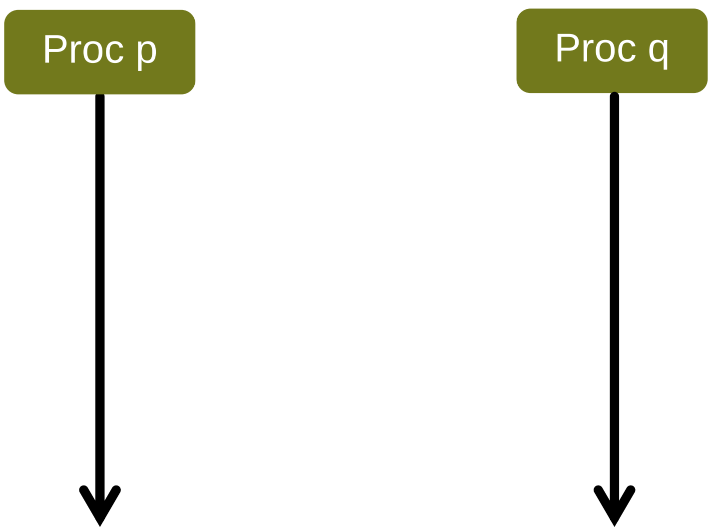

[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13

# Remote Memory Access Programming

- Is it ideal? 

- Consider an insert in a distributed hashtable...

No hash collision:




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

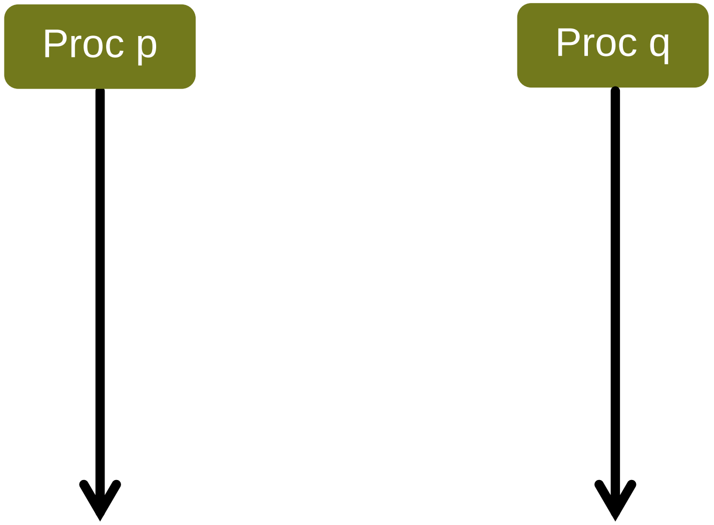
# Remote Memory Access Programming

- Is it ideal? 

- Consider an insert in a distributed hashtable...

No hash collision:

➔ 1 remote atomic




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

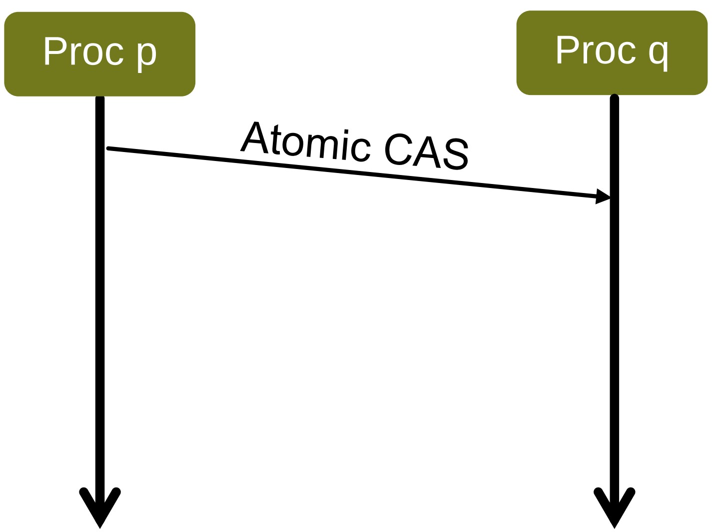
# REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal? 

- Consider an insert in a distributed hashtable...

No hash collision:

➔ 1 remote atomic



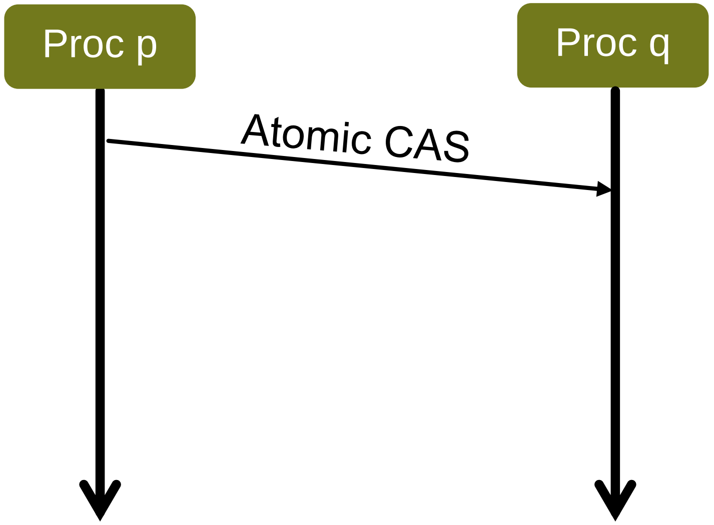

[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13

# REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal? NO!
- Consider an insert in a distributed hashtable...

No hash collision:

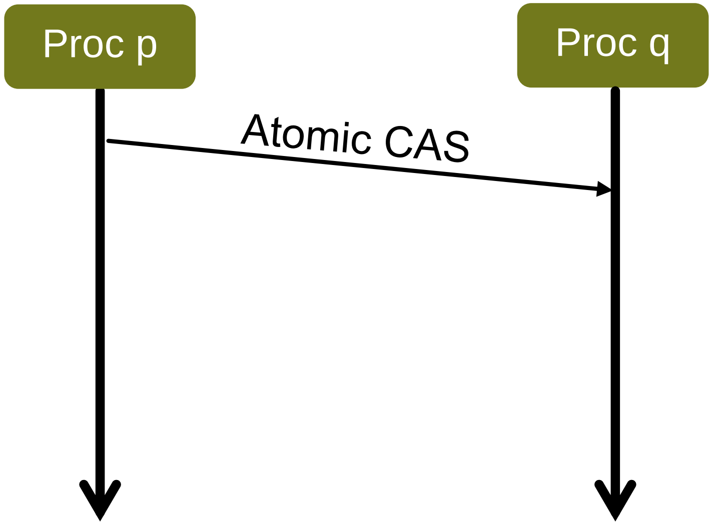
➔ 1 remote atomic
➔ Up to 5x speedup over MP [1]




[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? NO!
- Consider an insert in a distributed hashtable...




No hash collision:

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

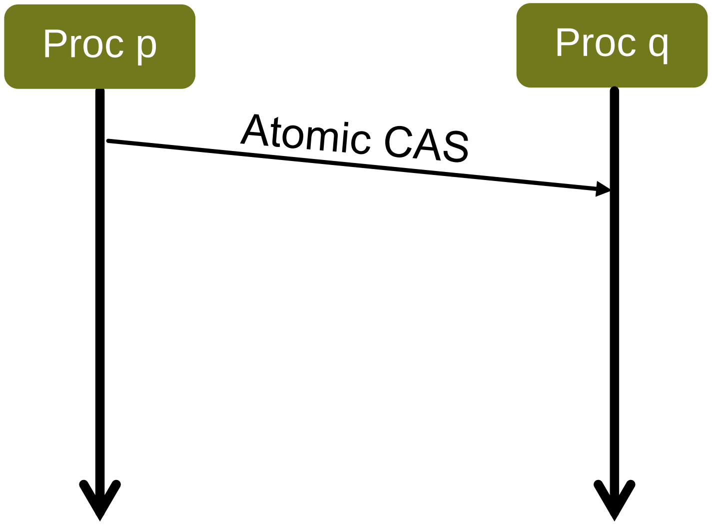
A hash collision:

[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?


- Consider an insert in a distributed hashtable...




No hash collision:

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

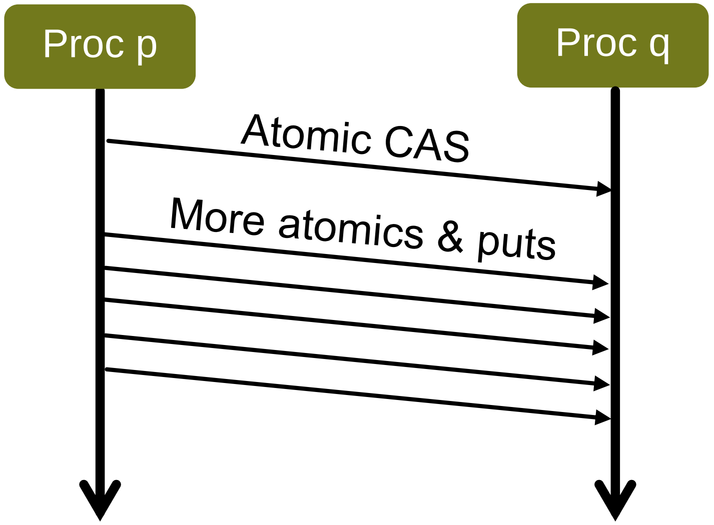
A hash collision:

➔ 4 remote atomics + 2 remote puts

[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? NO!
- Consider an insert in a distributed hashtable...




No hash collision:

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

A hash collision:

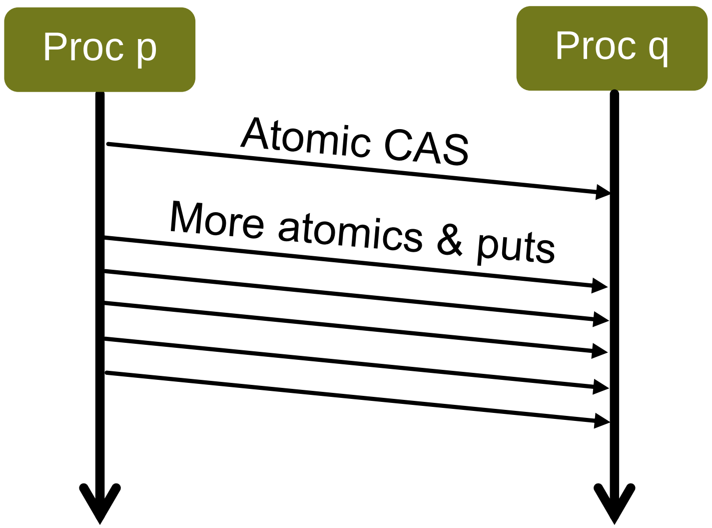
➔ 4 remote atomics + 2 remote puts

[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? 

- Consider an insert in a distributed hashtable...

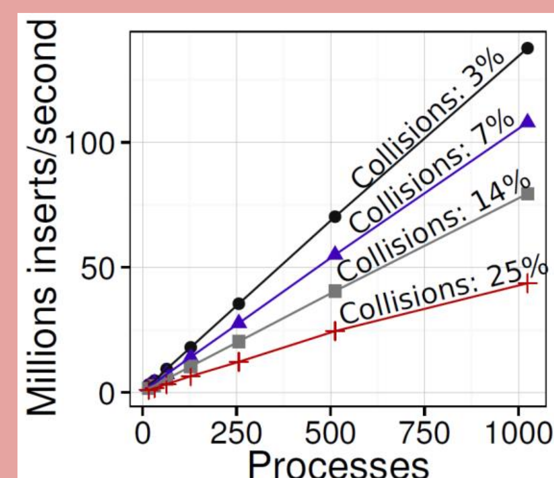



No hash collision:

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

A hash collision:

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

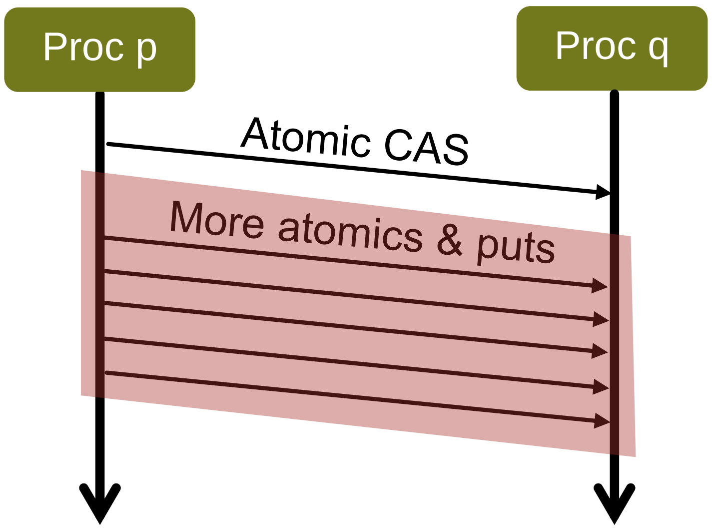



[1] R. Gerstenberger et al. Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. SC13

# Remote Memory Access Programming

- Is it ideal? 

- Consider an insert in a distributed hashtable...

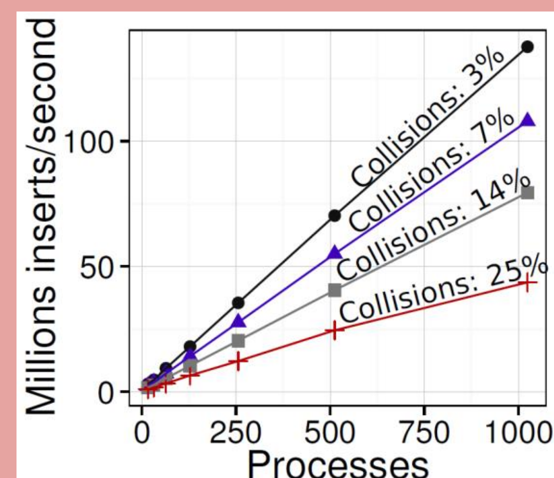



No hash collision:

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

A hash collision:

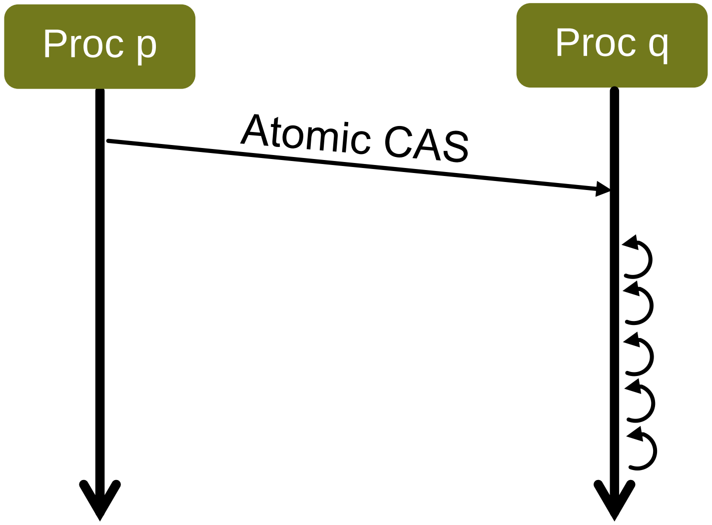
➔ 4 remote atomics + 2 remote puts
➔ Significant performance drops




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

# Remote Memory Access Programming

- Is it ideal? NO!
- Consider an insert in a distributed hashtable...

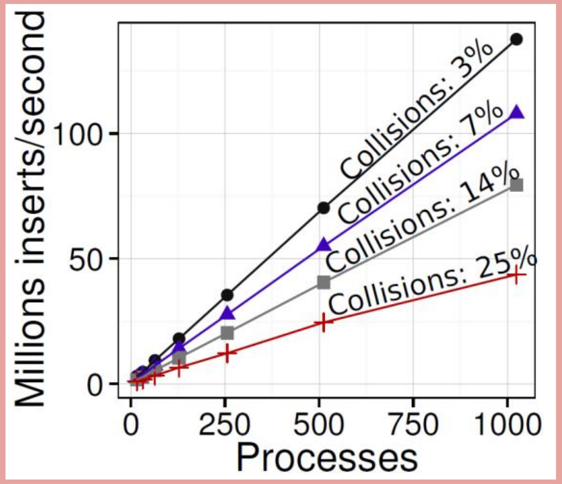



No hash collision:

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

A hash collision:

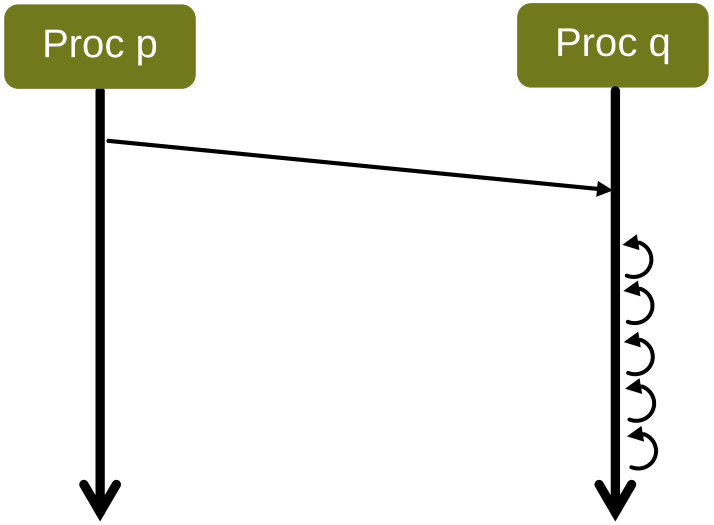
➔ 4 remote atomics + 2 remote puts
➔ Significant performance drops




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

# Remote Memory Access Programming

- Is it ideal? NO!
- Consider an insert in a distributed hashtable...

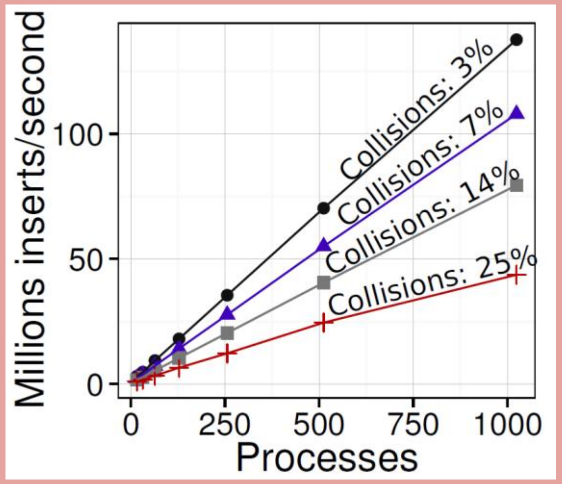



No hash collision:

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

A hash collision:

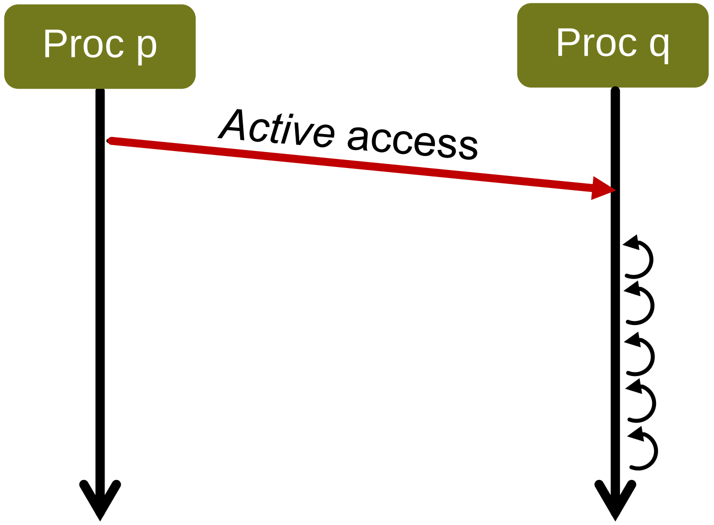
→ 4 remote atomics + 2 remote puts
→ Significant performance drops




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

# Remote Memory Access Programming

- Is it ideal? NO!
- Consider an insert in a distributed hashtable...

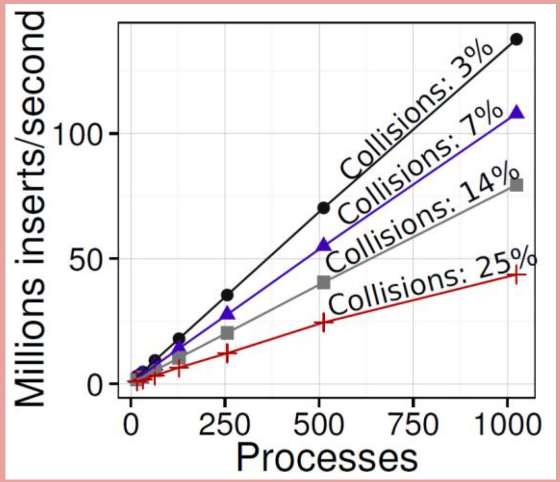



No hash collision:

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

A hash collision:

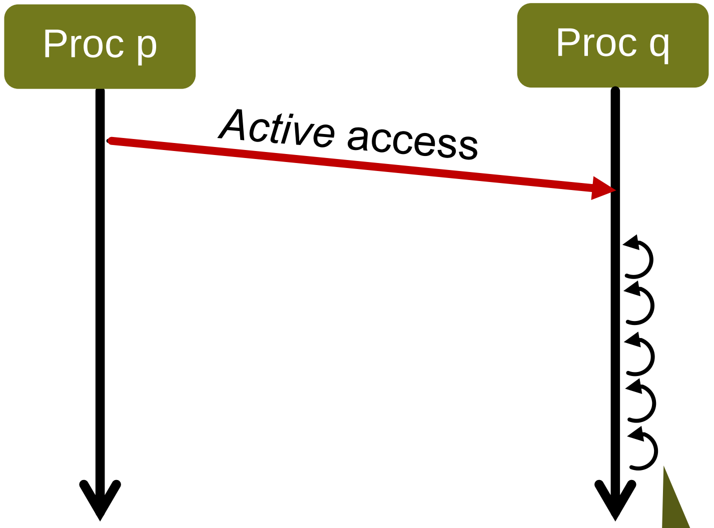
→ 4 remote atomics + 2 remote puts
→ Significant performance drops




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

# Remote Memory Access Programming

- Is it ideal? NO!
- Consider an insert in a distributed hashtable...



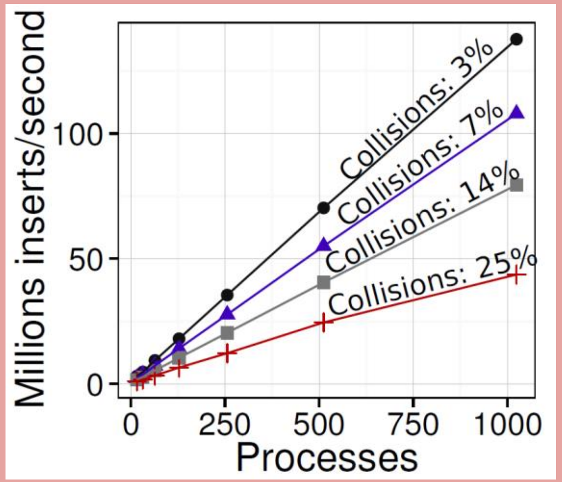

Local execution; triggered by an active *access*. In RMA?

No hash collision:

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

A hash 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

# Remote Memory Access Programming

How to enable it?

- Is i
- Co

distributed hashtable...




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

No hash collision:

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

A hash 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

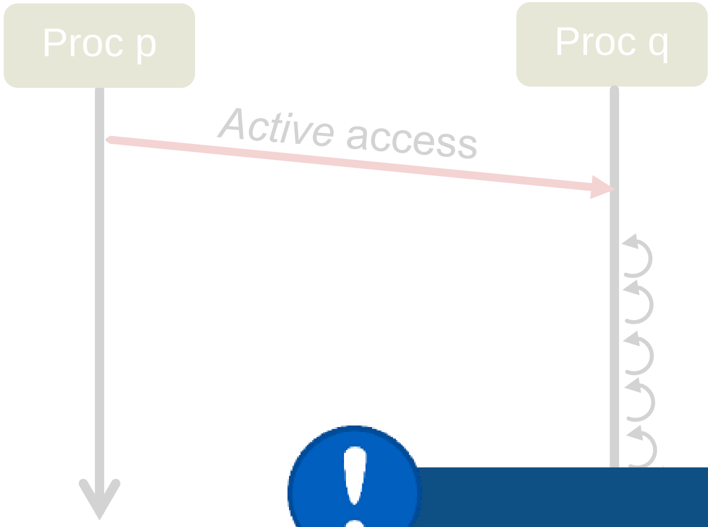
# Remote Memory Access Programming

- Is i
- Co
distributed hashtable...

**How to enable it?**

No hash collision:

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




Local executi
active access. In RMA?

**Use “active” semantics**

A hash 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

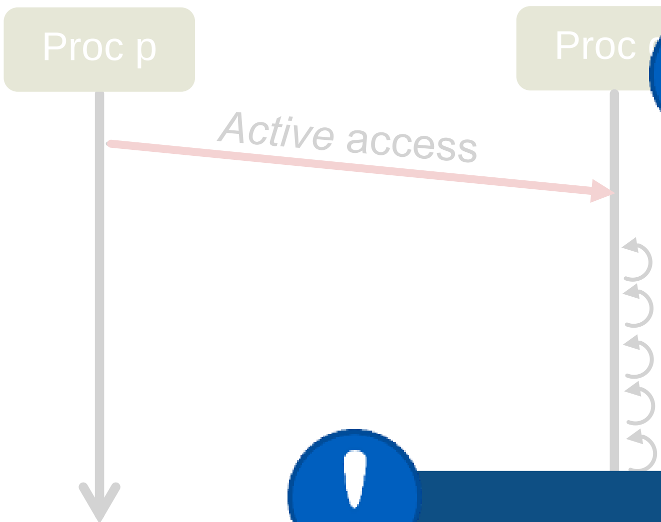
# Remote Memory Access Programming

How to enable it?

- Is i
- Co
  distributed hashtable...

No hash collision:

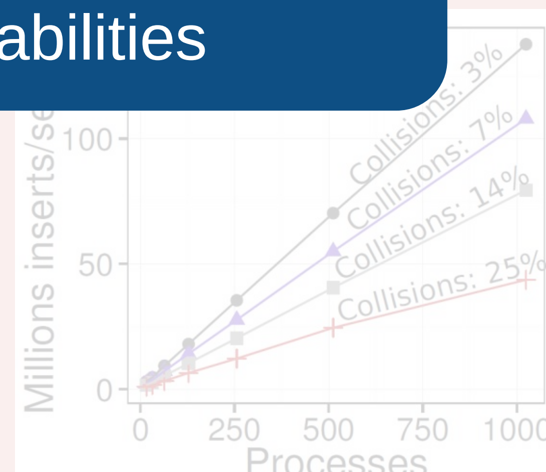
→ 1 remote atomic
→ Up to 5x speedup over MP [1]




Use and extend I/O MMUs and their paging capabilities

A hash collision:

emote puts
e drops




Use "active" semantics

Local executi
active access: In RMA?

[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]

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

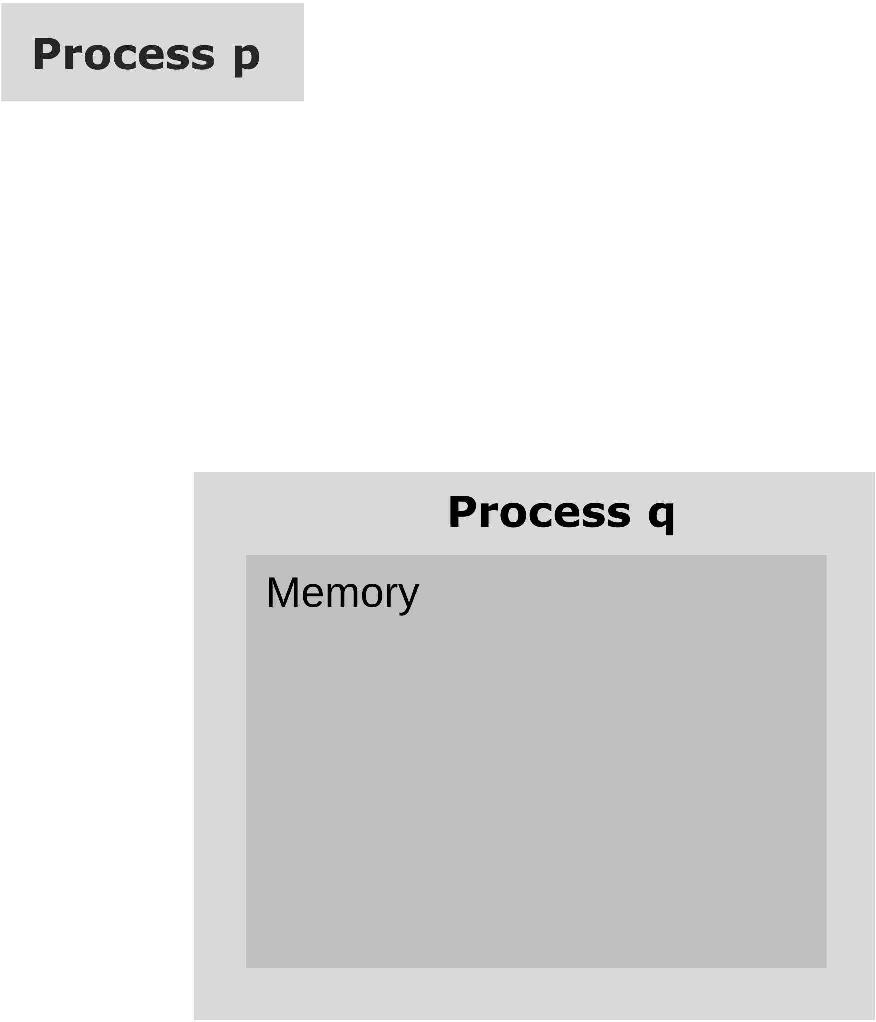
# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

Process p

Process q

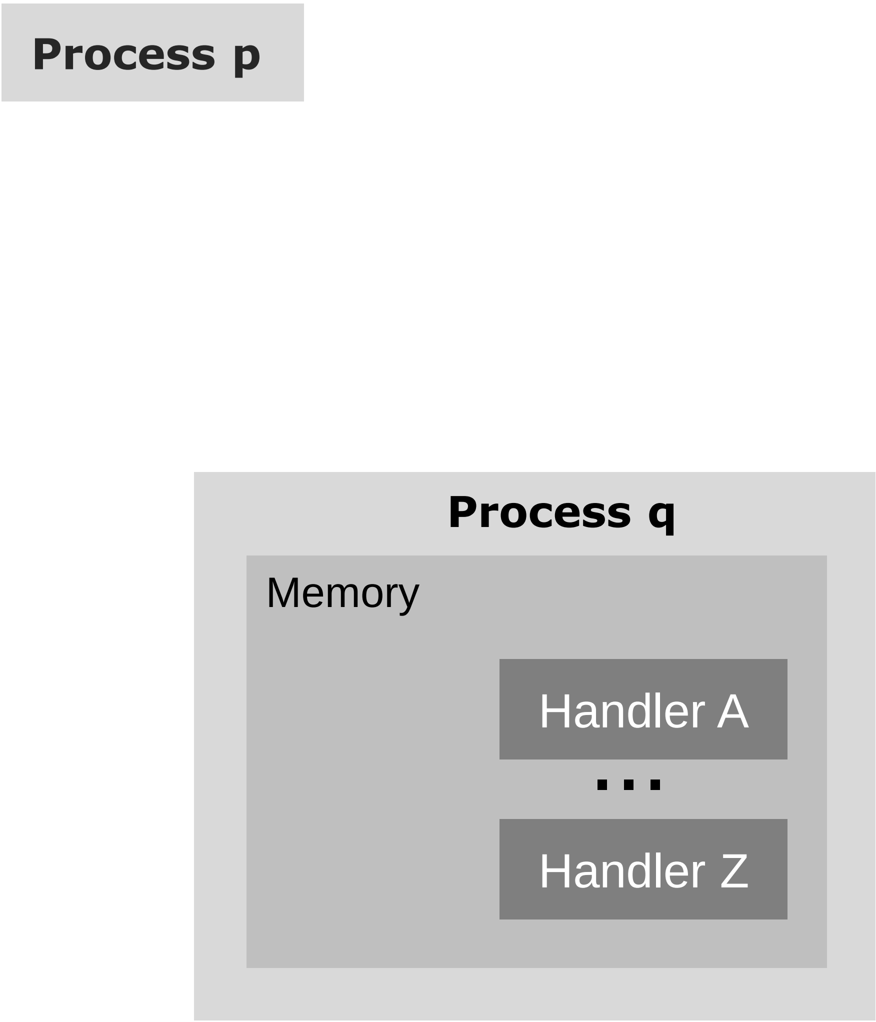
[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



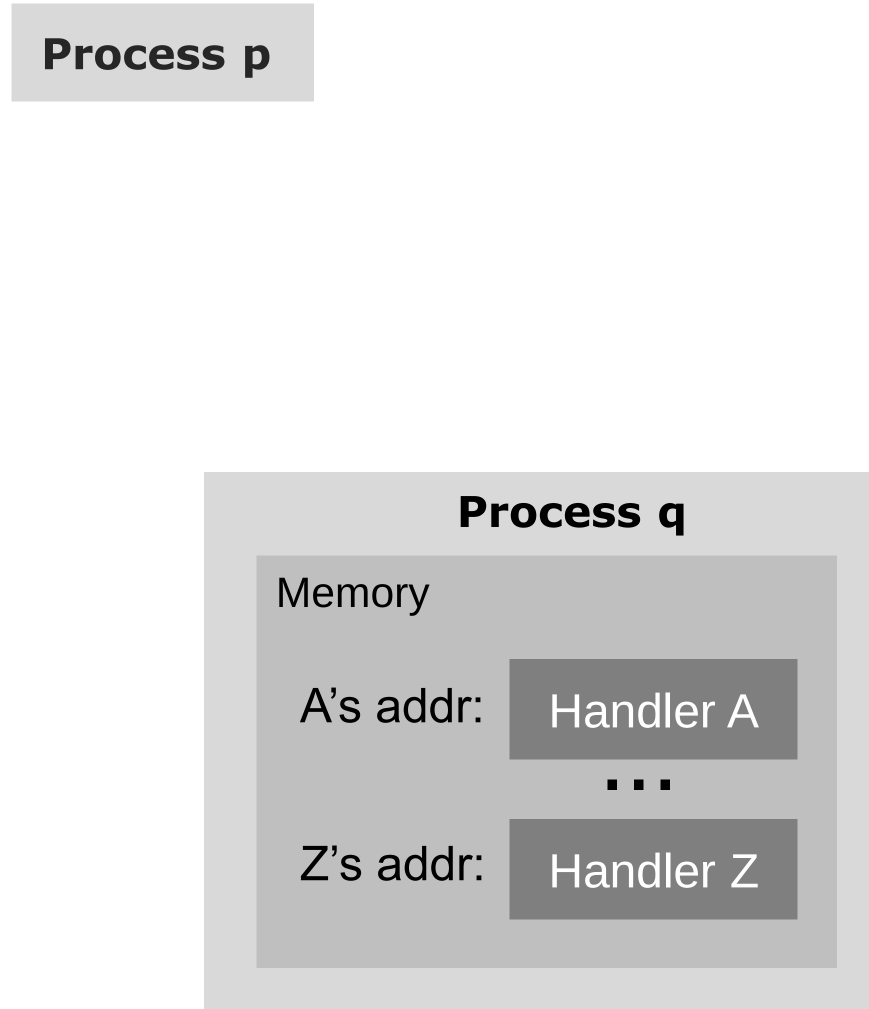

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



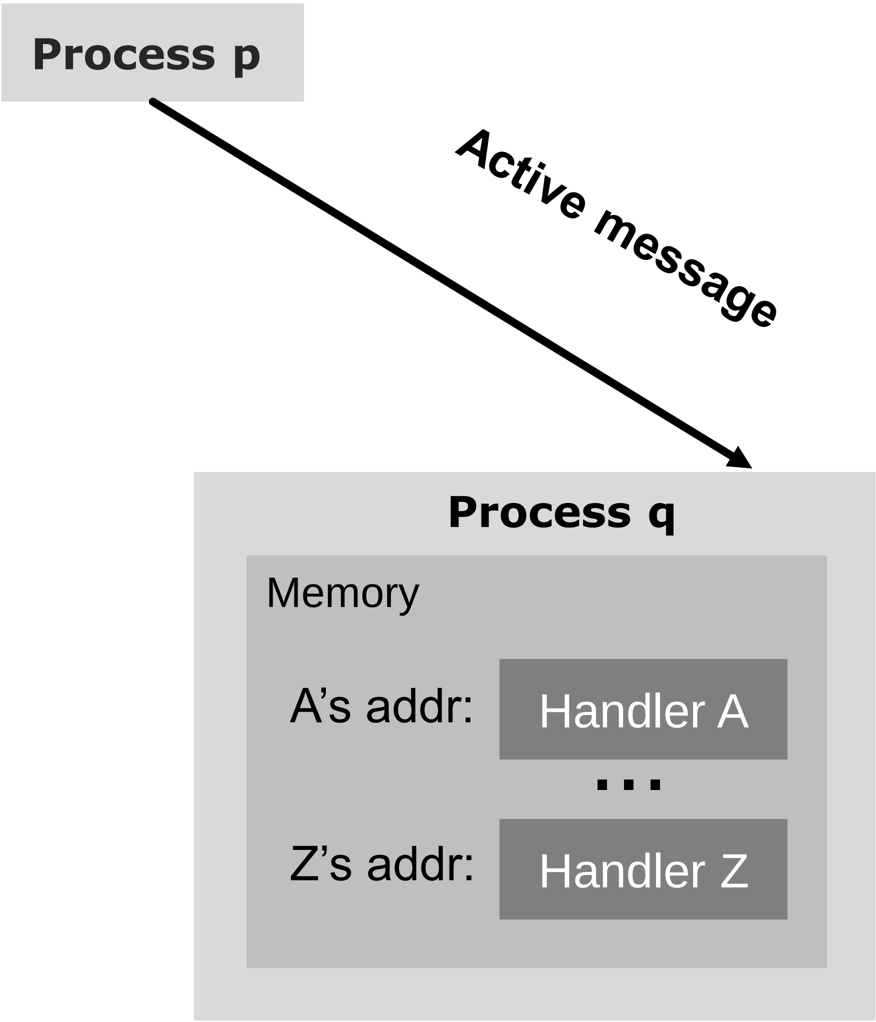

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



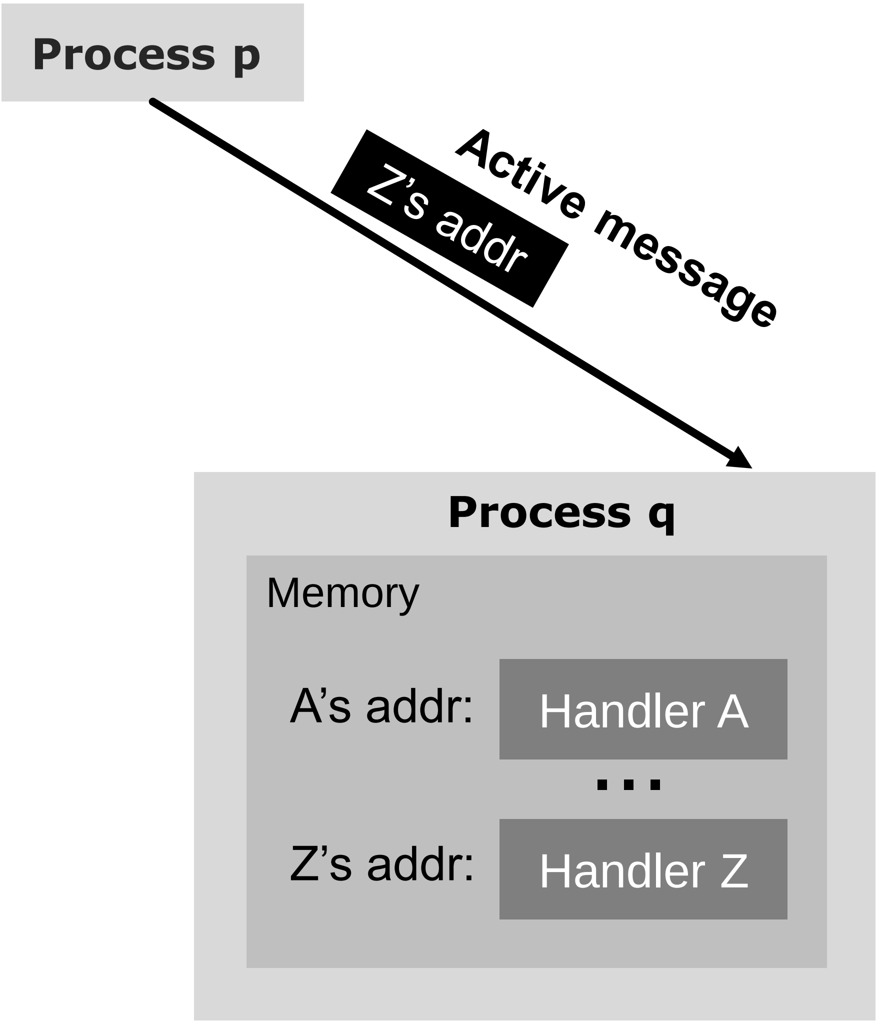

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



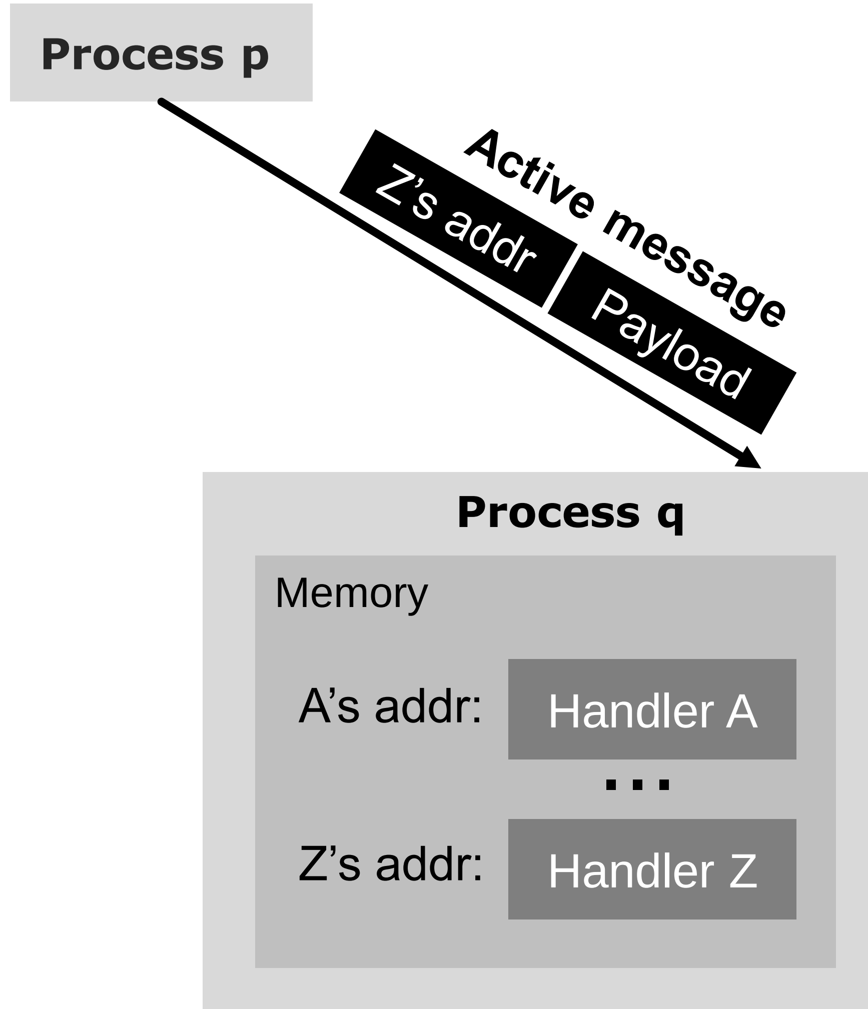

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



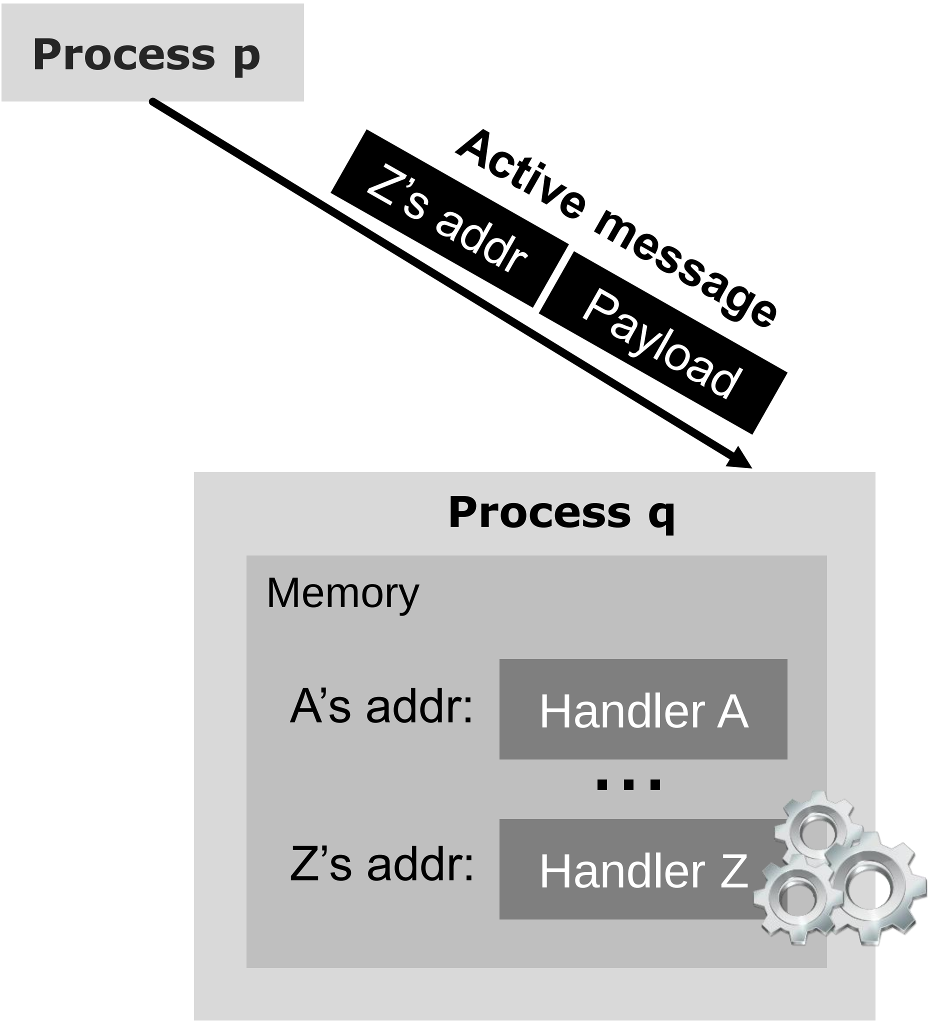

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



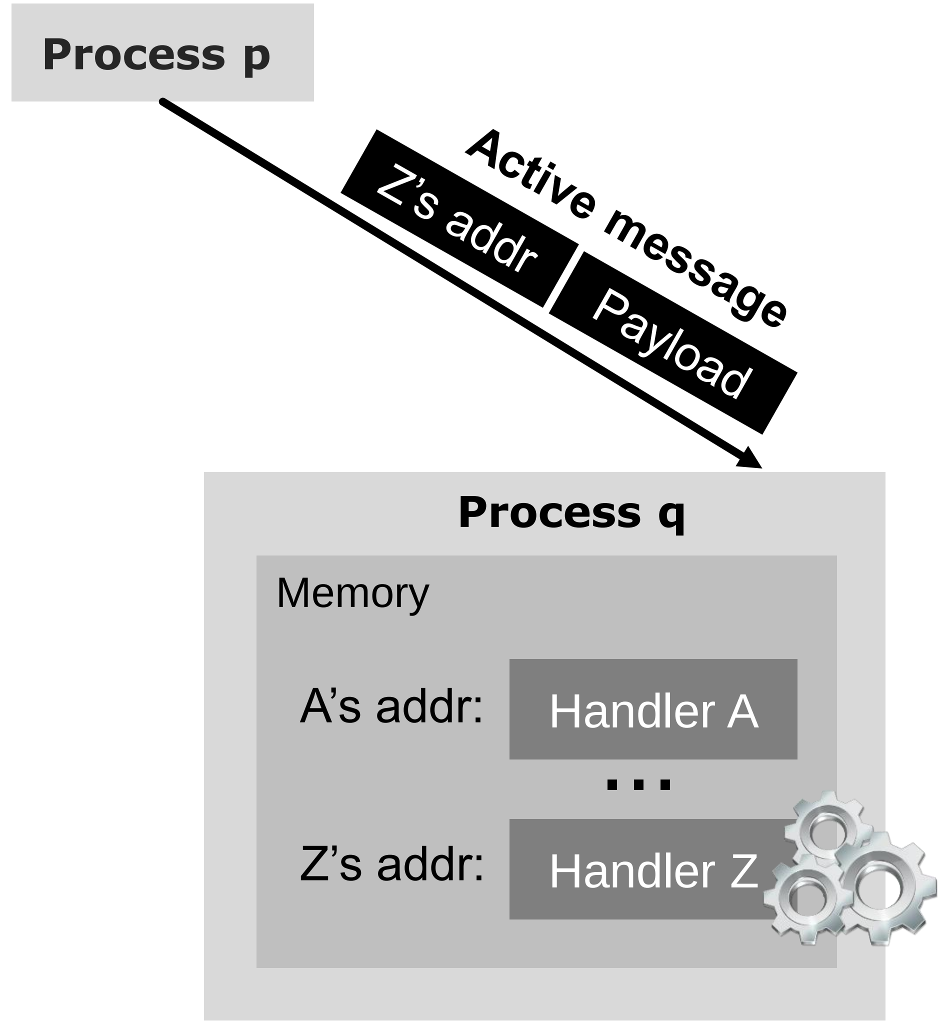

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]



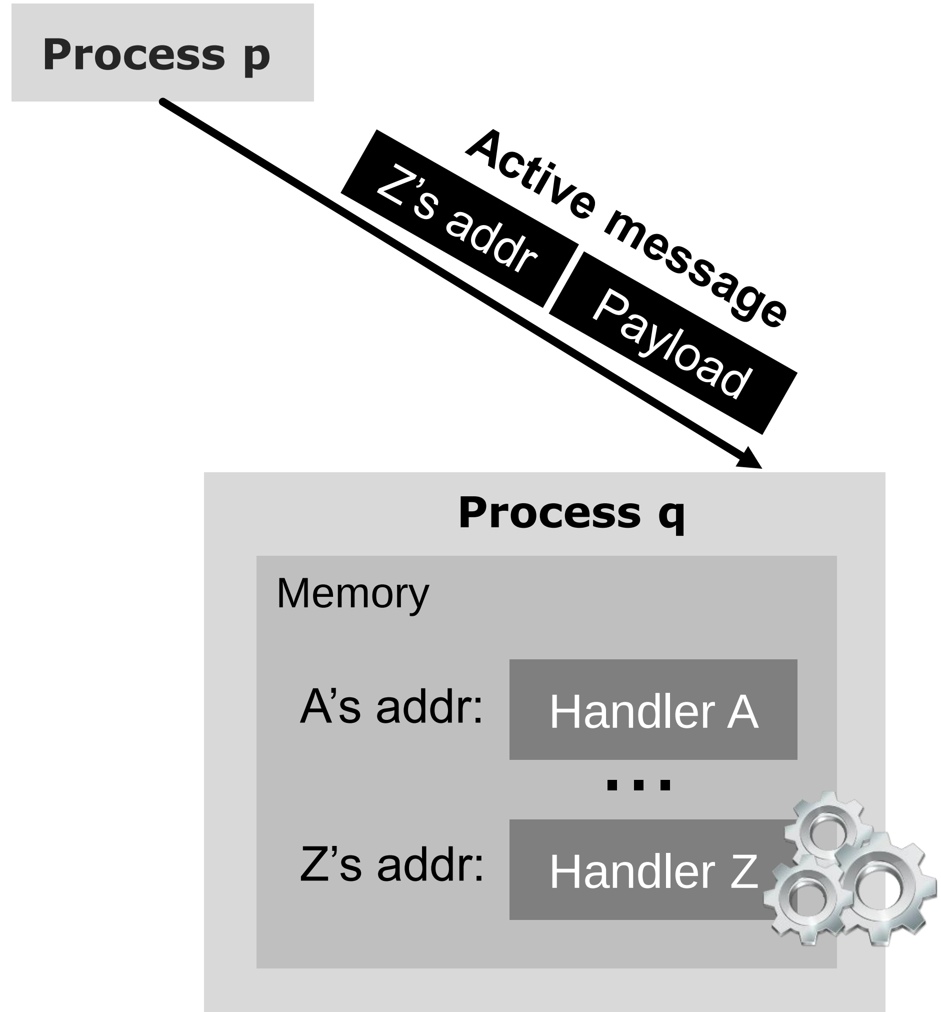

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]


[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

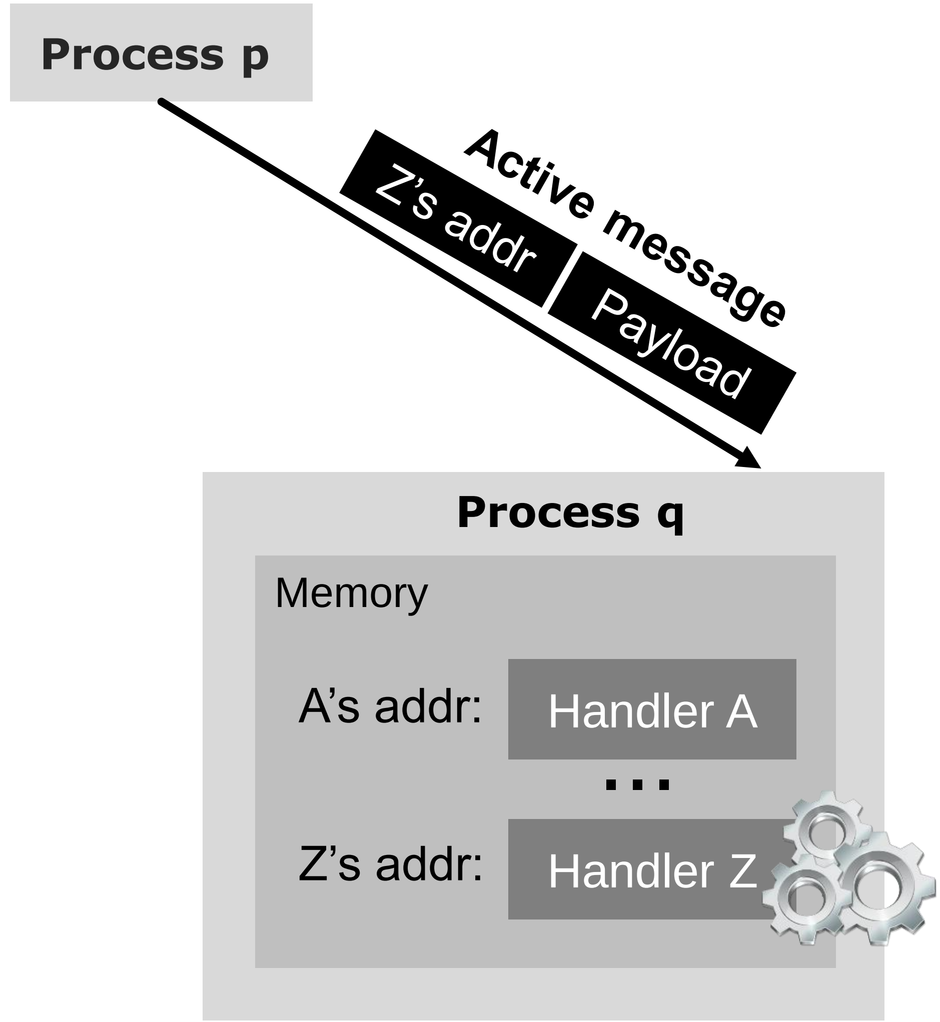
# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]


[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]


AM++[2]
GASNet [3]



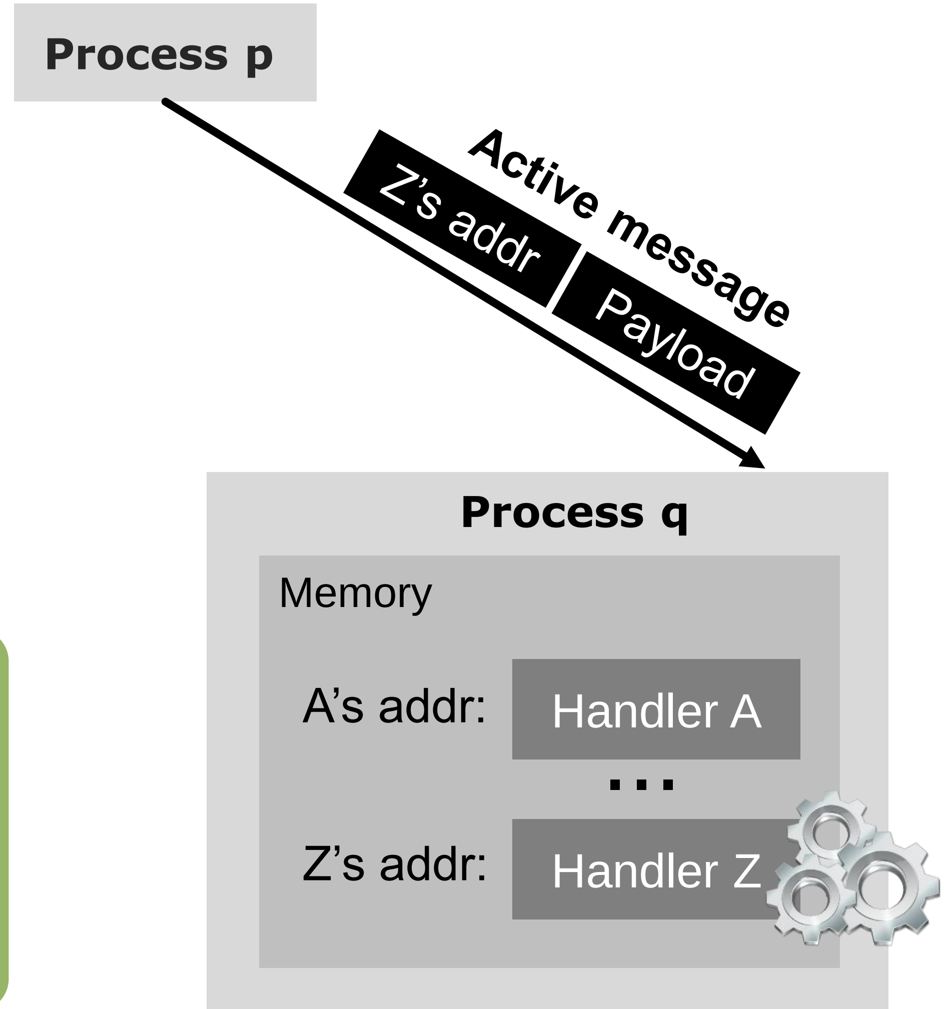

[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.
[2] J. J. Willcock et al. AM++: A generalized active message framework. PACT'10.
[3] D. Bonachea, GASNet Specification, v1.1. Berkeley Technical Report. 2002.

# USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

AM++[2]
GASNet [3]

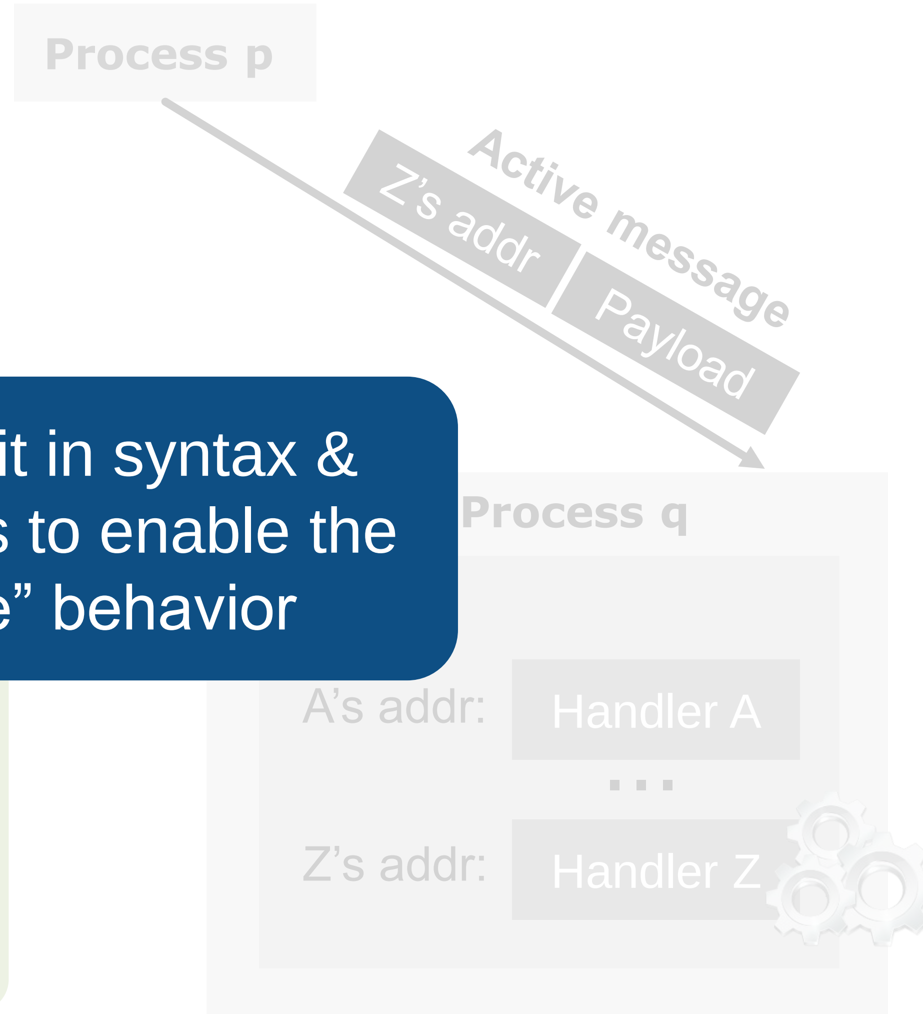
We need *active* puts/gets:

- 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 SEMANTICS FROM ACTIVE MESSAGES (AM) [1]




We need *active* puts/gets:

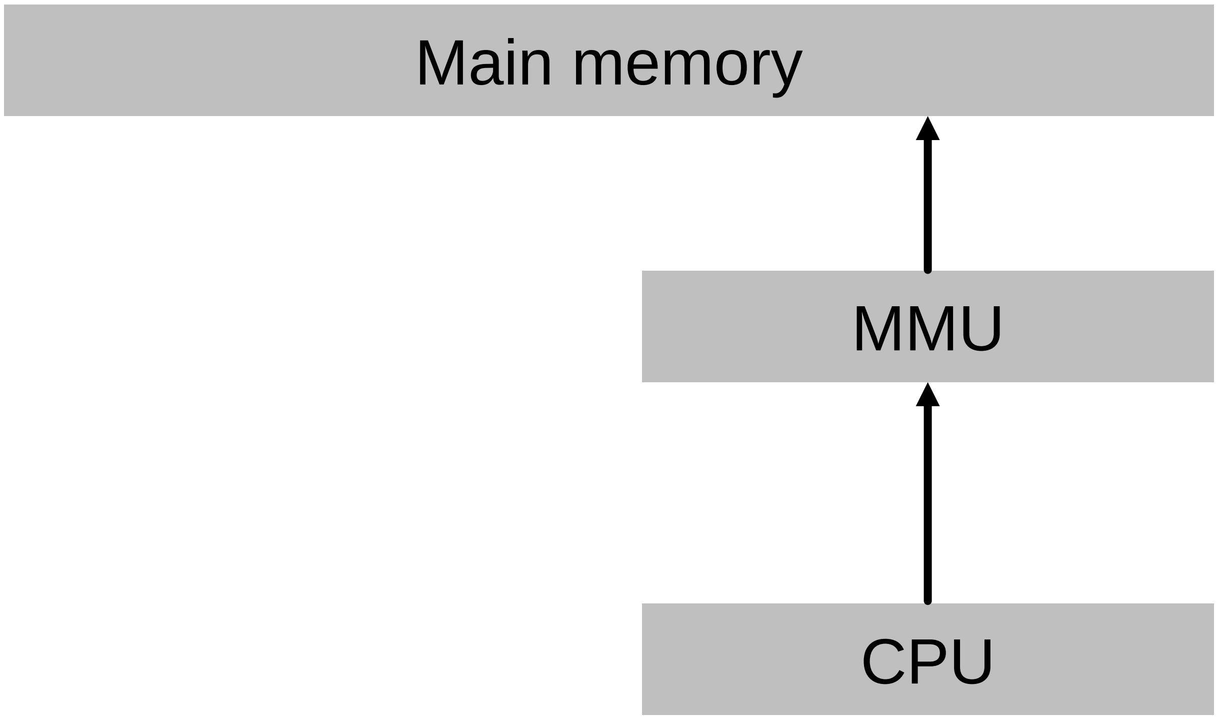
- Invoke a handler upon accessing a given page
- Preserve one-sided RMA behavior

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

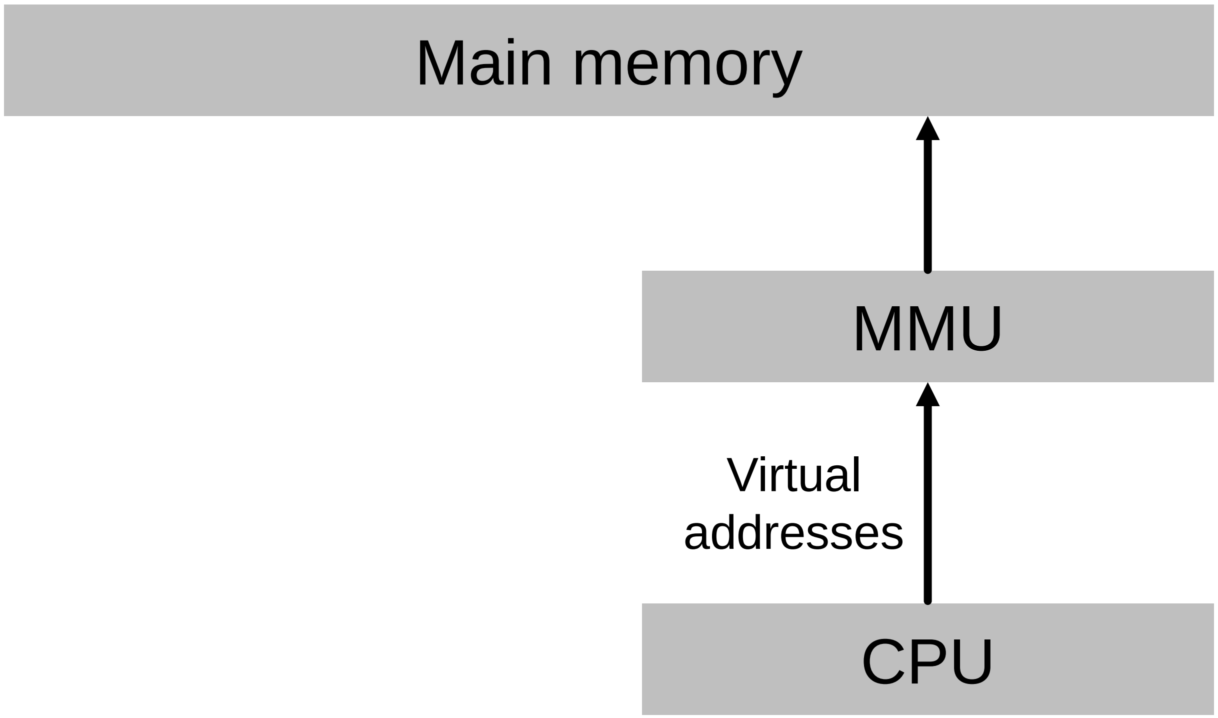
[1] T. von Eicken et al. Active messages: a mechanism for integrated communication and computation. ISCA'92.
[2] J. J. Willcock et al. AM++: A generalized active message framework. PACT'10.
[3] D. Bonachea, GASNet Specification, v1.1. Berkeley Technical Report. 2002.

# Use Input/Output Memory Management Units

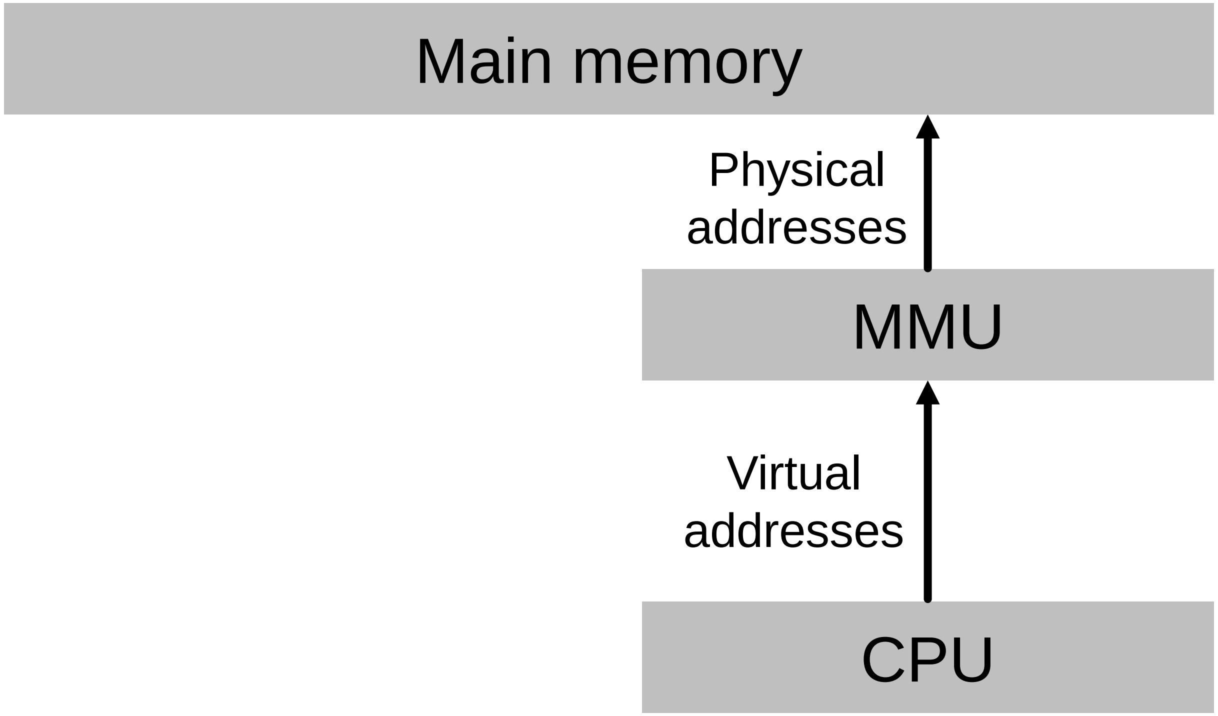
# Use Input/Output Memory Management Units



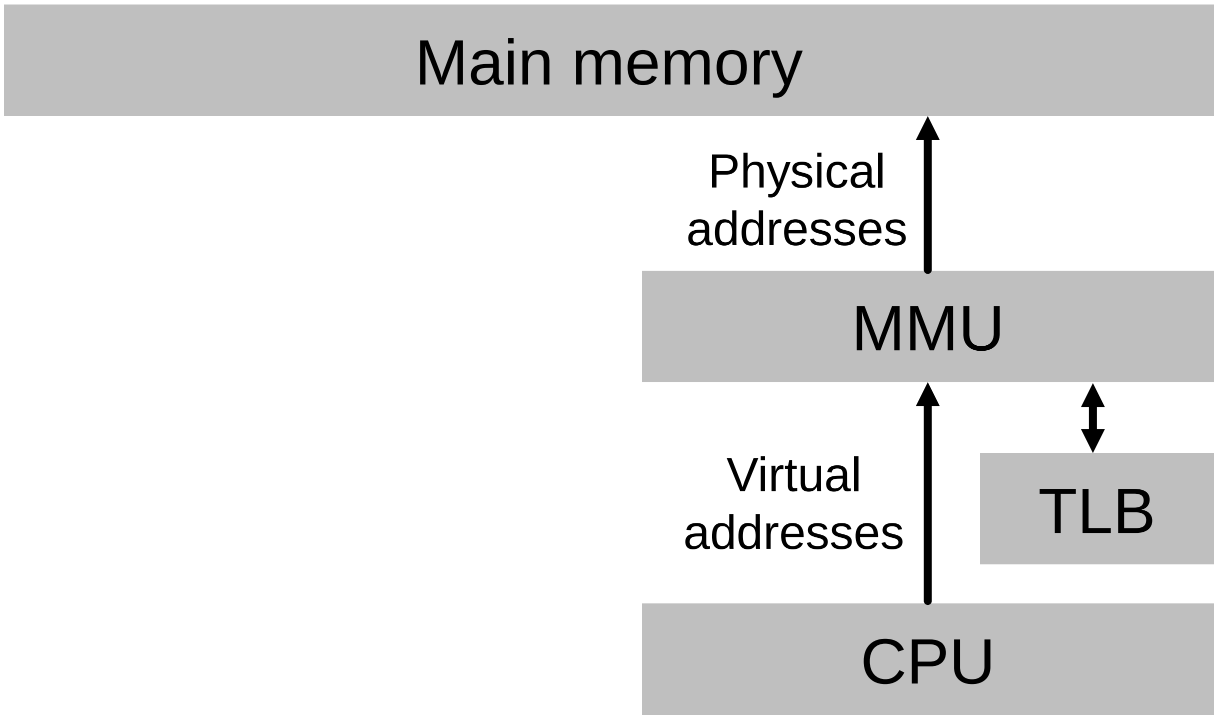
# USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS



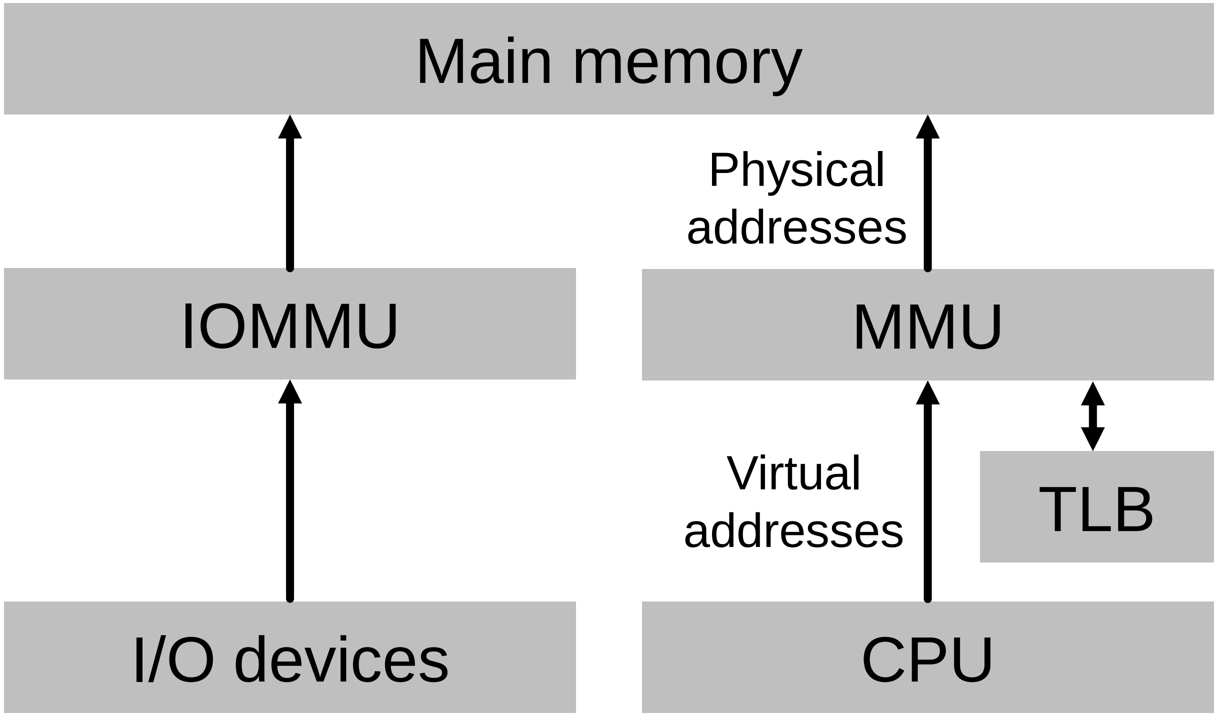
# Use Input/Output Memory Management Units



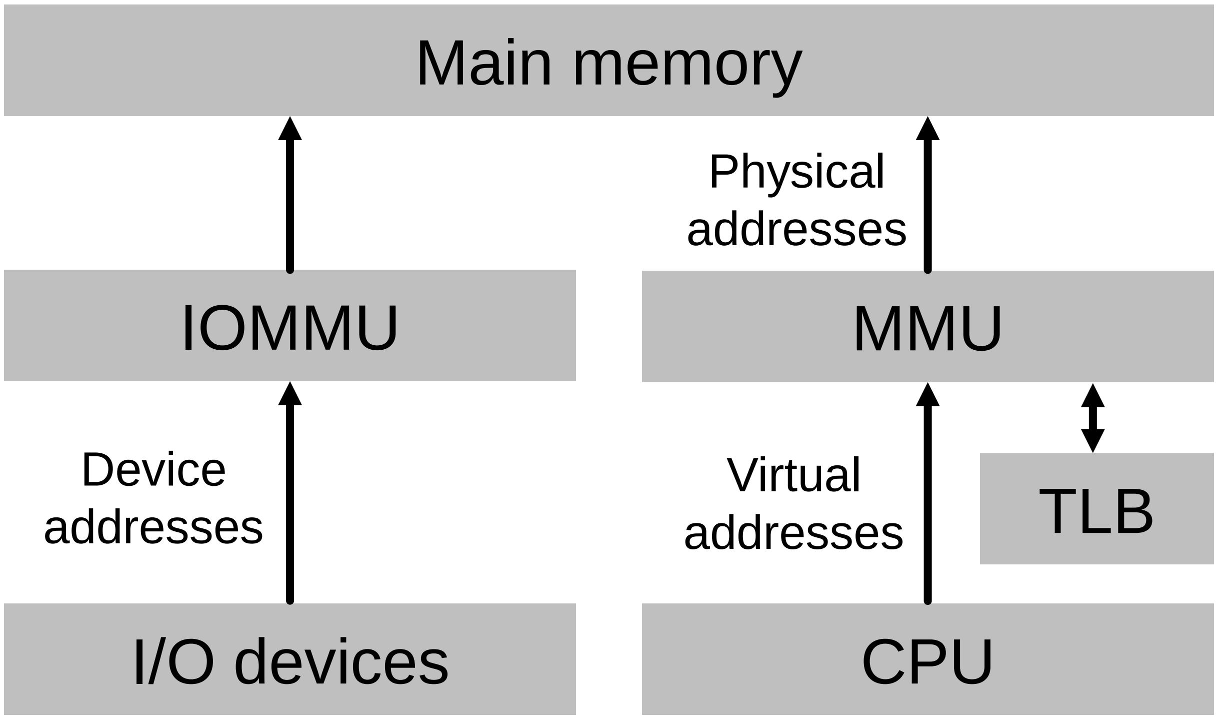
# Use Input/Output Memory Management Units



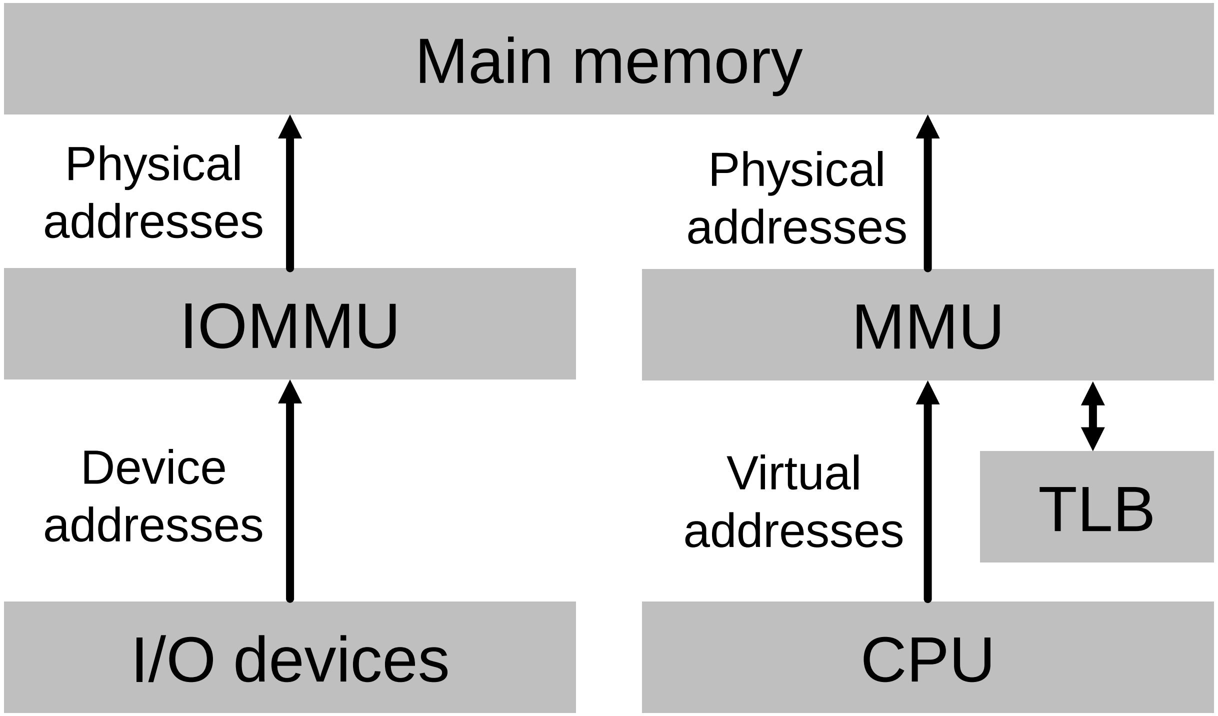
# USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS



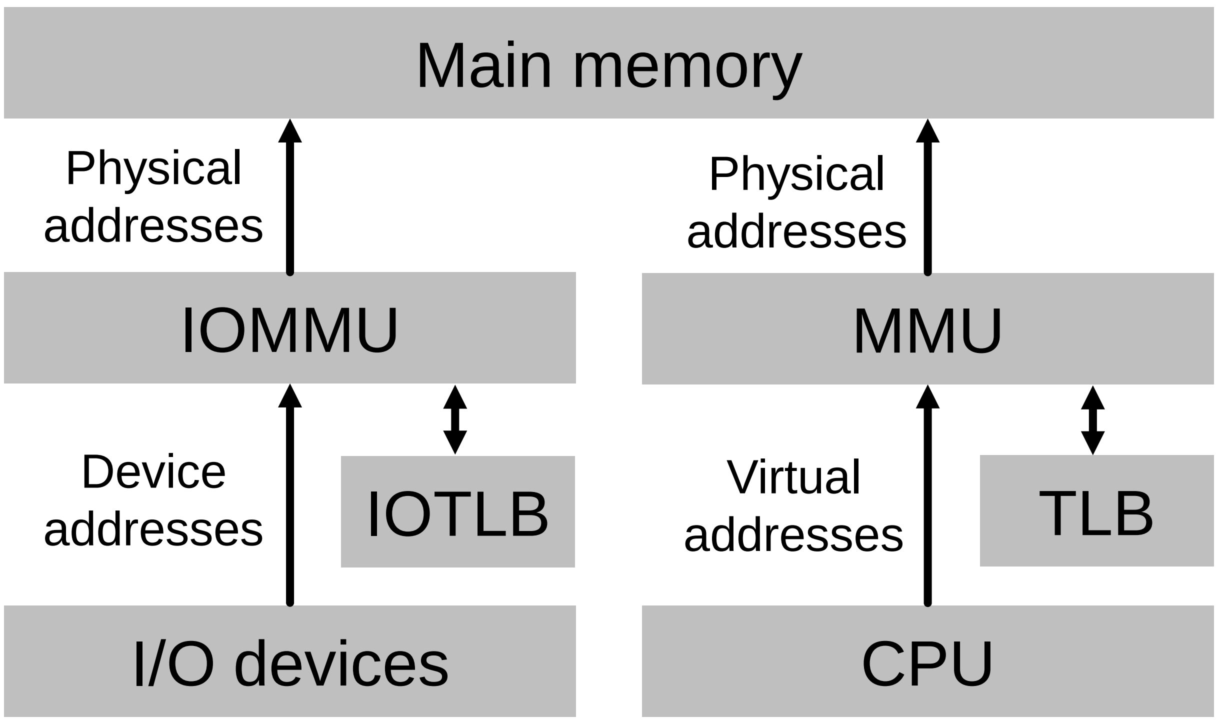
# USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS



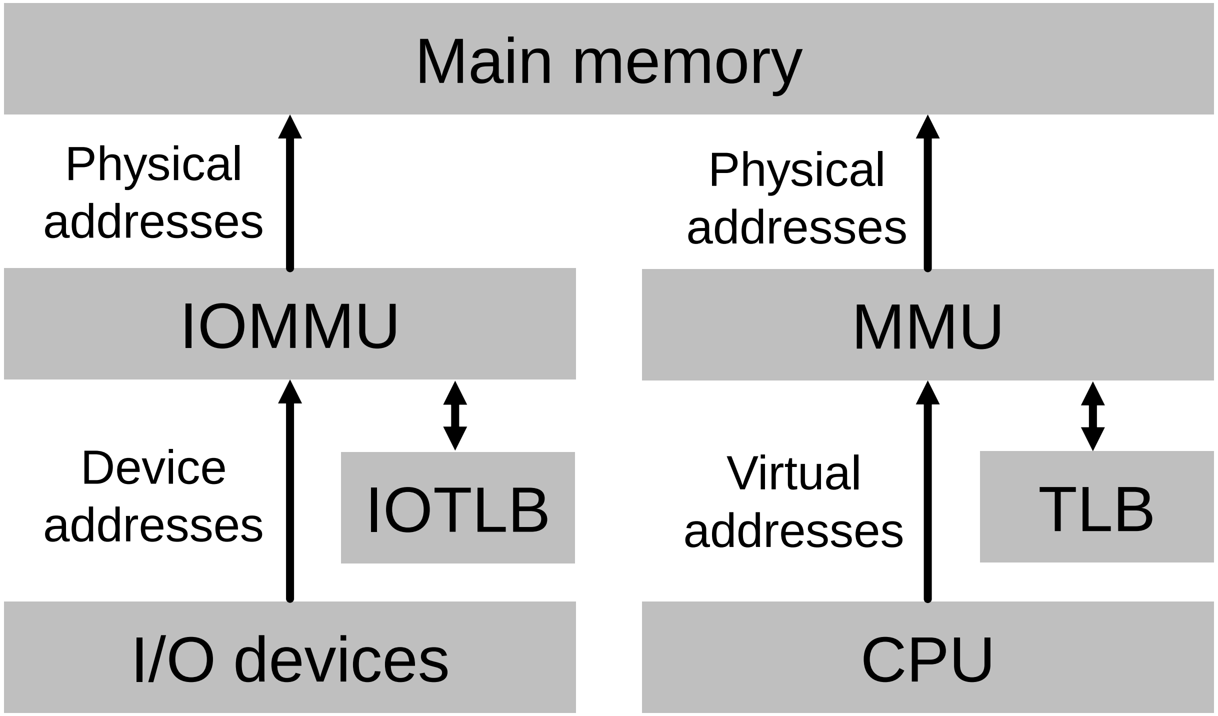
# Use Input/Output Memory Management Units



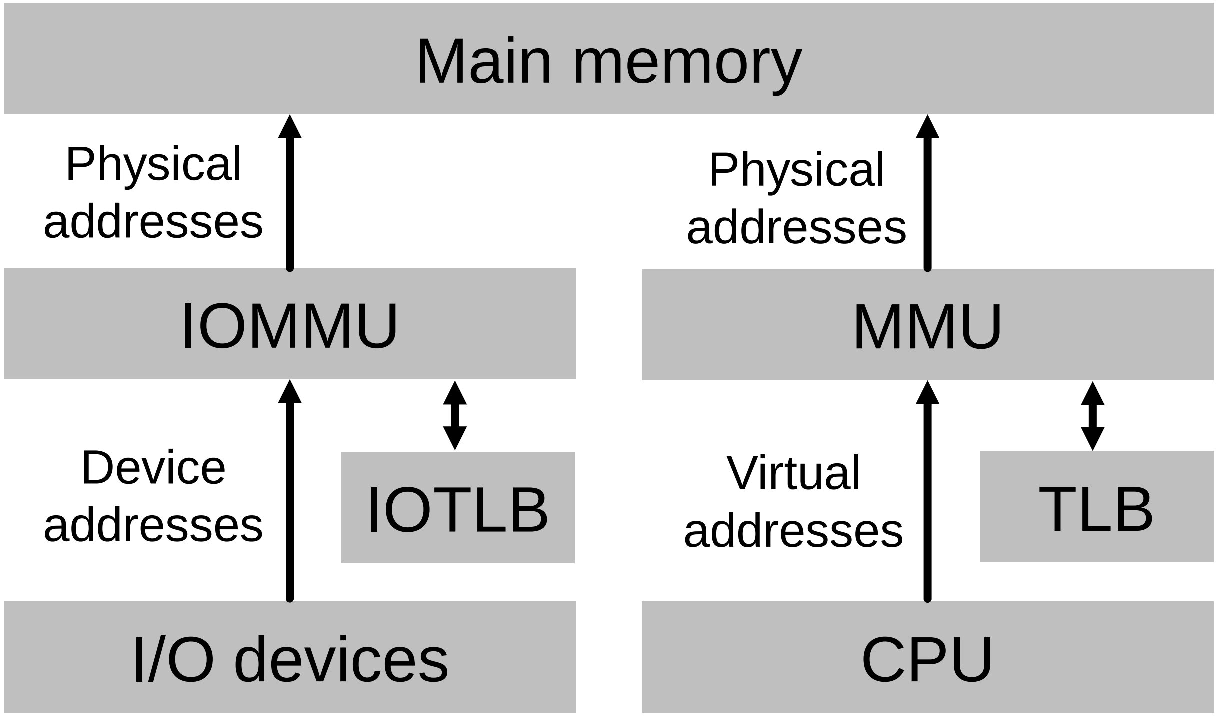
# USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS



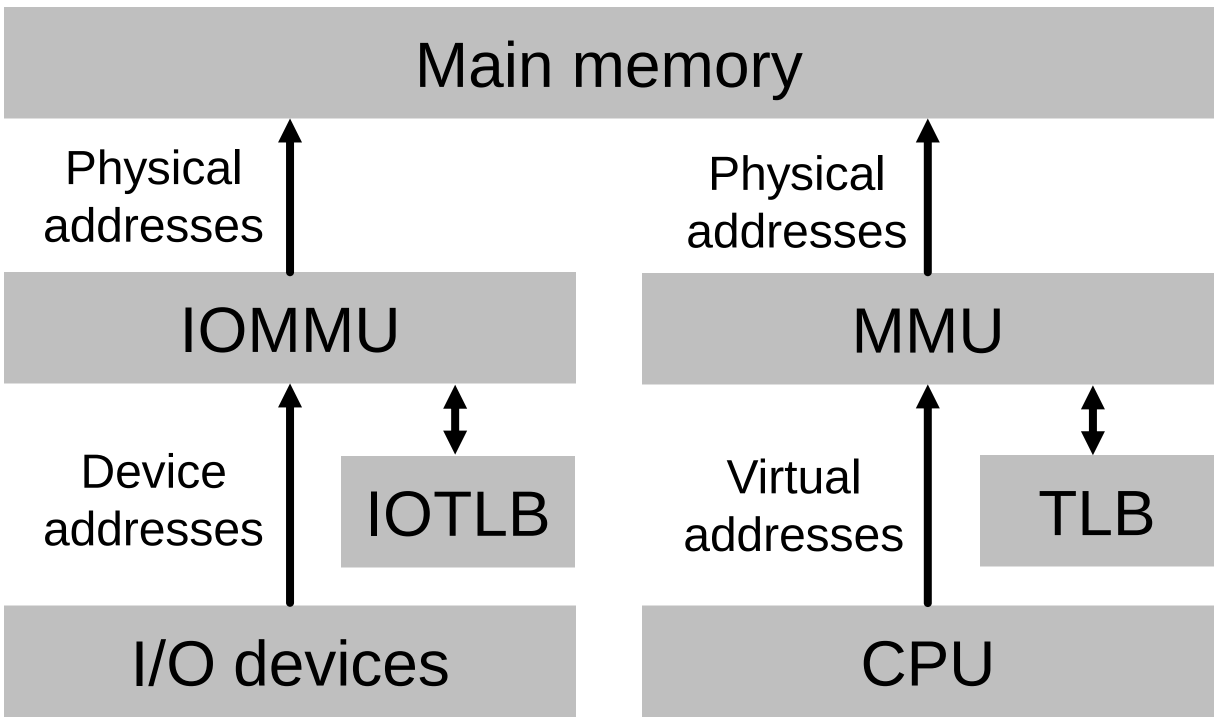
# Use Input/Output Memory Management Units



# Use Input/Output Memory Management Units

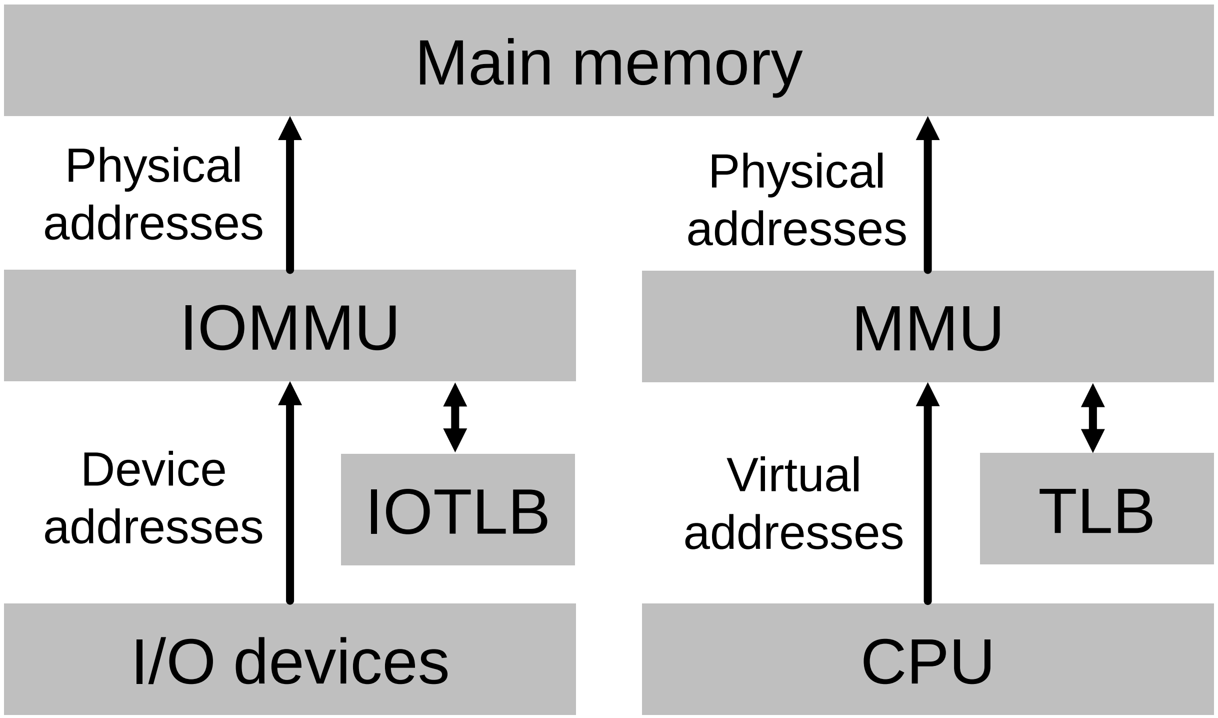


# Use Input/Output Memory Management Units



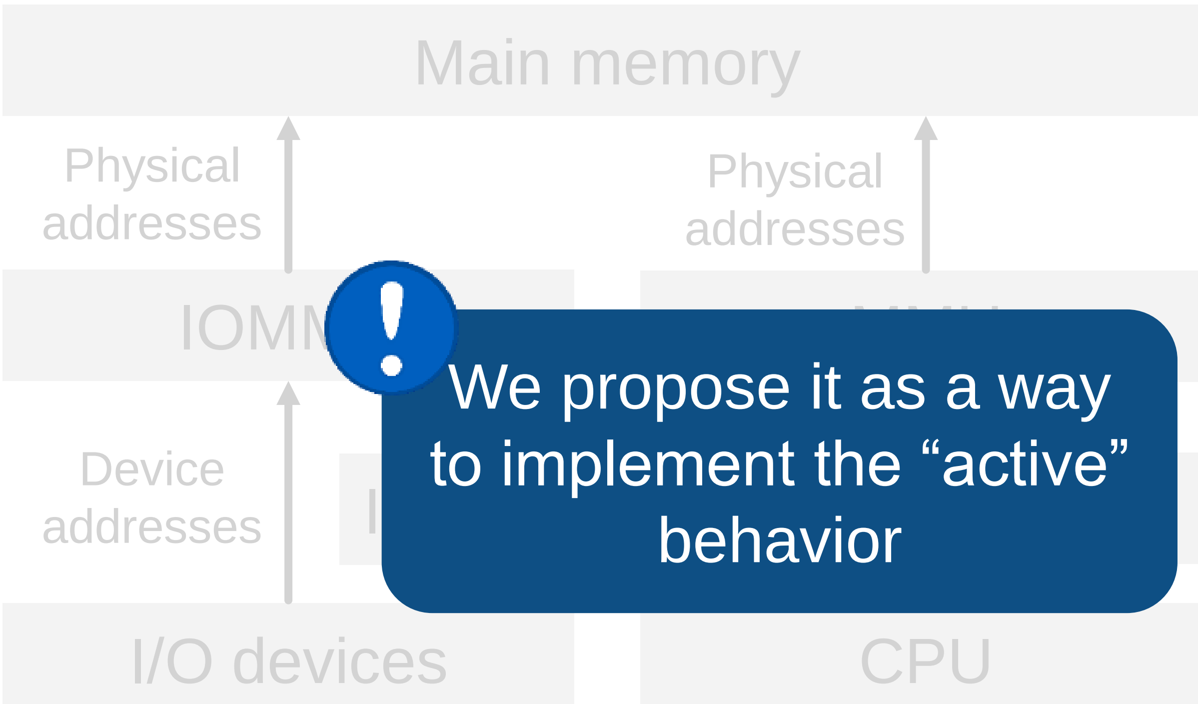

+

# Use Input/Output Memory Management Units




AMD
IBM
ARM
Sun microsystems
SOLARFLARE
intel + PCI EXPRESS

# Use Input/Output Memory Management Units



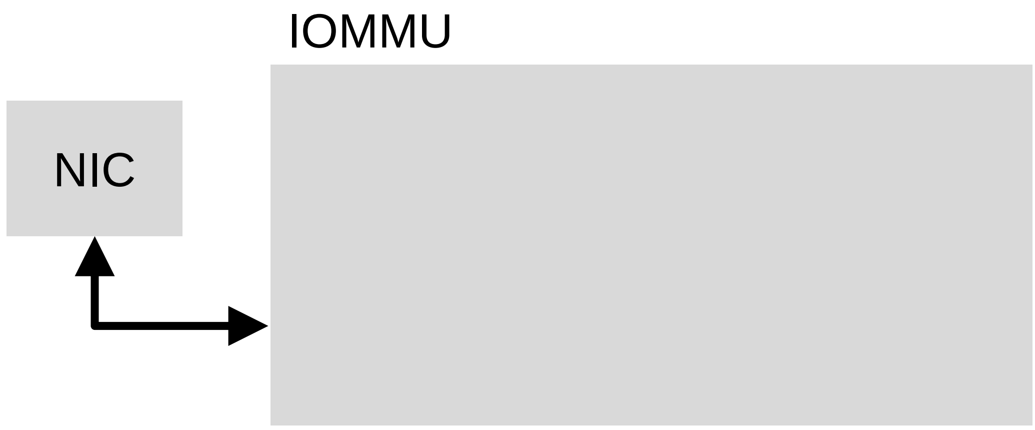

We propose it as a way to implement the “active” behavior

# IOMMUS AND RMA

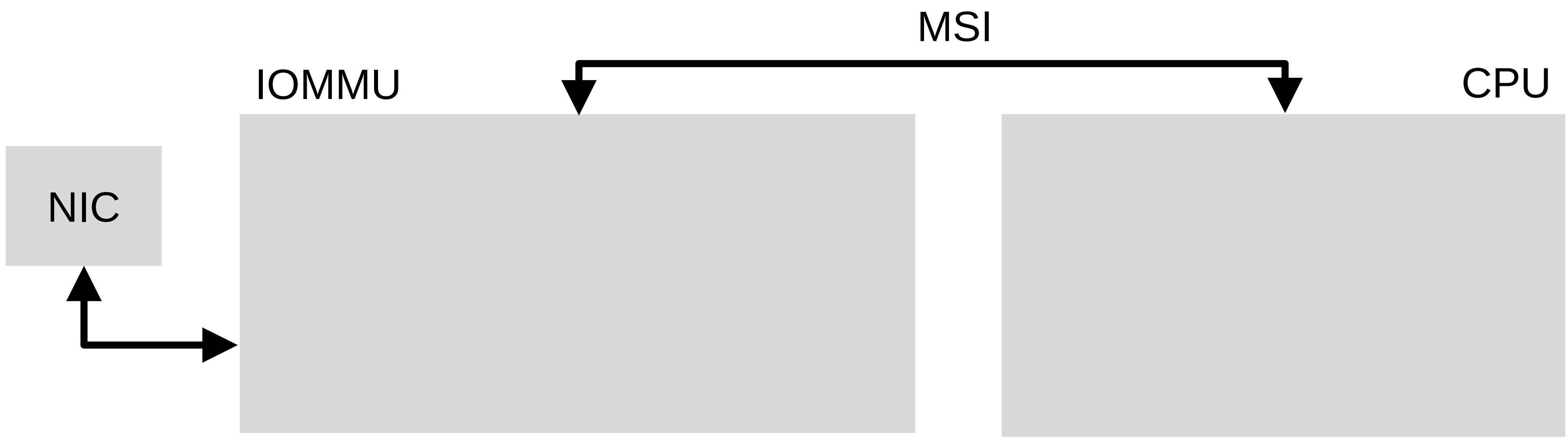
# IOMMUs and RMA

NIC

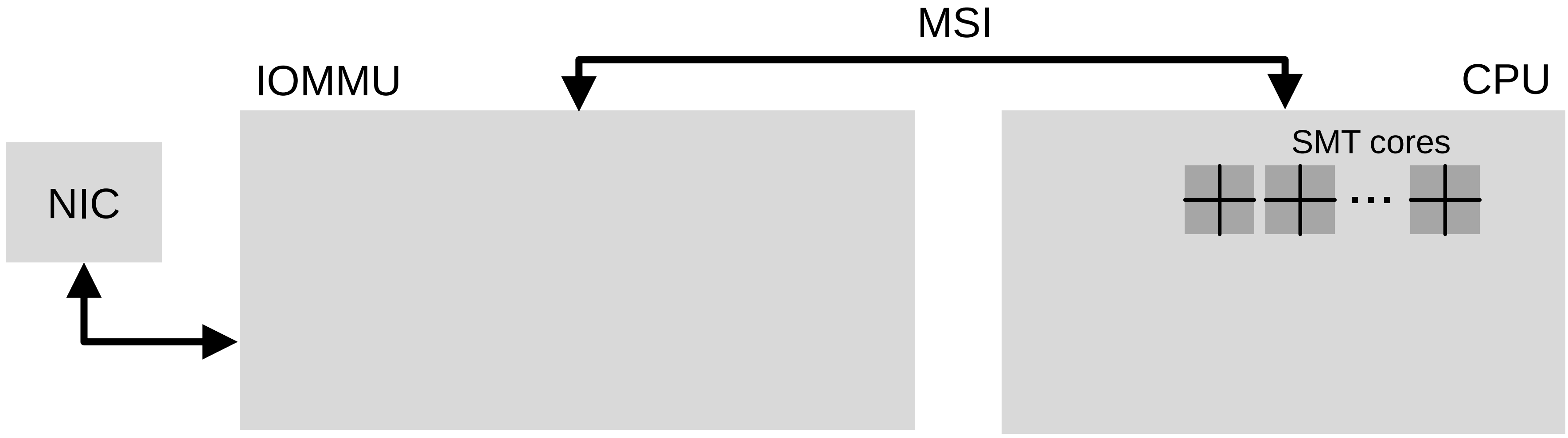
# IOMMUs and RMA



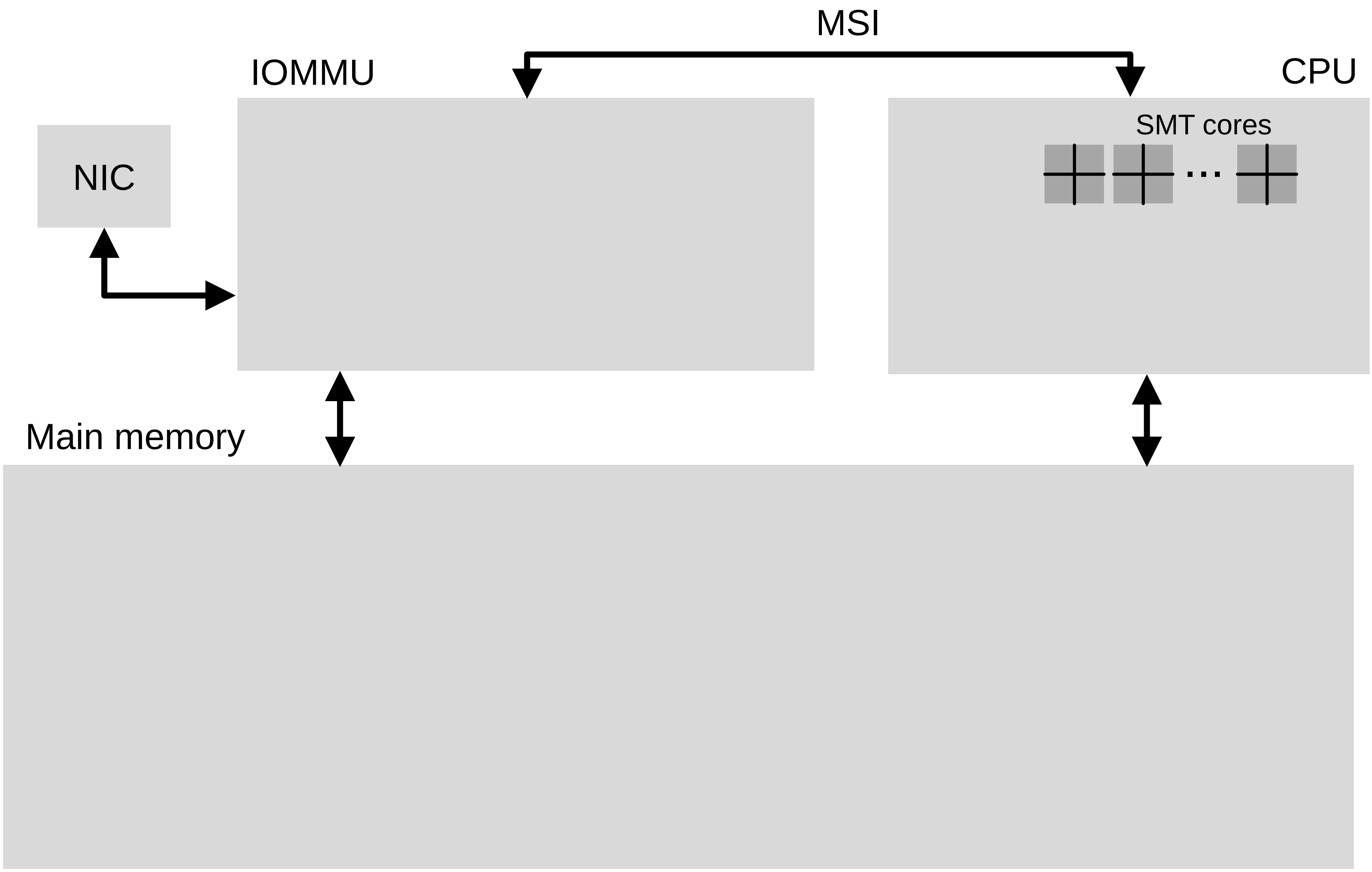
# IOMMUs and RMA



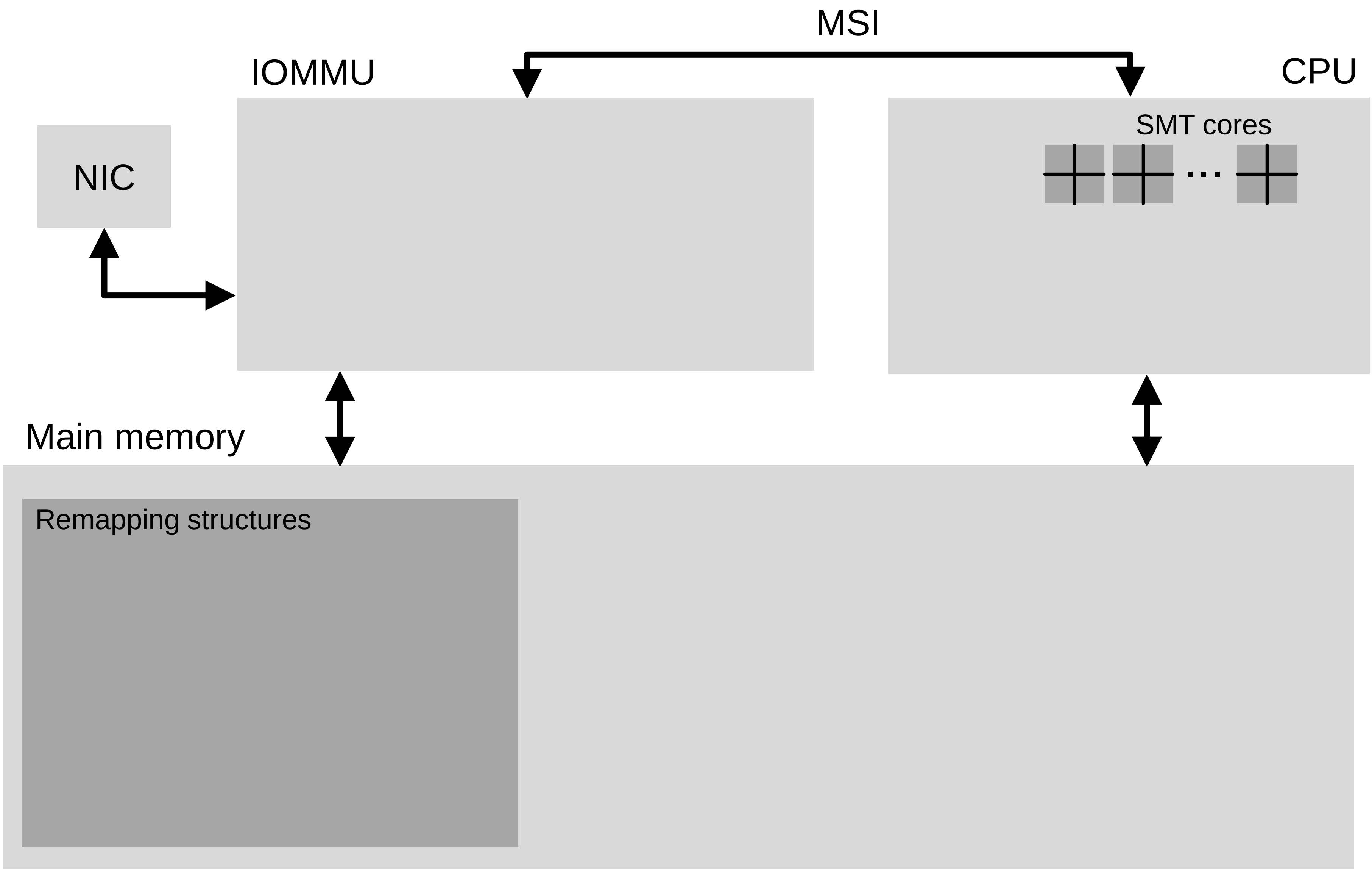
# IOMMUs and RMA



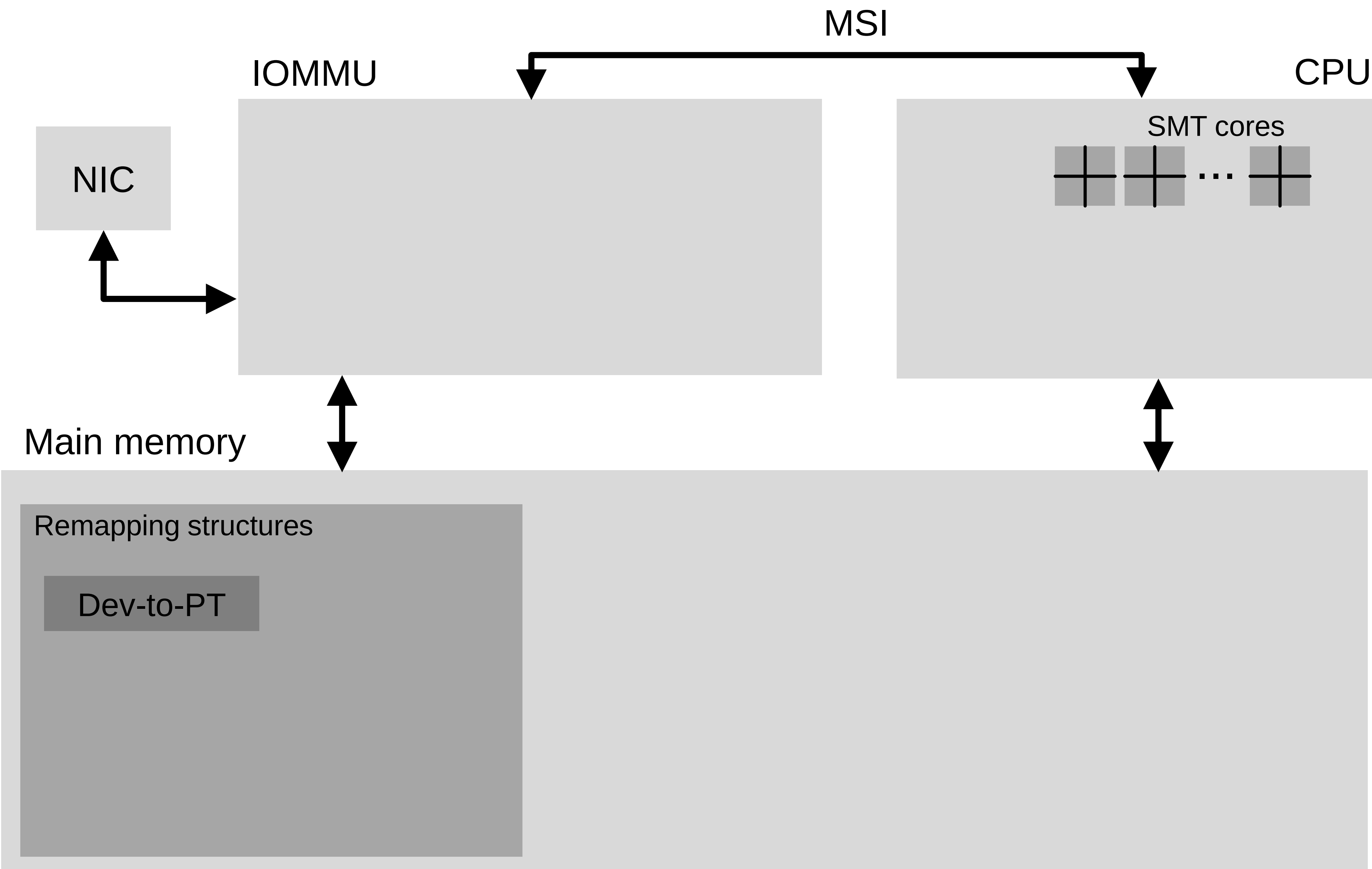
# IOMMUs and RMA



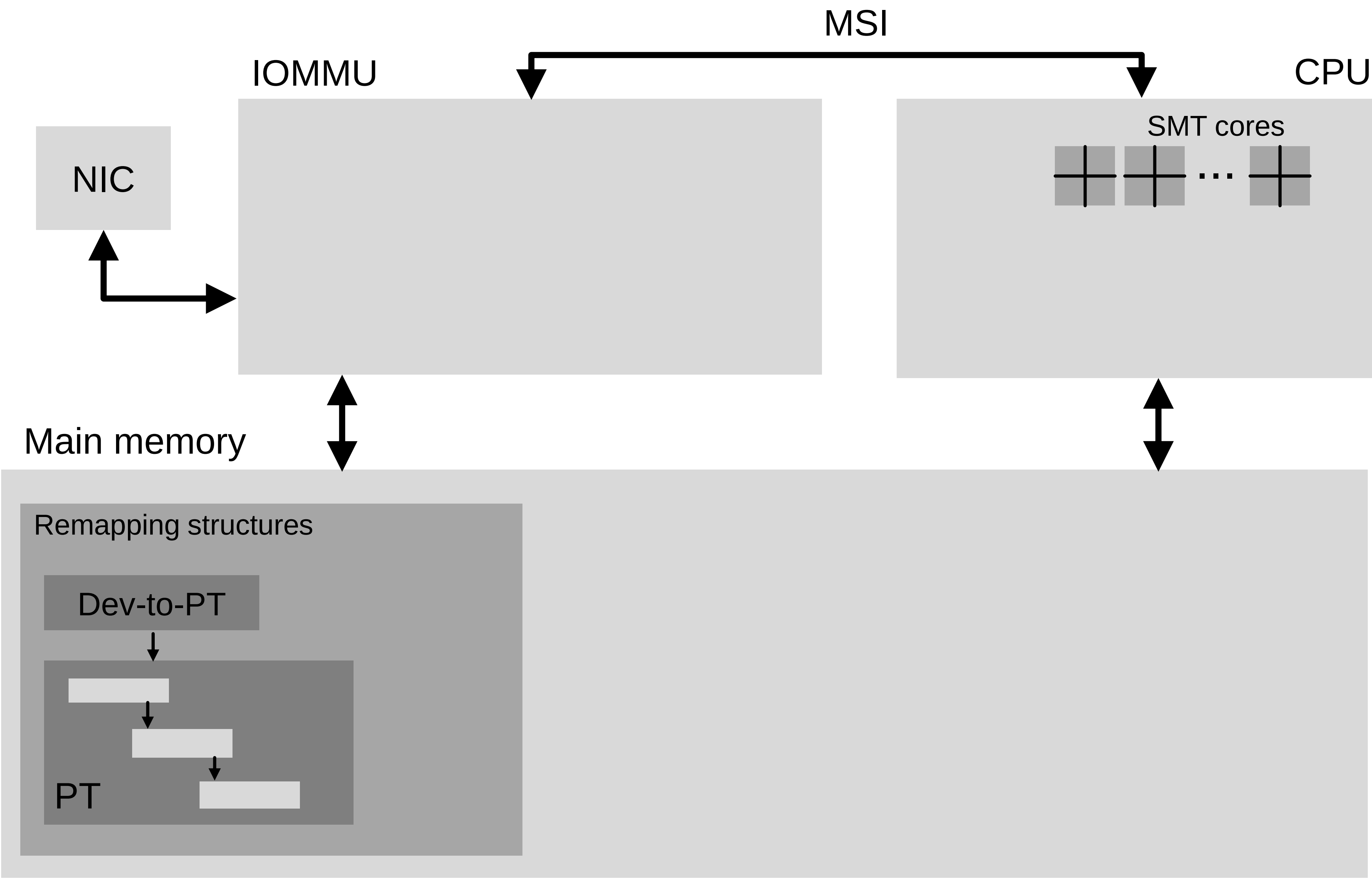
# IOMMUs and RMA



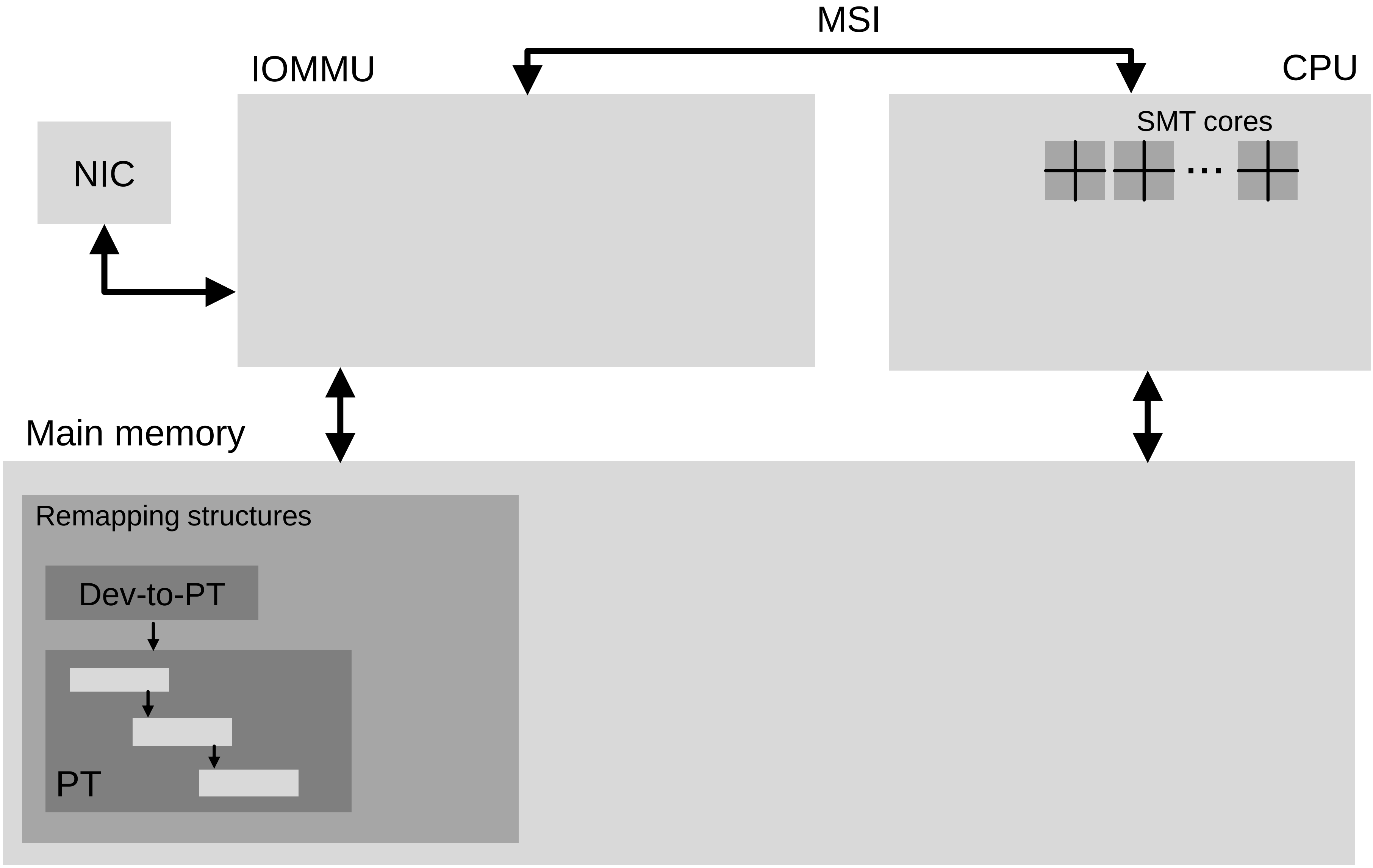
# IOMMUs and RMA



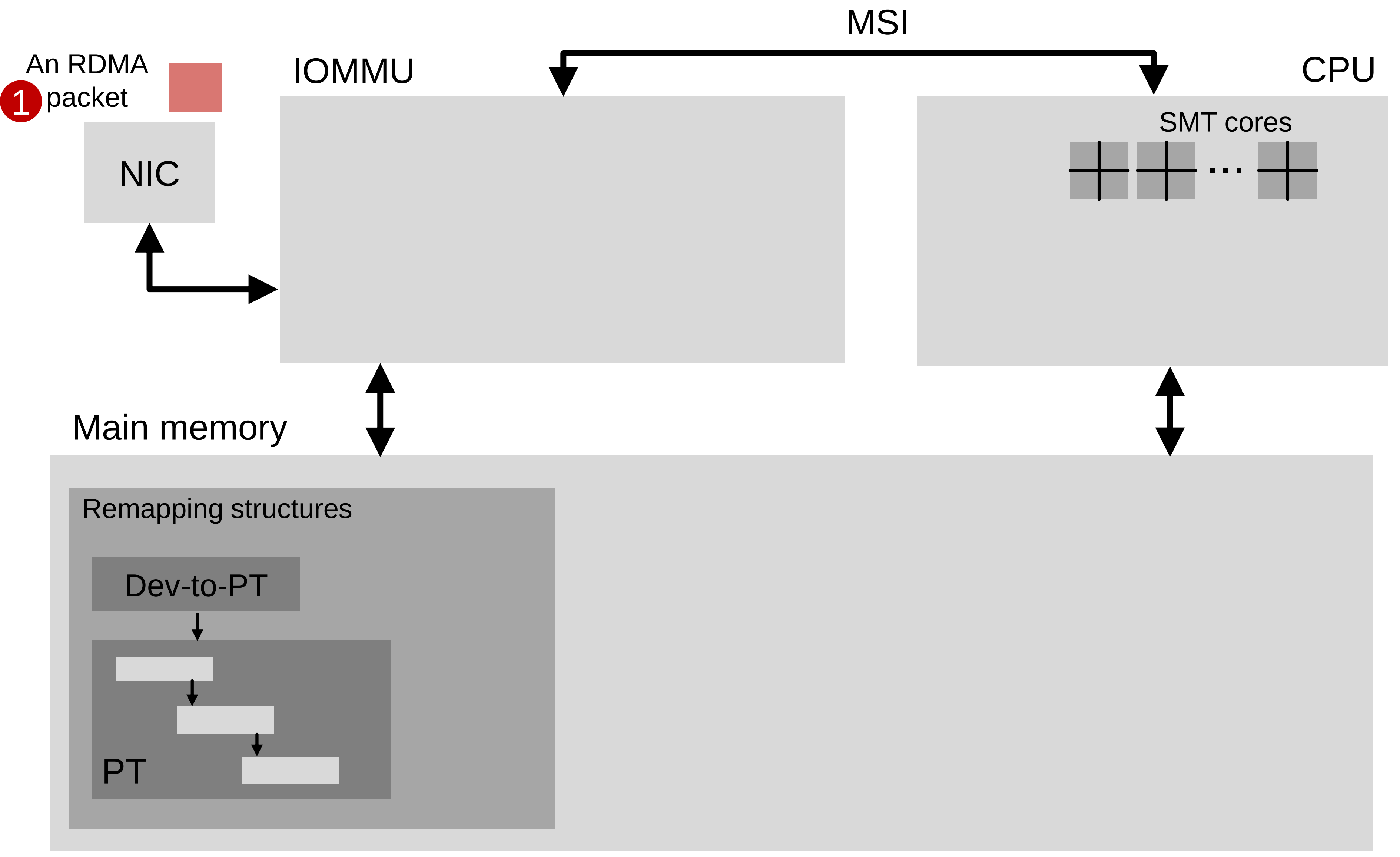
# IOMMUs and RMA



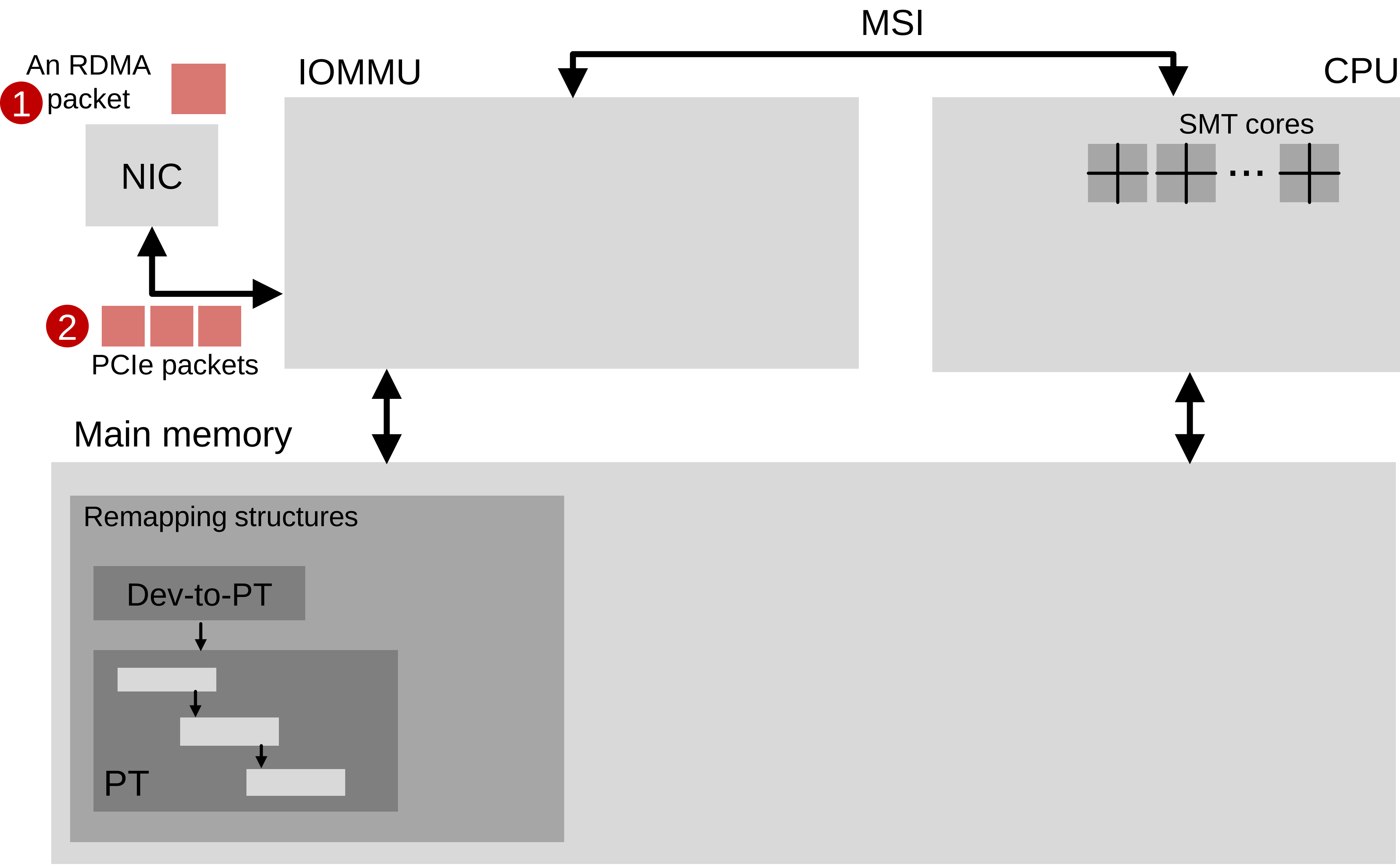
# IOMMUs and RMA



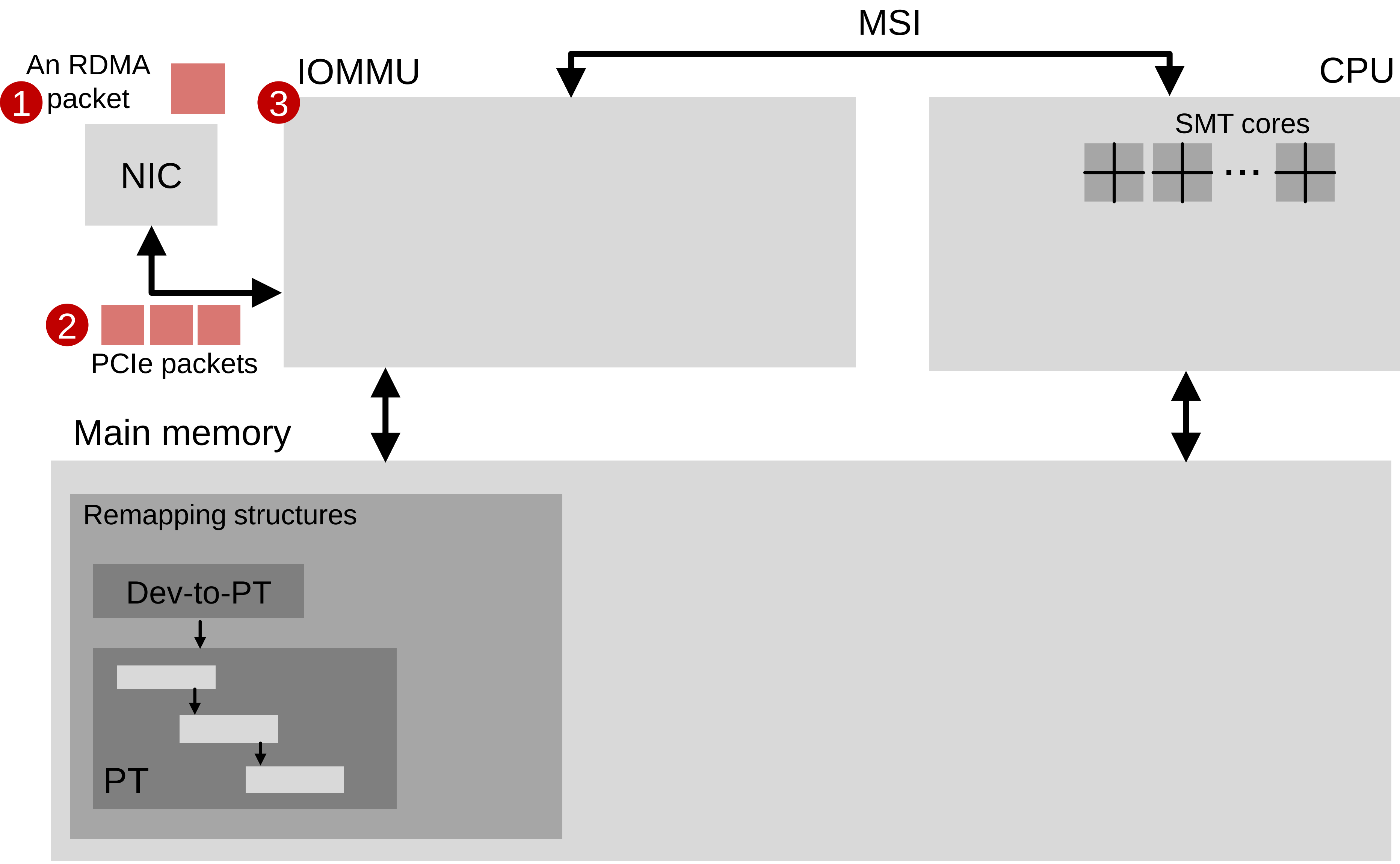
# IOMMUs and RMA



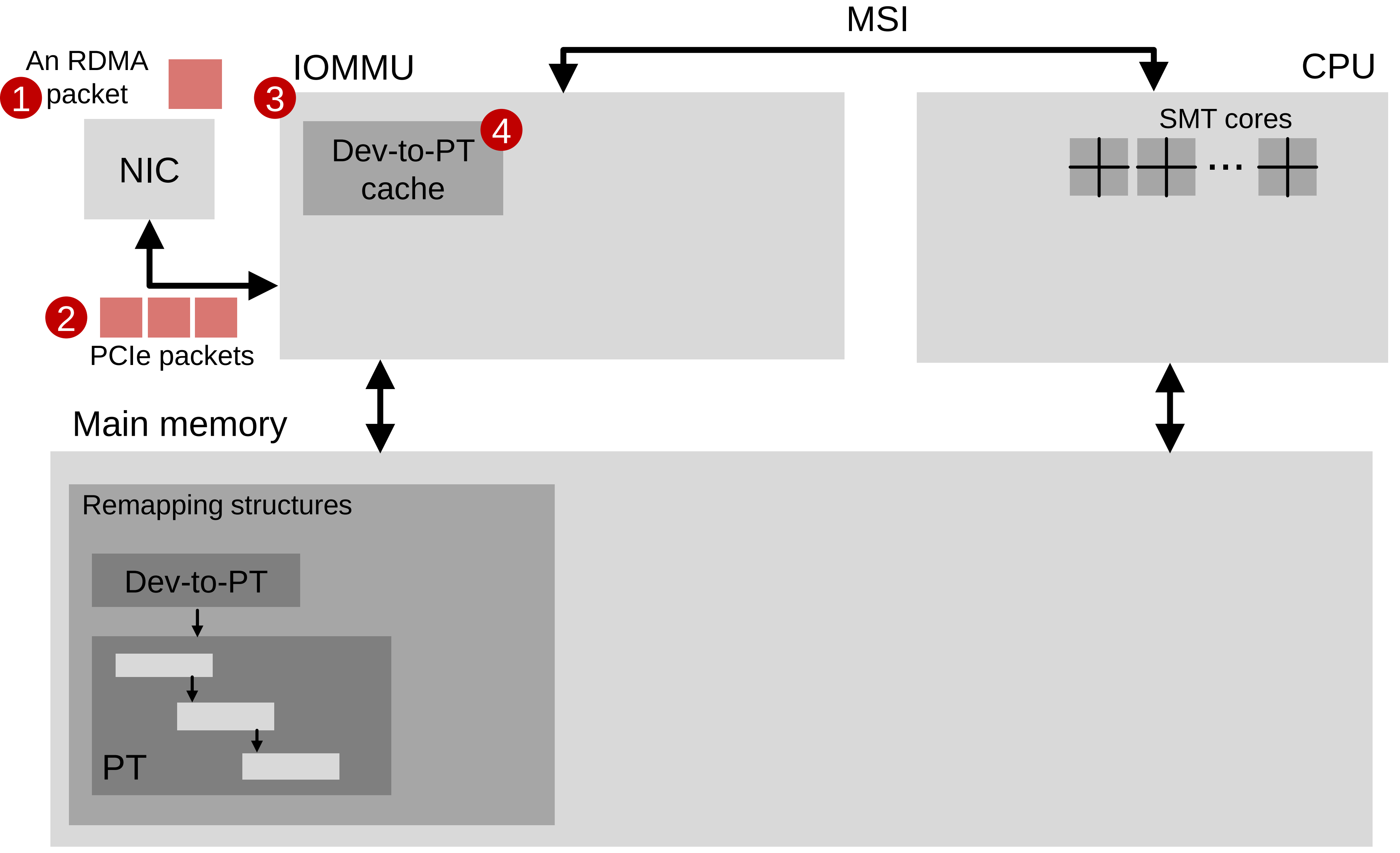
# IOMMUs and RMA



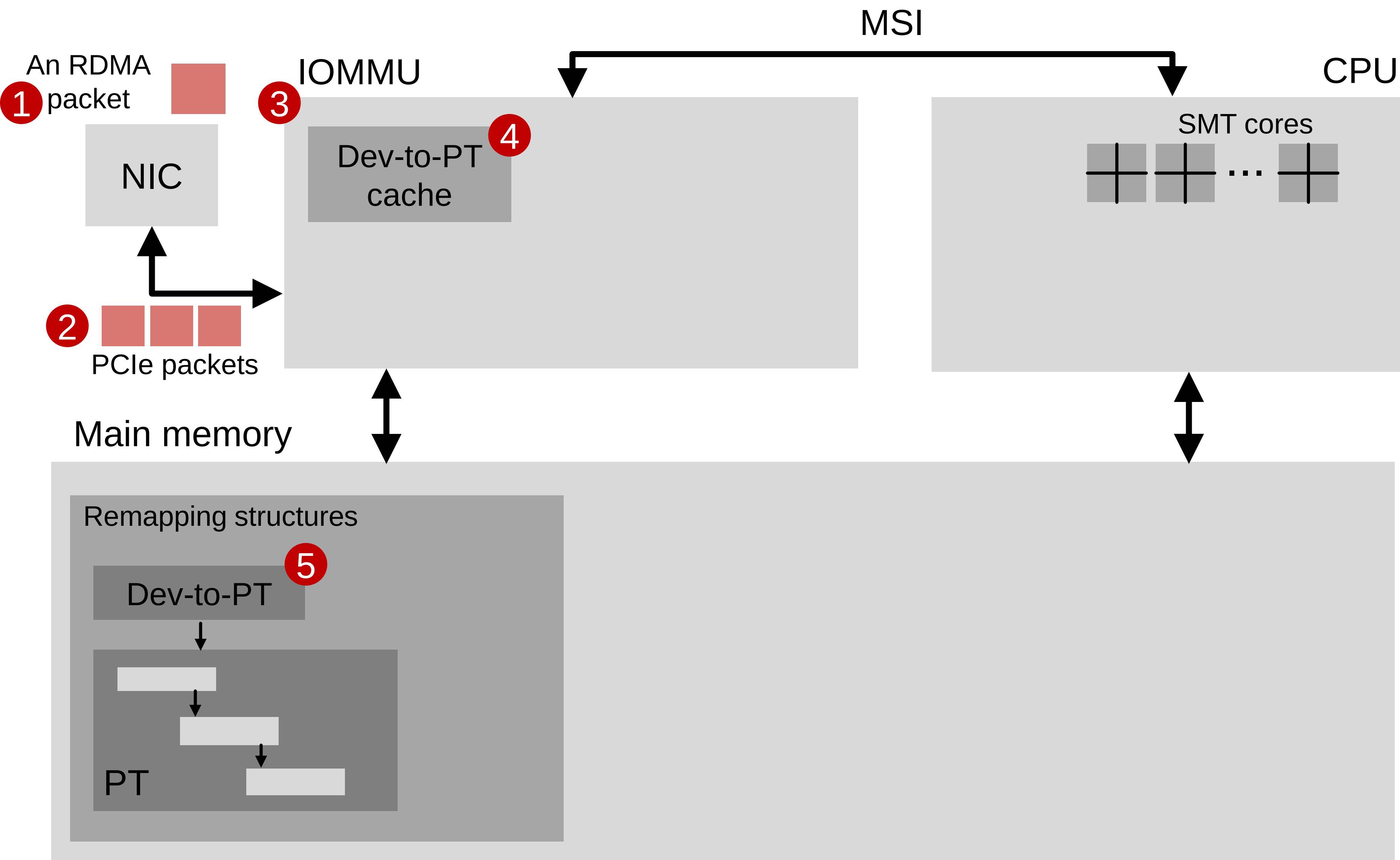
# IOMMUs and RMA



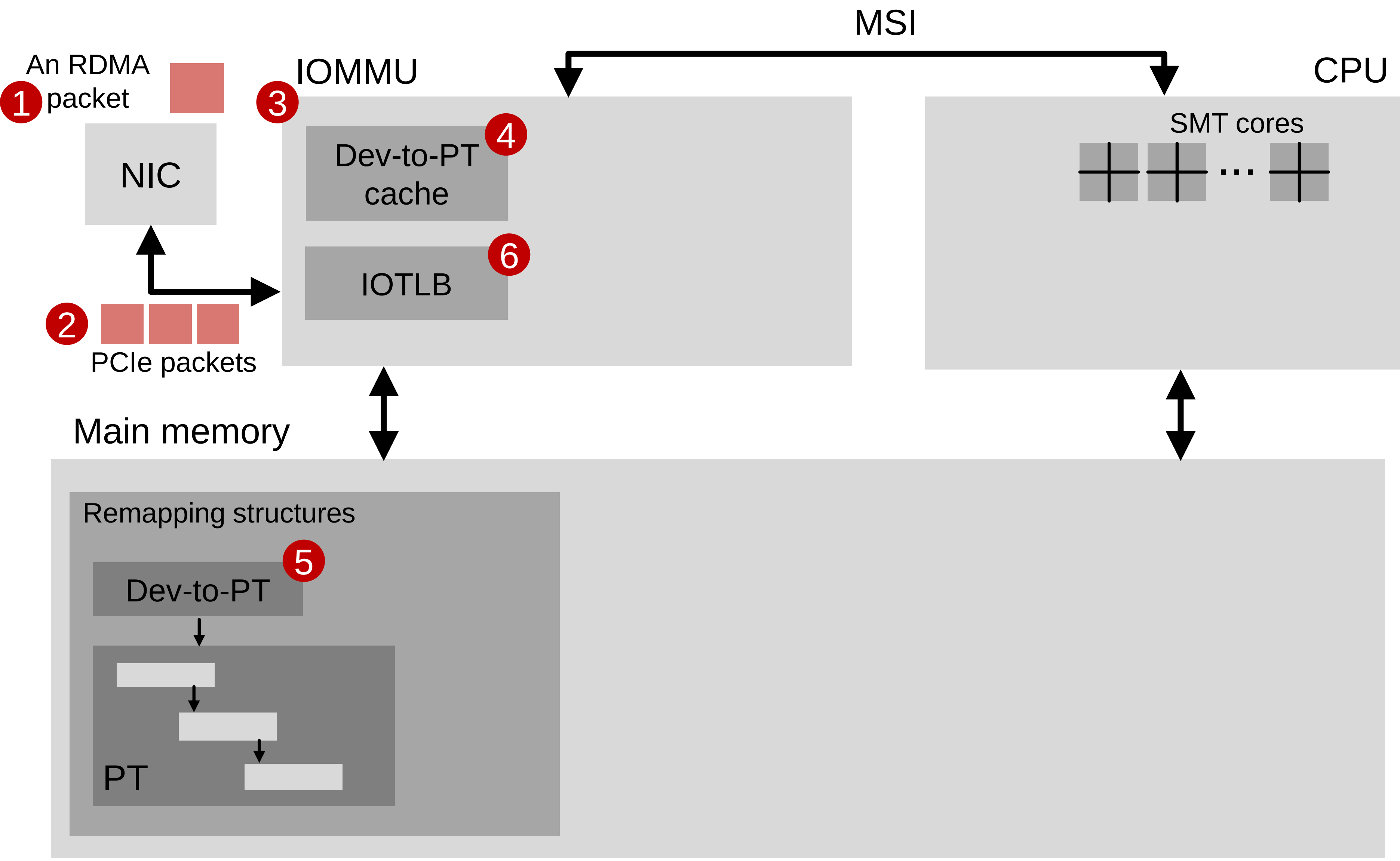
# IOMMUS AND RMA



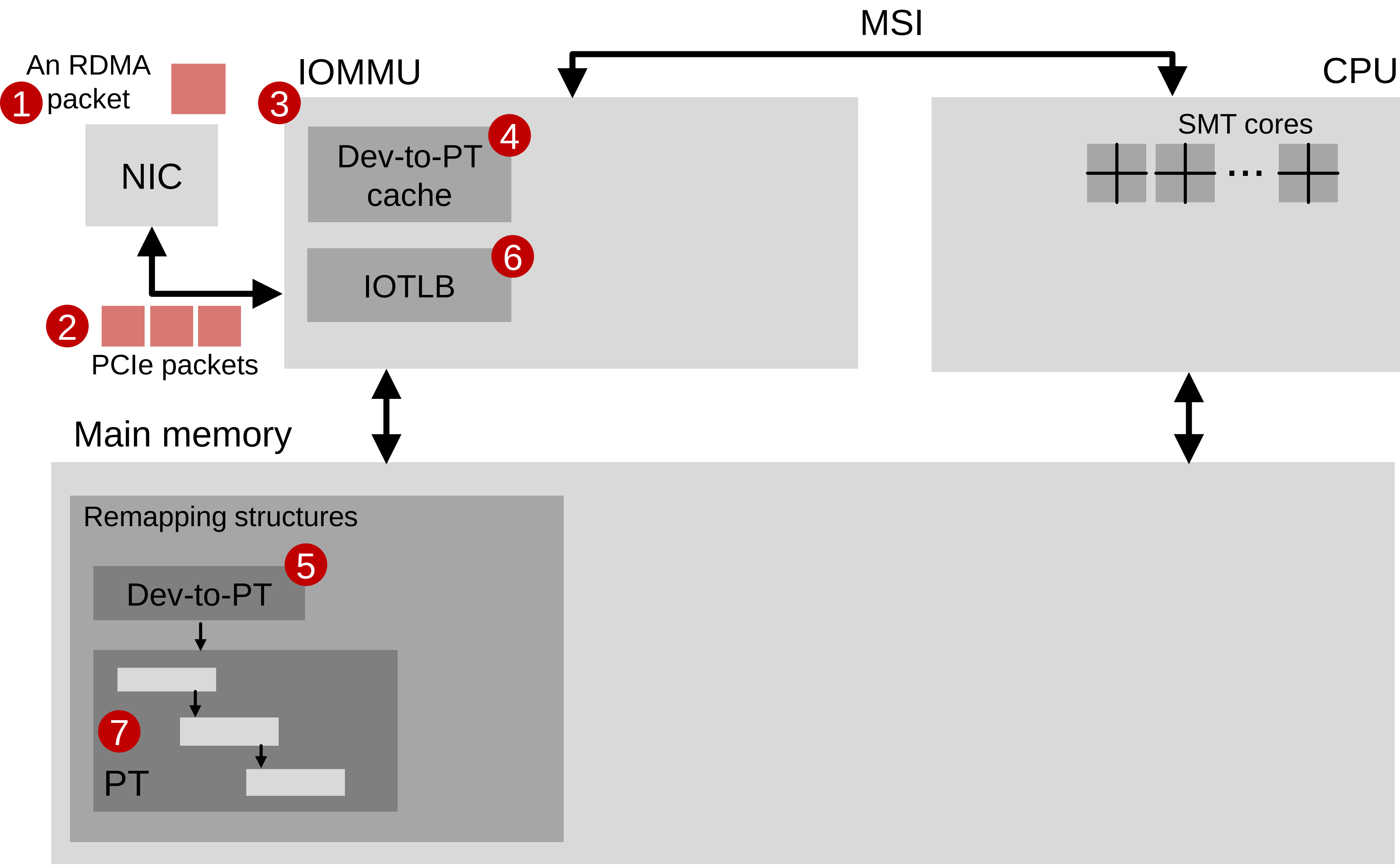
# IOMMUs and RMA



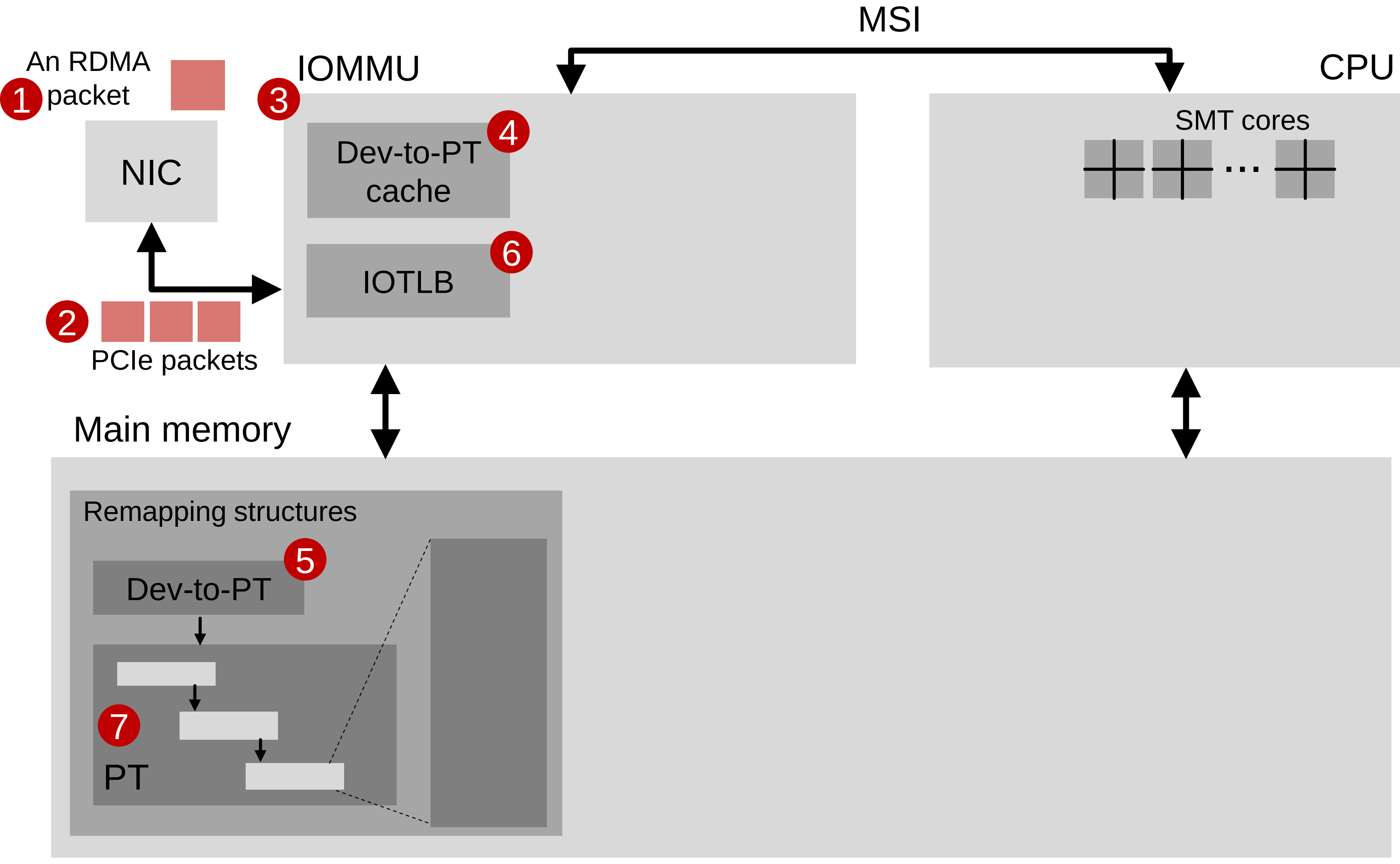
# IOMMUs and RMA



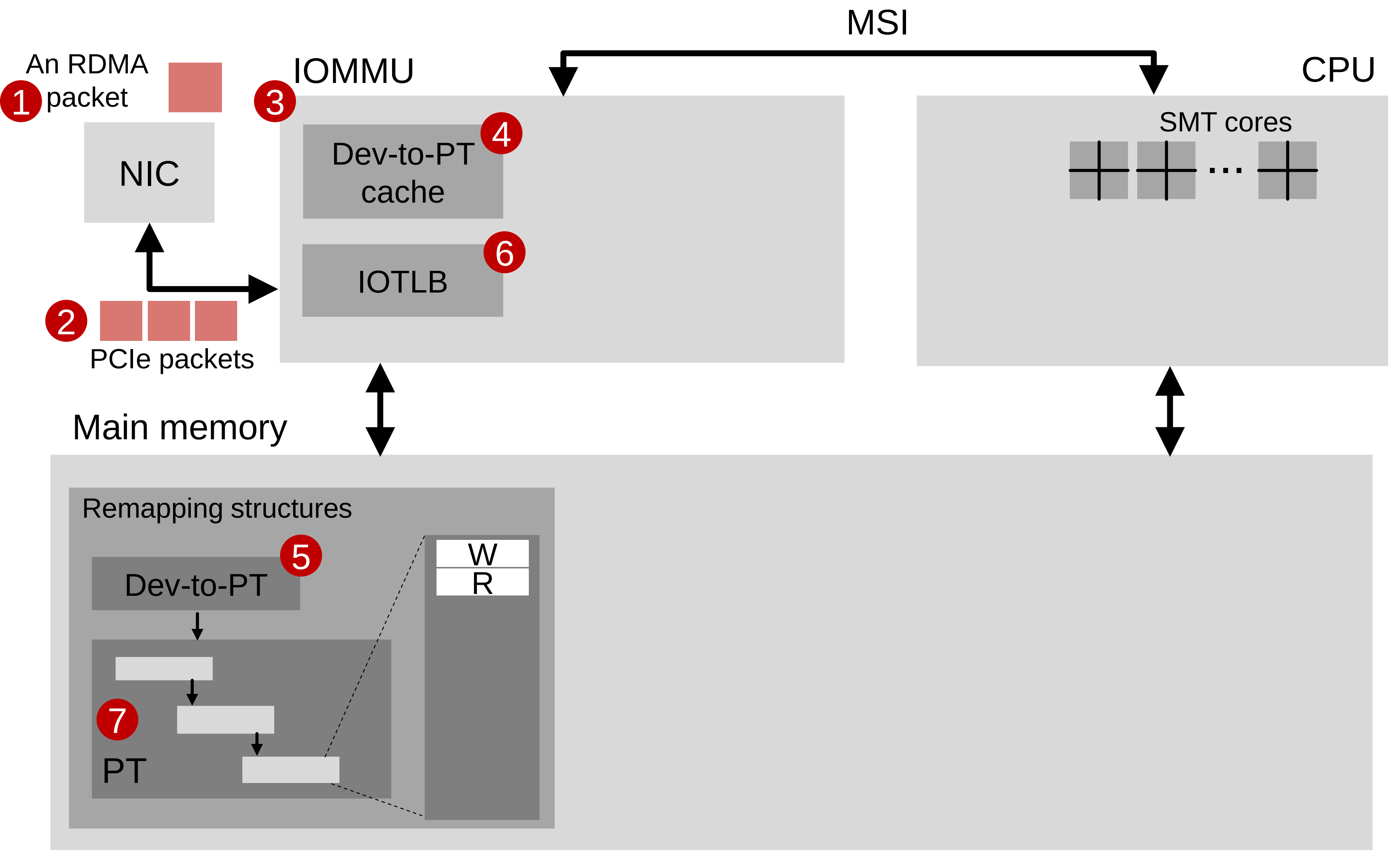
# IOMMUs and RMA



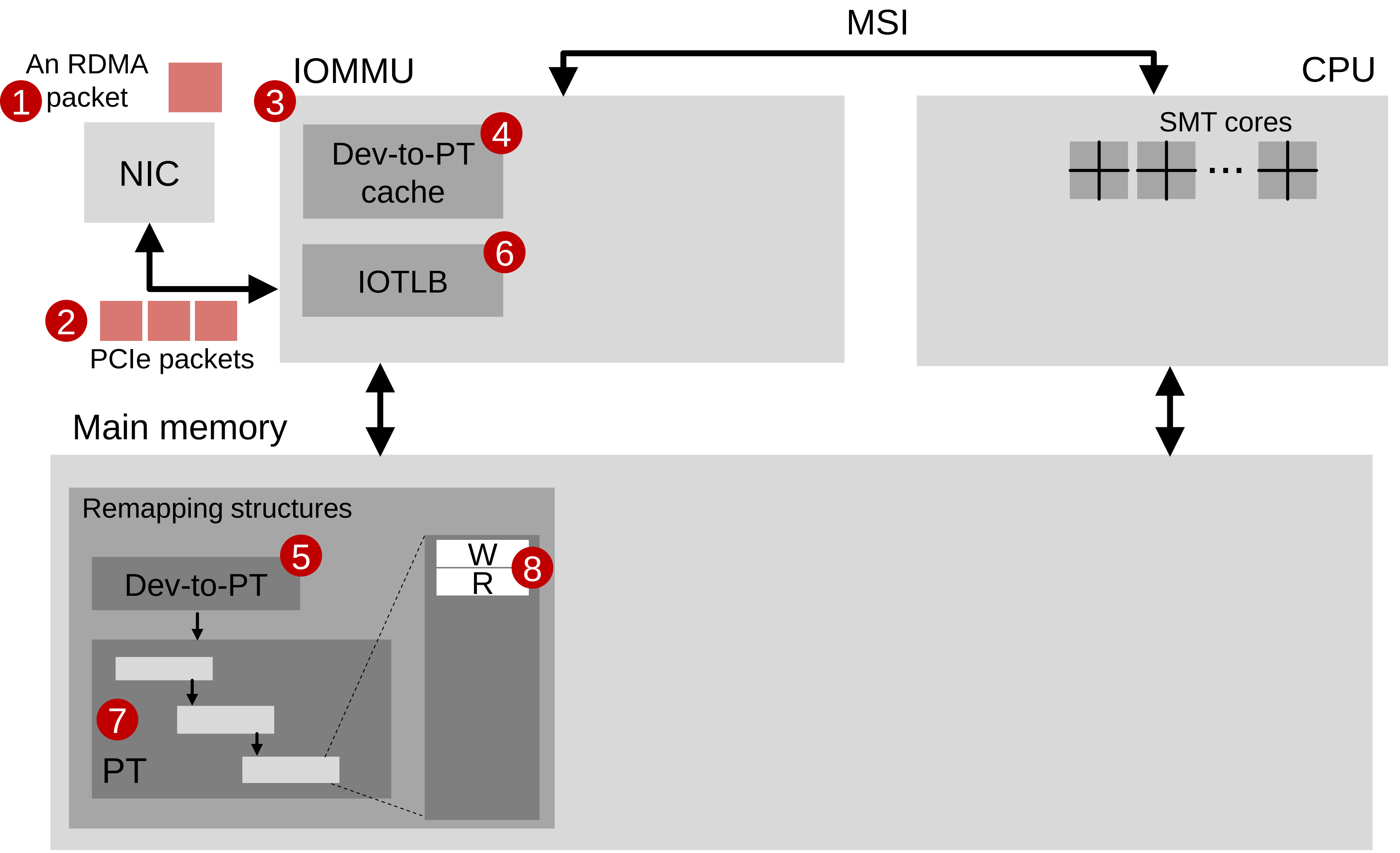
# IOMMUs and RMA



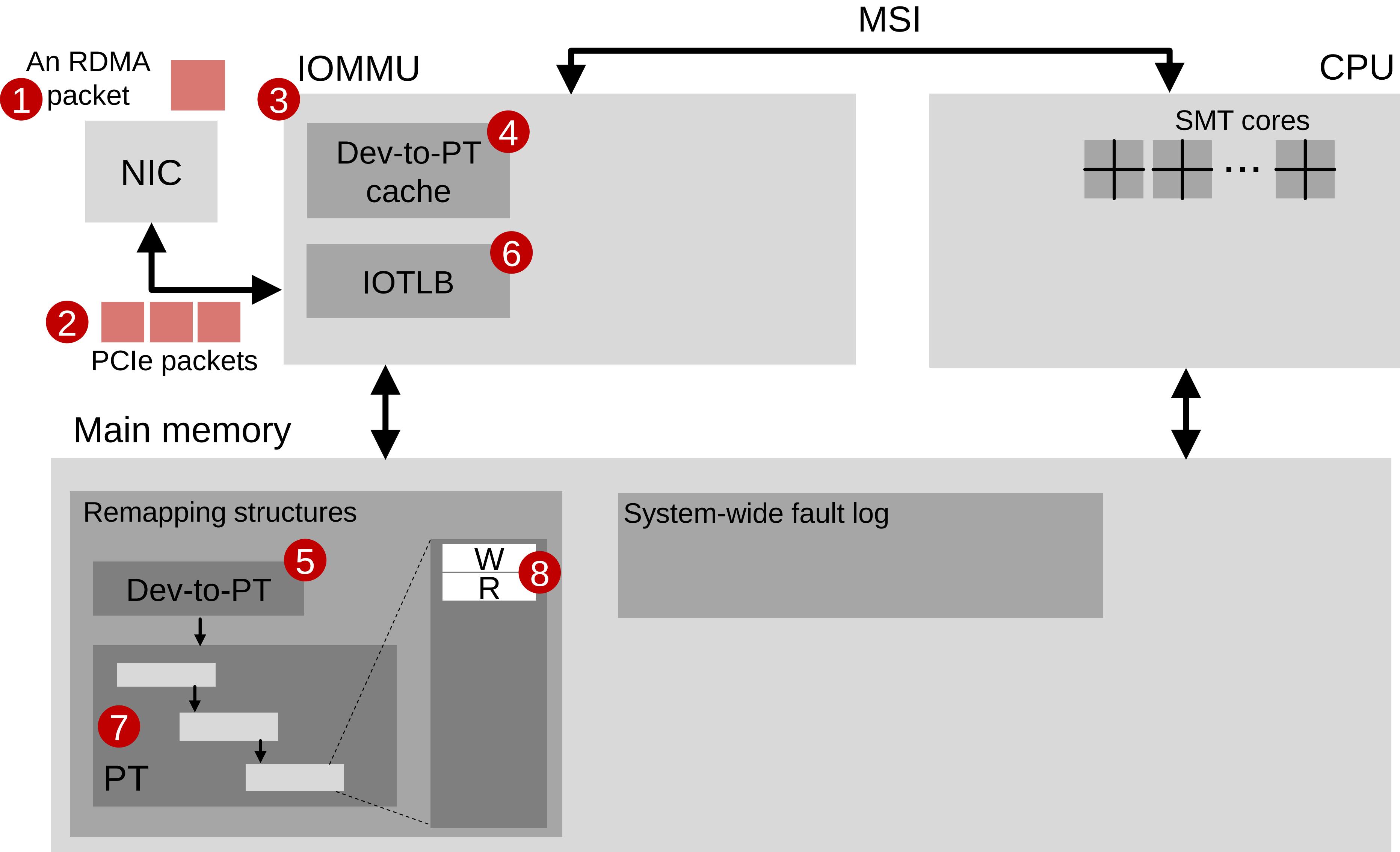
# IOMMUs and RMA



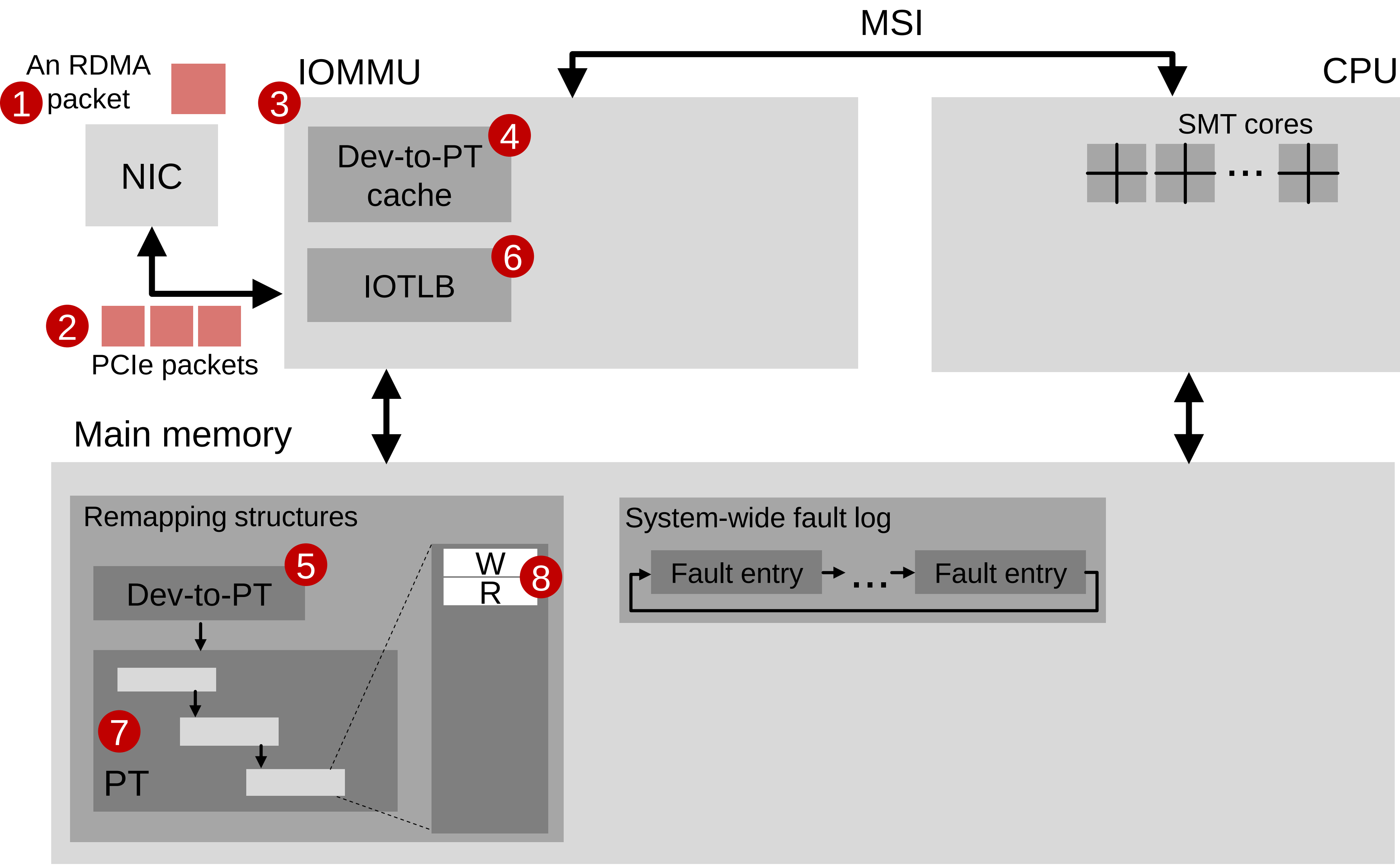
# IOMMUs and RMA



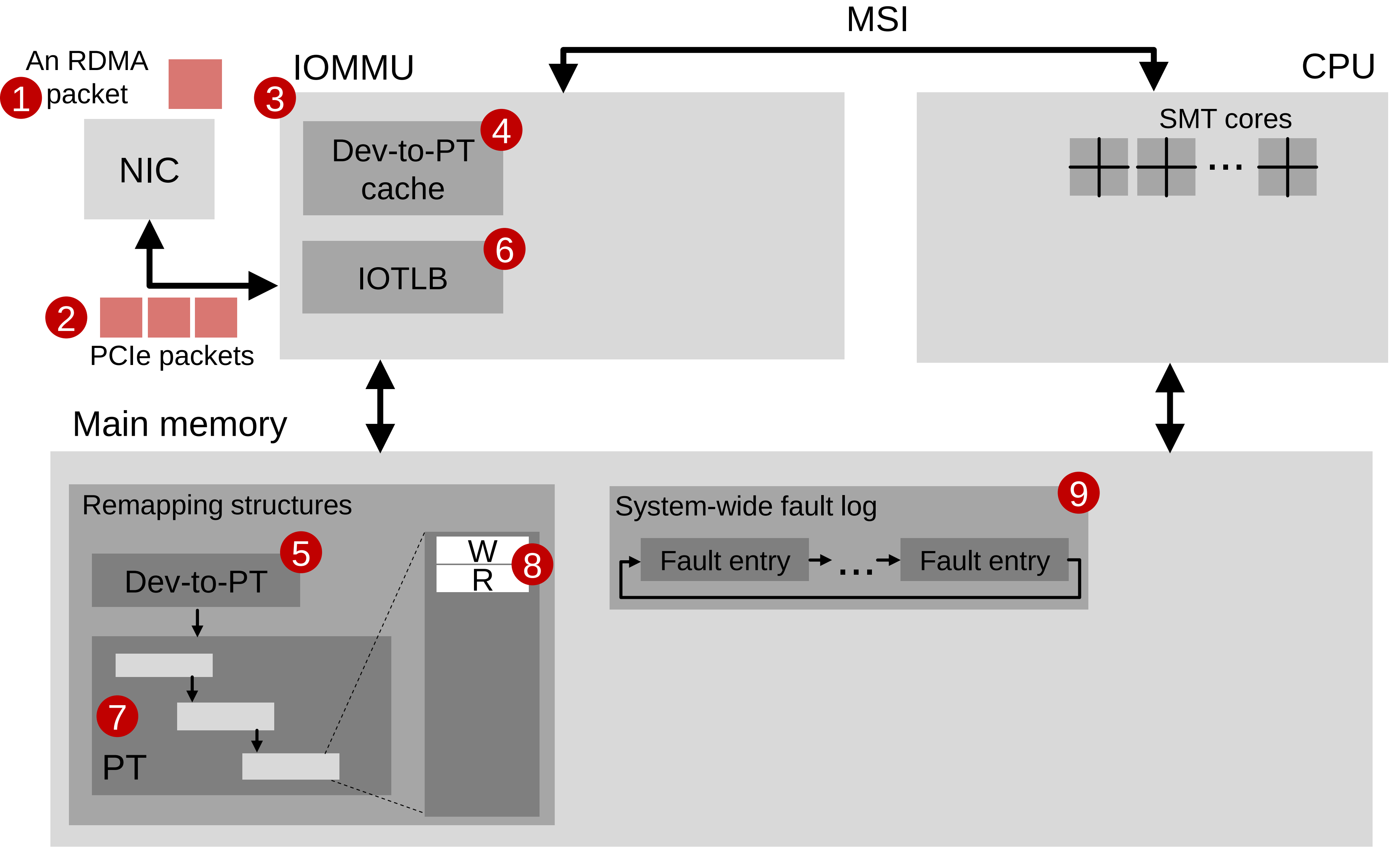
# IOMMUs and RMA



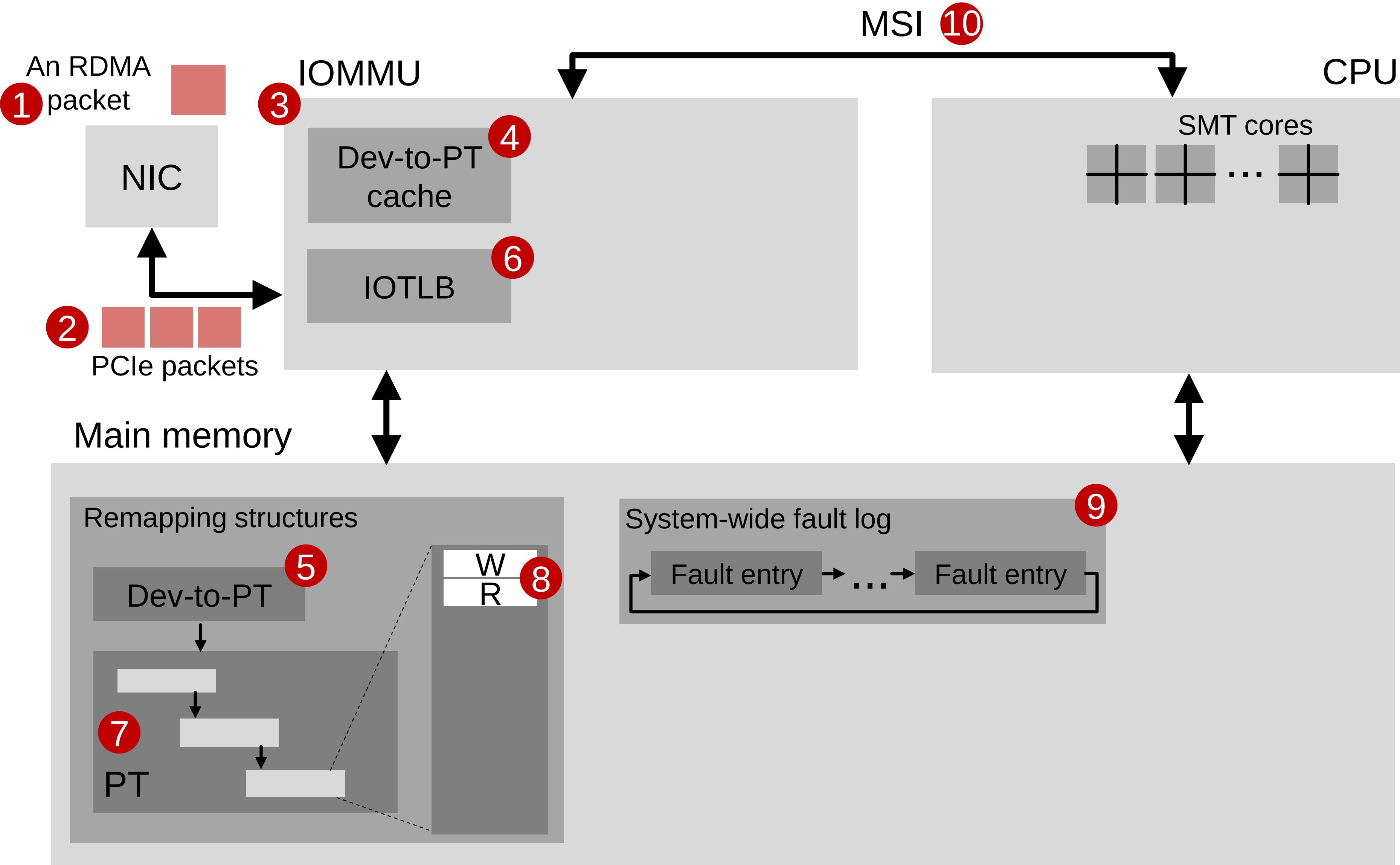
# IOMMUs and RMA



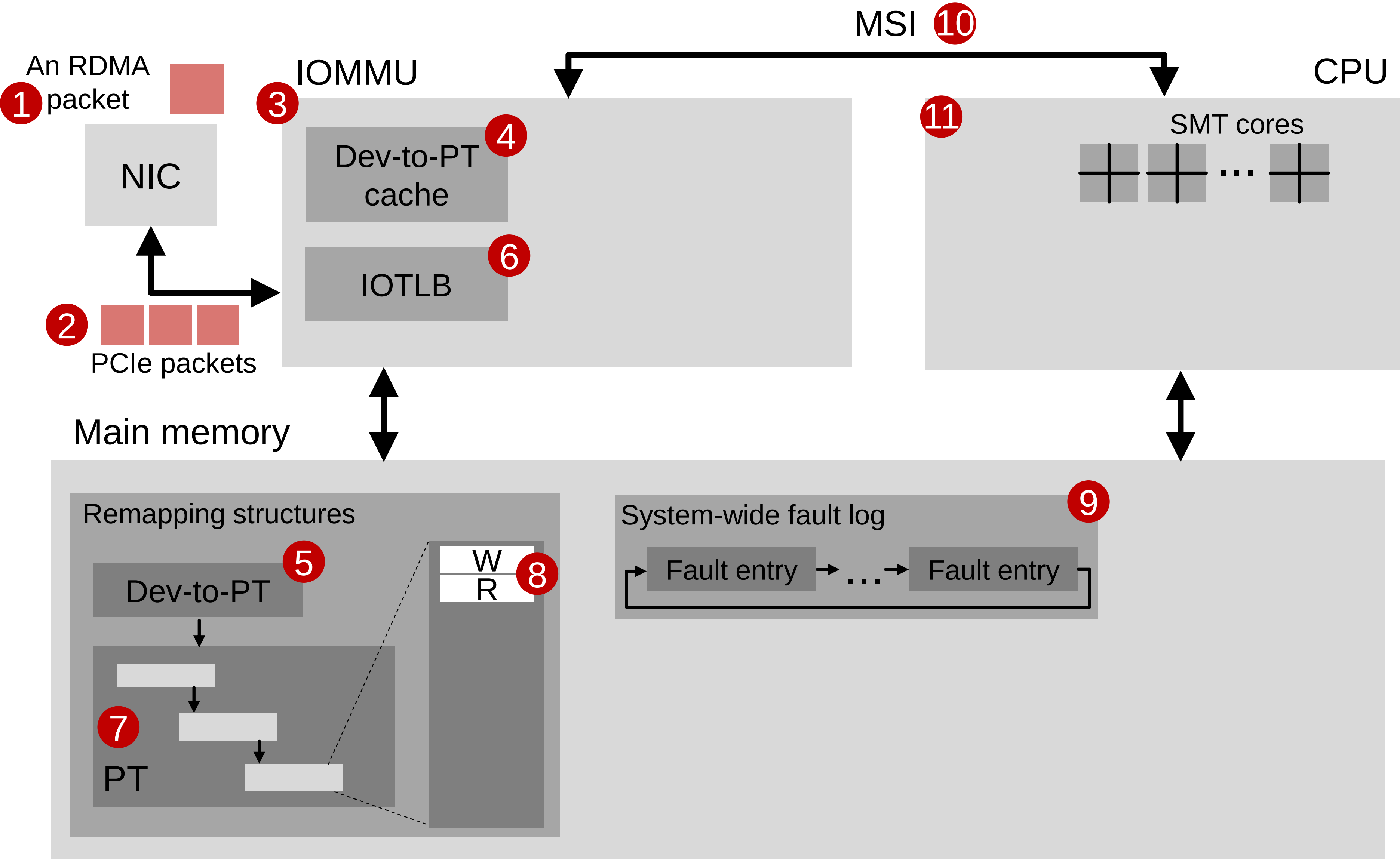
# IOMMUs and RMA



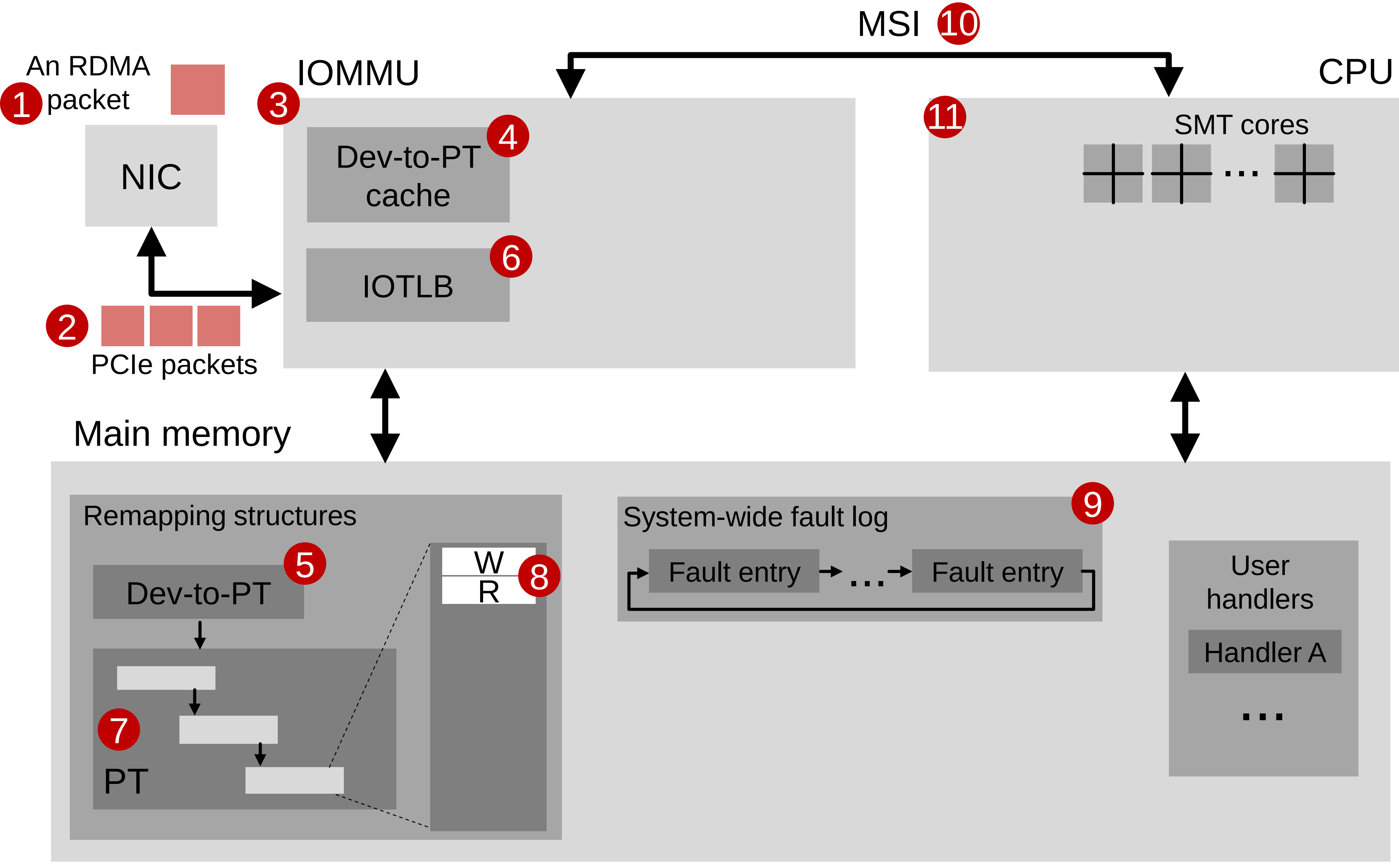
# IOMMUs and RMA



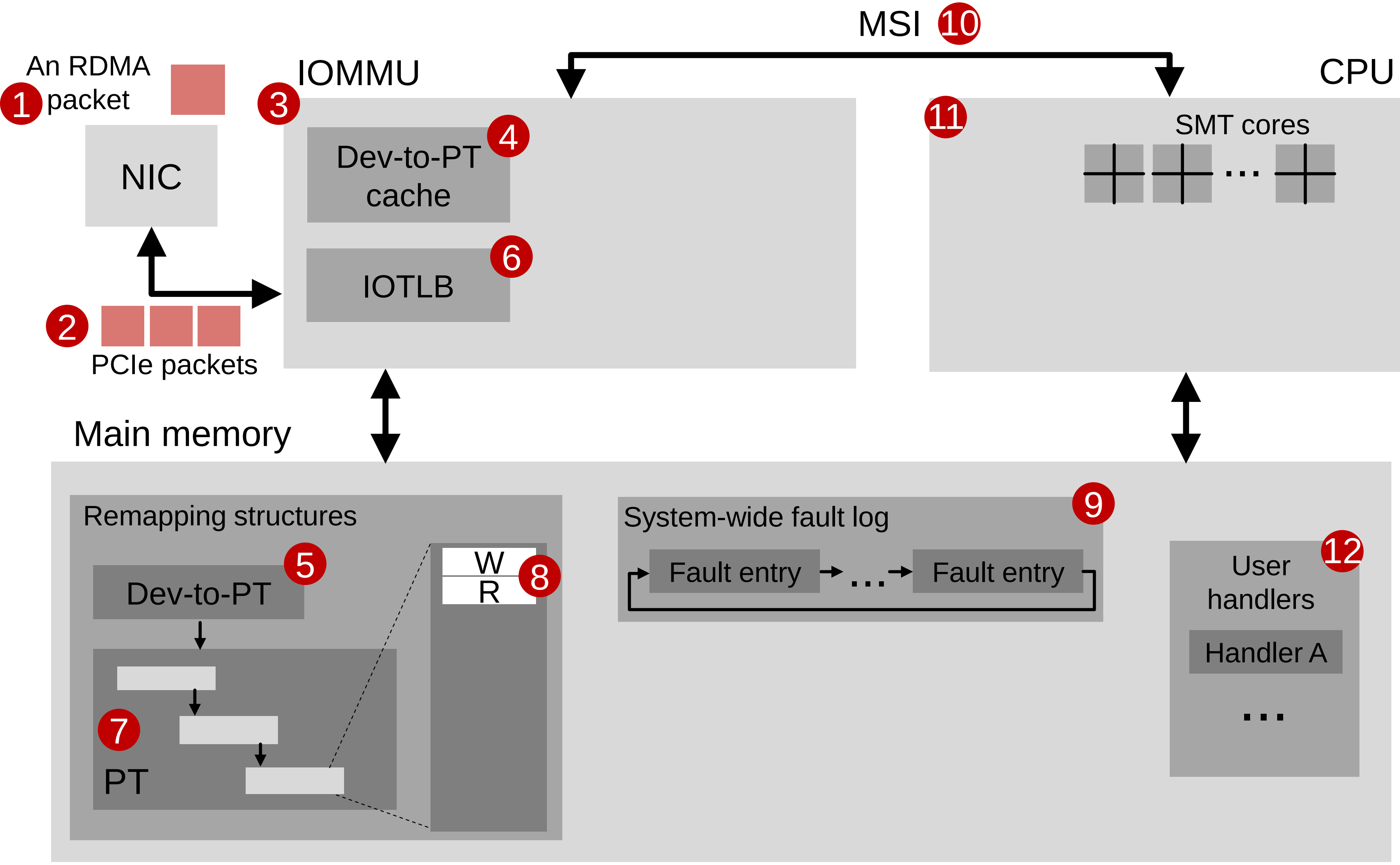
# IOMMUs and RMA



# IOMMUs and RMA

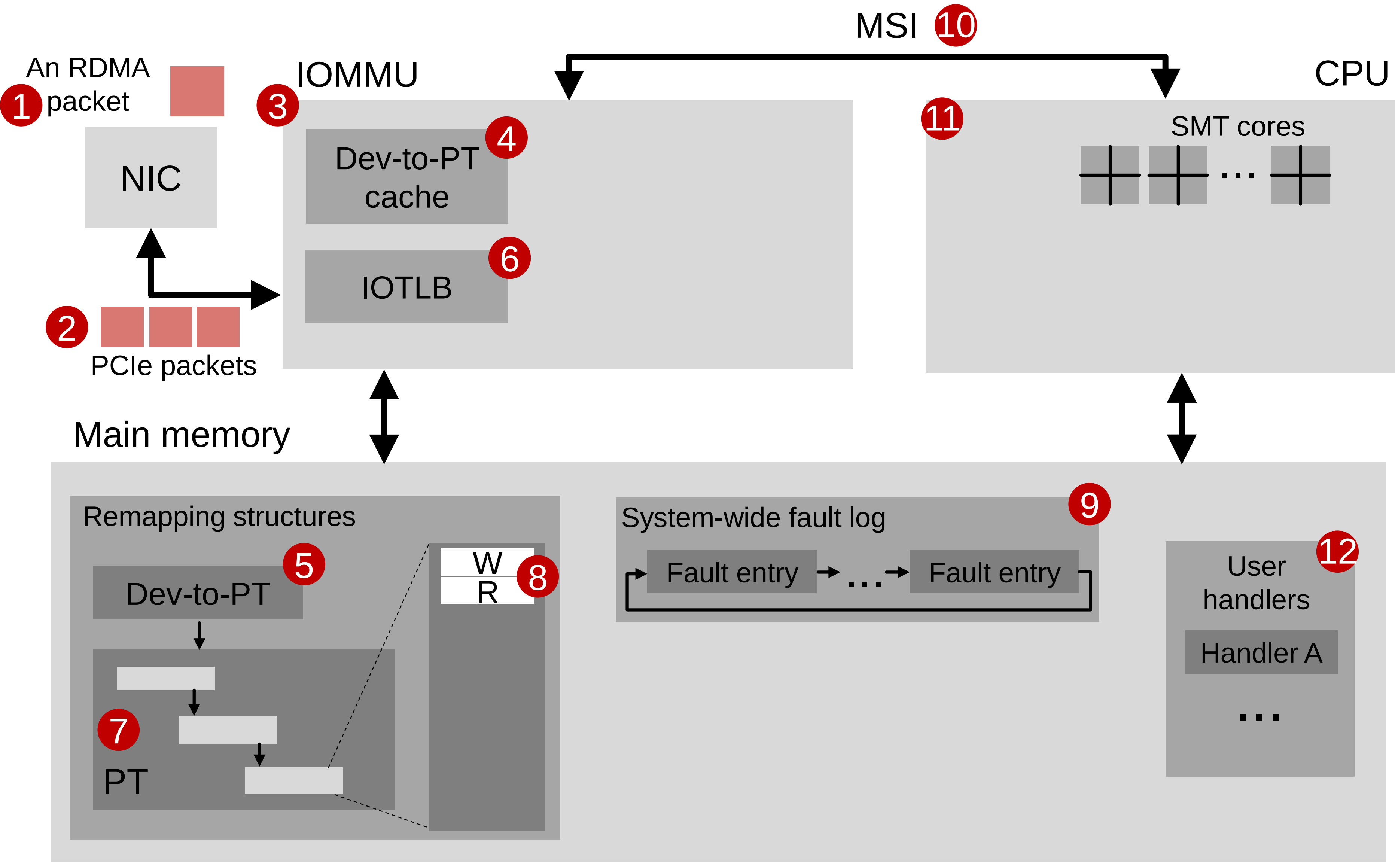


# IOMMUs and RMA



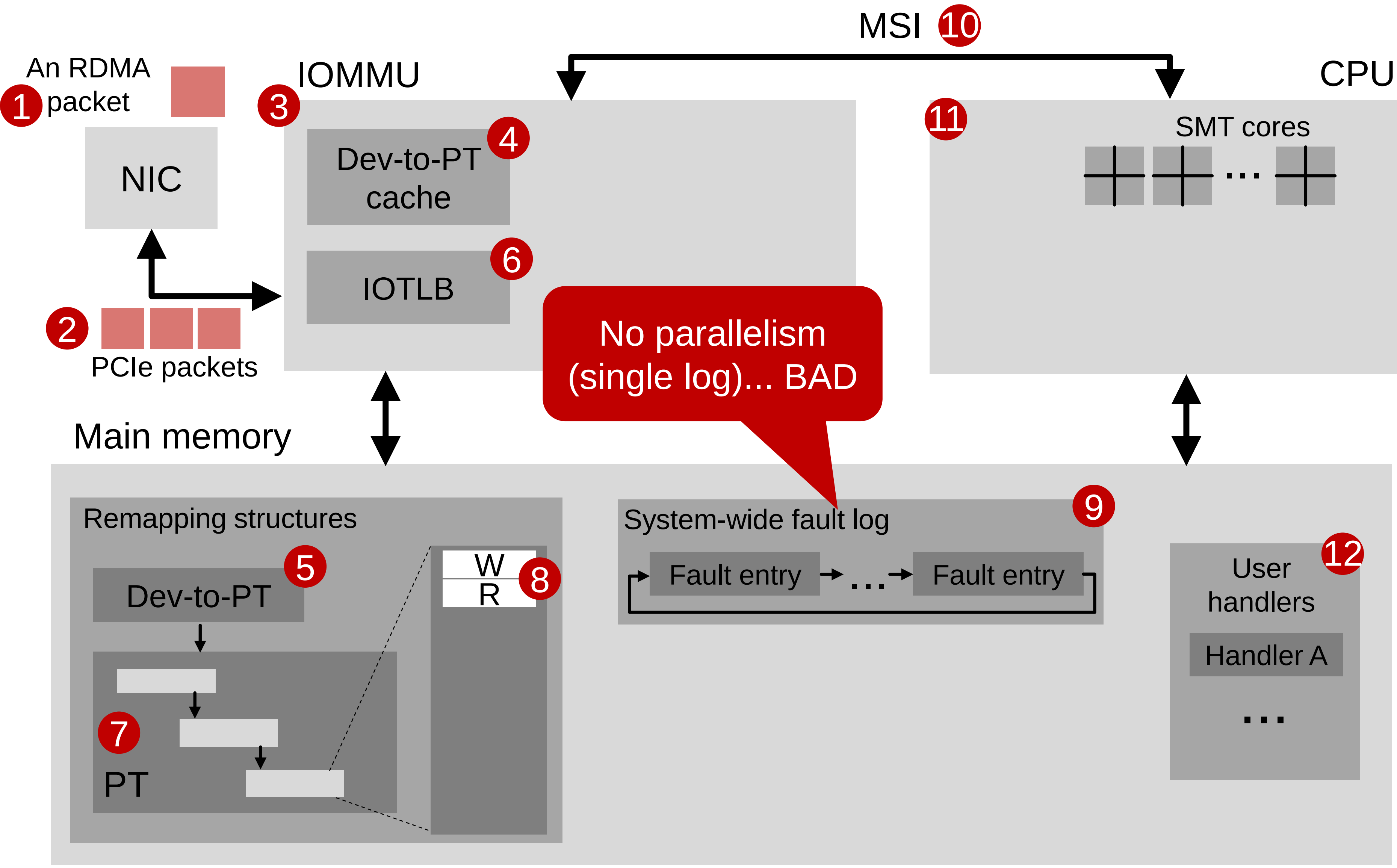
# IOMMUs and RMA

We could use it somehow. But…



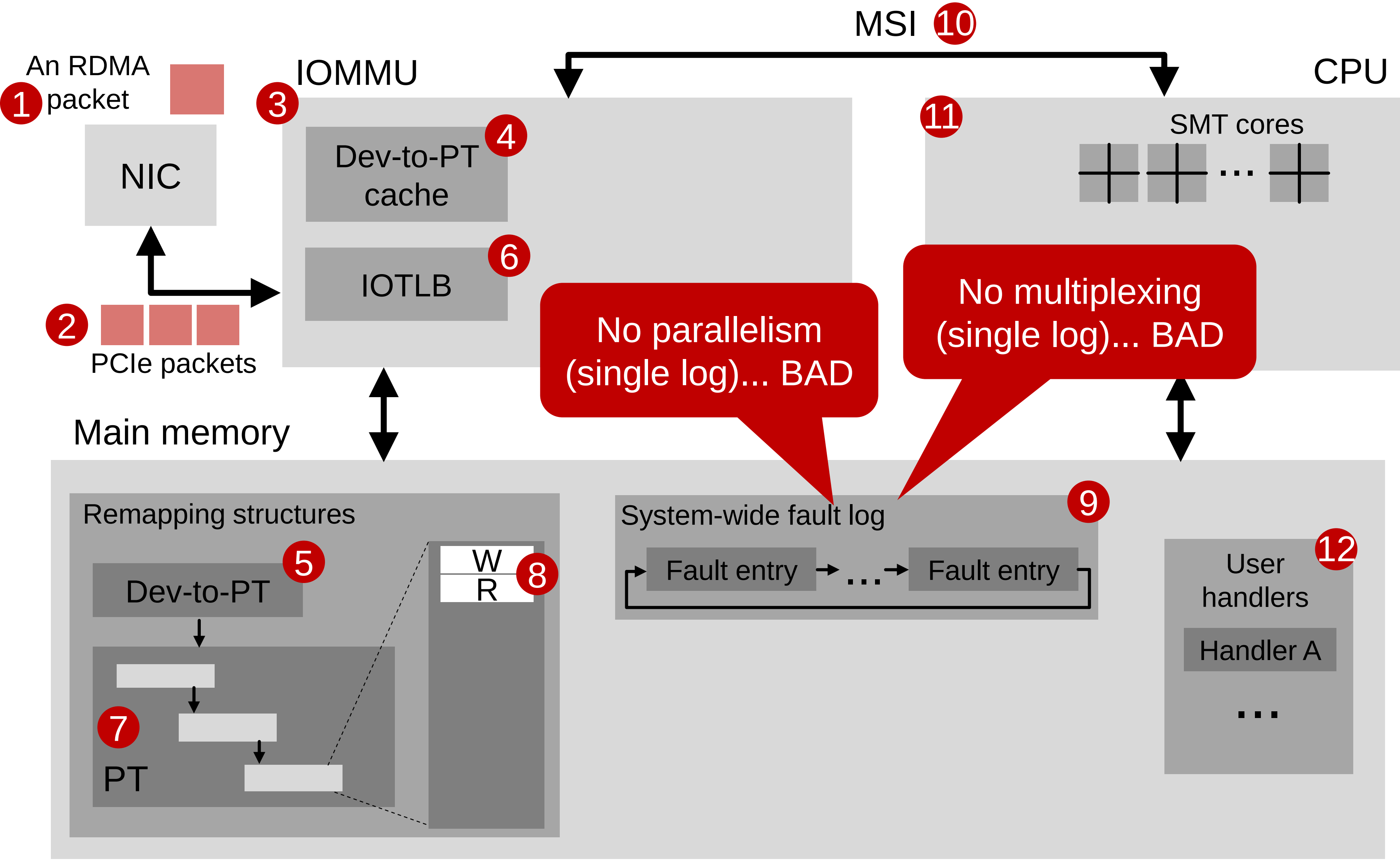
# IOMMUs and RMA

We could use it somehow. But…



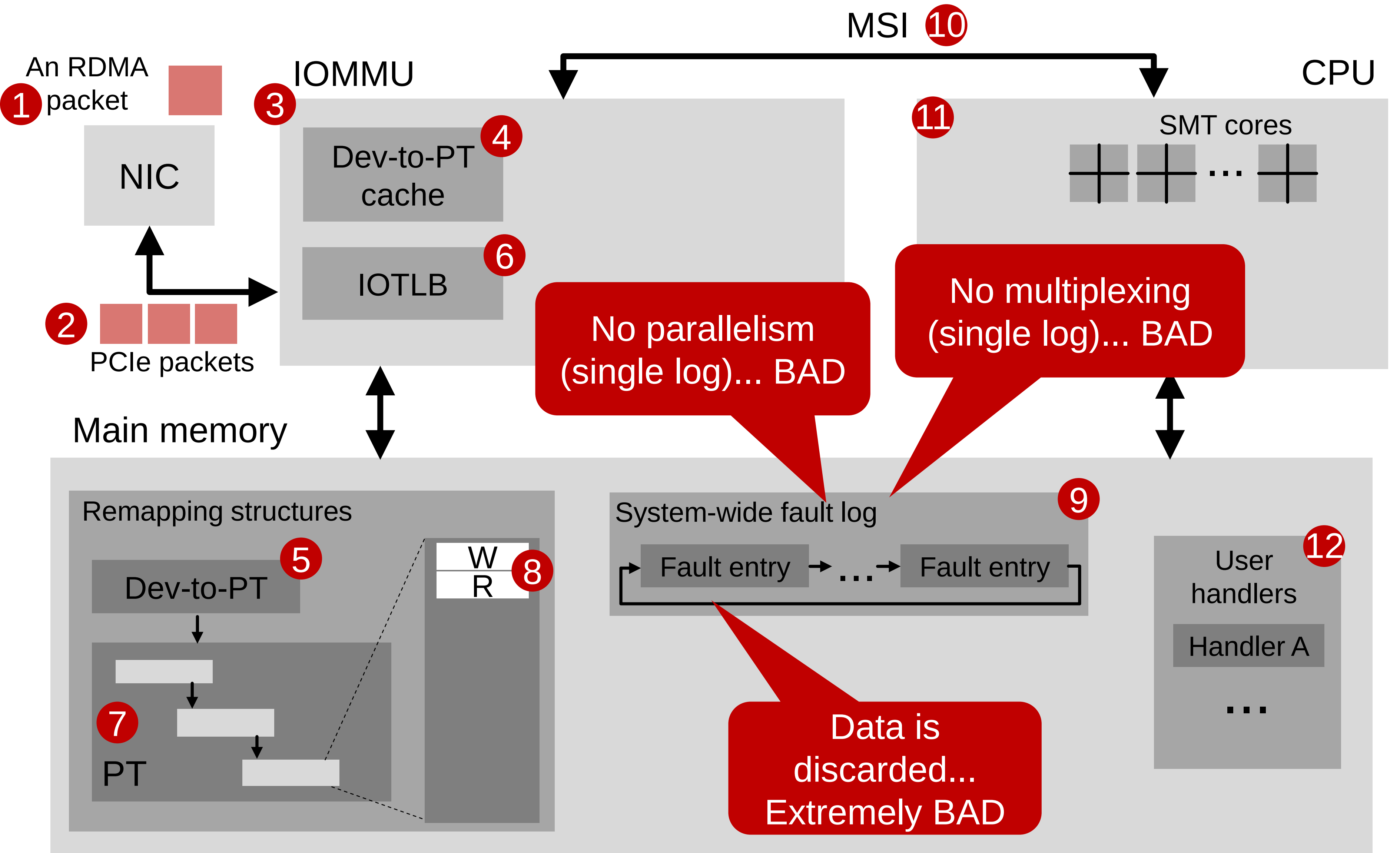
# IOMMUs and RMA

We could use it somehow. But…

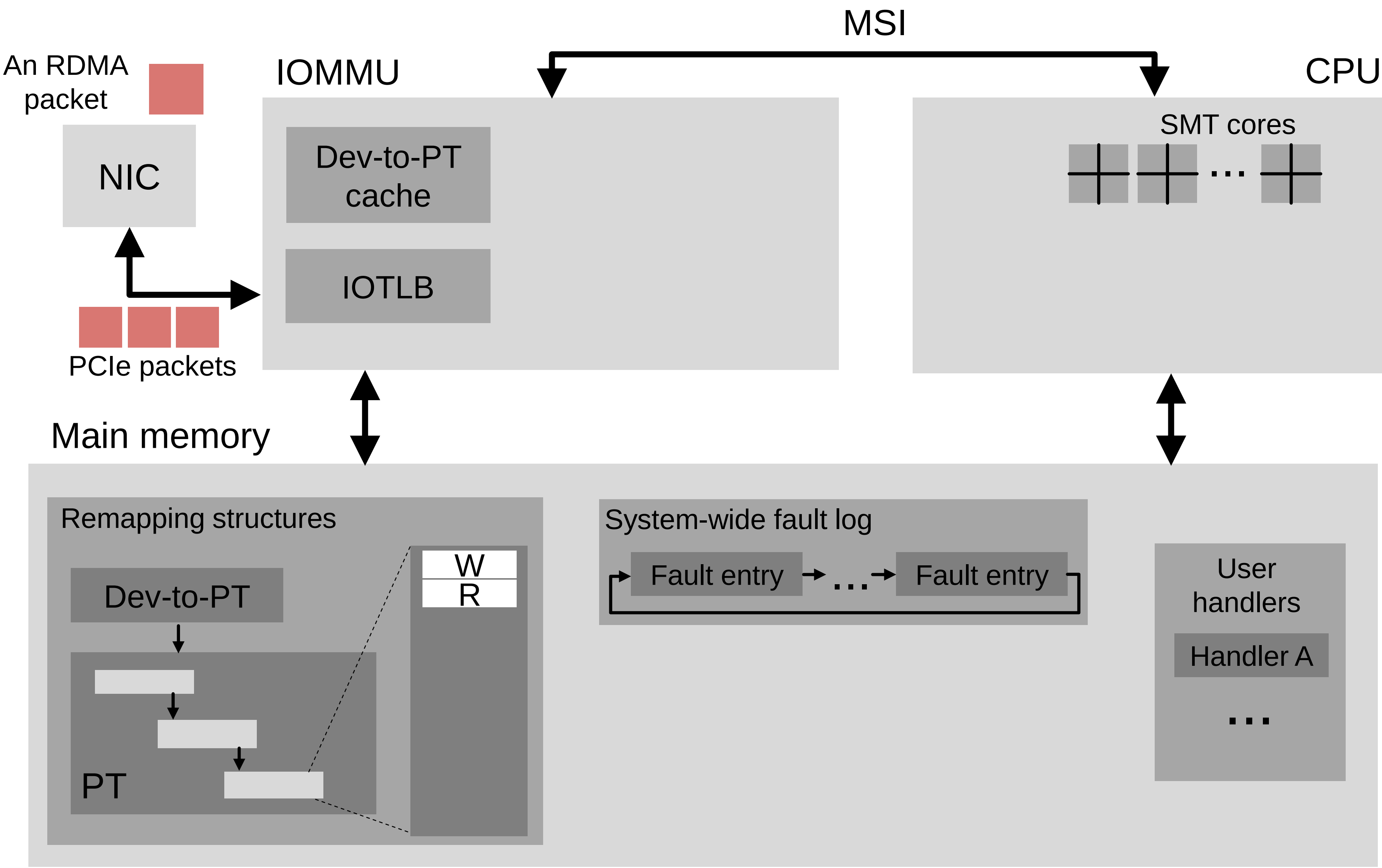


We could use it somehow. But…

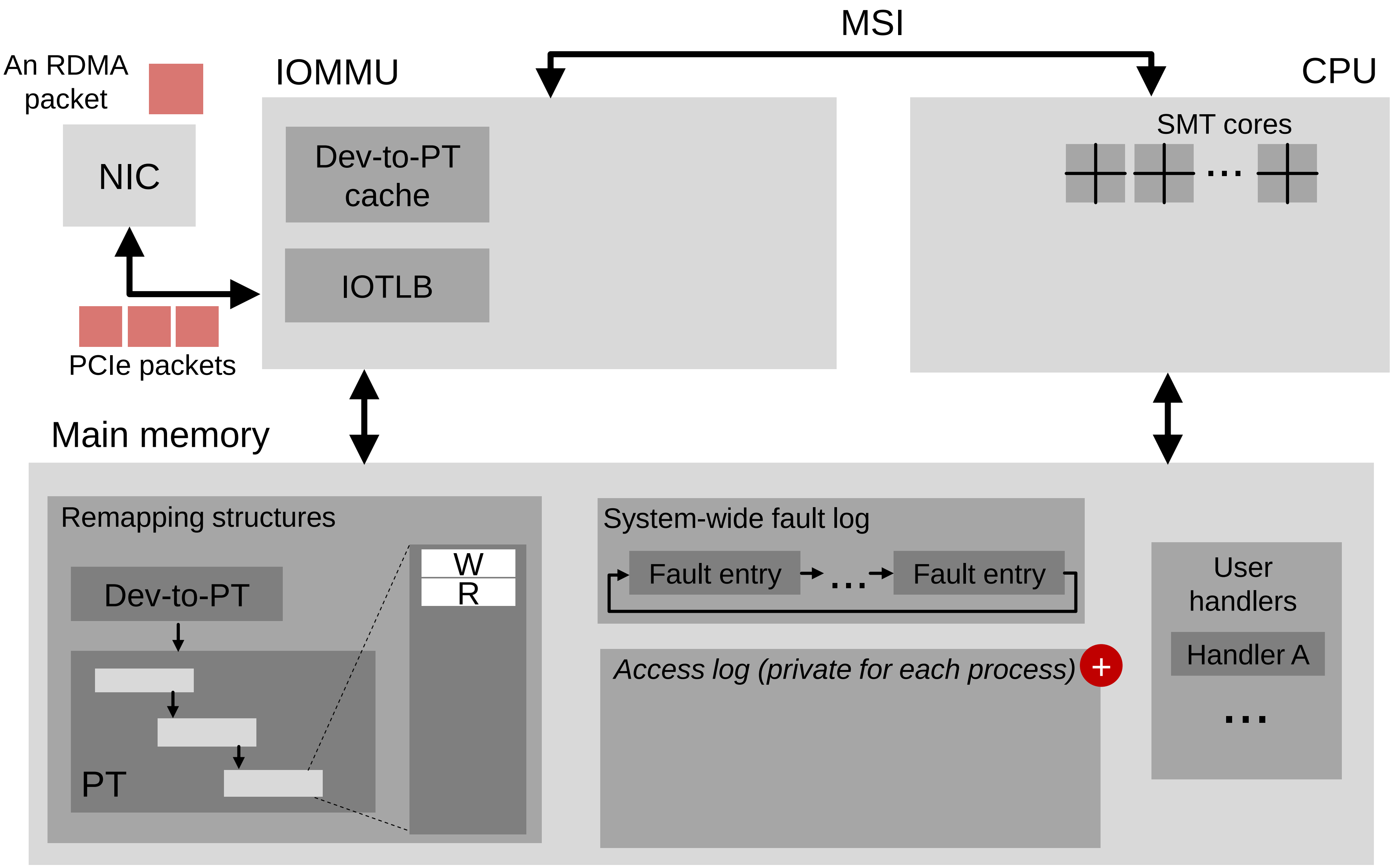
# IOMMUs and RMA



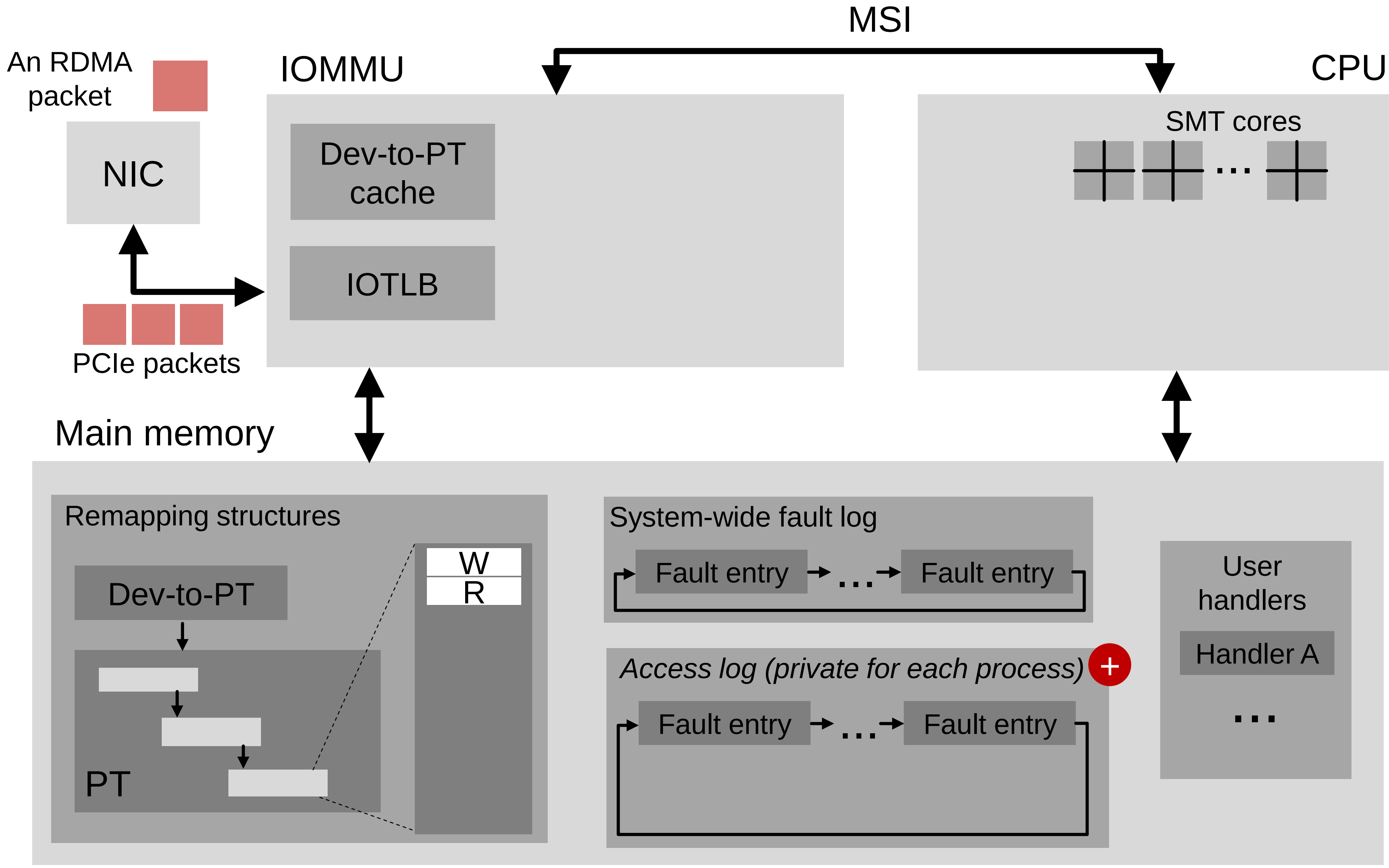
# Active Puts


MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
SMT cores
...
PCIe packets
Main memory
Remapping structures
W
R
Dev-to-PT
PT
System-wide fault log
Fault entry
...
Fault entry
User handlers
Handler A
...

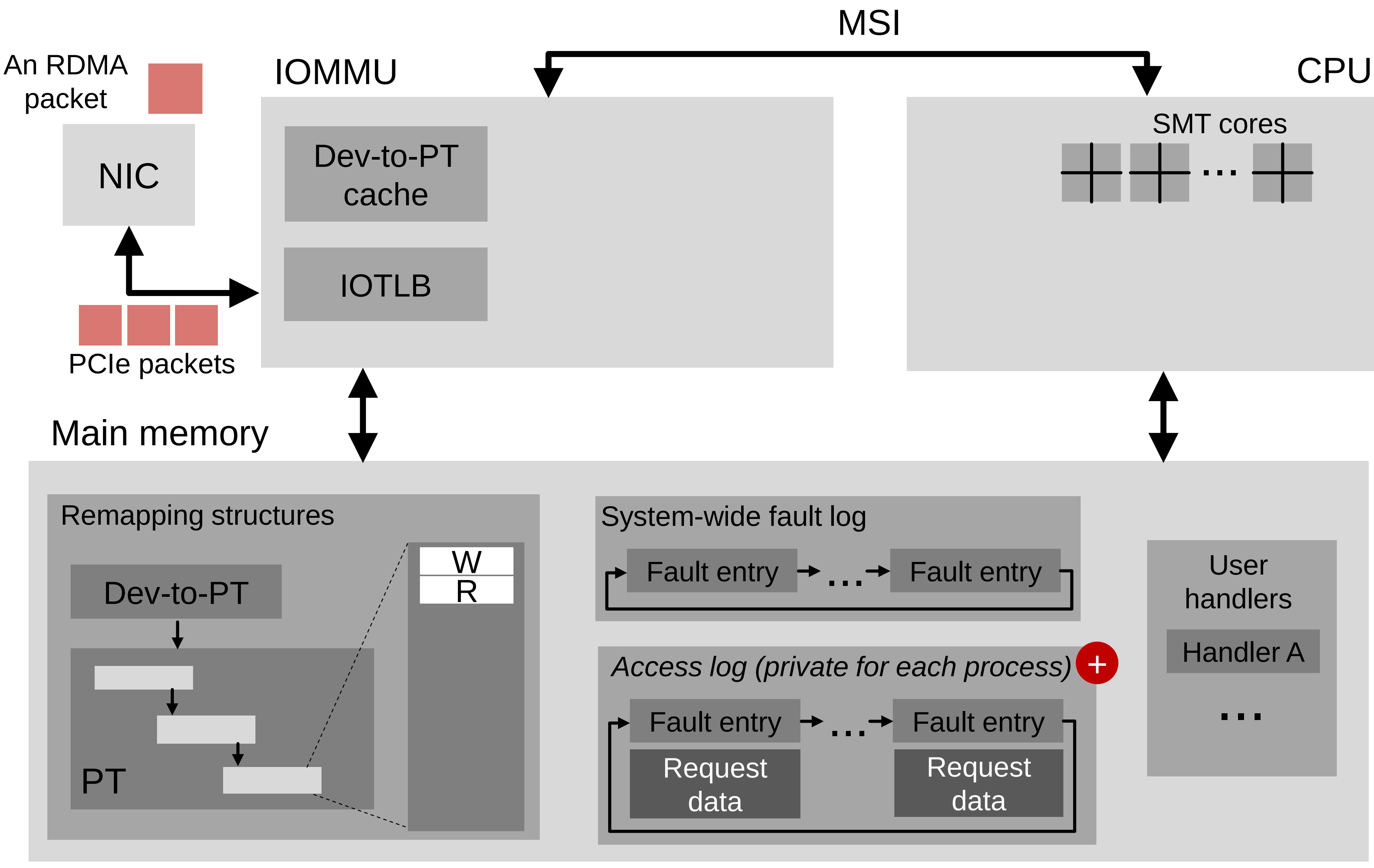
# Active Puts


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

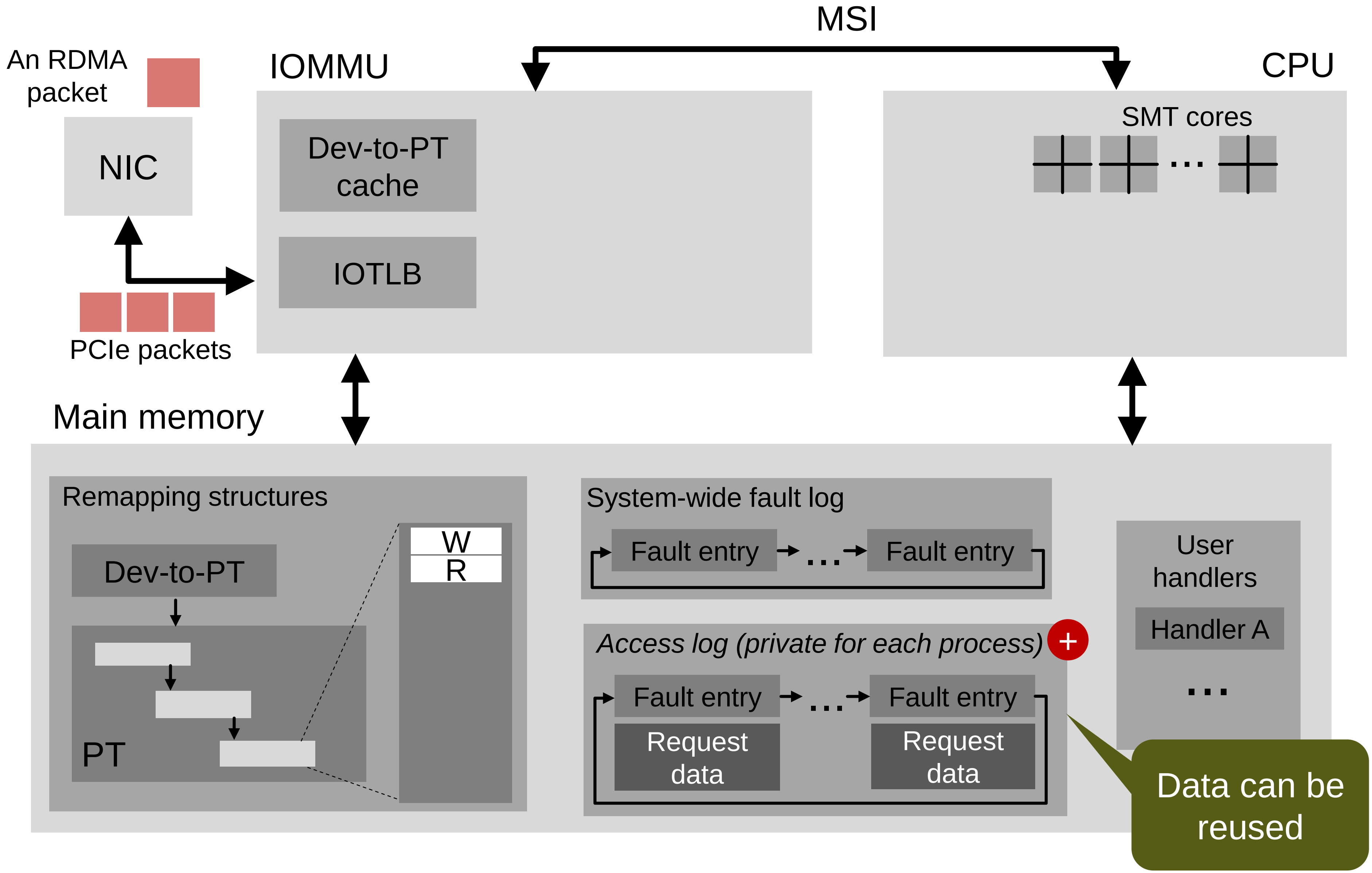
# ACTIVE PUTS


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

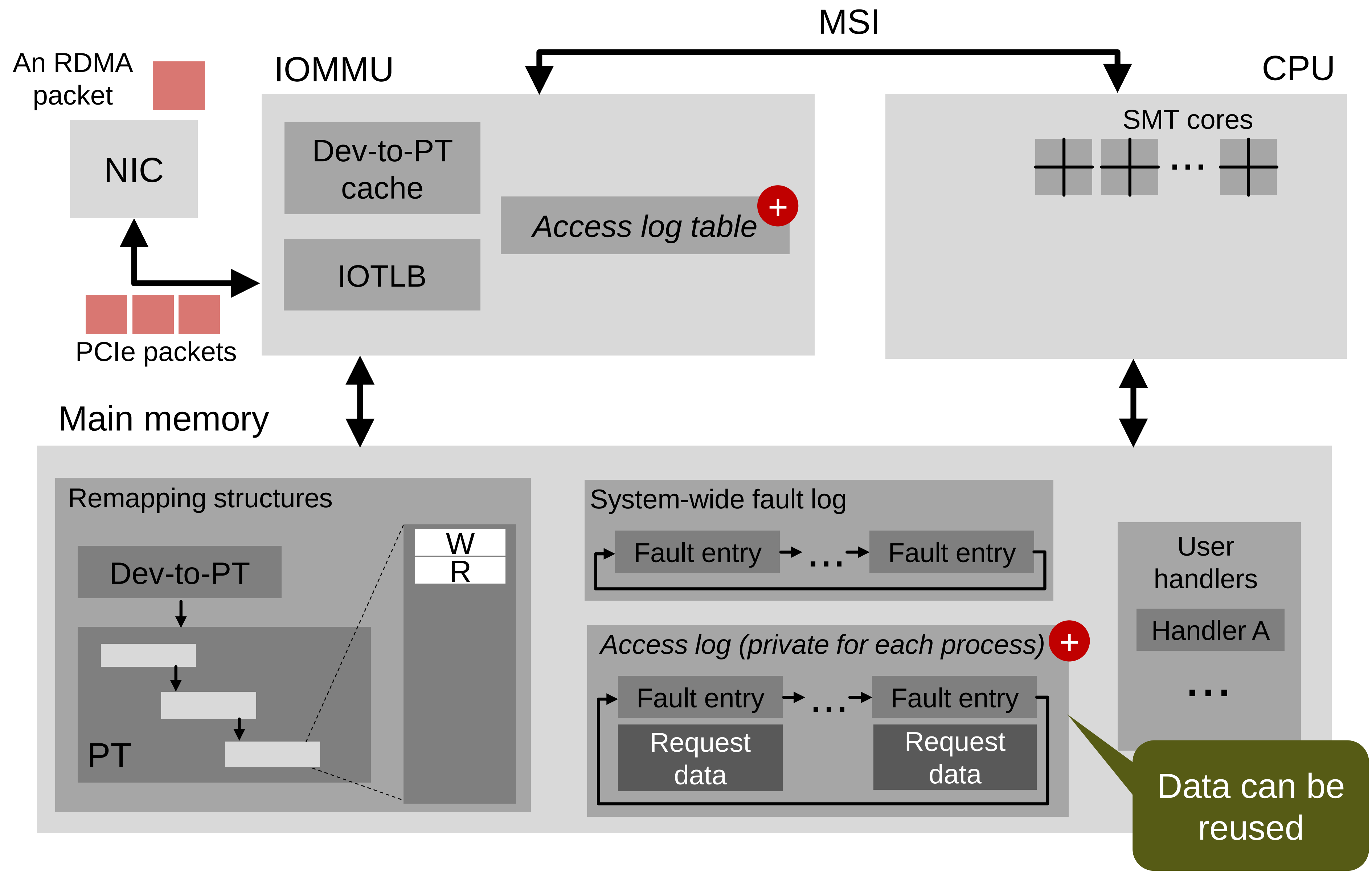
# Active Puts


MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
SMT cores
...
PCIe packets
Main memory
Remapping structures
Dev-to-PT
PT
W
R
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
...

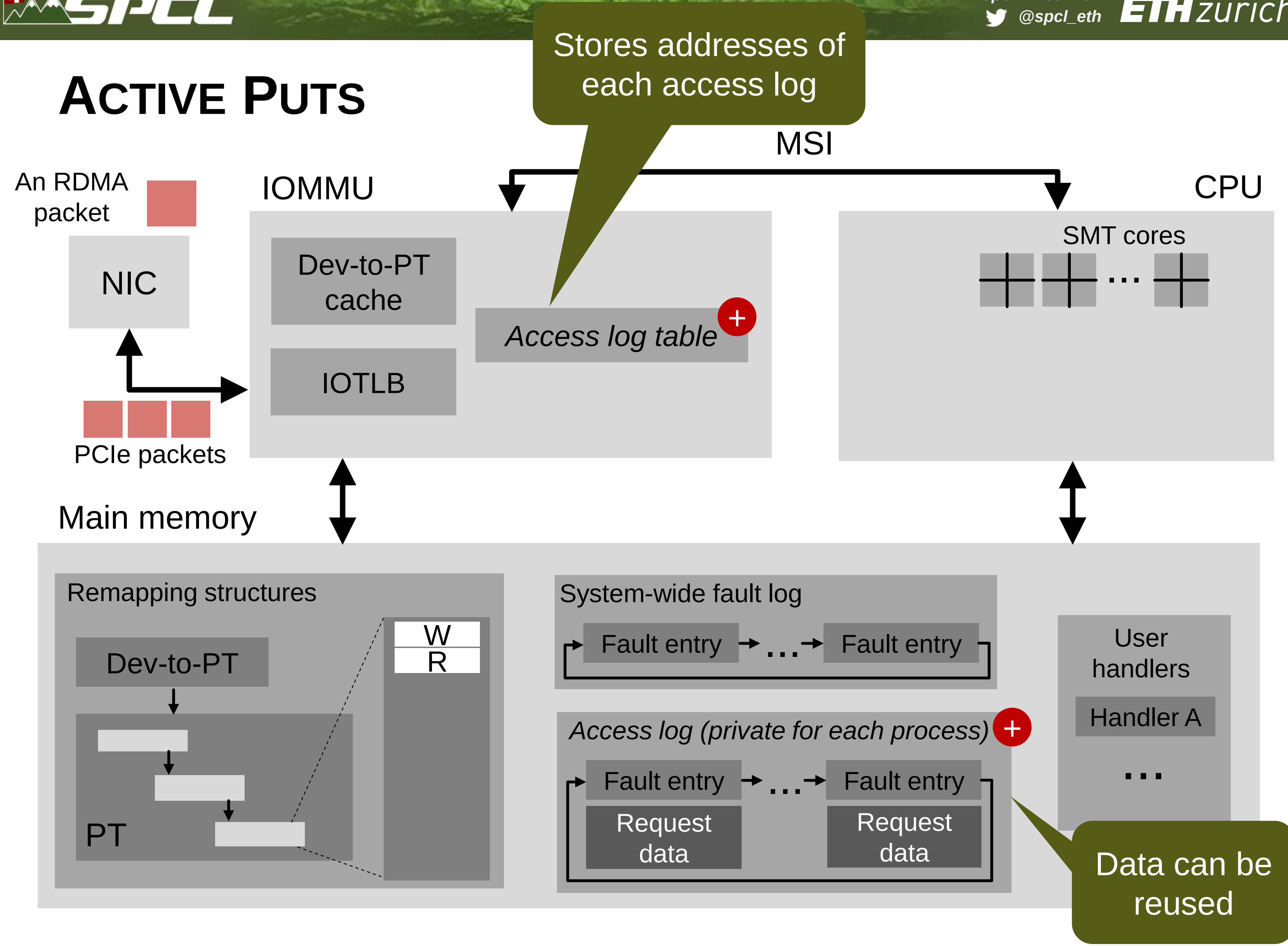
# Active Puts


MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
SMT cores
PCIe packets
Main memory
Remapping structures
Dev-to-PT
W
R
PT
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
Data can be reused

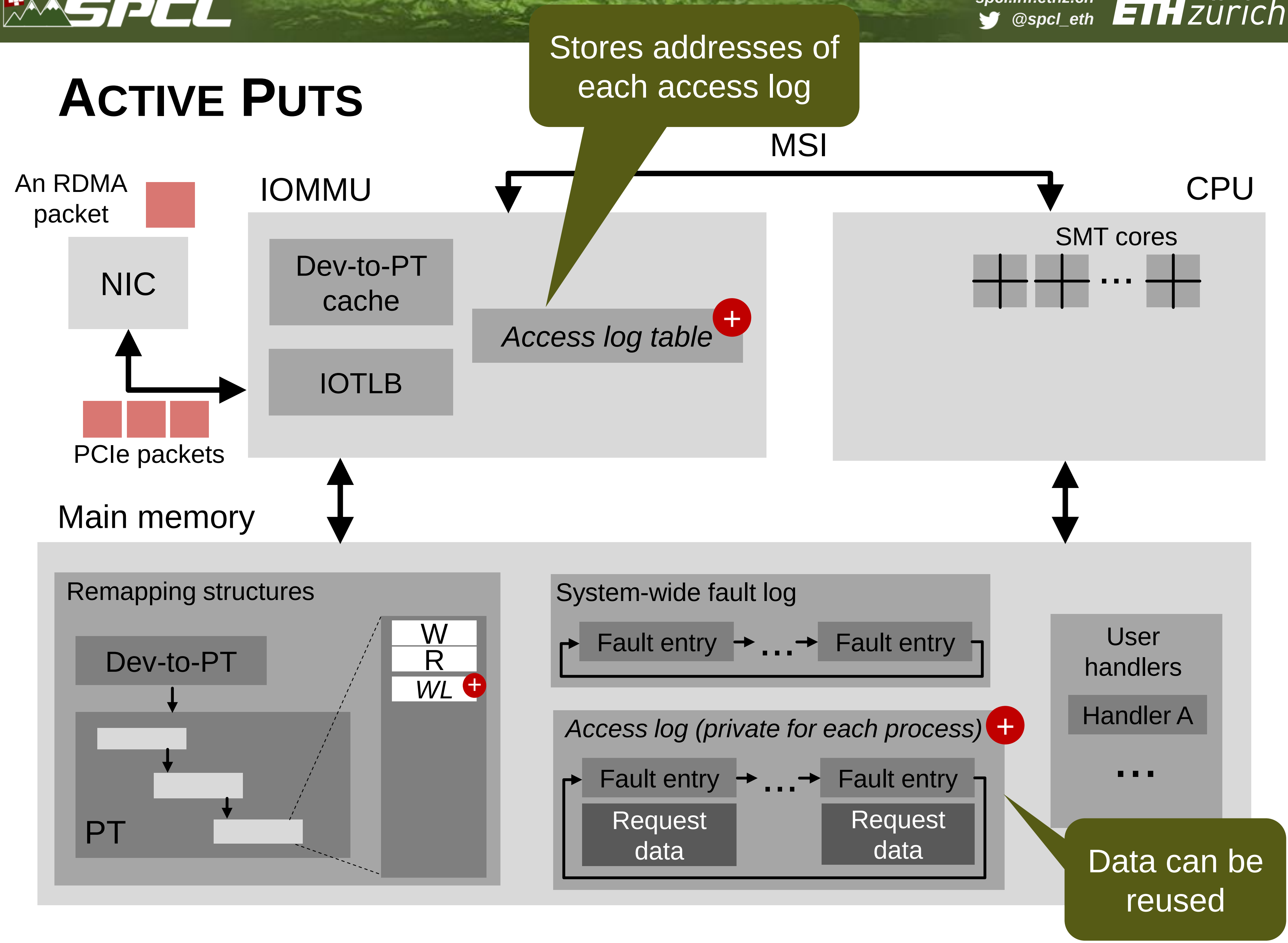
# Active Puts


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

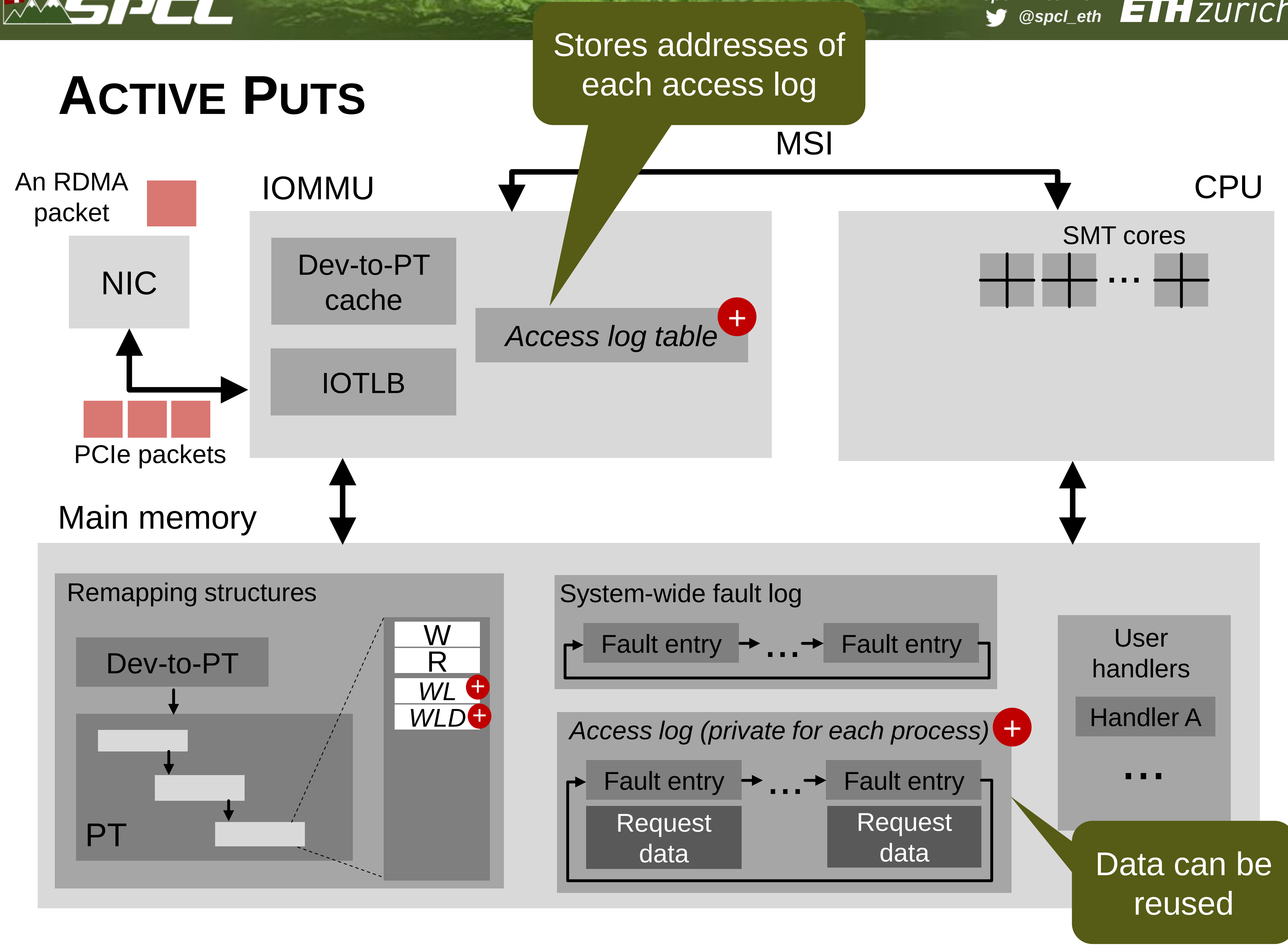
# ACTIVE PUTS


Stores addresses of each access log
MSI
An RDMA packet
NIC
PCIe packets
IOMMU
Dev-to-PT cache
IOTLB
Access log table
+
CPU
SMT cores
...
Main memory
Remapping structures
Dev-to-PT
PT
W
R
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
...
Data can be reused

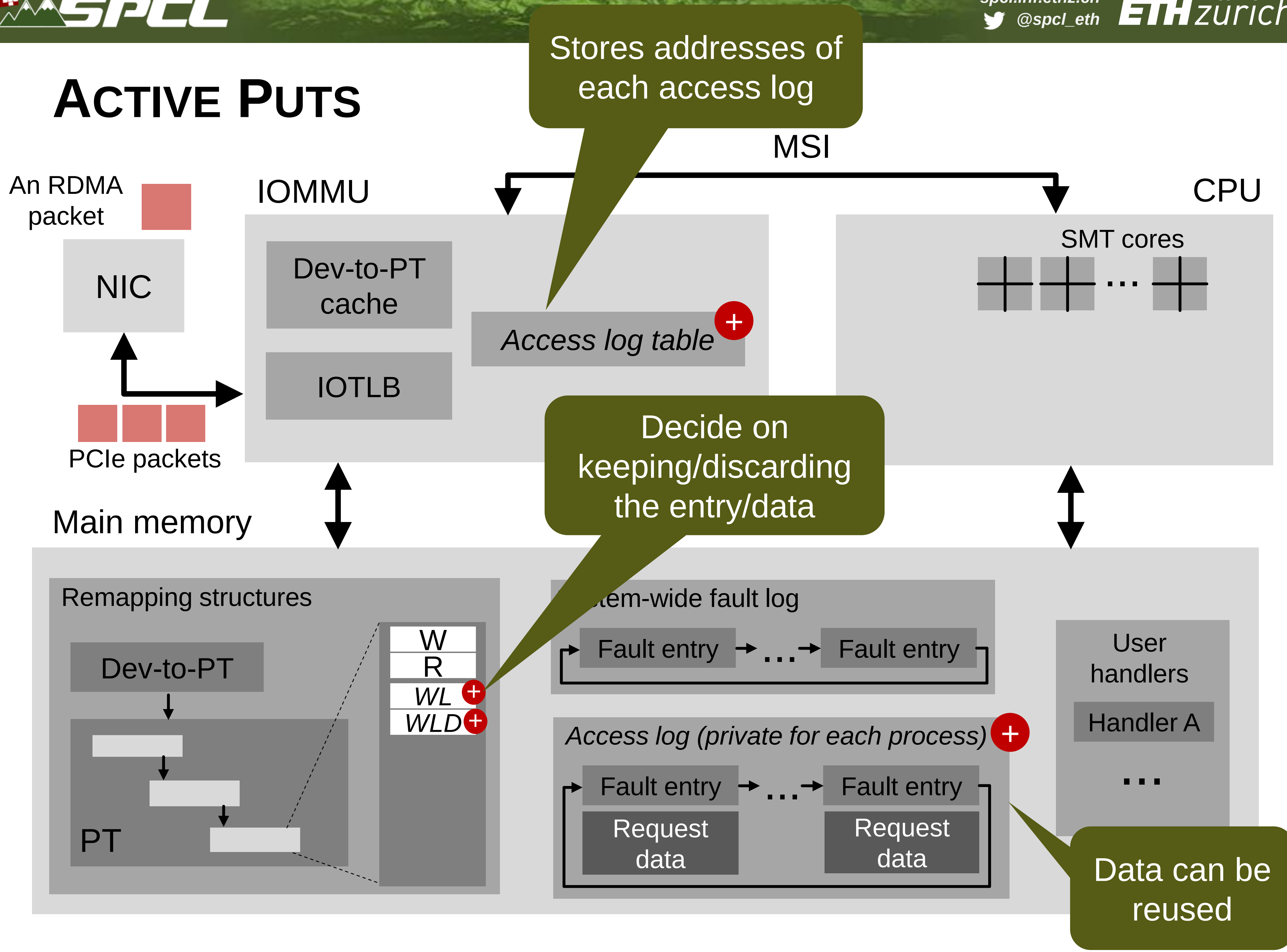
# Active Puts


Stores addresses of each access log
MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
SMT cores
…
Access log table
+
IOTLB
PCIe packets
Main memory
Remapping structures
W
R
WL
+
Dev-to-PT
PT
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
…
Data can be reused

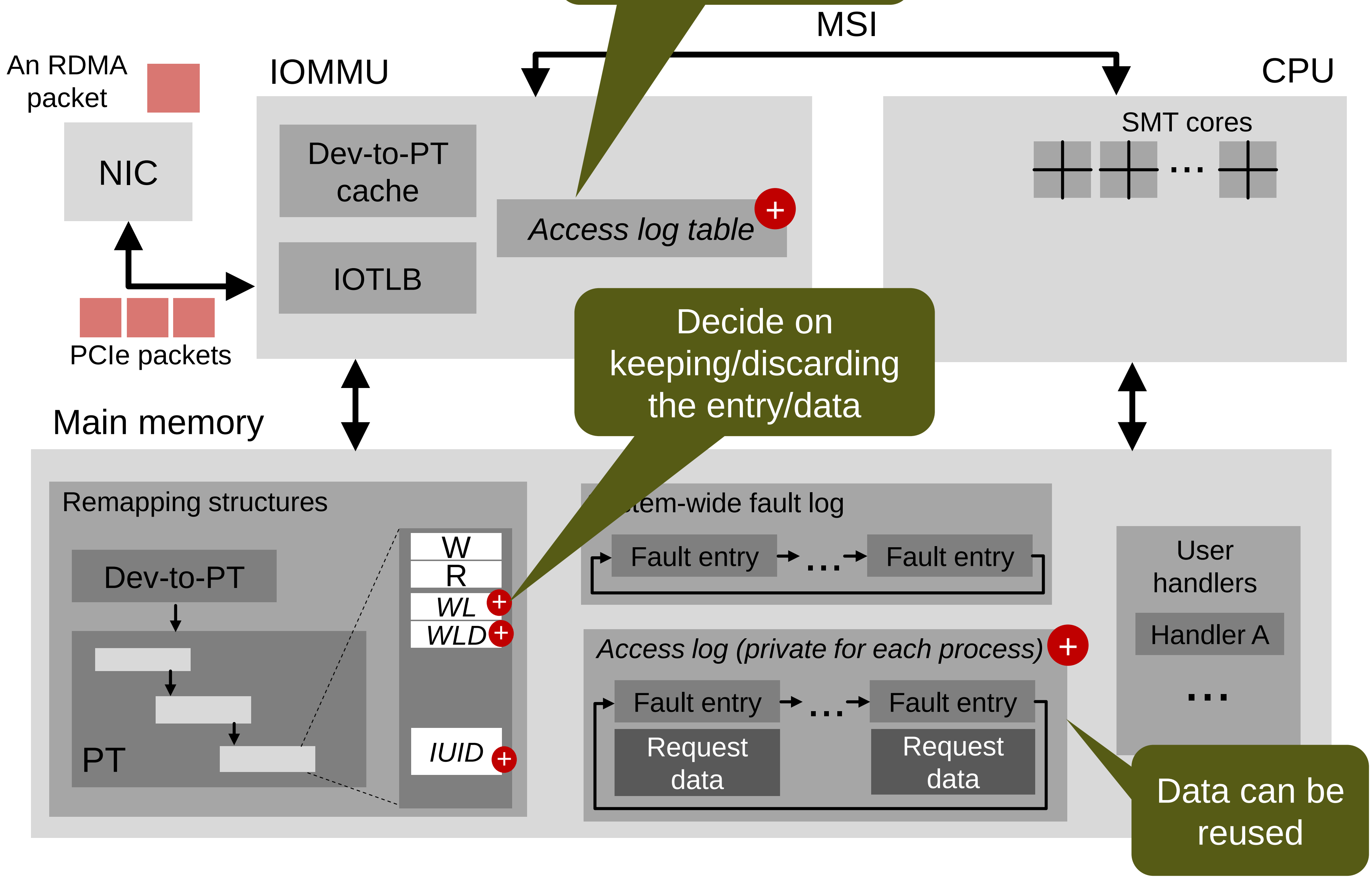
# Active Puts


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
Main memory
Remapping structures
Dev-to-PT
PT
W
R
WL
WLD
System-wide fault log
Fault entry
...
Fault entry
User handlers
Handler A
...
Access log (private for each process)
Fault entry
...
Fault entry
Request data
Request data
Data can be reused

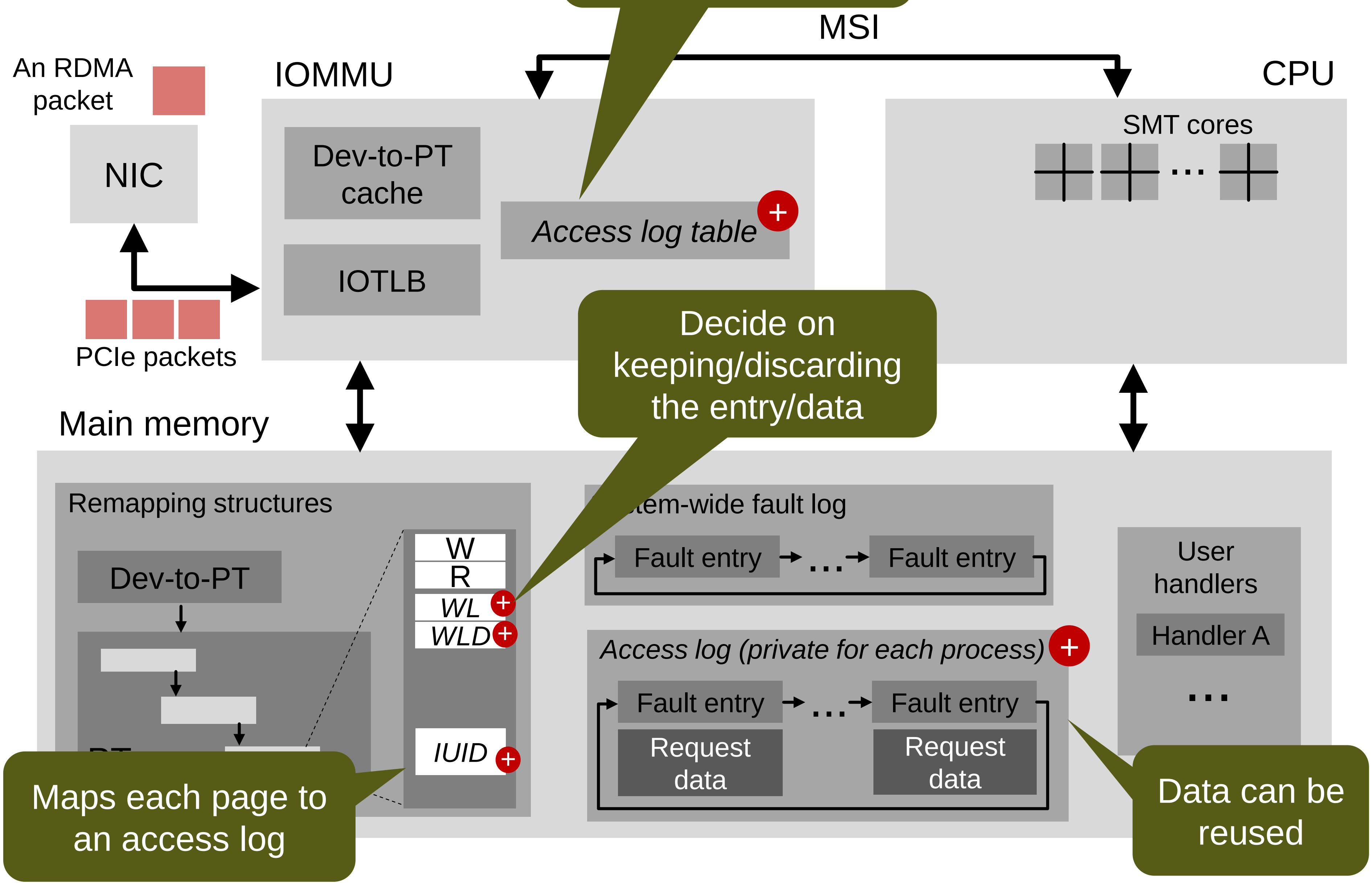
# Active Puts


Stores addresses of each access log
MSI
An RDMA packet
NIC
PCIe packets
IOMMU
Dev-to-PT cache
IOTLB
Access log table
CPU
SMT cores
...
Decide on keeping/discarding the entry/data
Main memory
Remapping structures
Dev-to-PT
PT
W
R
WL
WLD
tem-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
...
Data can be reused

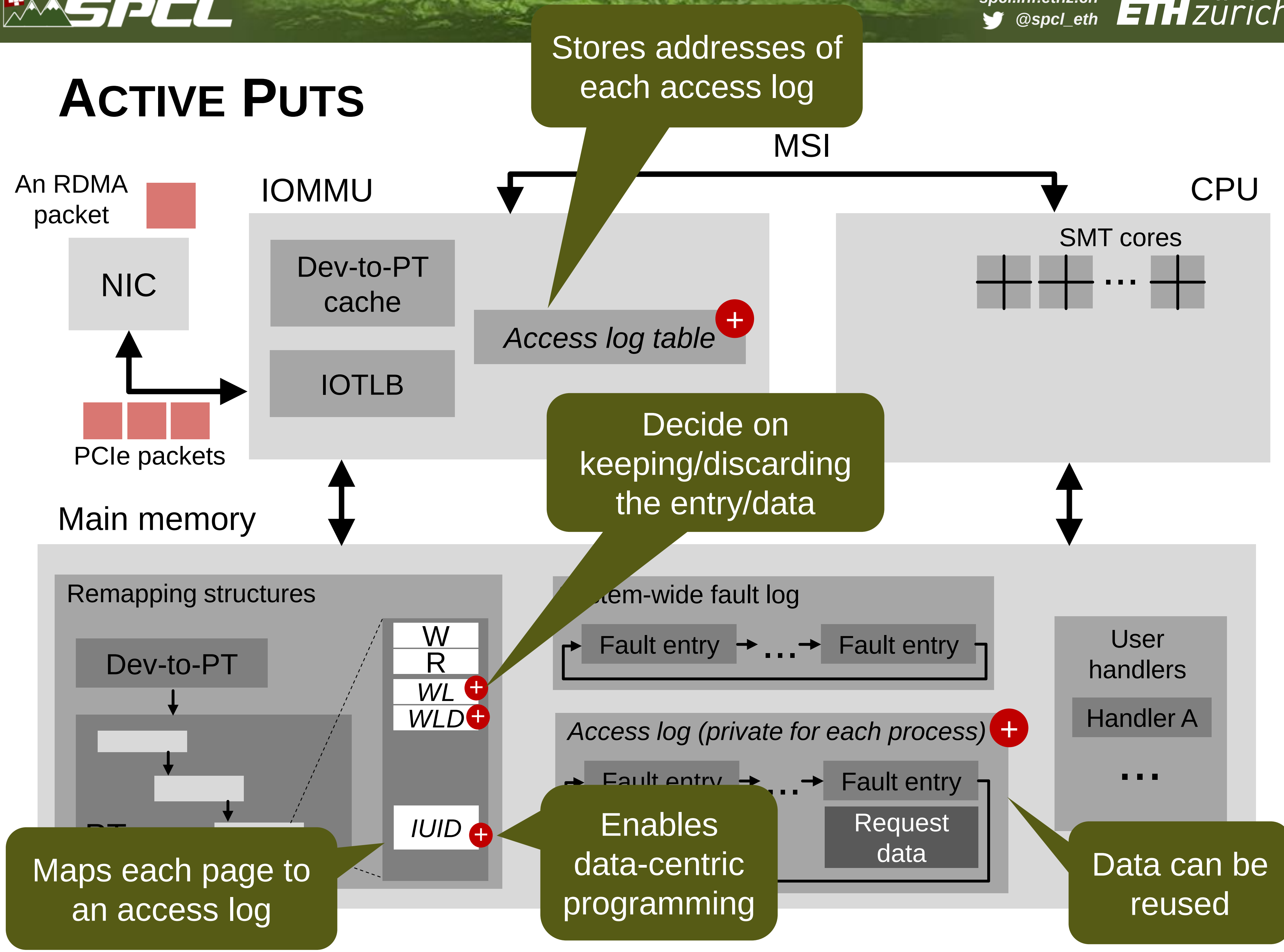
# Active Puts


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
PT
W
R
WL
WLD
IUID
tem-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
...
Data can be reused

# ACTIVE PUTS

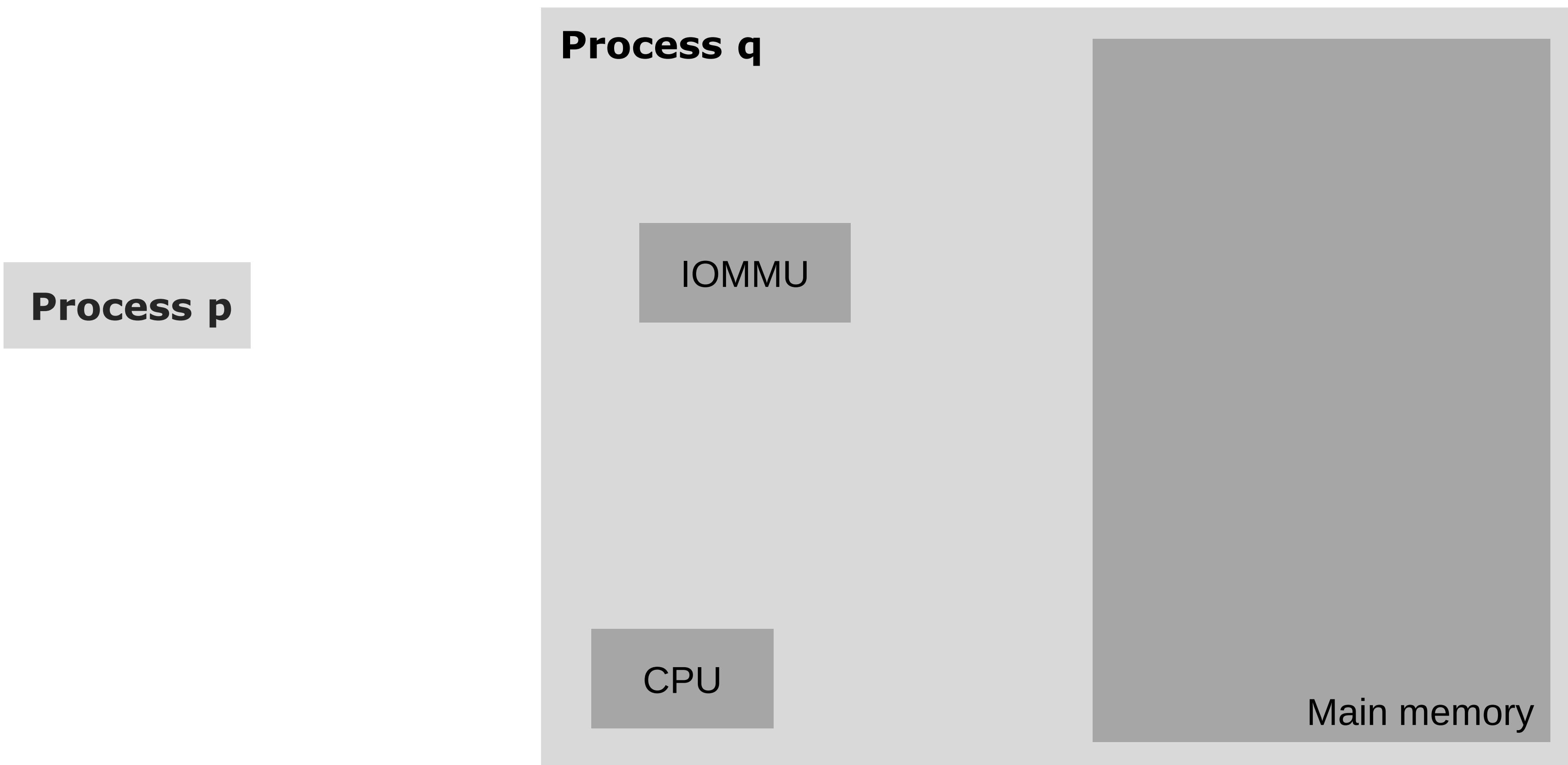

Stores addresses of each access log
MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
Access log table
IOTLB
SMT cores
PCIe packets
Decide on keeping/discarding the entry/data
Main memory
Remapping structures
tem-wide fault log
Dev-to-PT
W
R
WL
WLD
IUID
Fault entry
Fault entry
User handlers
Handler A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

# Active Puts

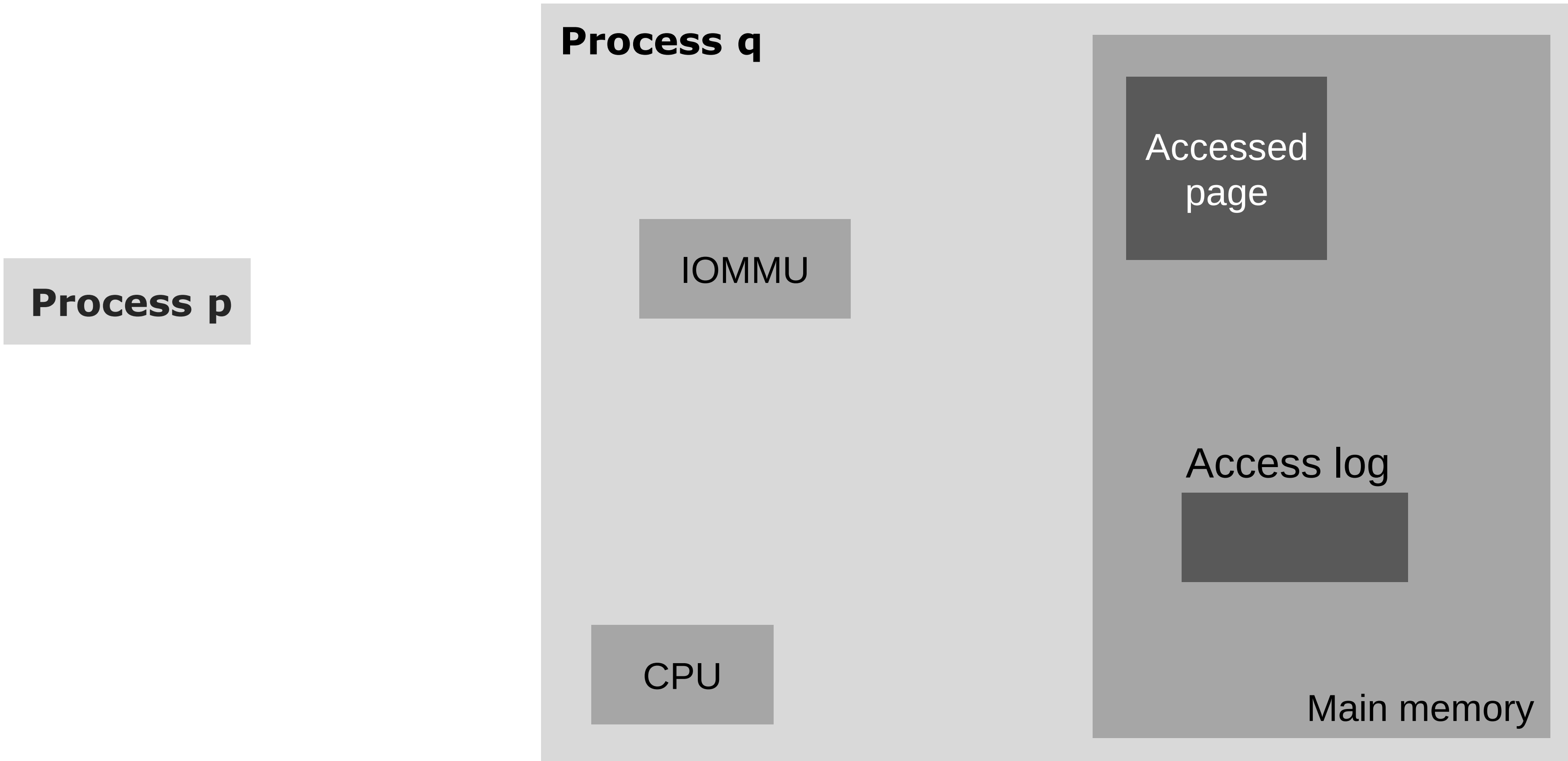

Stores addresses of each access log
MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
Access log table
SMT cores
PCIe packets
Decide on keeping/discarding the entry/data
Main memory
Remapping structures
Dev-to-PT
W
R
WL
WLD
IUID
tem-wide fault log
Fault entry
Fault entry
Access log (private for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

# ACTIVE PUTS

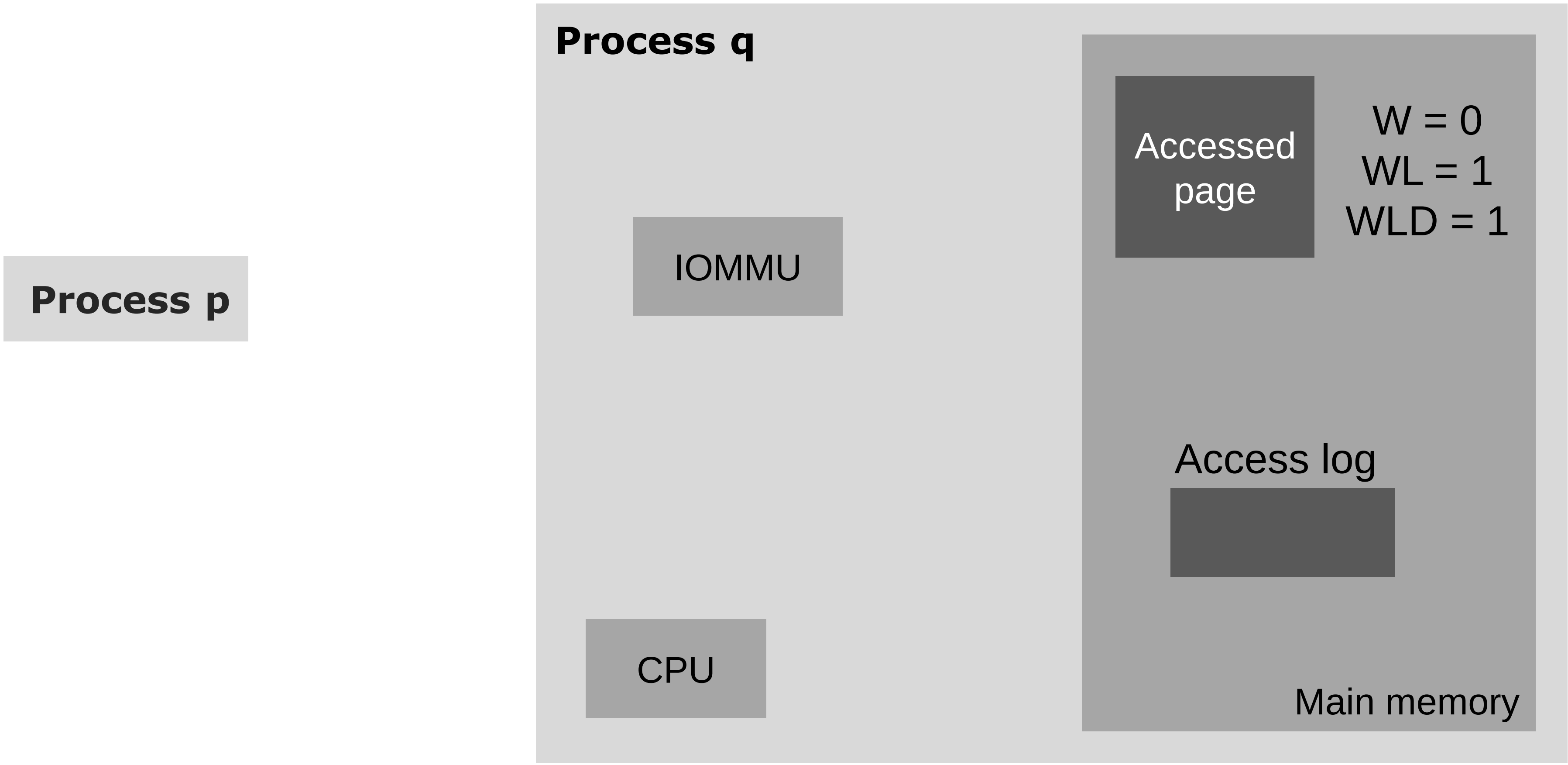
# ACTIVE PUTS



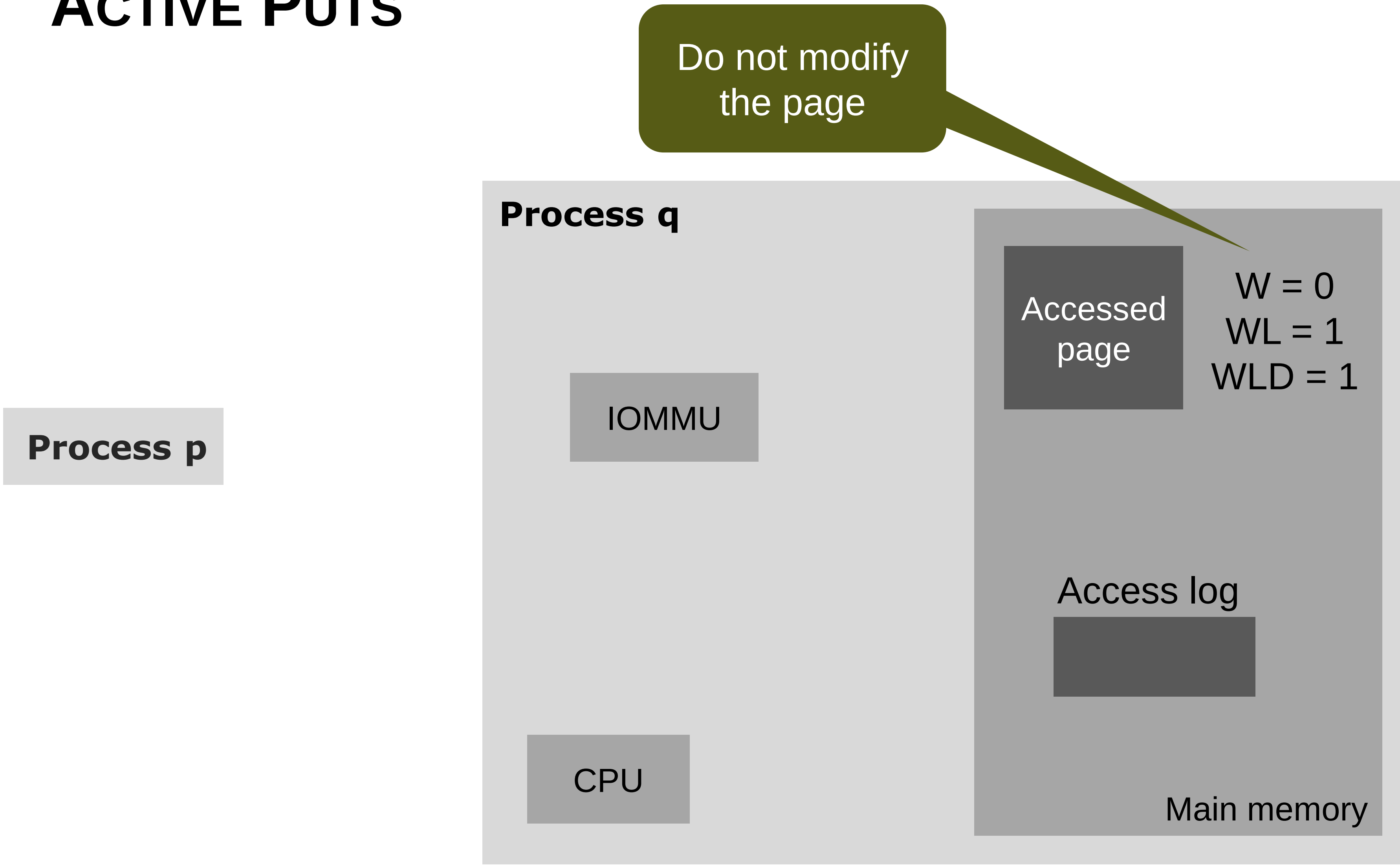
# Active Puts


Process p
Process q
IOMMU
CPU
Accessed page
Access log
Main memory

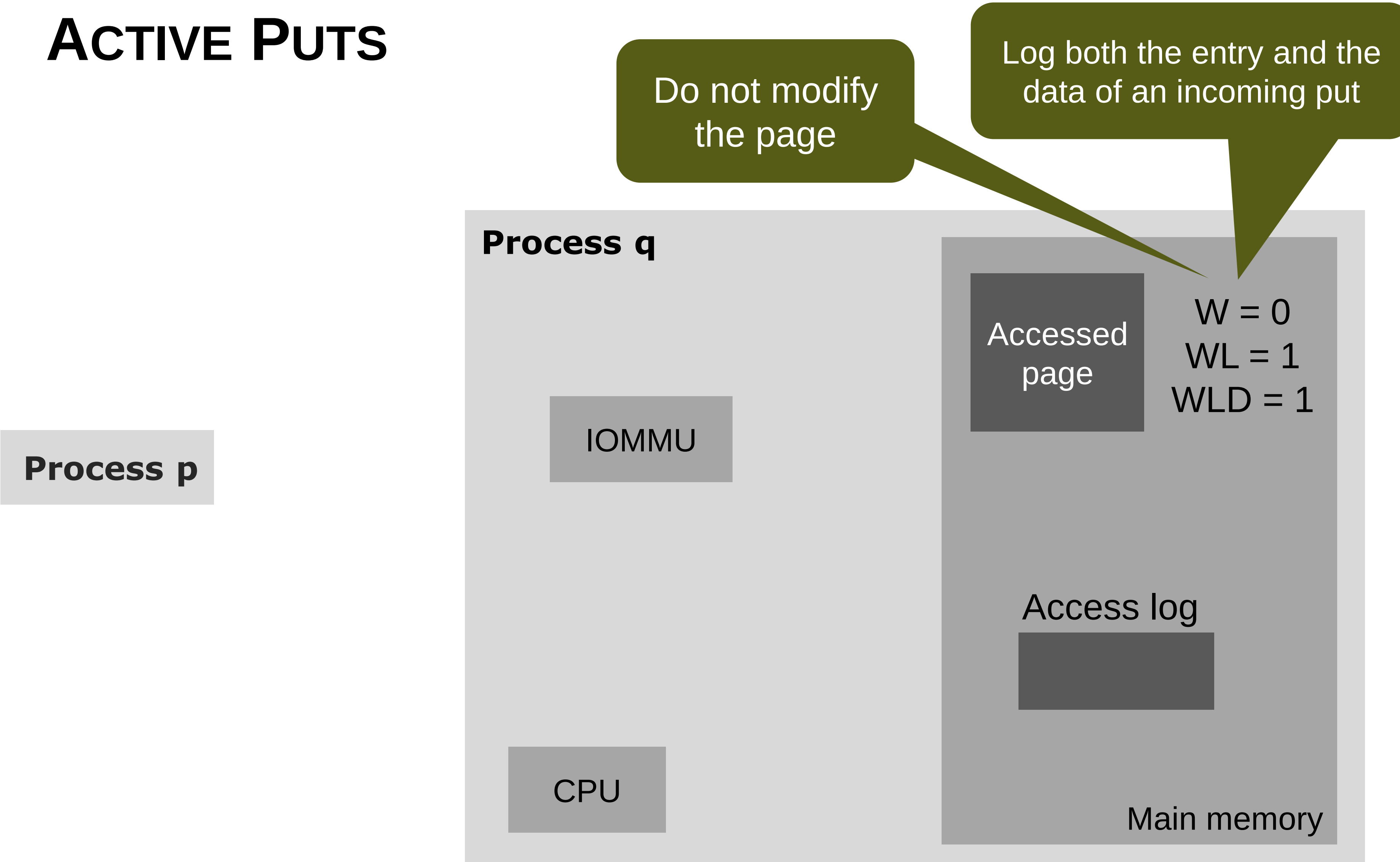
# Active Puts


Process q
Accessed page
W = 0
WL = 1
WLD = 1
IOMMU
Process p
Access log
CPU
Main memory

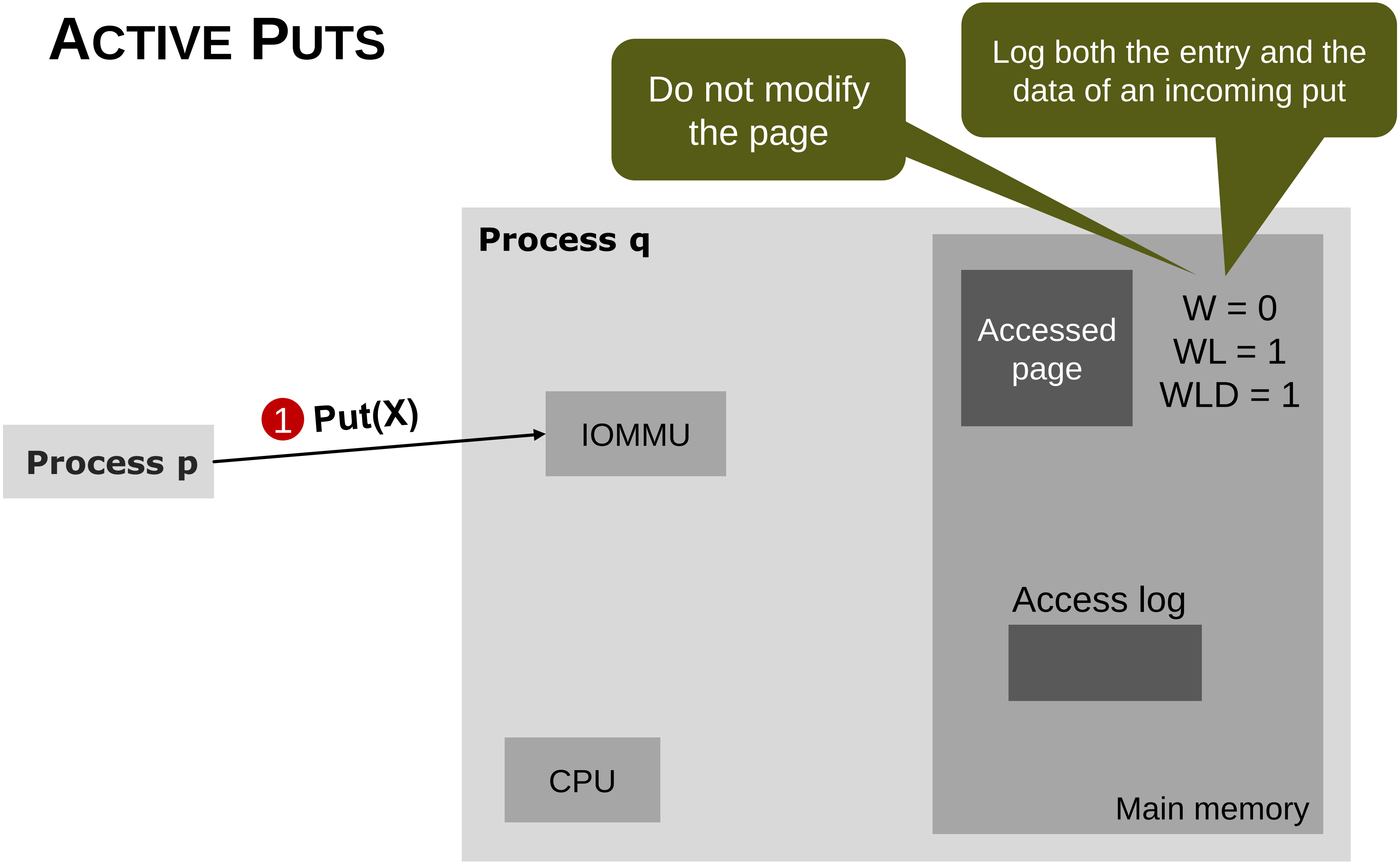
# Active Puts


Do not modify the page
Process q
Accessed page
W = 0
WL = 1
WLD = 1
IOMMU
Process p
Access log
CPU
Main memory

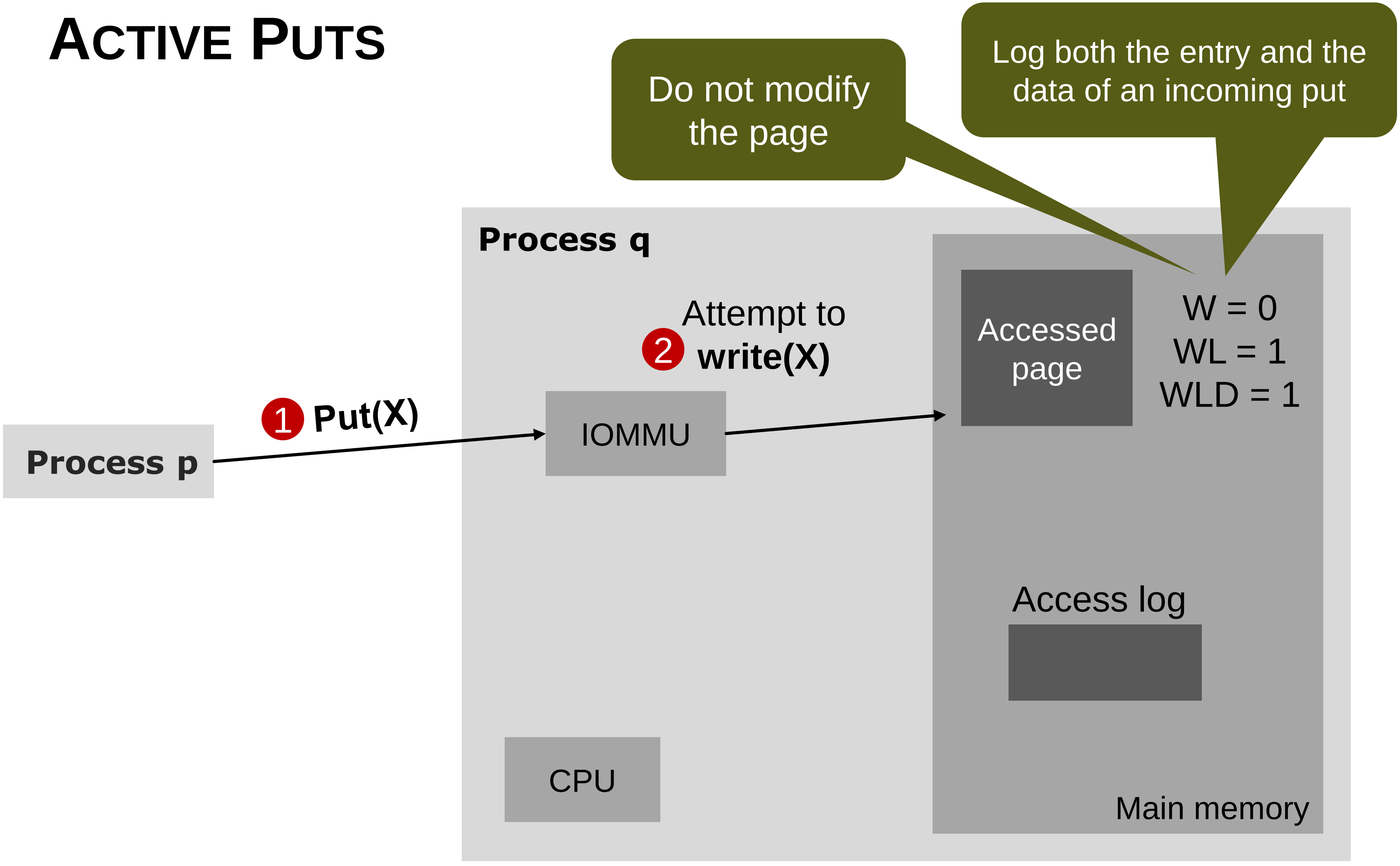
# Active Puts


Do not modify the page
Log both the entry and the data of an incoming put
Process q
Accessed page
W = 0
WL = 1
WLD = 1
IOMMU
Process p
Access log
CPU
Main memory

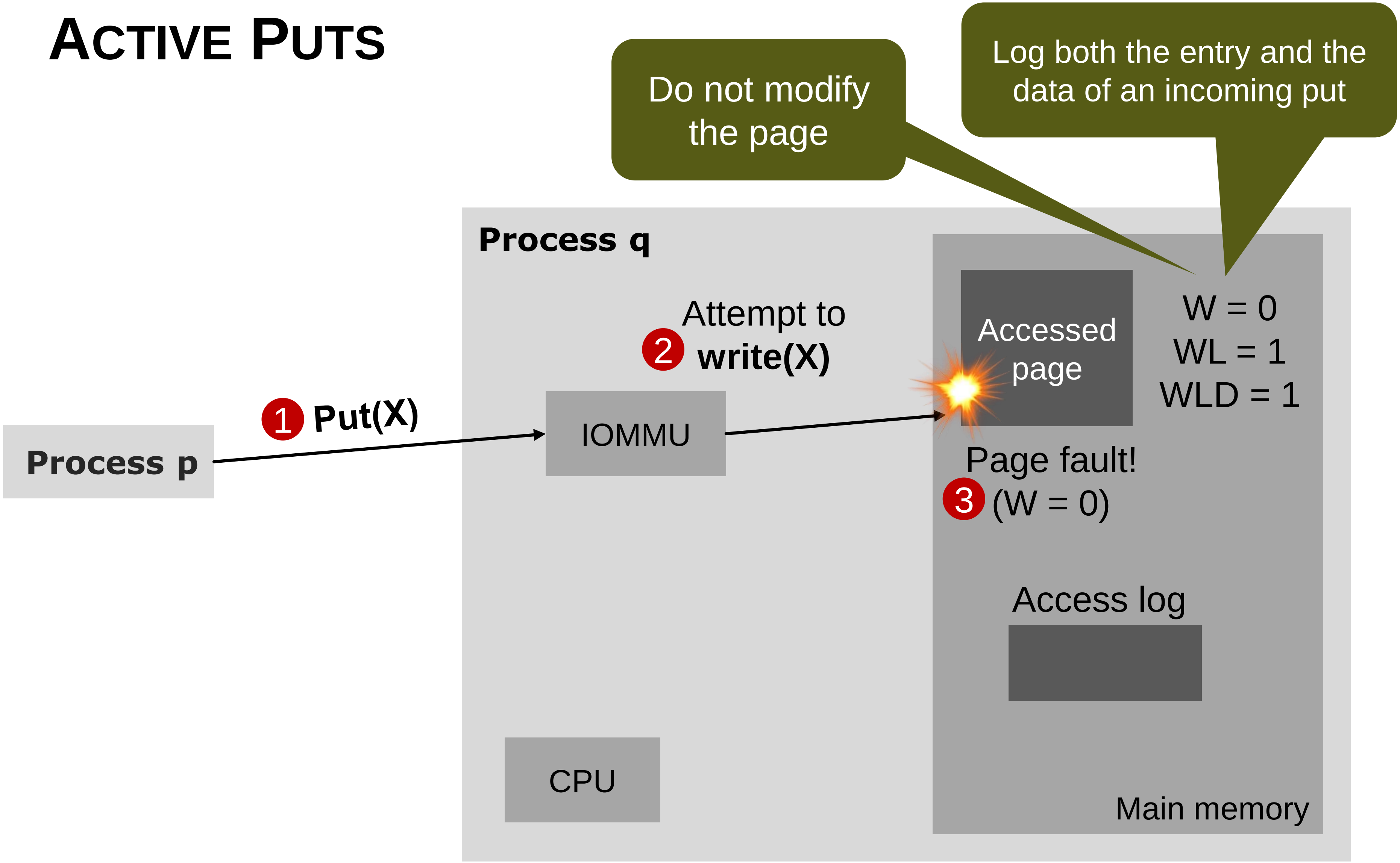
# Active Puts


Do not modify the page
Log both the entry and the data of an incoming put
Process q
Accessed page
W = 0
WL = 1
WLD = 1
1 Put(X)
Process p
IOMMU
Access log
CPU
Main memory

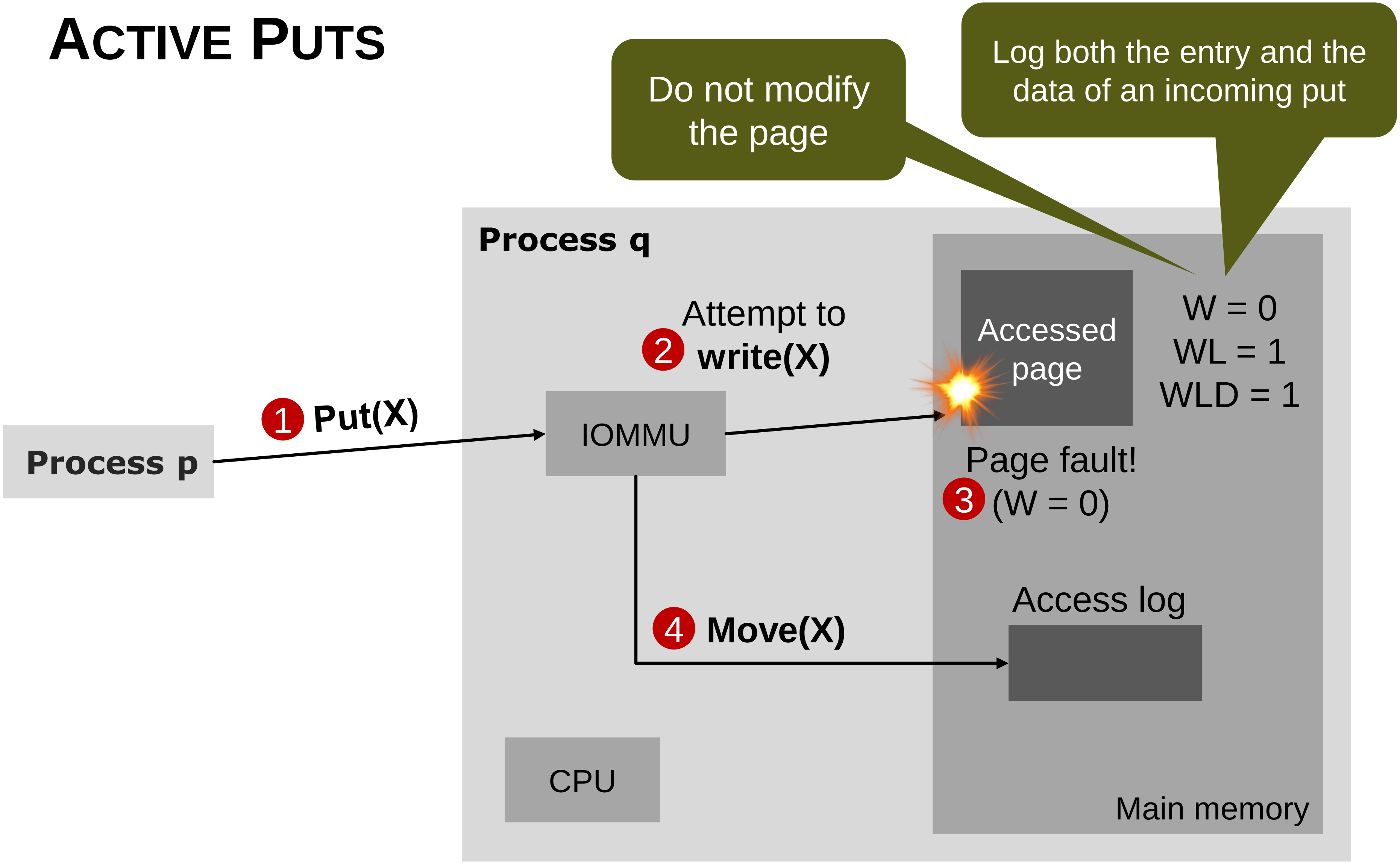
# Active Puts


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
Access log
CPU
Main memory

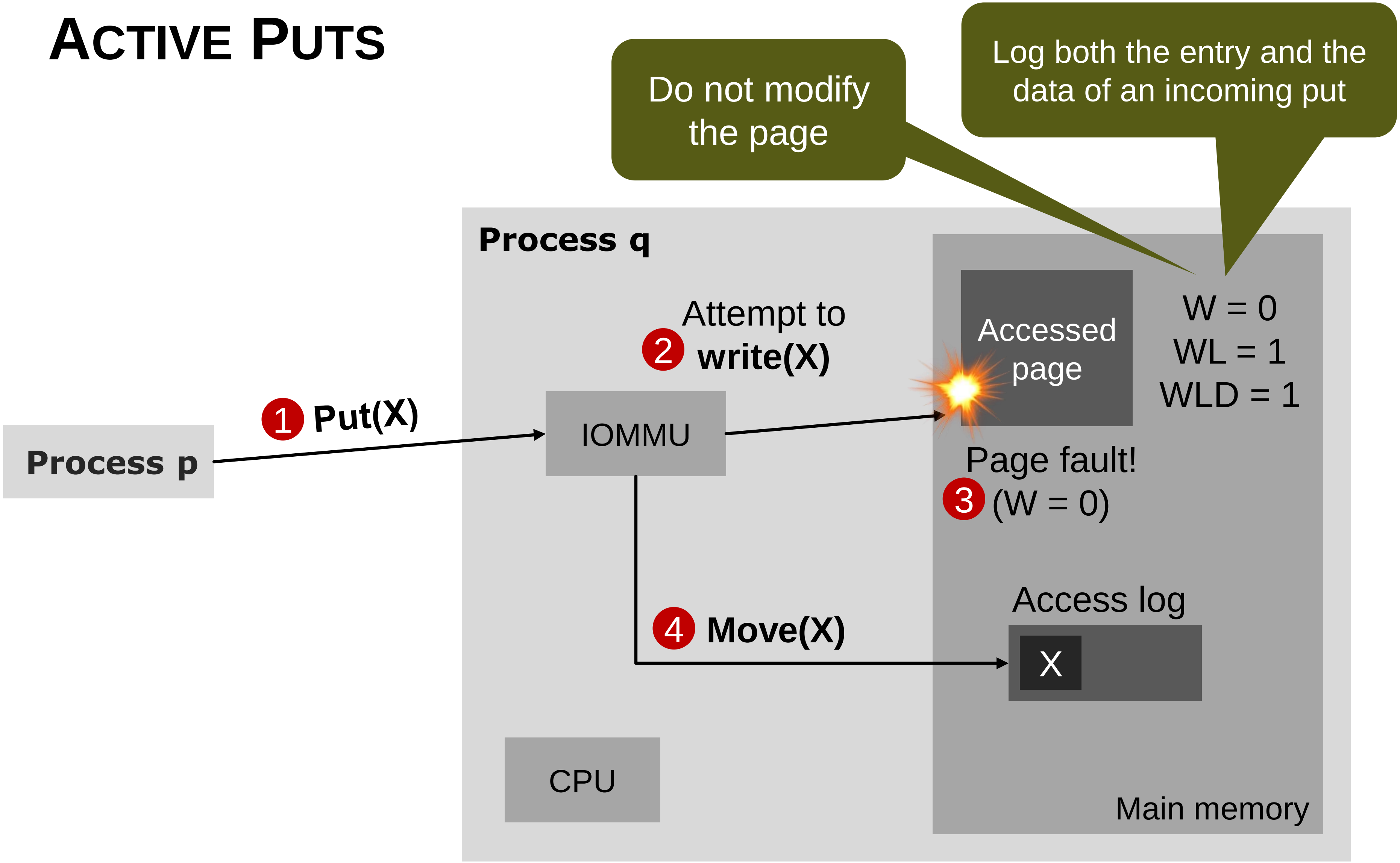
# Active Puts


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

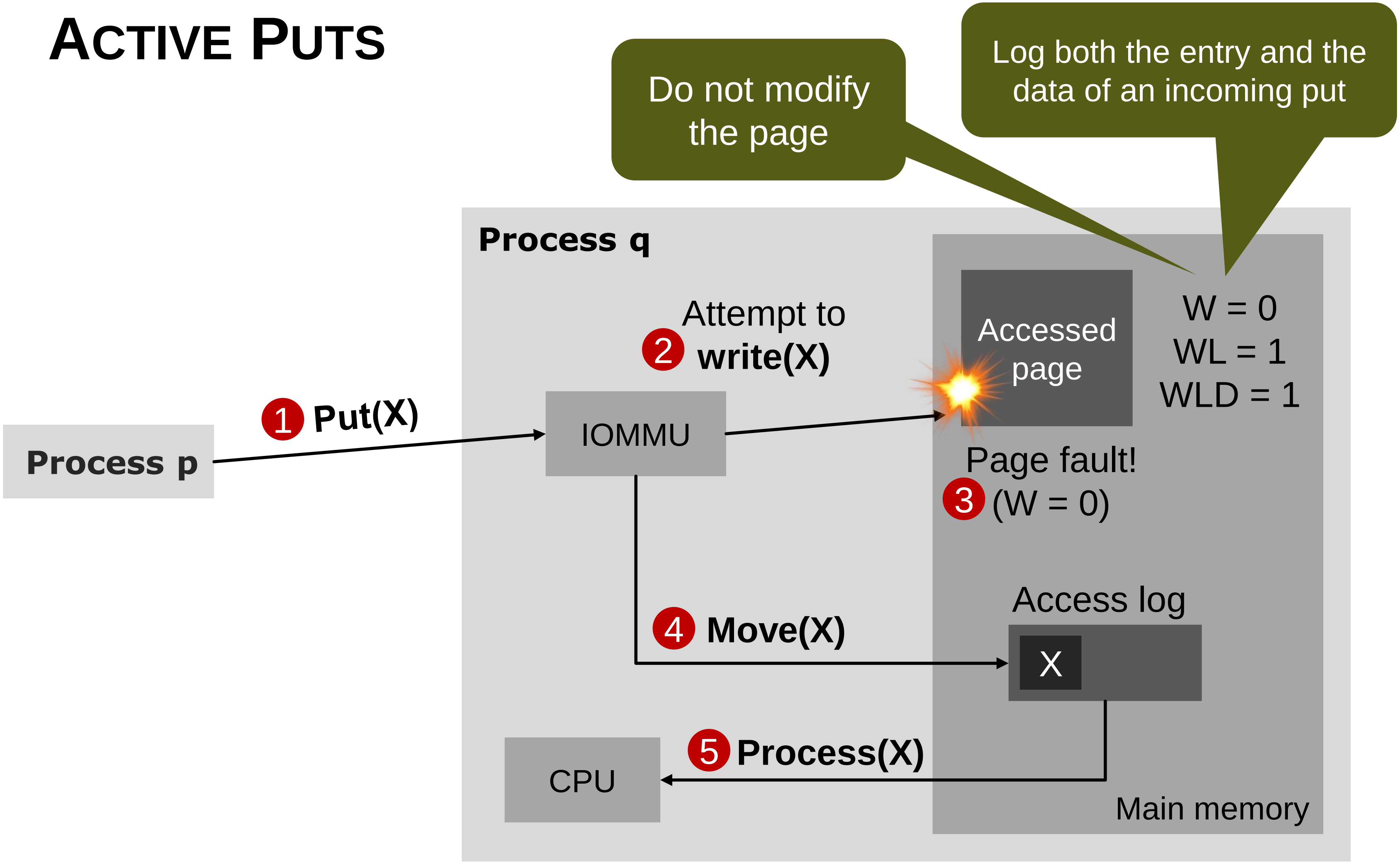
# Active Puts


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

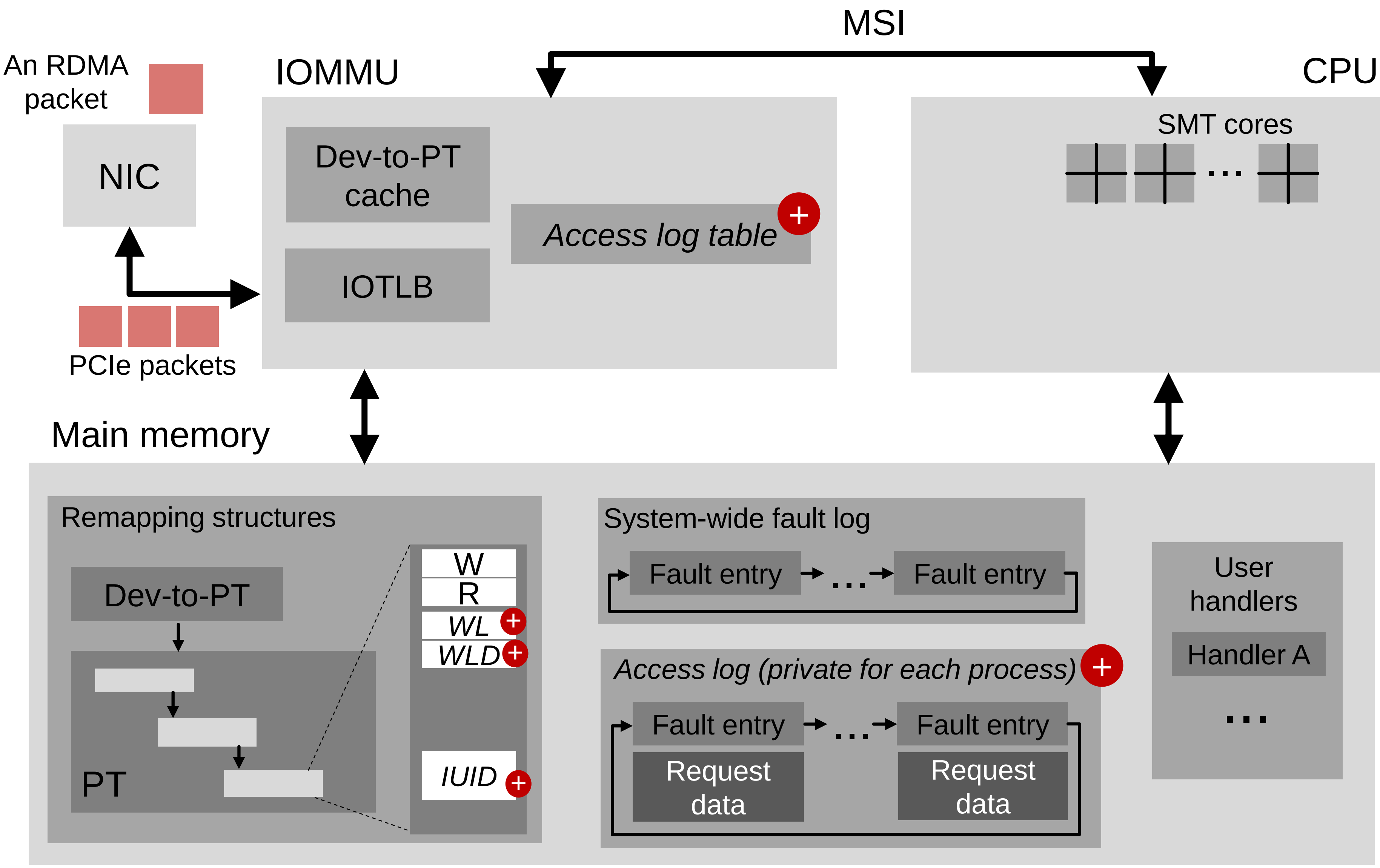
# Active Puts


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

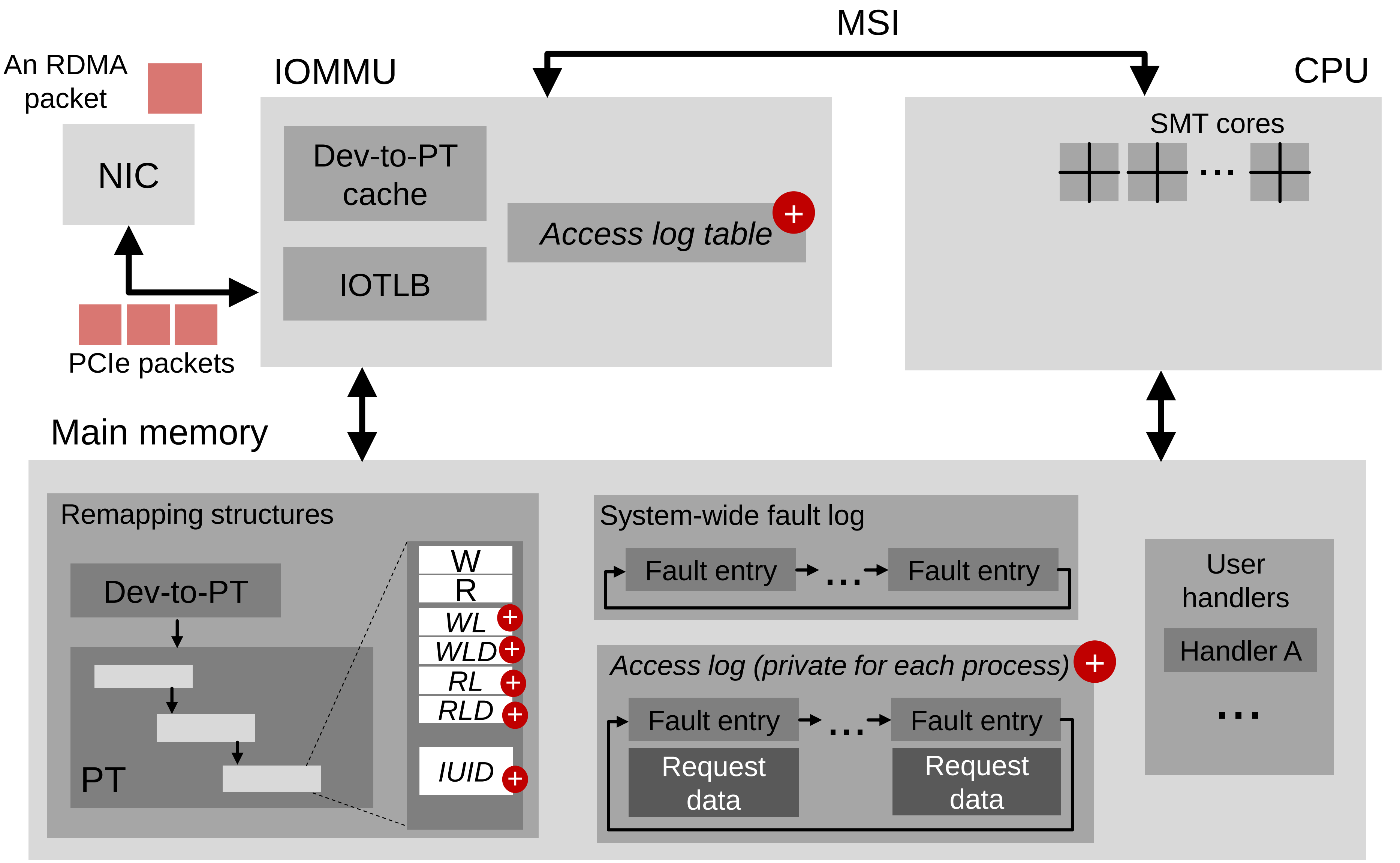
# Active Puts


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

# Active Gets

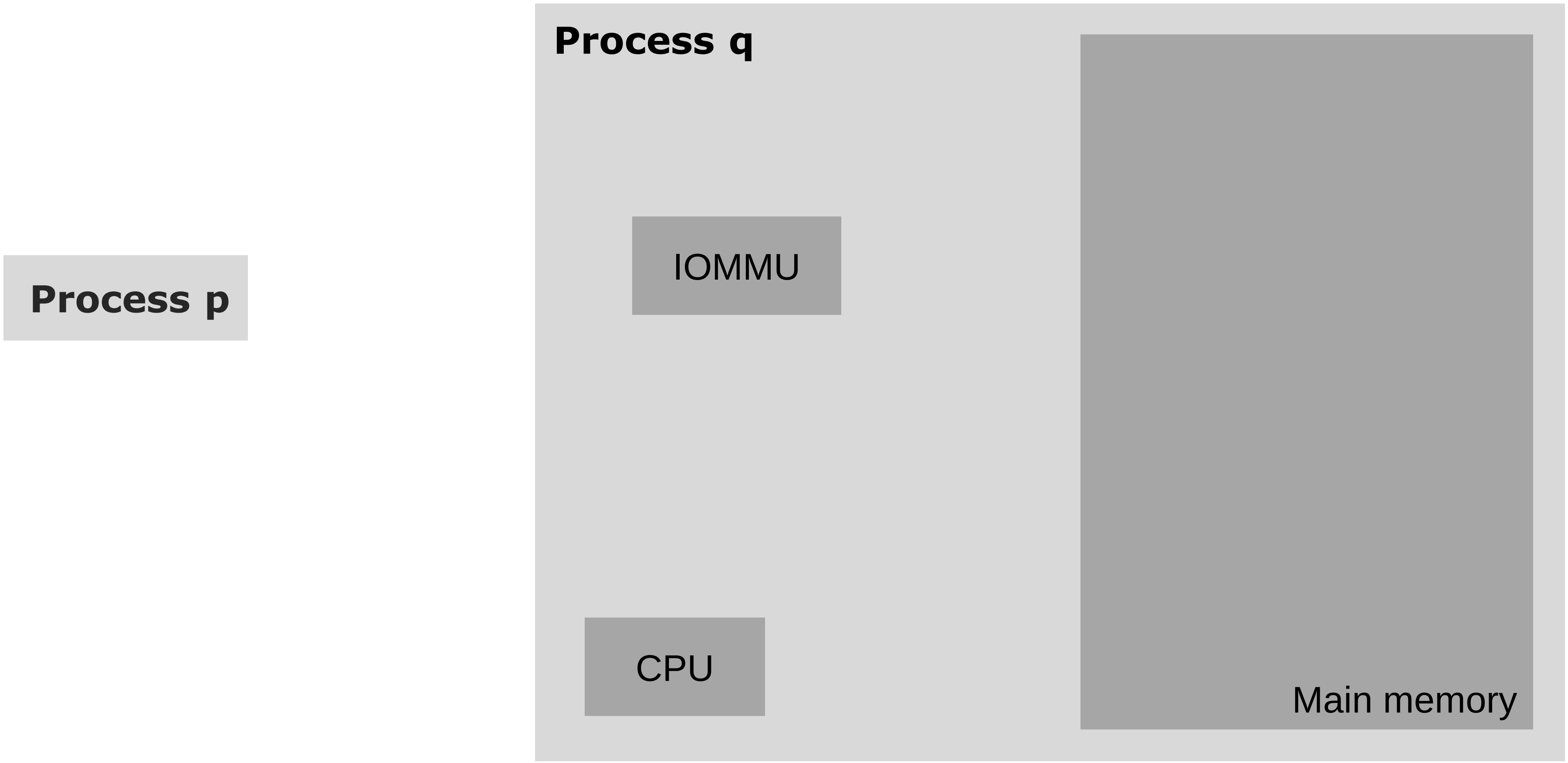

MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
Access log table
SMT cores
...
PCIe packets
Main memory
Remapping structures
Dev-to-PT
PT
W
R
WL
WLD
IUID
System-wide fault log
Fault entry
...
Fault entry
Access log (private for each process)
Fault entry
...
Fault entry
Request data
Request data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

# Active Gets

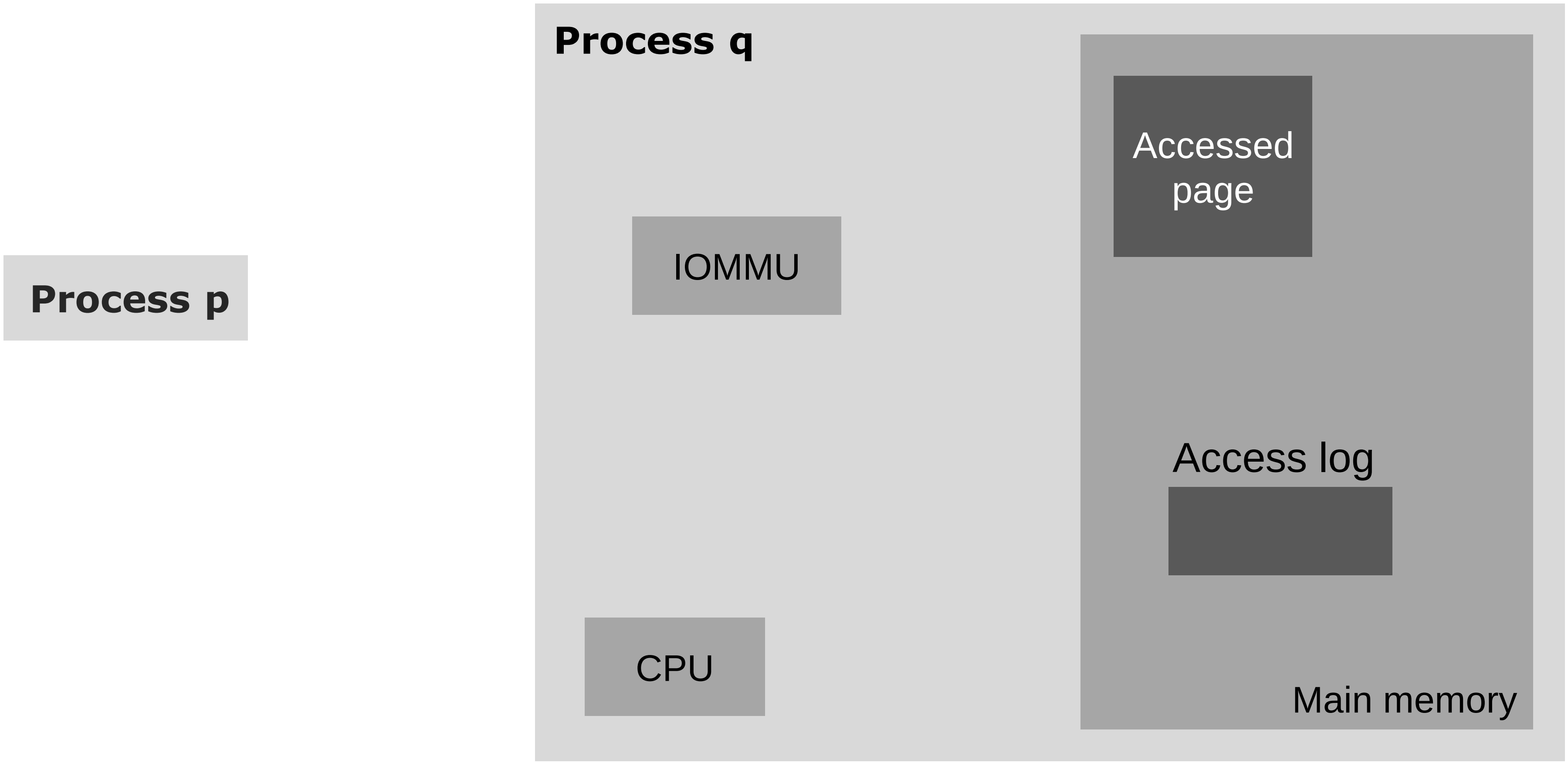

MSI
An RDMA packet
IOMMU
CPU
NIC
Dev-to-PT cache
IOTLB
Access log table
SMT cores
PCIe packets
Main memory
Remapping structures
Dev-to-PT
PT
W
R
WL
WLD
RL
RLD
IUID
System-wide fault log
Fault entry
Fault entry
Access log (private for each process)
Fault entry
Fault entry
Request data
Request data
User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

# Active Gets

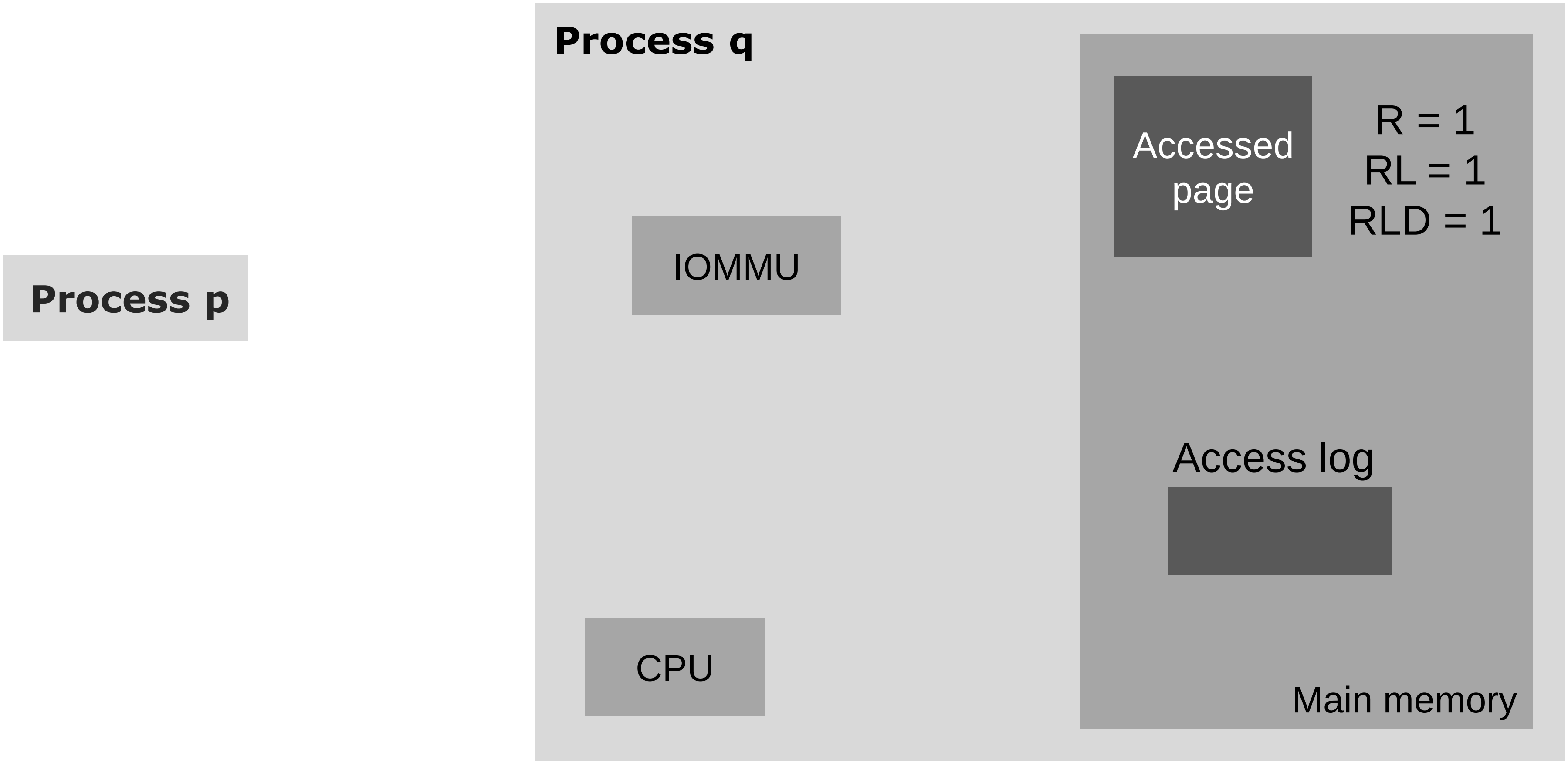
# ACTIVE GETS



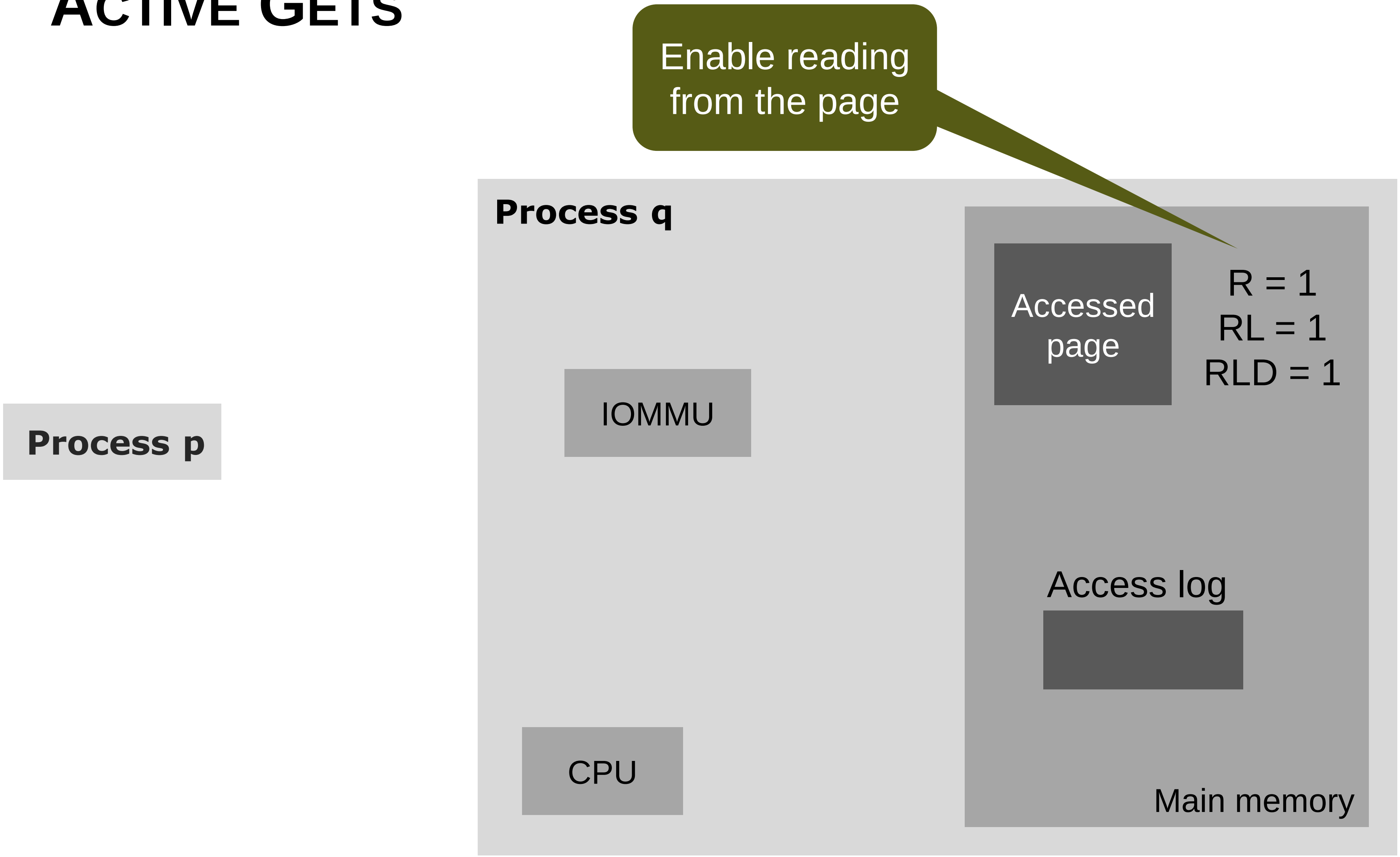
# ACTIVE GETS


Process p
Process q
IOMMU
CPU
Accessed page
Access log
Main memory

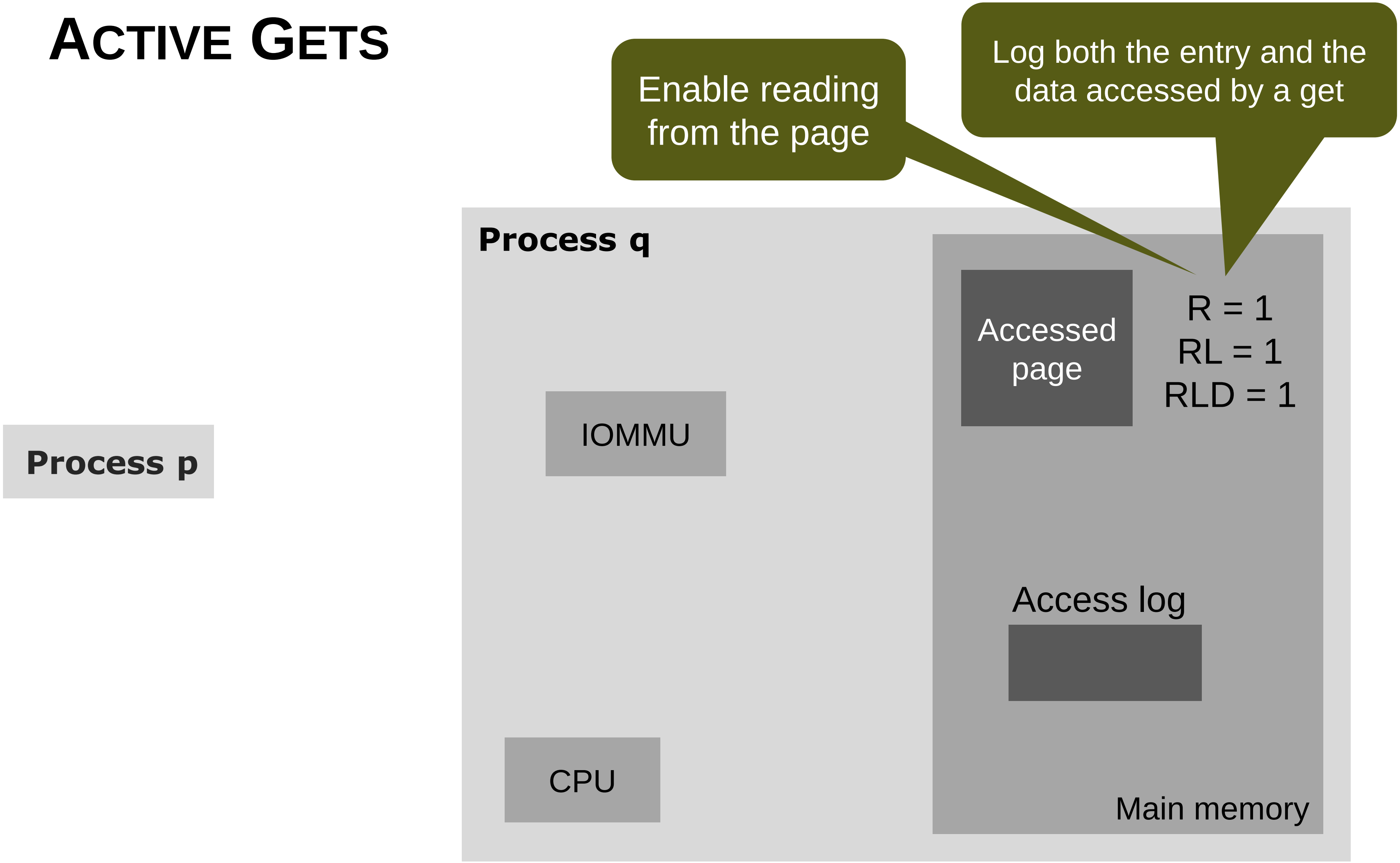
# ACTIVE GETS


Process p
Process q
Accessed page
R = 1
RL = 1
RLD = 1
IOMMU
Access log
CPU
Main memory

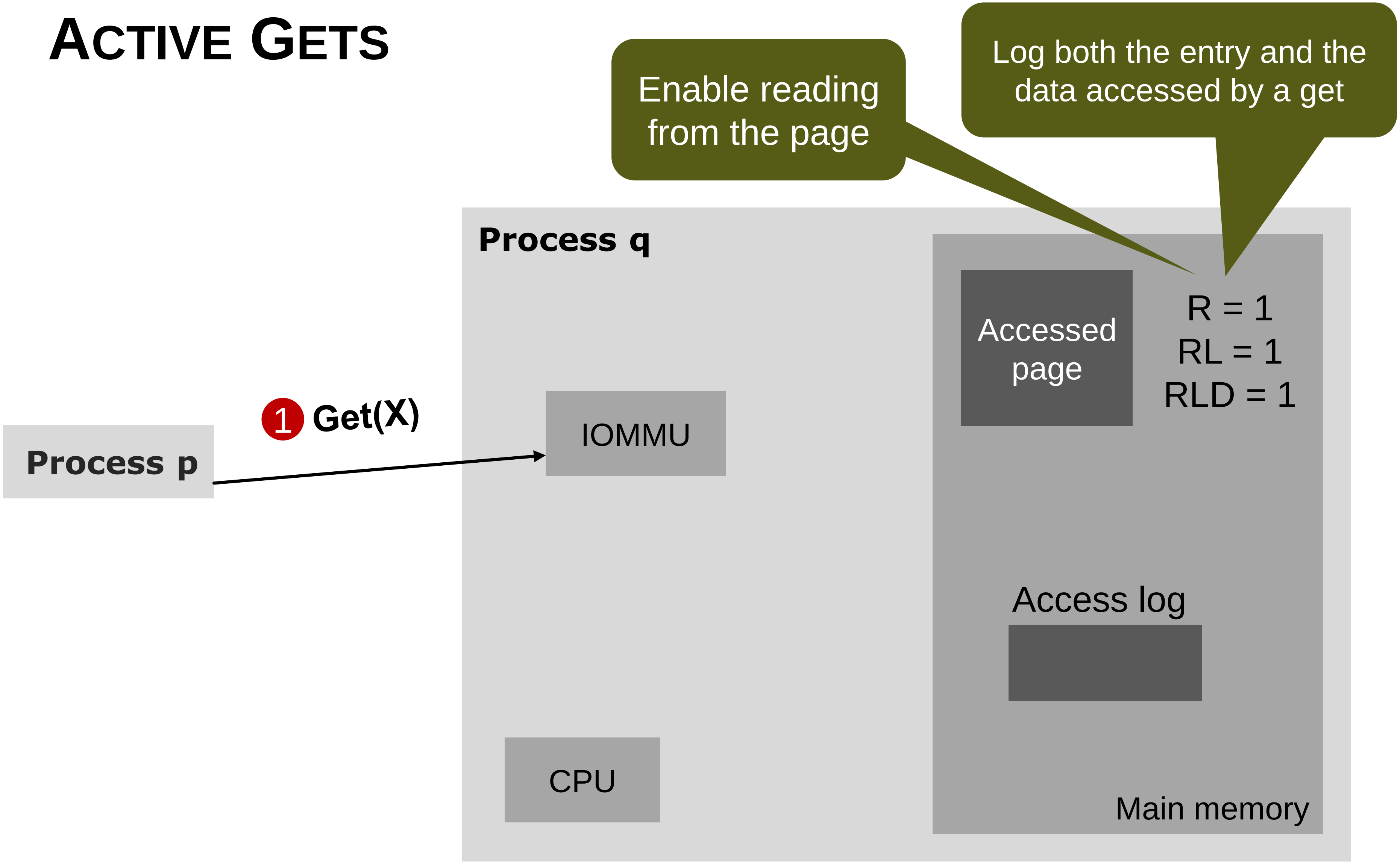
# Active Gets


Enable reading
from the page
Process q
Accessed
page
R = 1
RL = 1
RLD = 1
IOMMU
Process p
Access log
CPU
Main memory

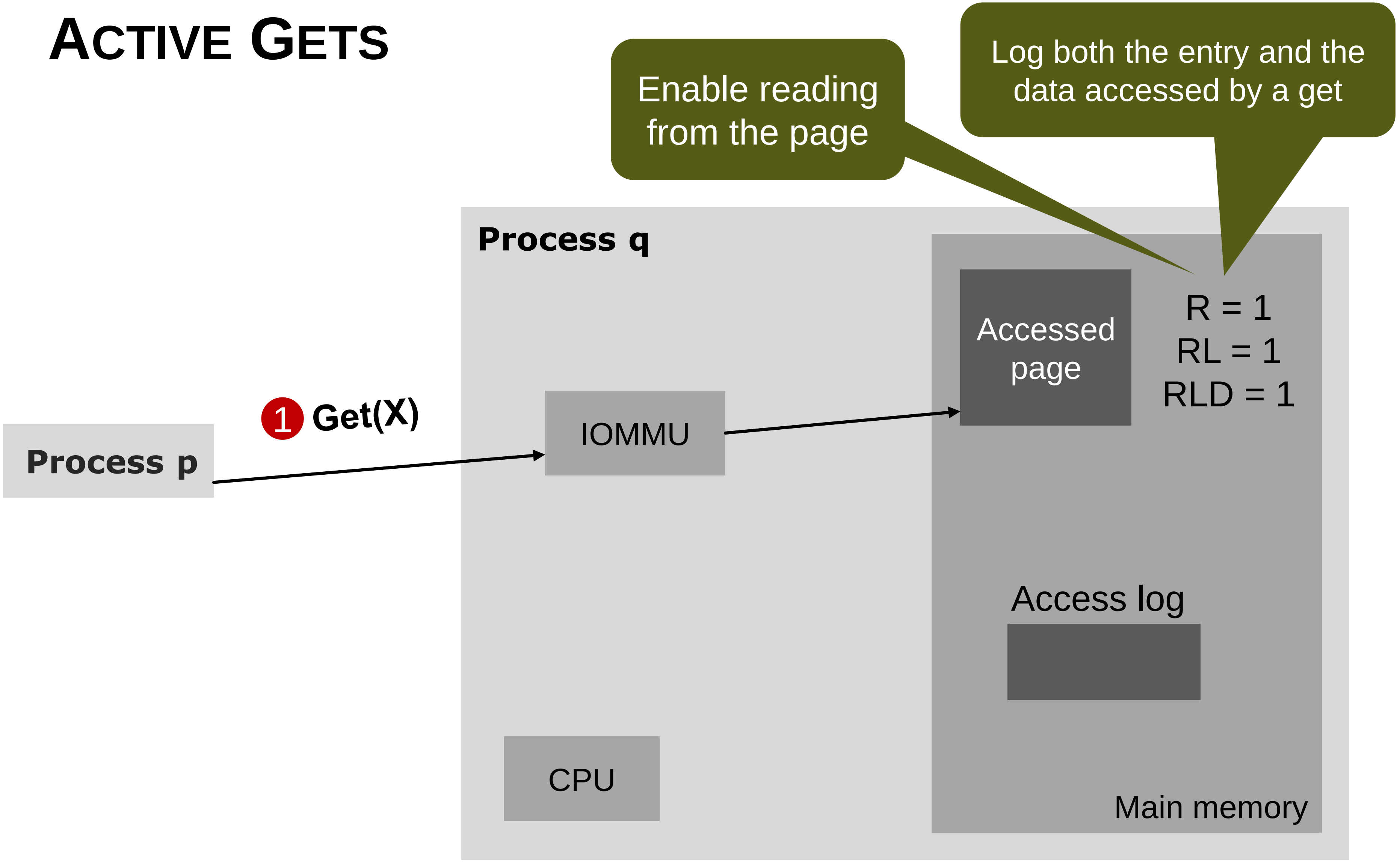
# ACTIVE GETS


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Process p
Accessed page
R = 1
RL = 1
RLD = 1
IOMMU
Access log
CPU
Main memory

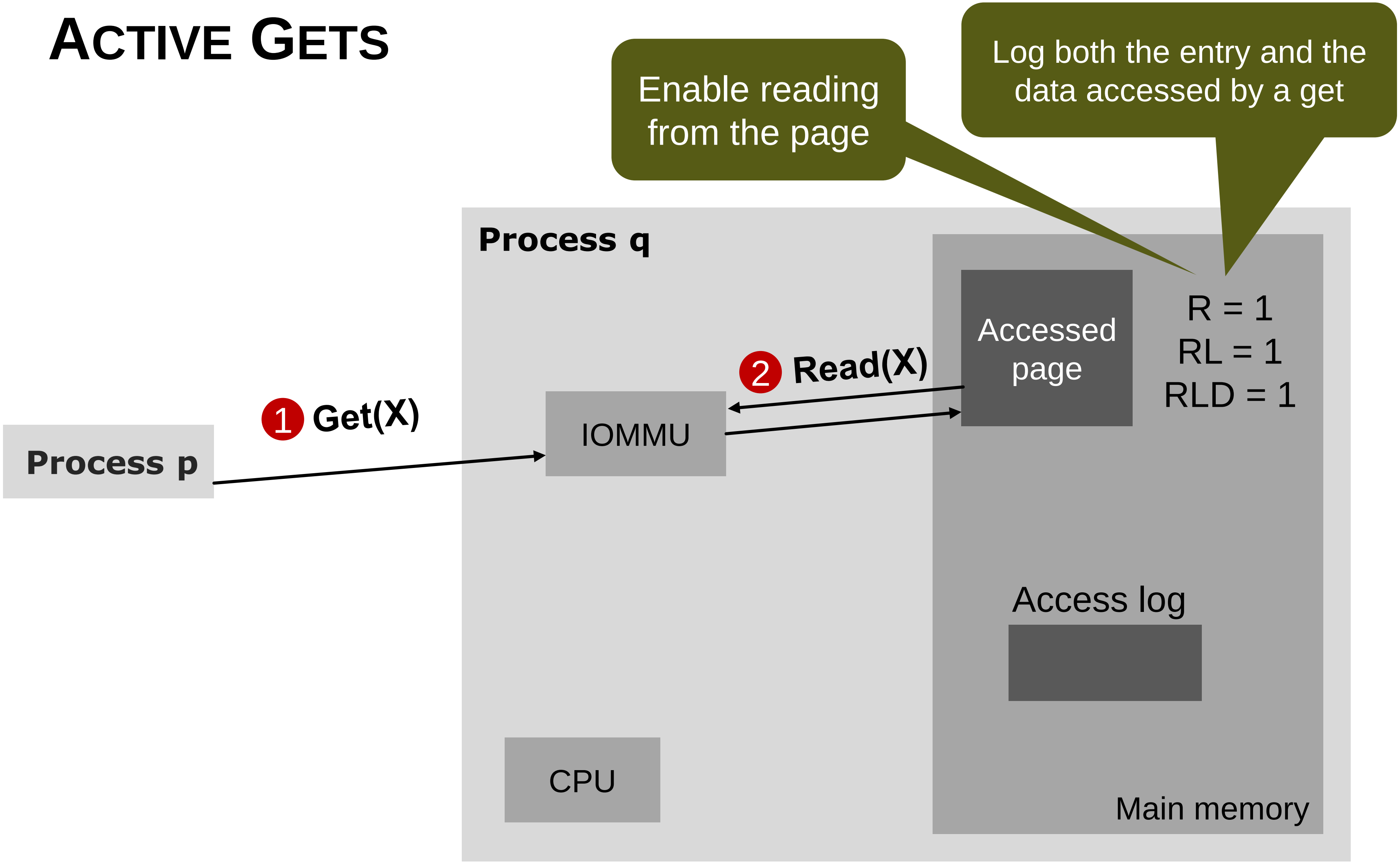
# ACTIVE GETS


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
1 Get(X)
Process p
IOMMU
Access log
CPU
Main memory

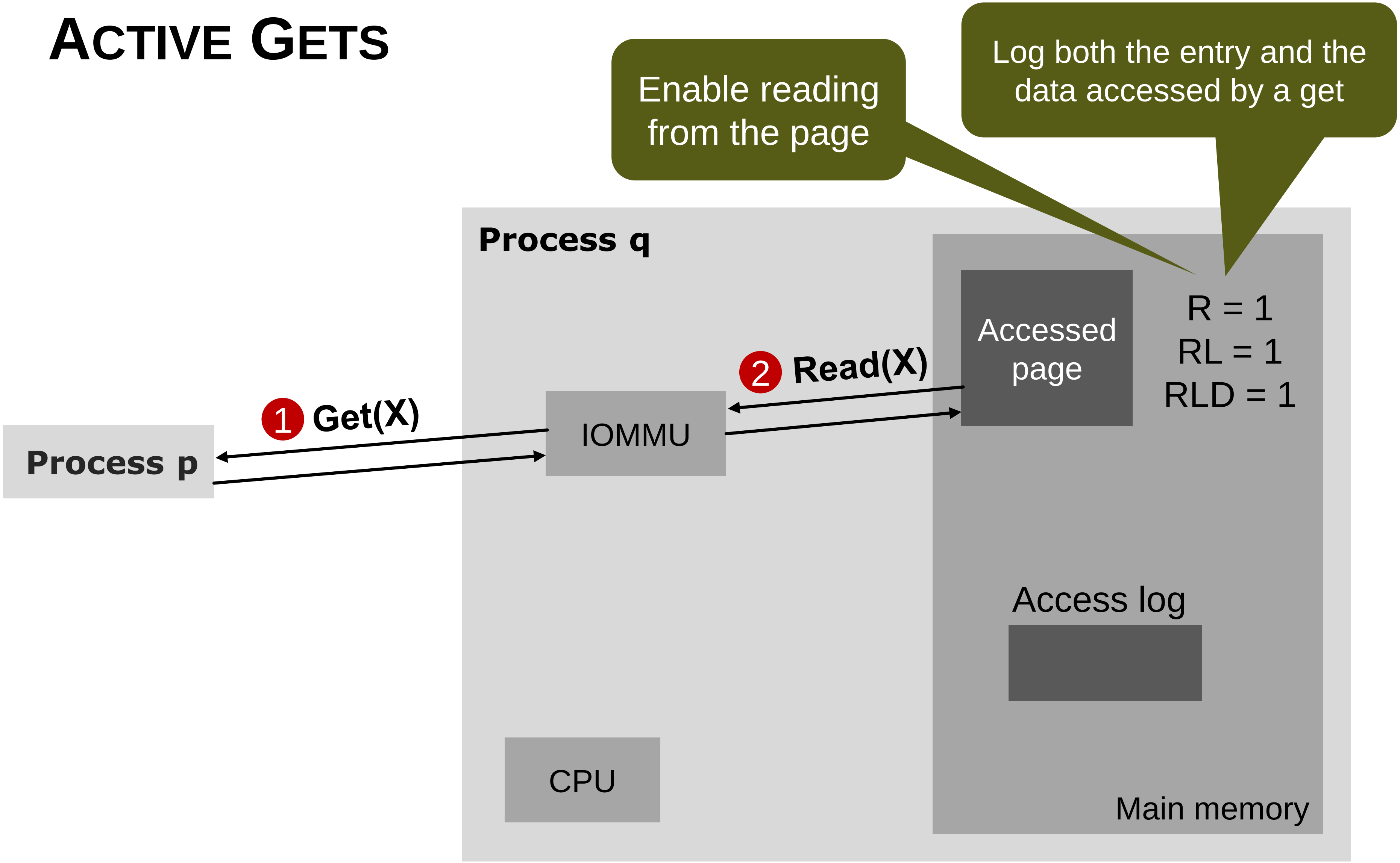
# ACTIVE GETS


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
1 Get(X)
IOMMU
Process p
Access log
CPU
Main memory

# ACTIVE GETS

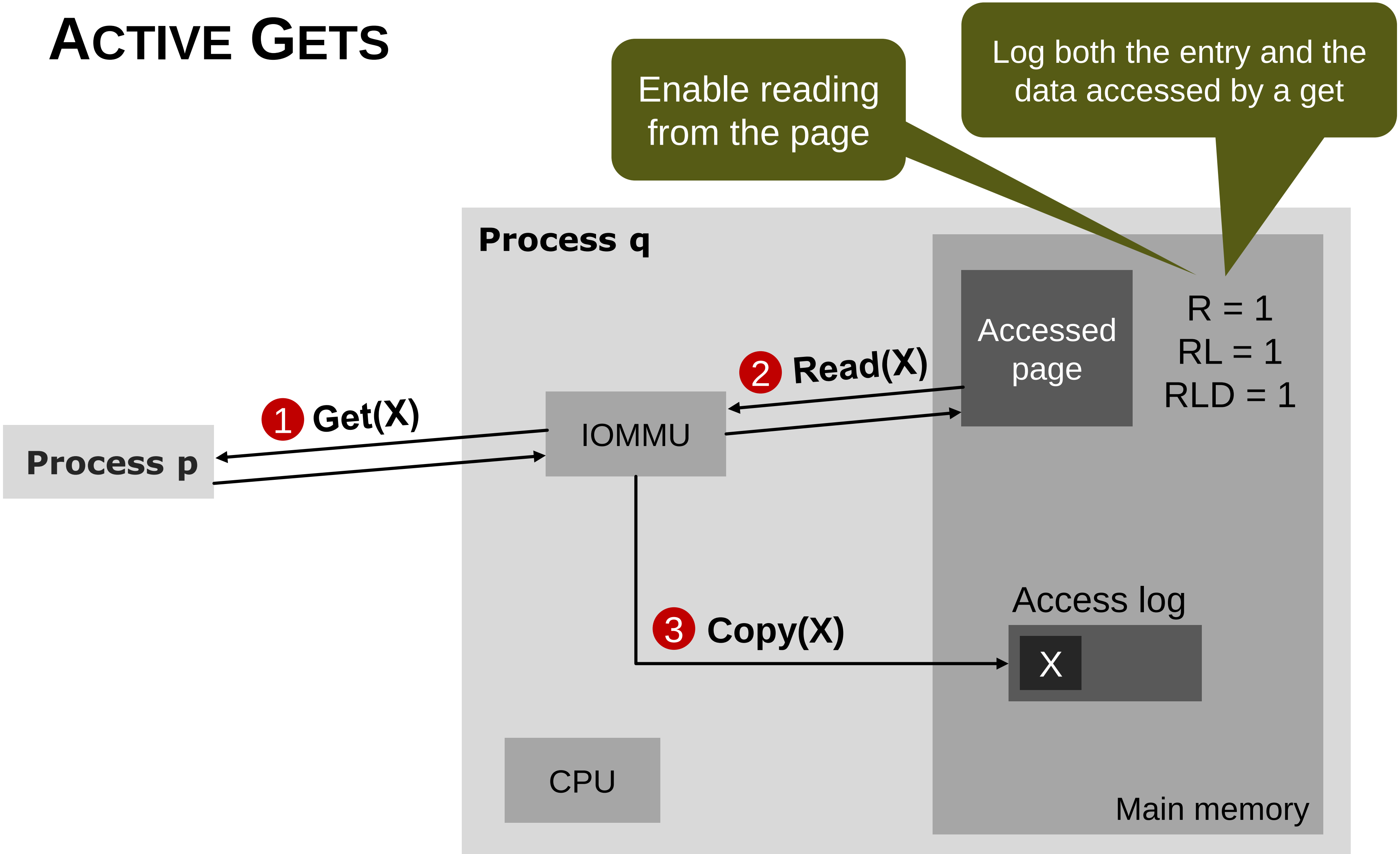

Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
2 Read(X)
1 Get(X)
IOMMU
Process p
Access log
CPU
Main memory

# Active Gets

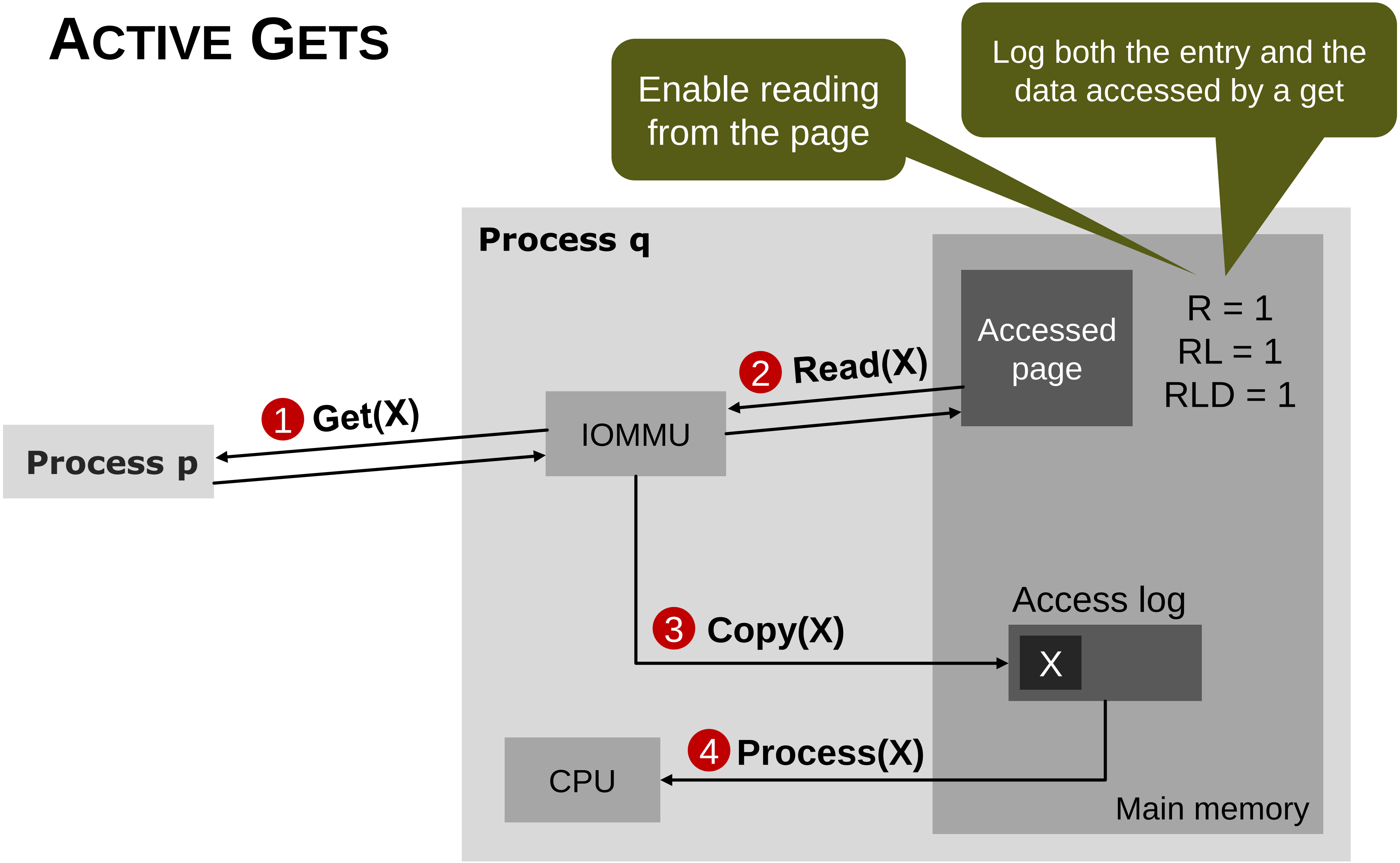


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
2 Read(X)
1 Get(X)
Process p
IOMMU
Access log
CPU
Main memory

# ACTIVE GETS


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
2 Read(X)
1 Get(X)
Process p
IOMMU
3 Copy(X)
Access log
X
CPU
Main memory

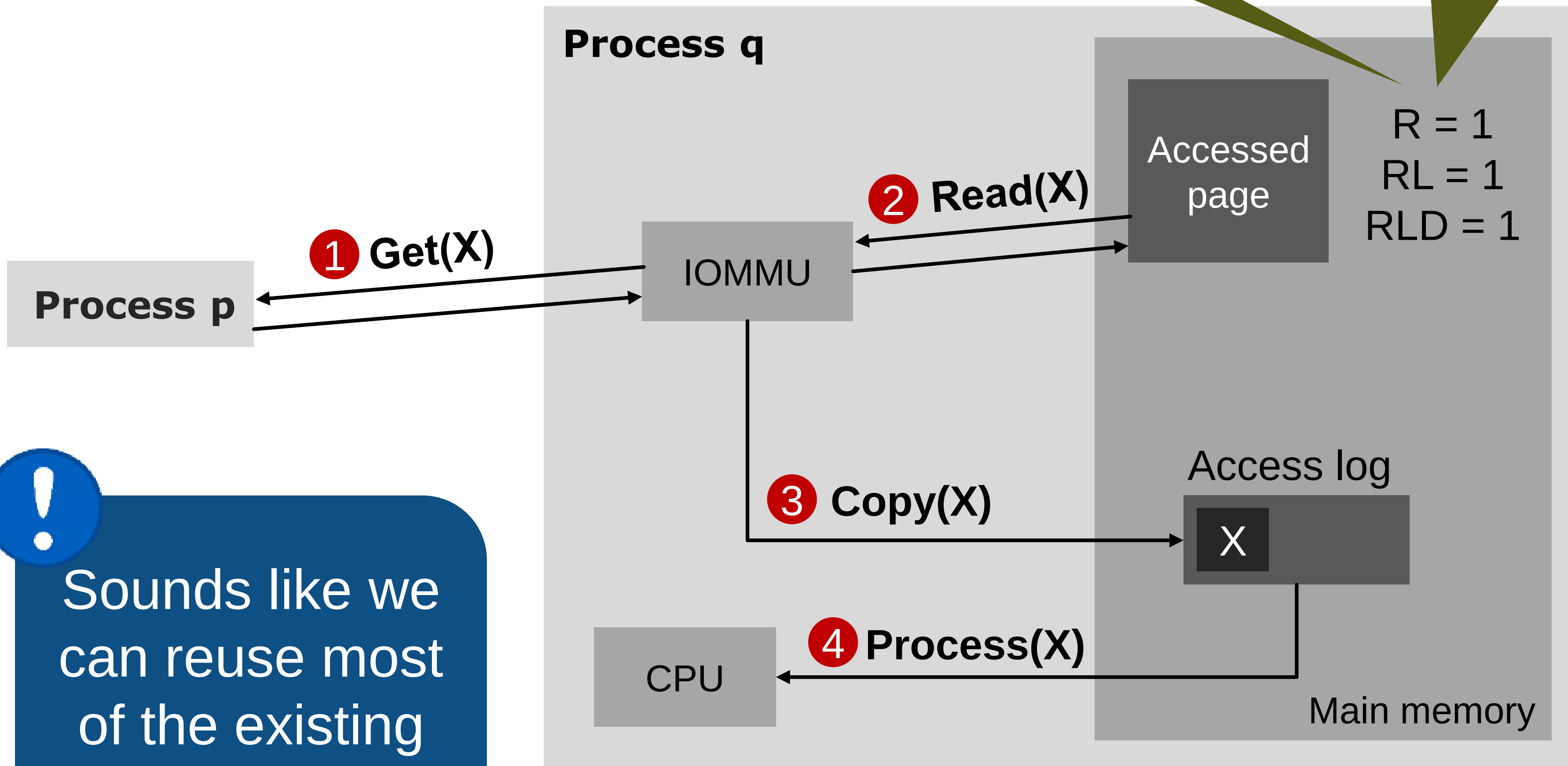
# Active Gets


Enable reading from the page
Log both the entry and the data accessed by a get
Process q
Accessed page
R = 1
RL = 1
RLD = 1
2 Read(X)
1 Get(X)
Process p
IOMMU
Access log
X
3 Copy(X)
4 Process(X)
CPU
Main memory

# ACTIVE GETS

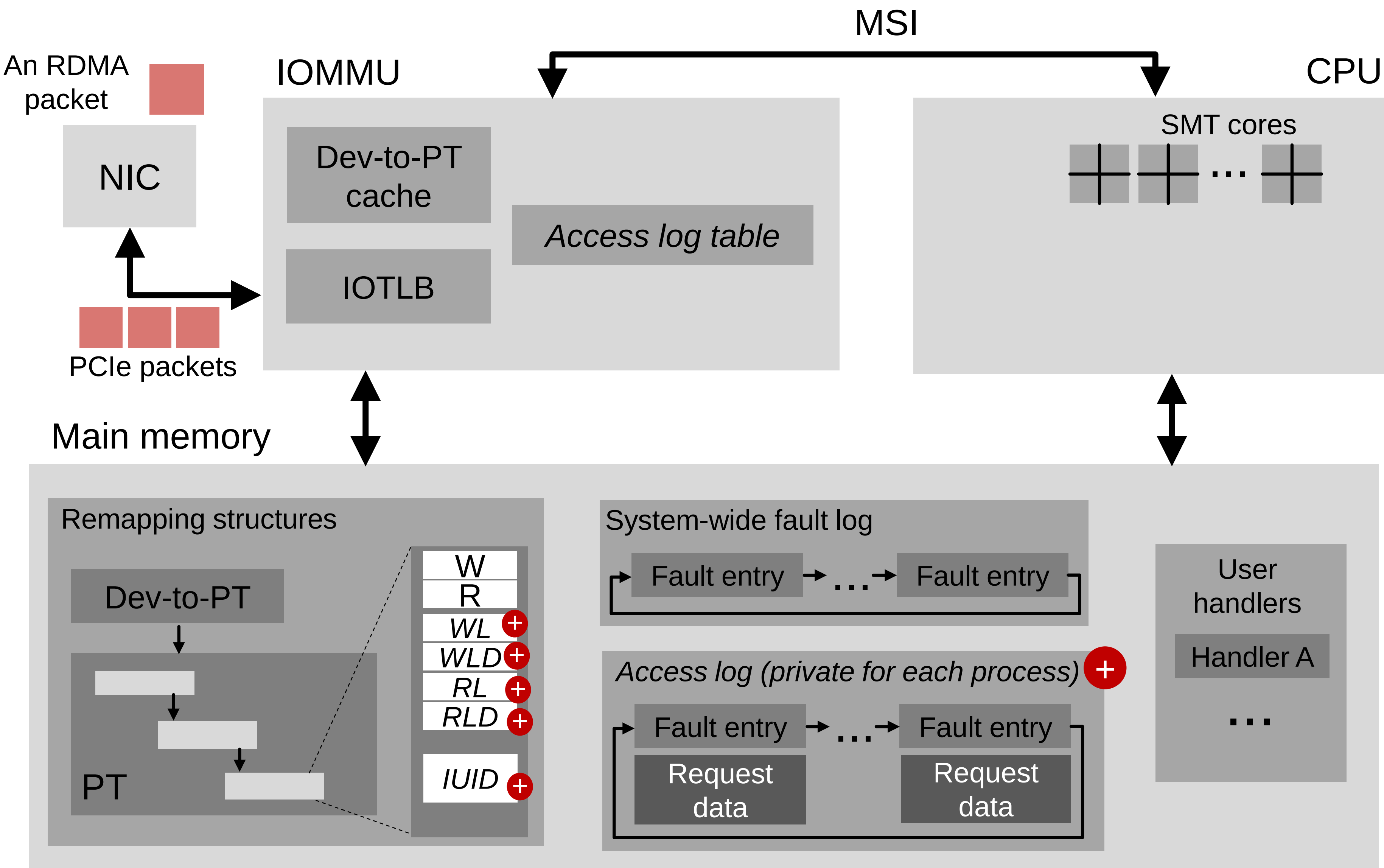
Enable reading from the page

Log both the entry and the data accessed by a get

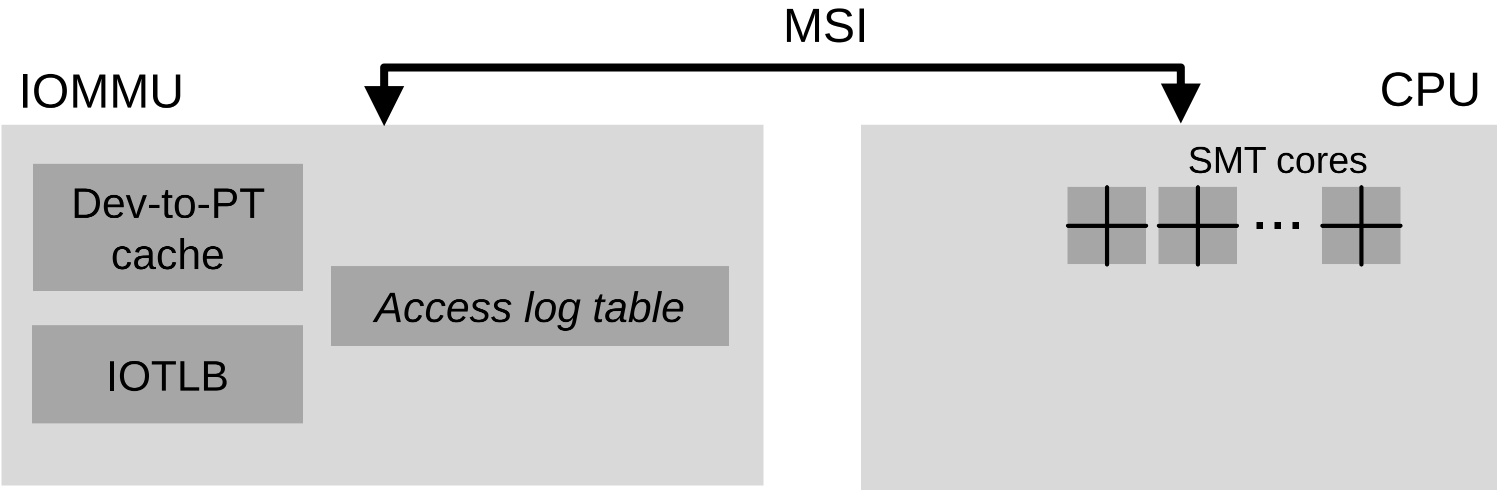



Sounds like we can reuse most of the existing stuff!

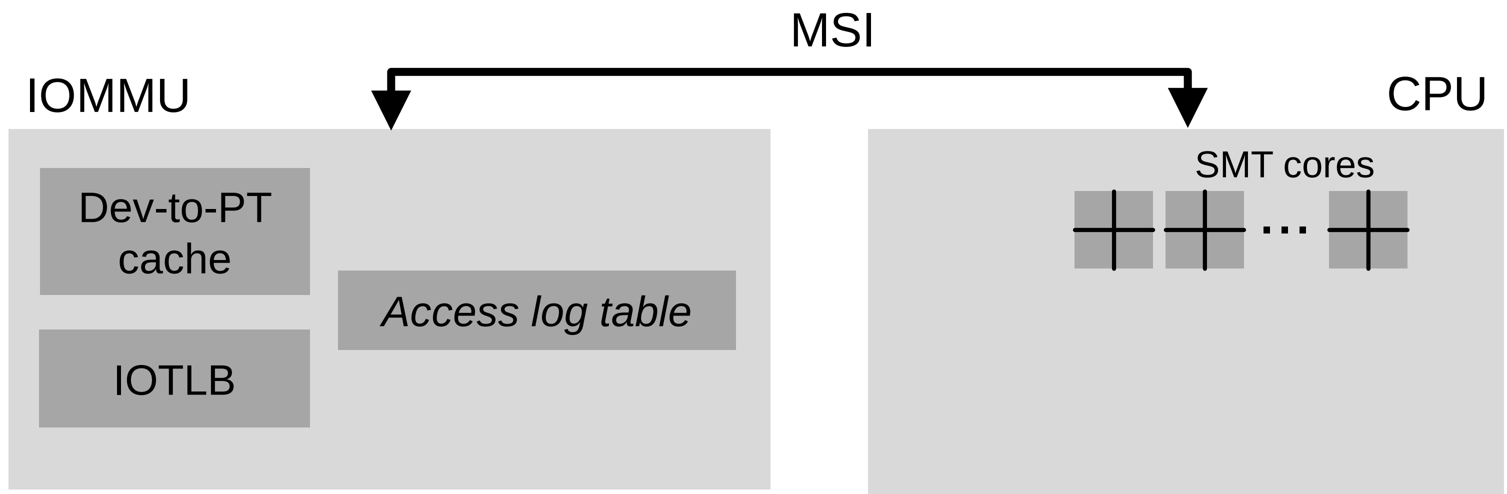
# Interactions with the CPU



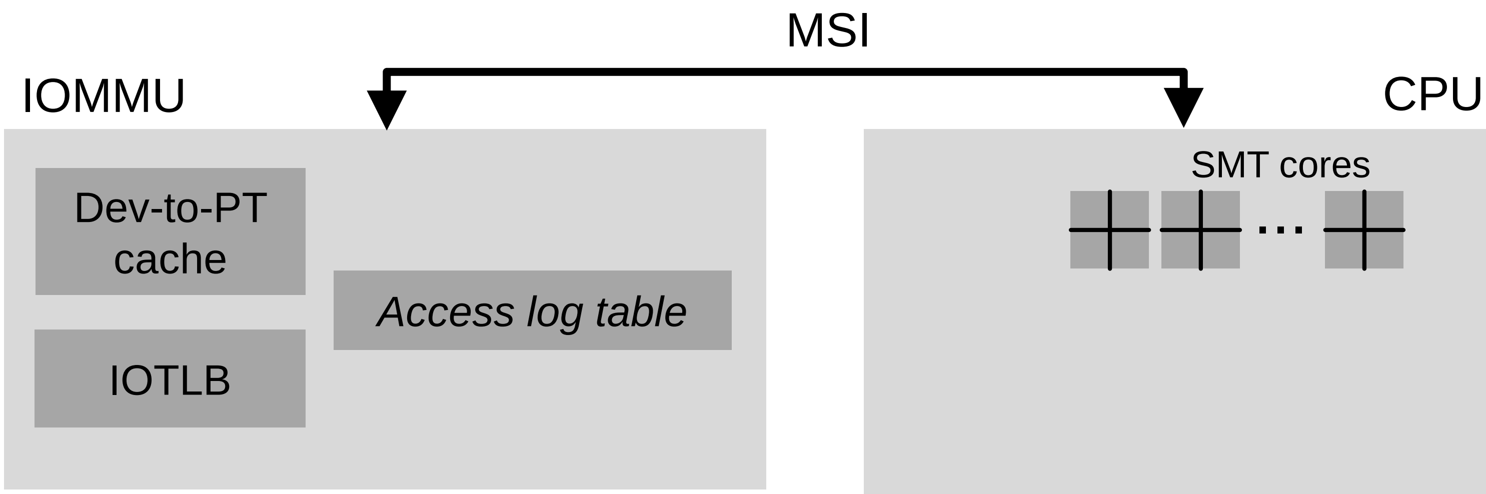
# INTERACTIONS WITH THE CPU



# INTERACTIONS WITH THE CPU

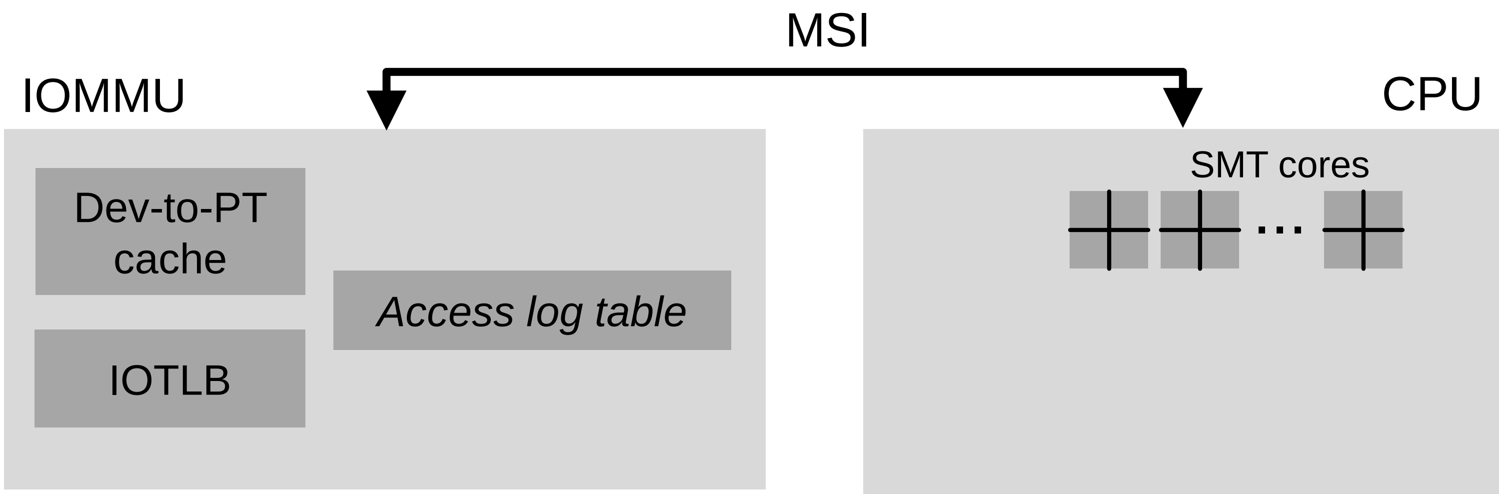


# INTERACTIONS WITH THE CPU



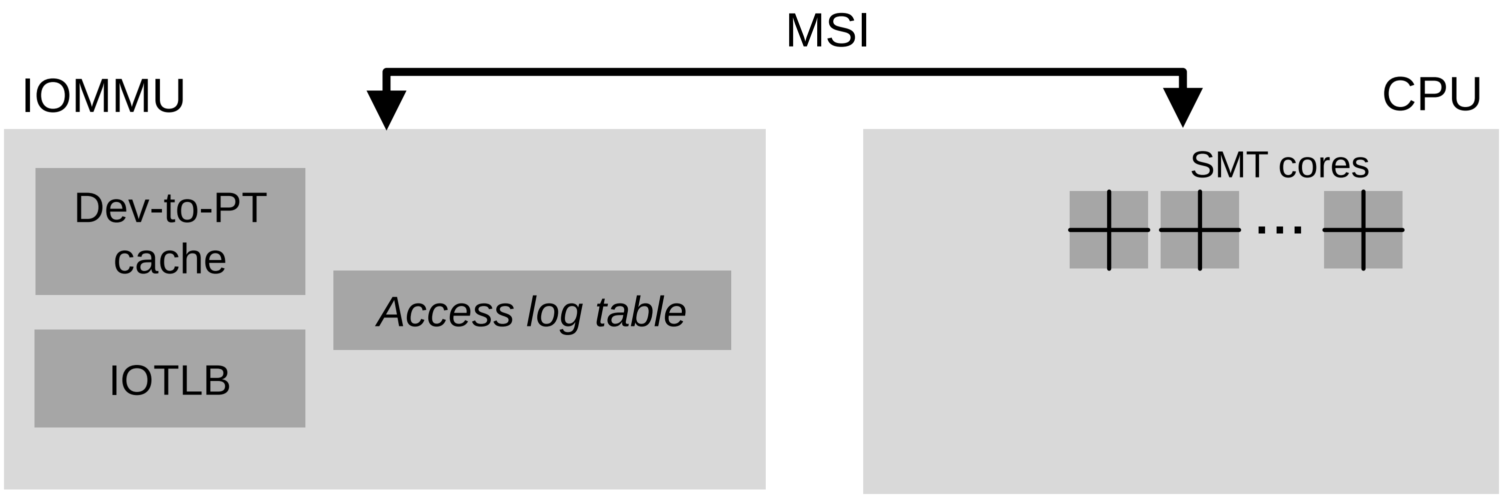

- Interrupts

# INTERACTIONS WITH THE CPU



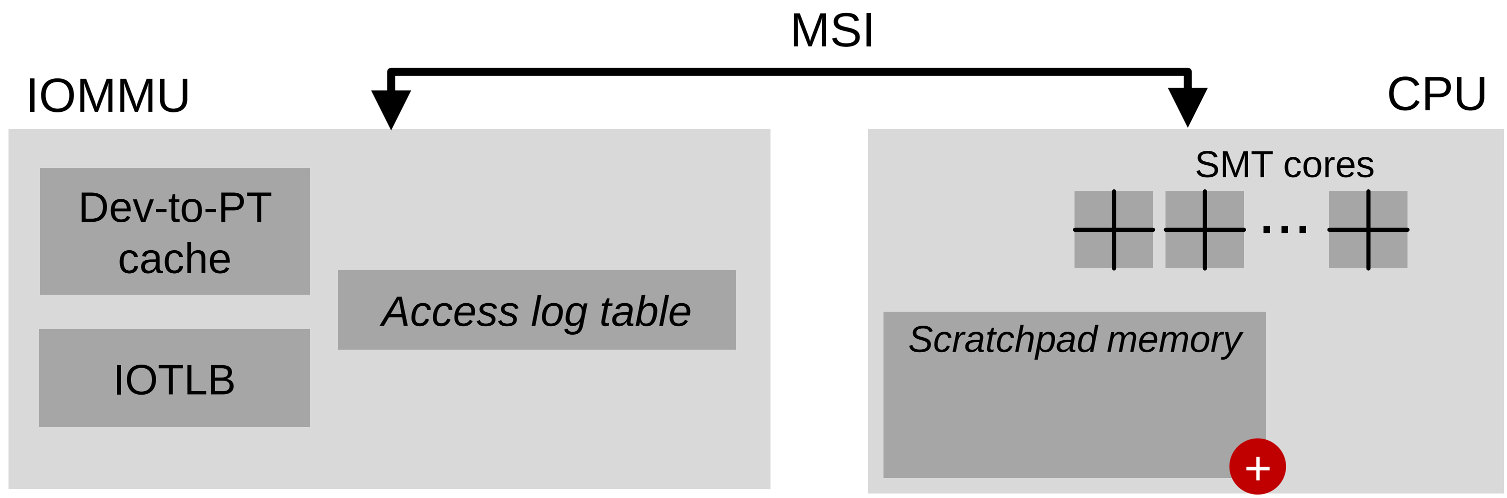

- Interrupts
- Polling

# INTERACTIONS WITH THE CPU



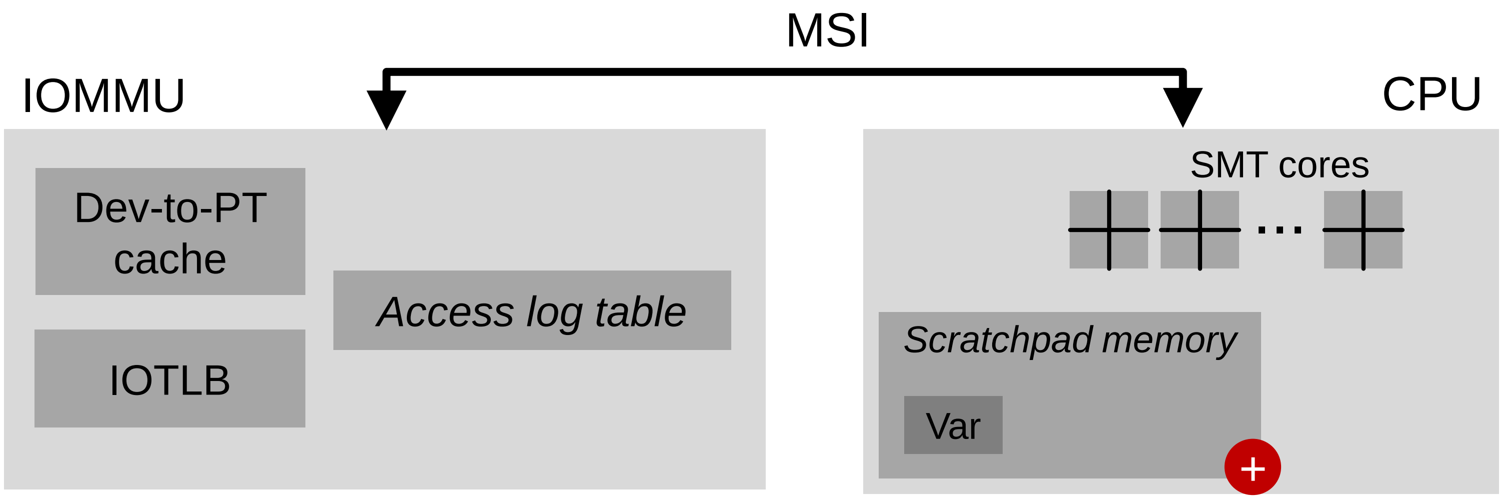

- Interrupts
- Polling
- Direct notifications via scratchpads

# INTERACTIONS WITH THE CPU



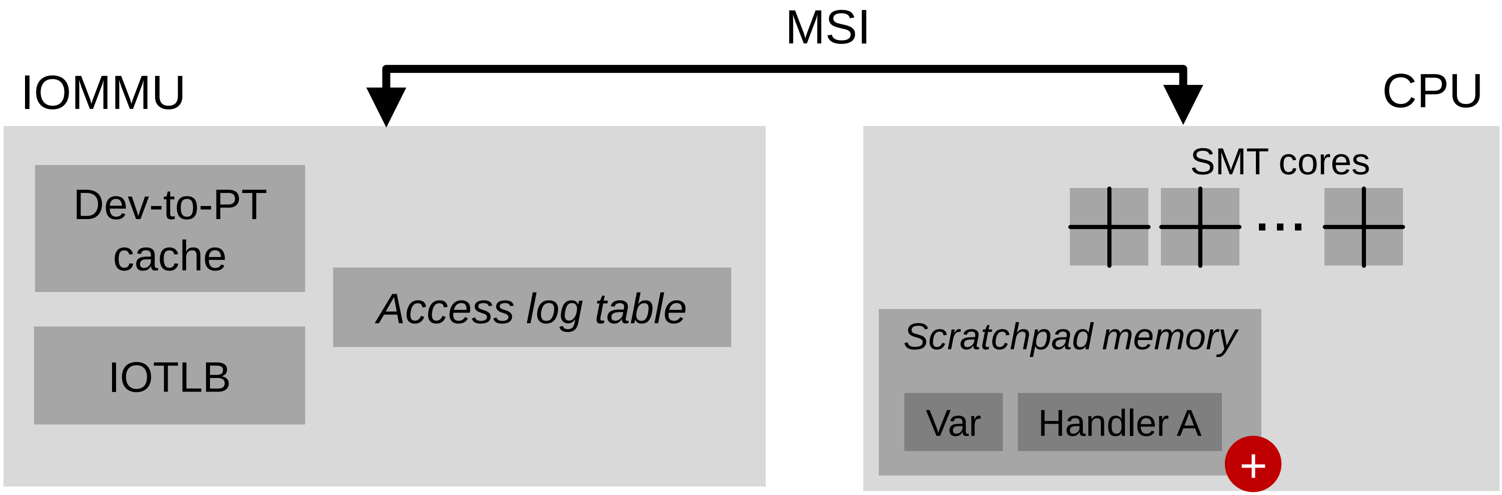

- Interrupts
- Polling
- Direct notifications via scratchpads

# INTERACTIONS WITH THE CPU



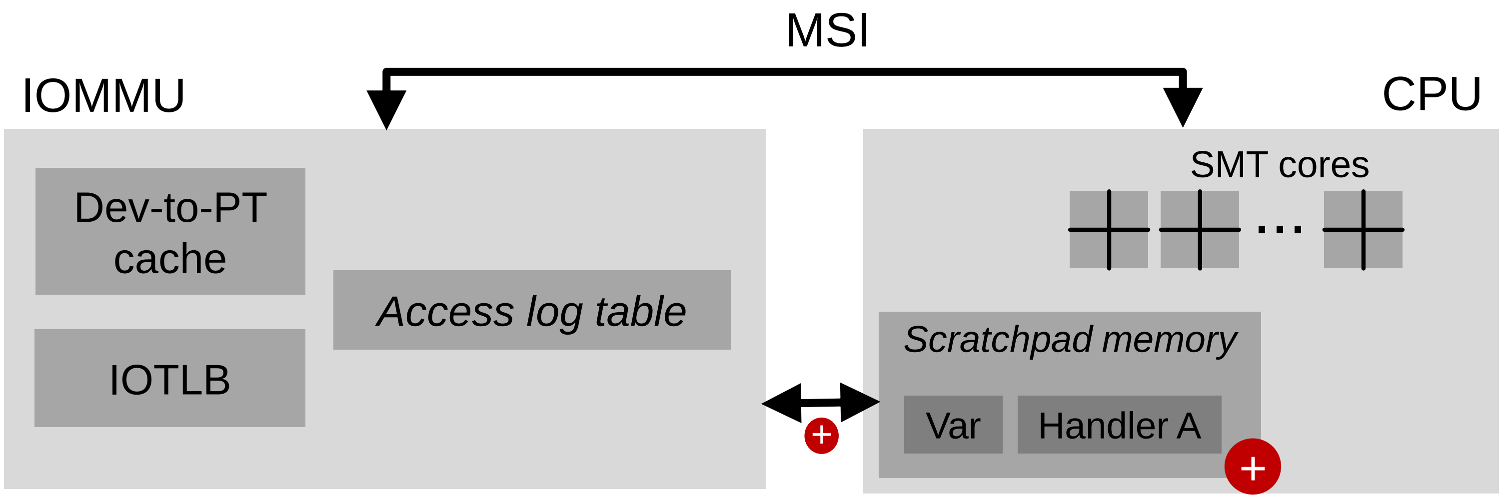

- Interrupts
- Polling
- Direct notifications via scratchpads

# INTERACTIONS WITH THE CPU




- Interrupts
- Polling
- Direct notifications via scratchpads

# INTERACTIONS WITH THE CPU



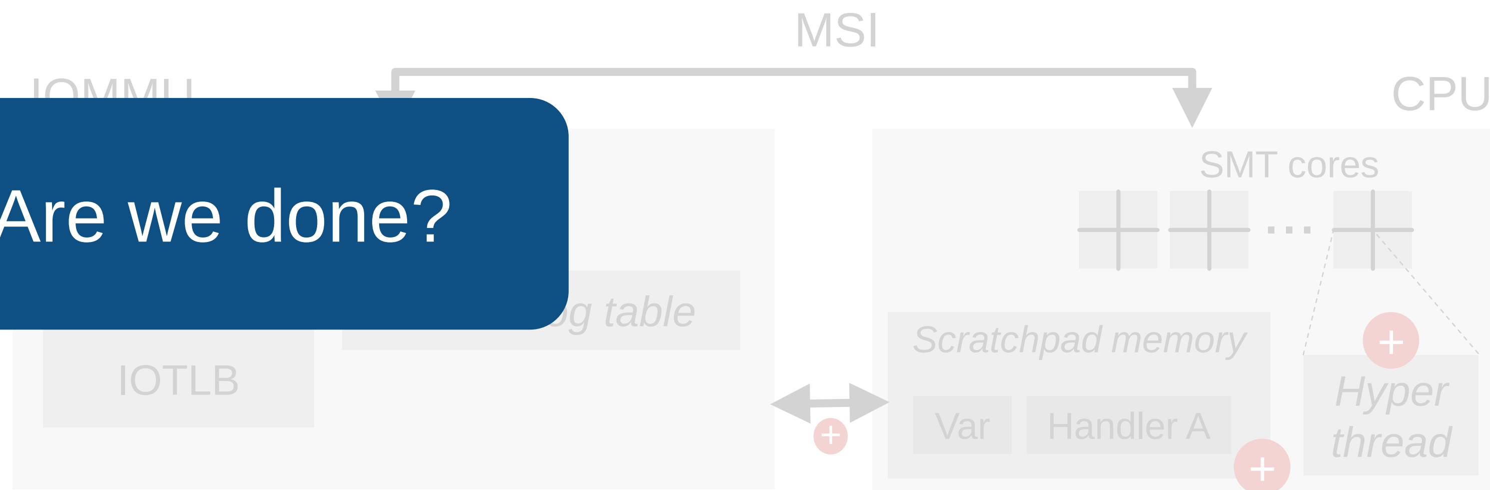

- Interrupts
- Polling
- Direct notifications via scratchpads

# Interactions with the CPU




- Interrupts
- Polling
- Direct notifications via scratchpads

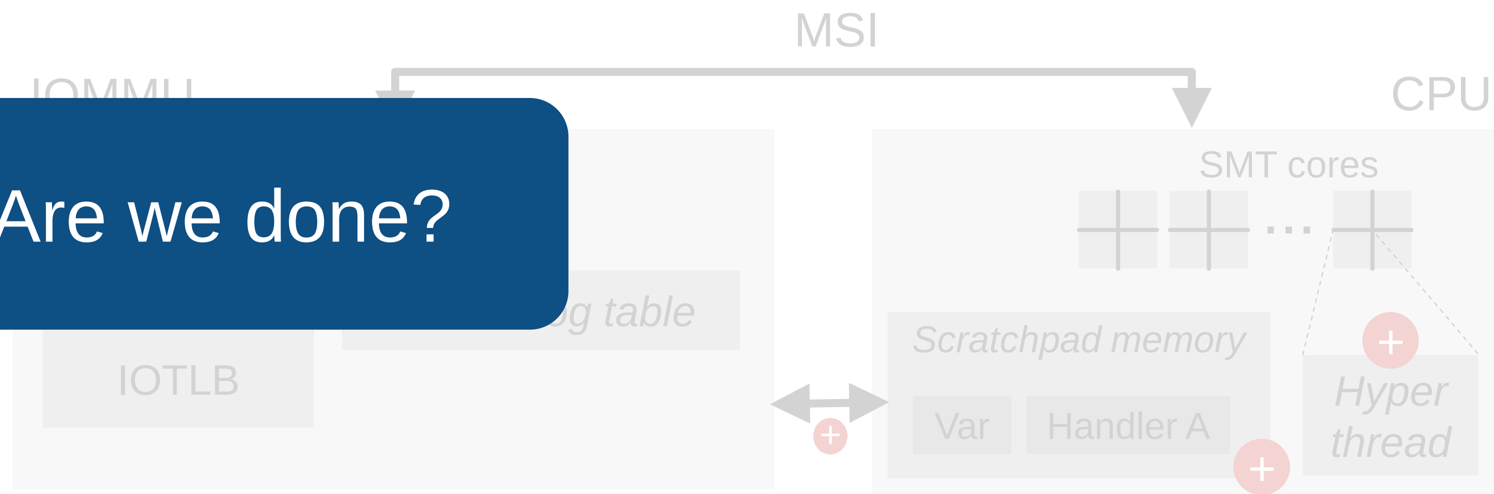
# Interactions with the CPU




Are we done?

- Interrupts
- Polling
- Direct notifications via scratchpads

# Interactions with the CPU




Are we done?

- Interrupts
- Polling
- Direct notifications via sc

Well…

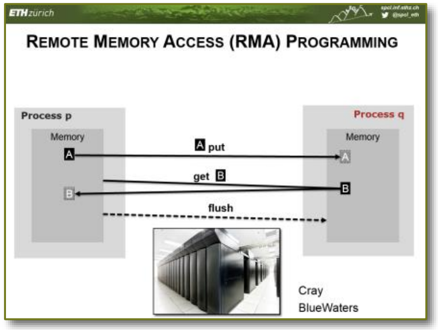
# CONSISTENCY

# CONSISTENCY

- A weak consistency model [1]

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

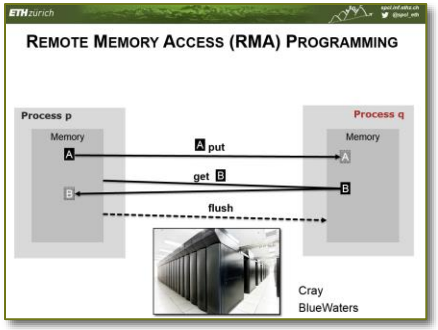
# CONSISTENCY




- A weak consistency model [1]
  - Consistency on-demand

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

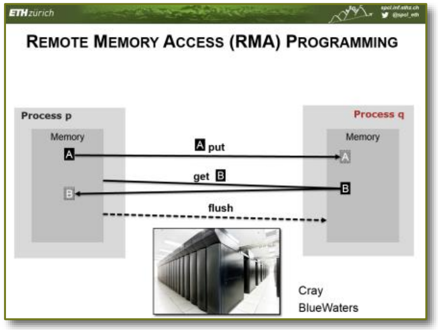
# CONSISTENCY




- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)

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

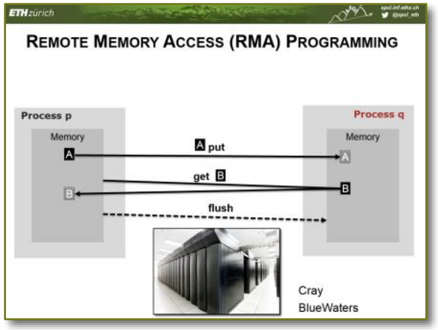
# CONSISTENCY




- 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

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

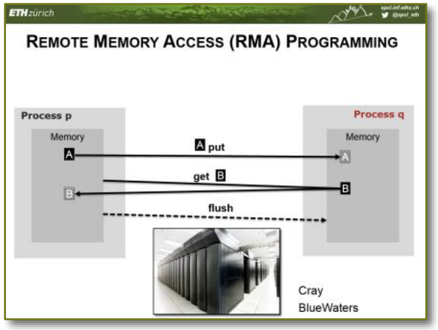
# Consistency



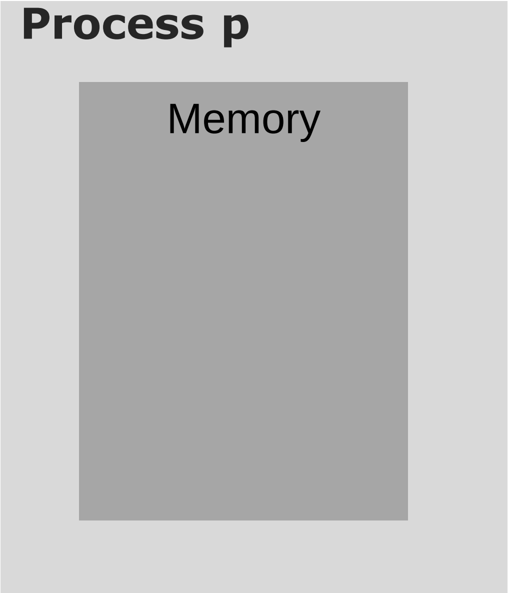

- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

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

# CONSISTENCY

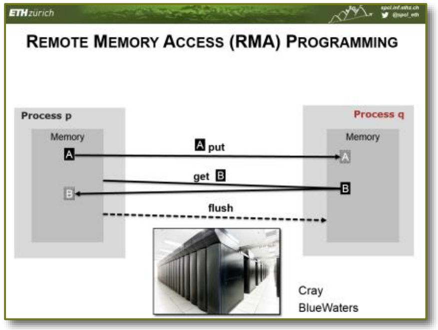



- A weak consistency model [1]
  - Consistency on-demand
- `active_flush(int target_id)`
  - Enforces the completion of active accesses issued by the calling process and targeted at `target_id`
  - Implemented with an active get issued at a special *flushing page*




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

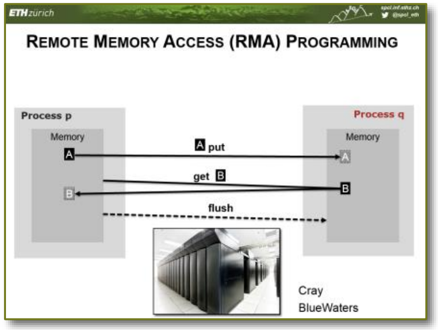
# CONSISTENCY




- 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

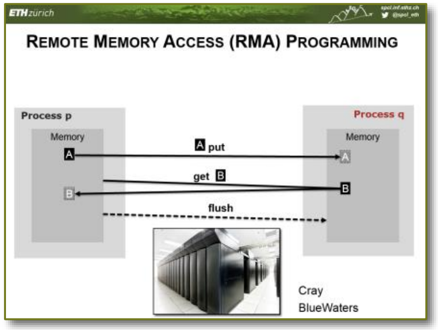
# CONSISTENCY




- 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

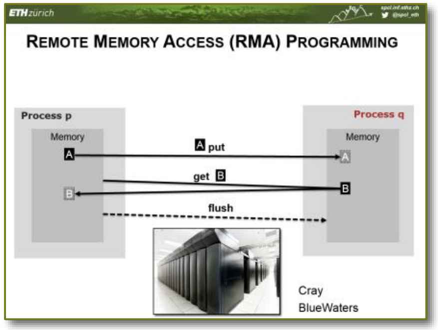
# CONSISTENCY



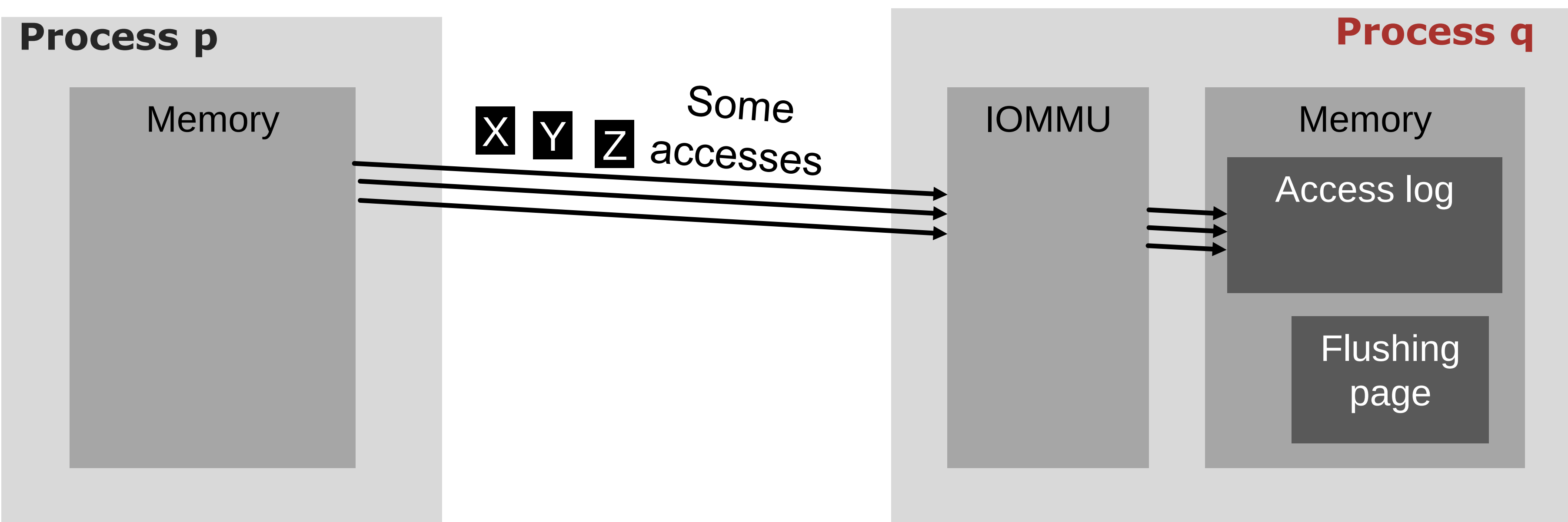

- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*


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

# Consistency

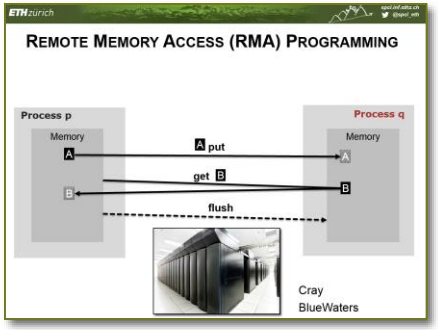



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

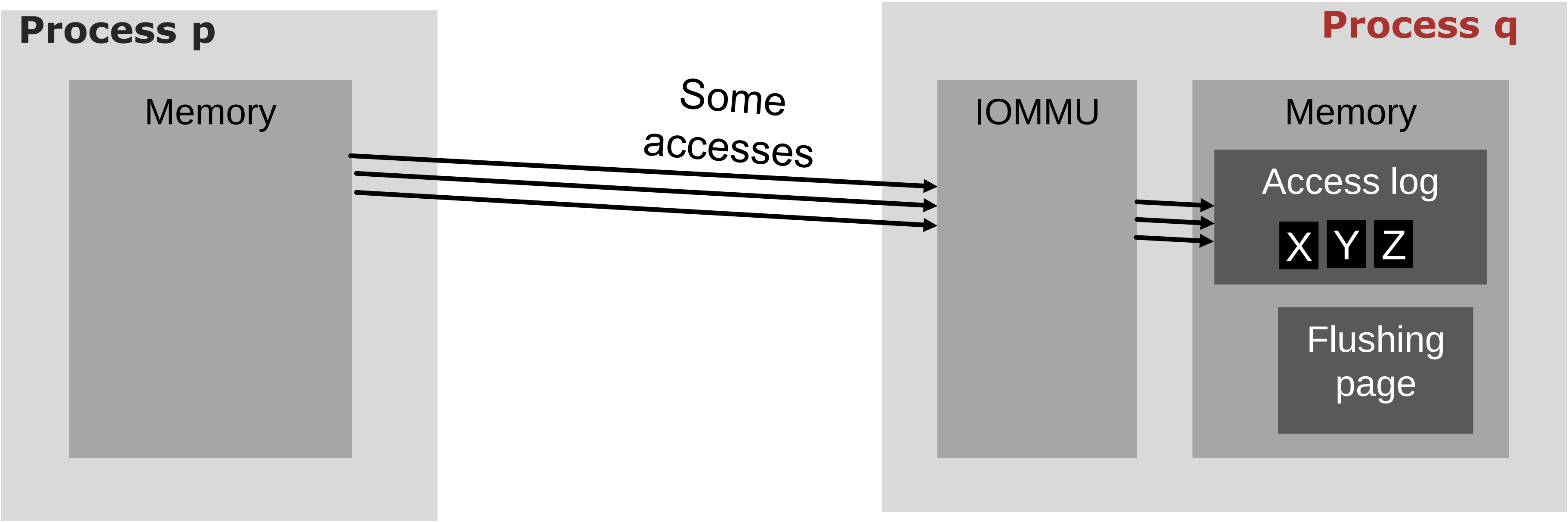



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency



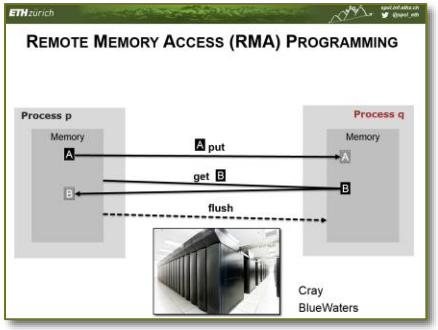

- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

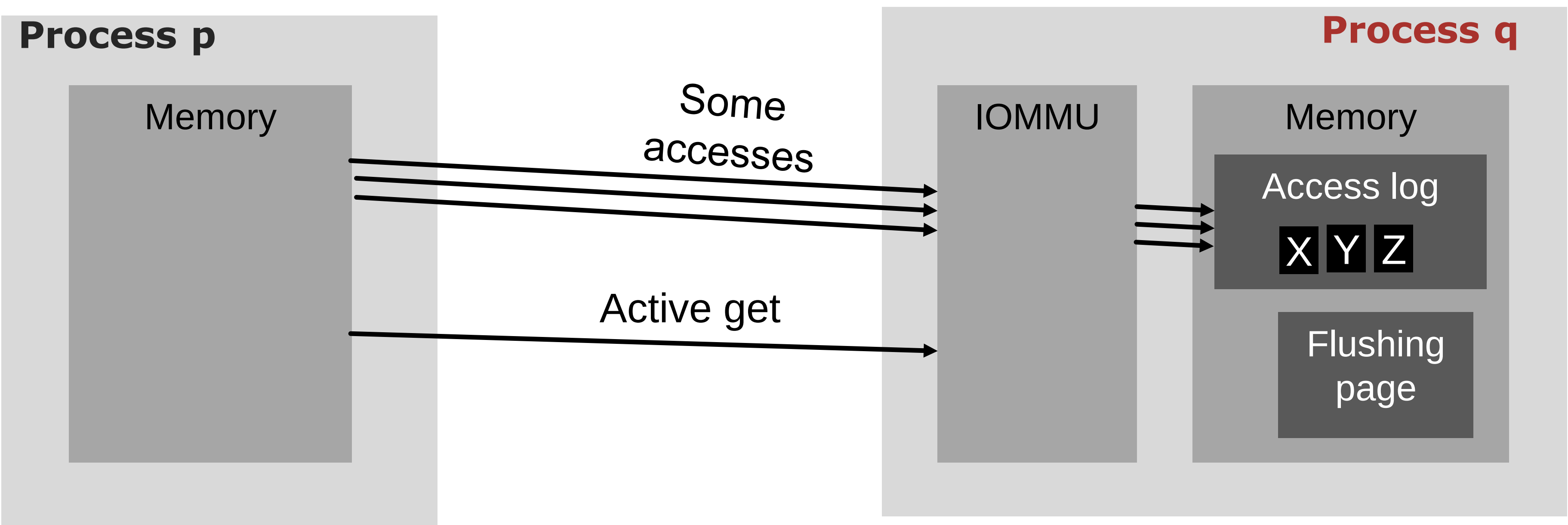



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency

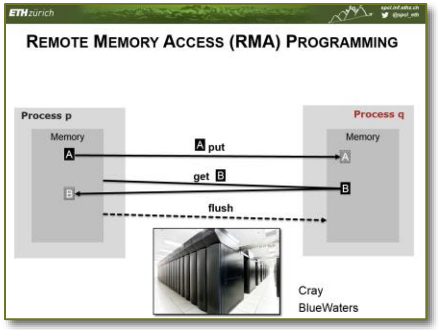
- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*



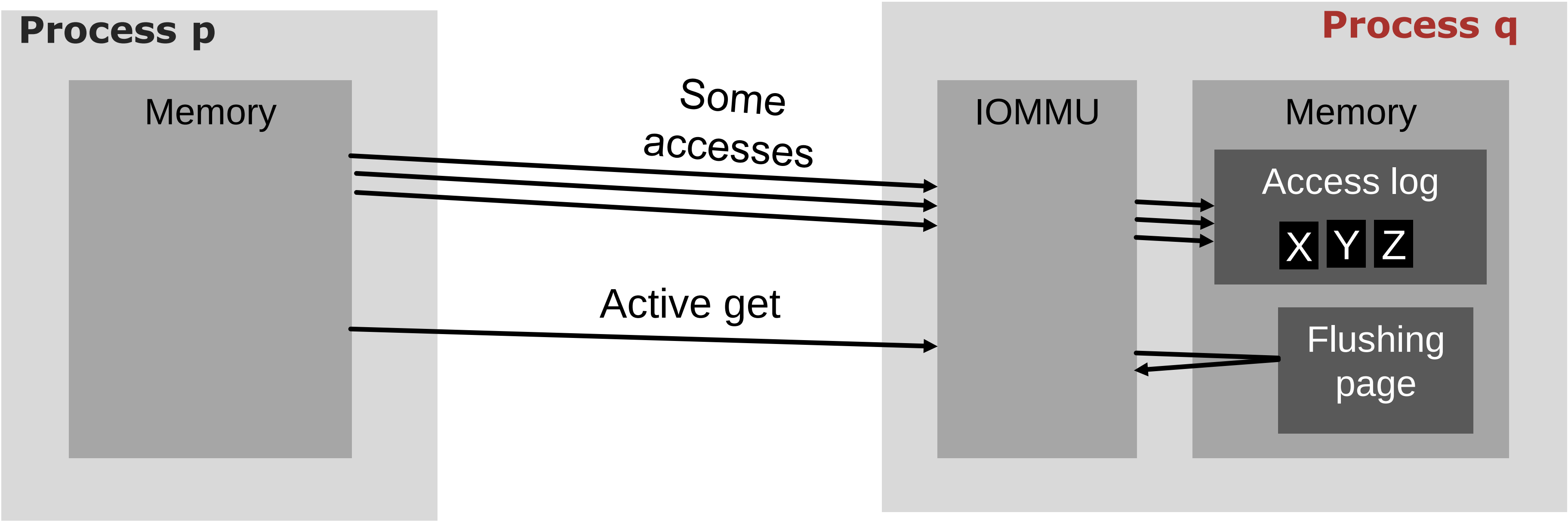




[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# CONSISTENCY

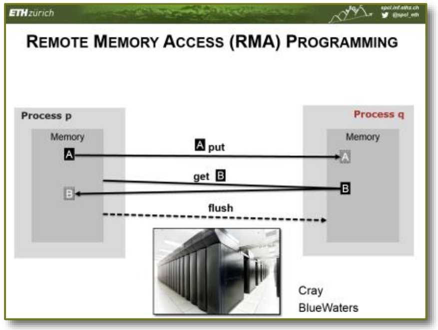



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

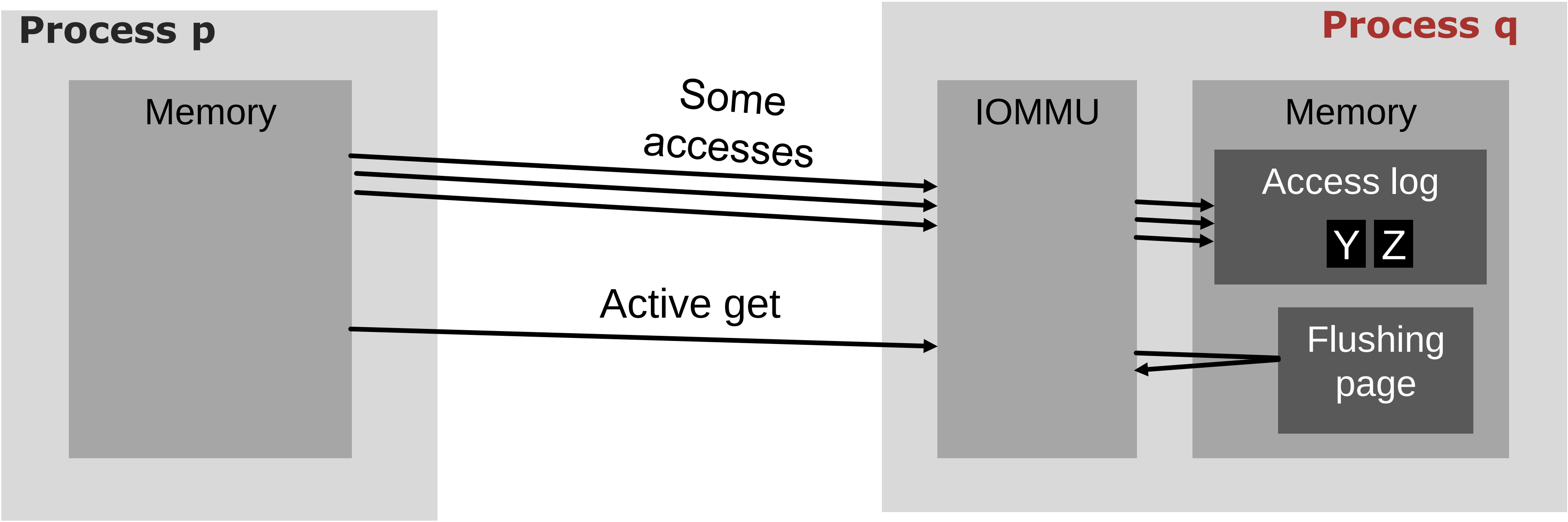



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency

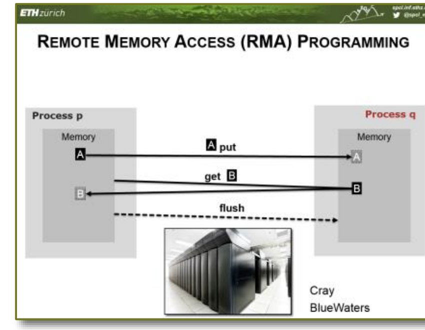



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

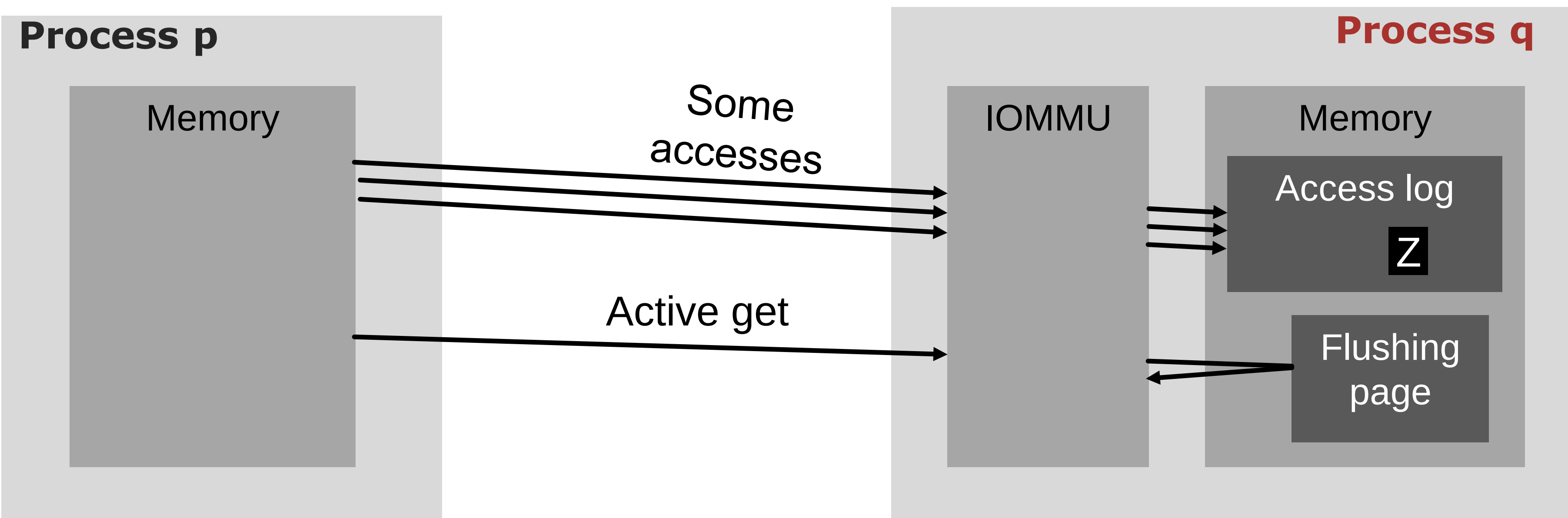



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency

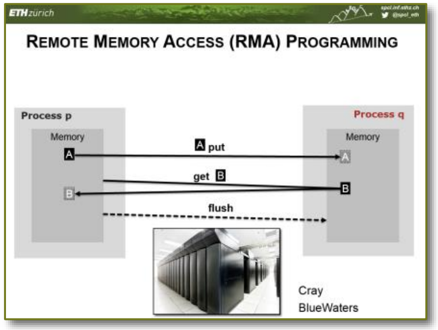



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

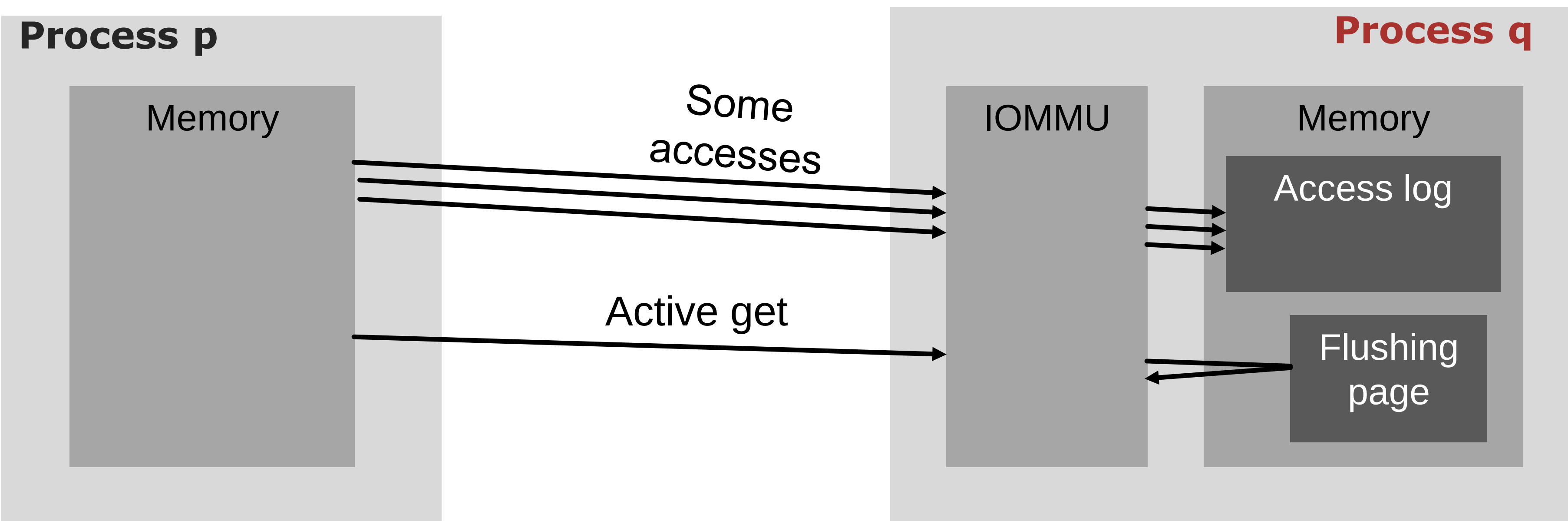



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency

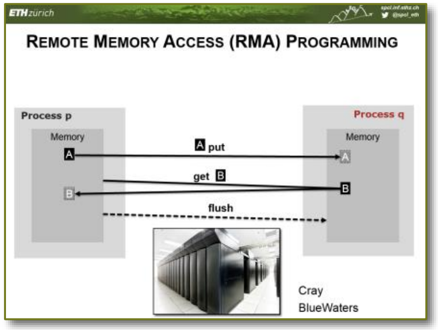



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

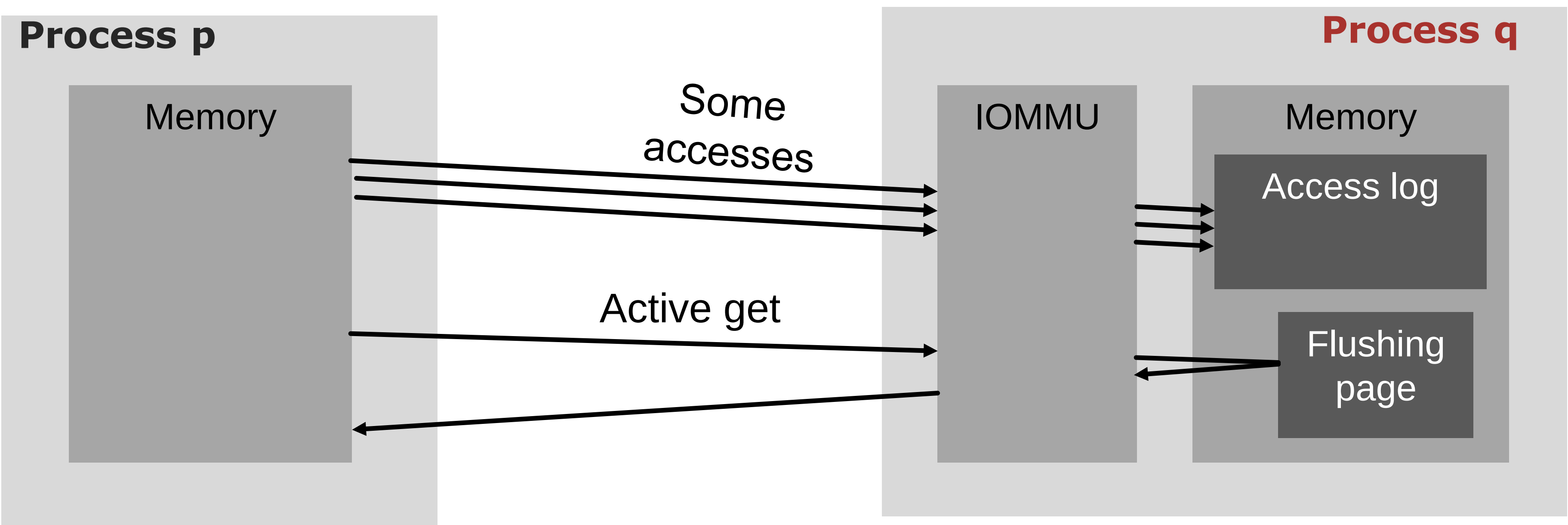



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

# Consistency

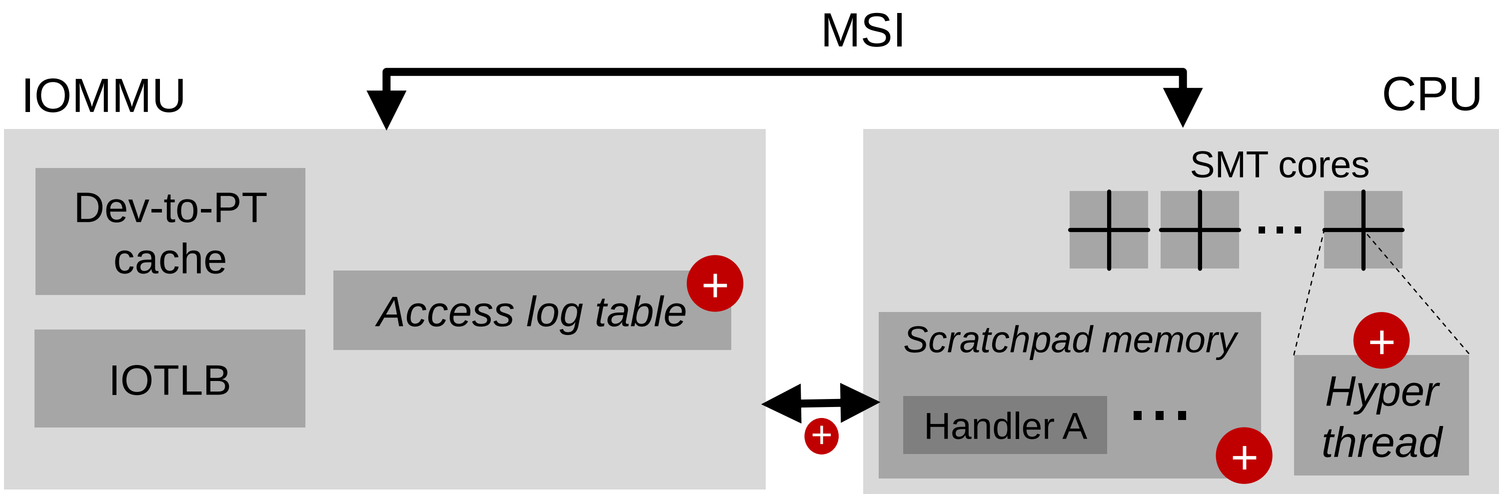



- A weak consistency model [1]
  - Consistency on-demand
- active_flush(int target_id)
  - Enforces the completion of active accesses issued by the calling process and targeted at target_id
  - Implemented with an active get issued at a special *flushing page*

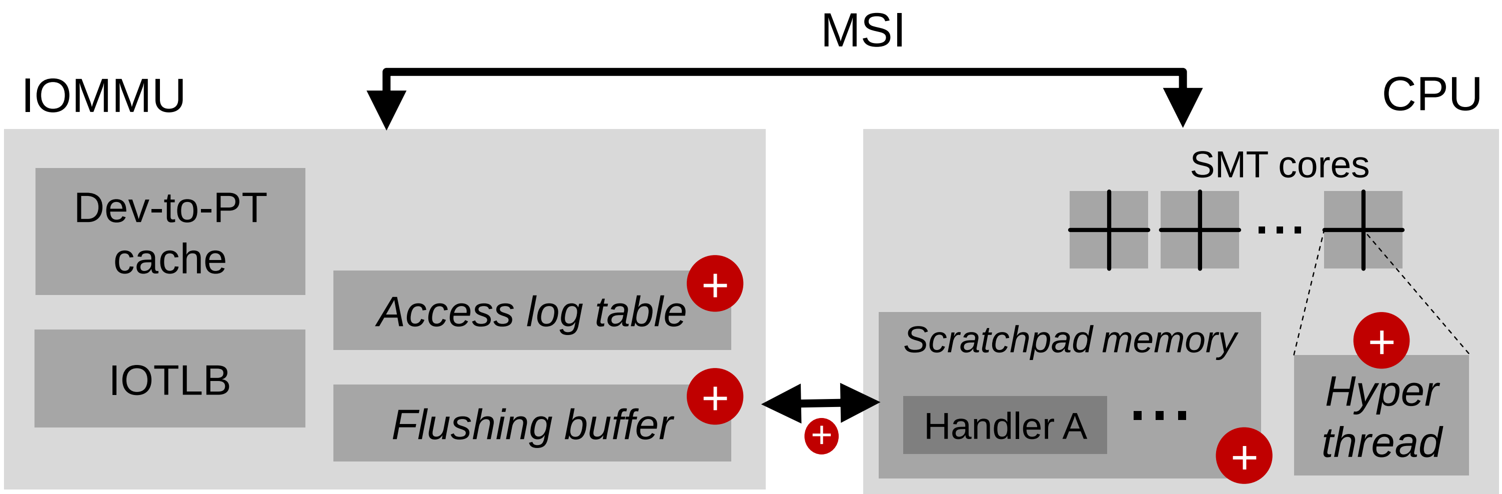



[1] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. ISCA '90

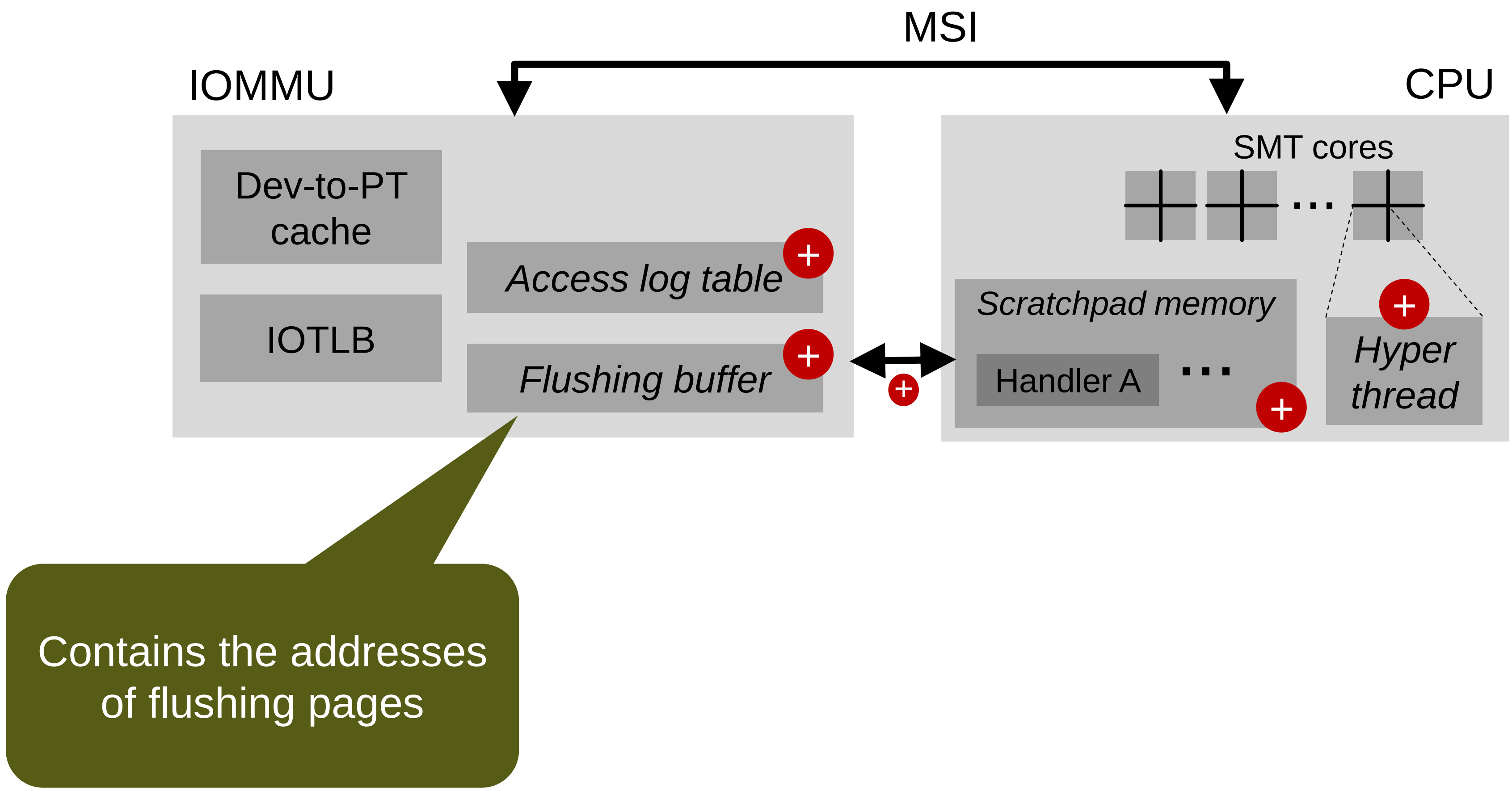
# Consistency



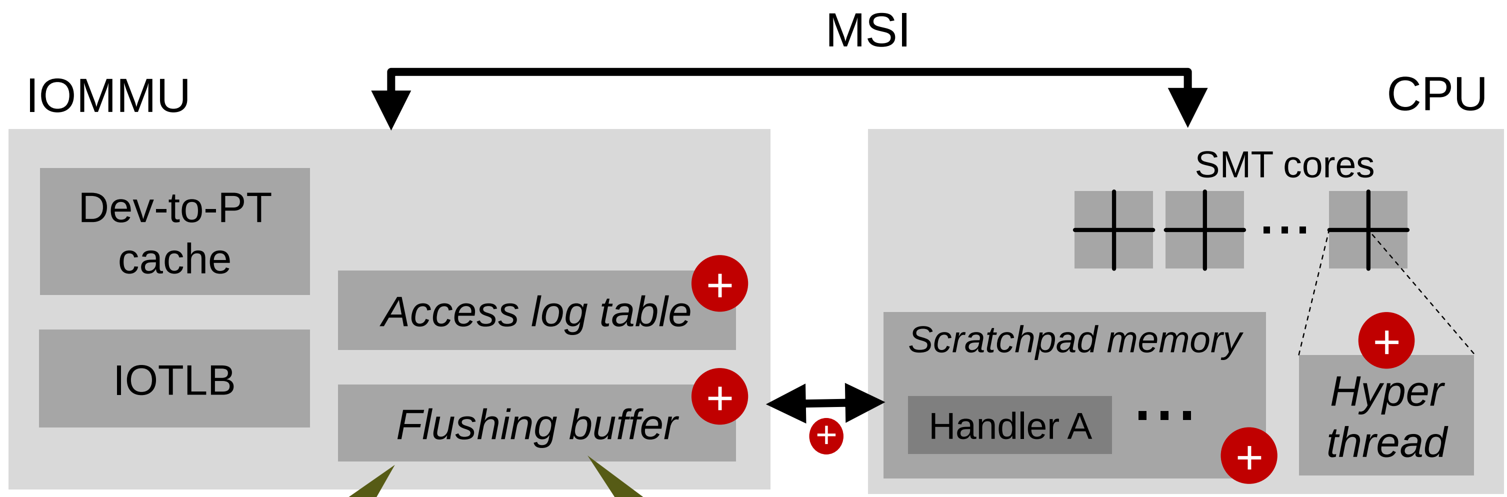
# CONSISTENCY



# Consistency



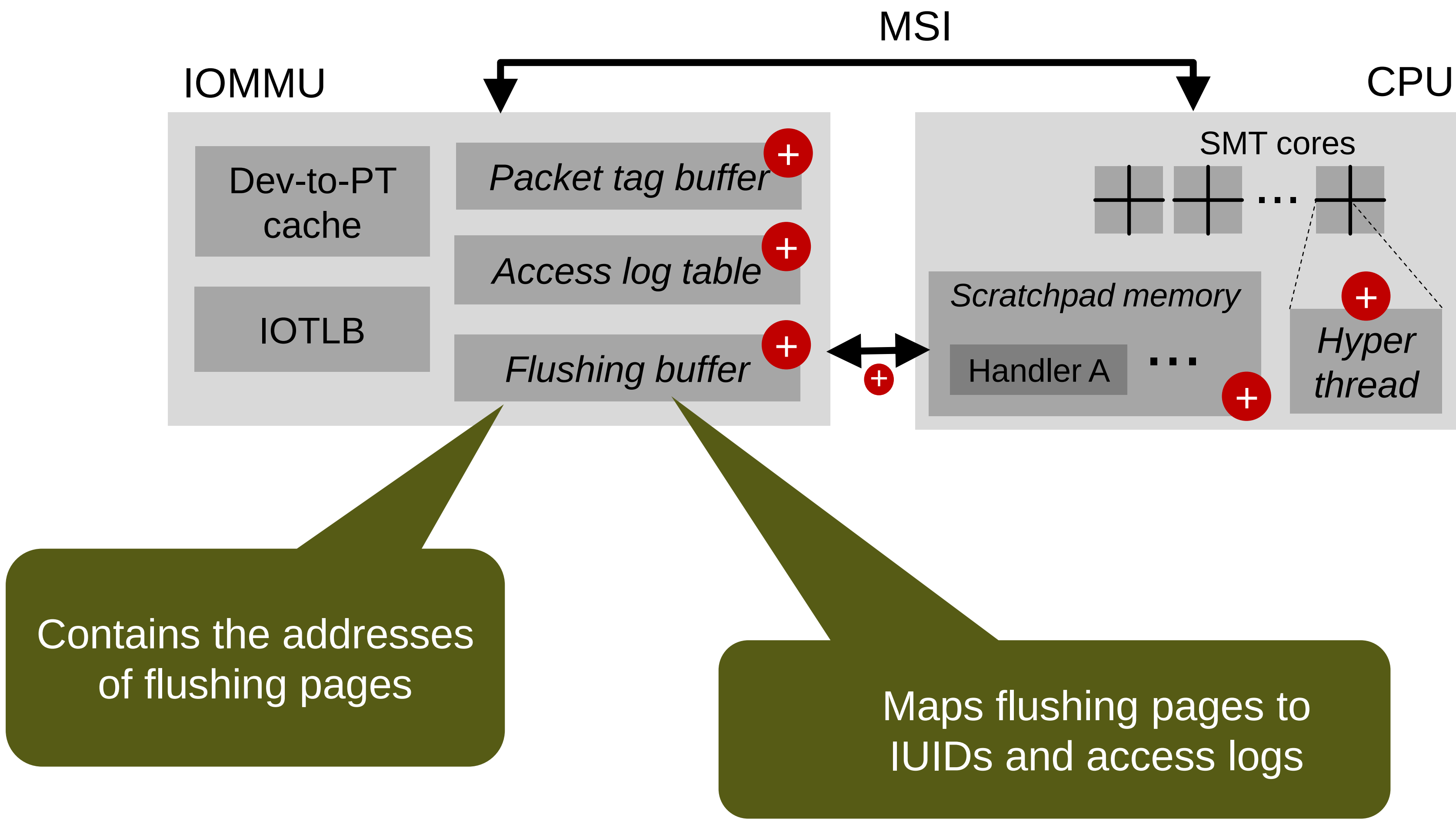
# CONSISTENCY




Contains the addresses of flushing pages

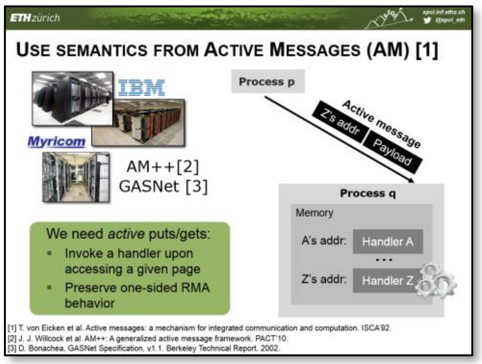
Maps flushing pages to IUIDs and access logs

# Consistency



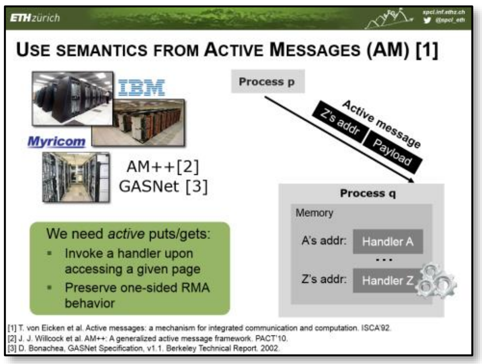
# Let’s summarize…

# Let’s summarize…



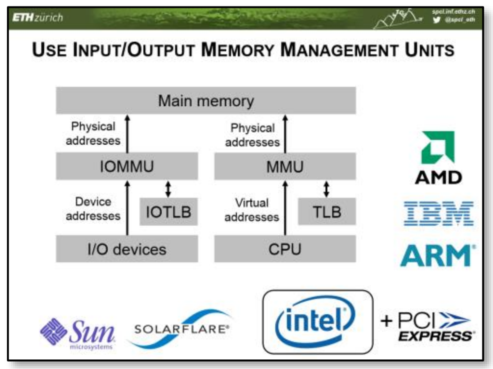

Active Messages

# Let's summarize…

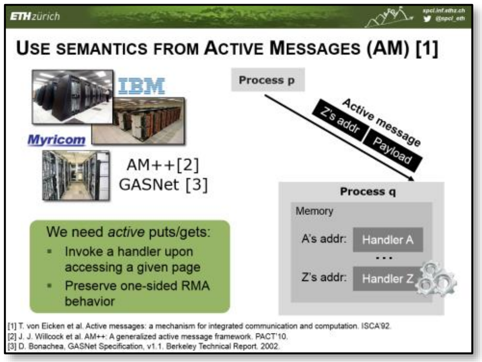



Active Messages

IOMMUs

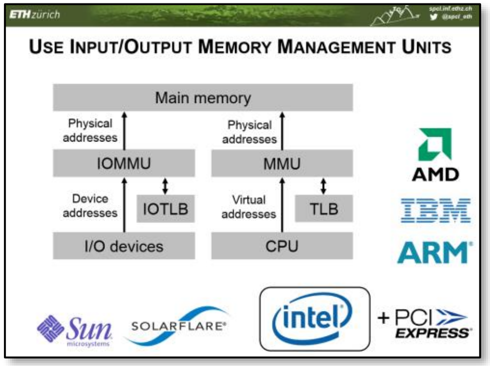


# Let's summarize…

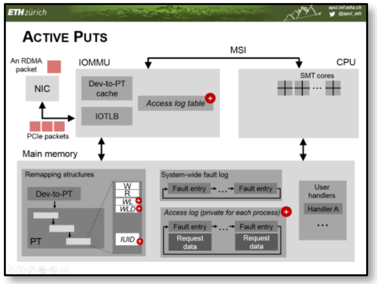


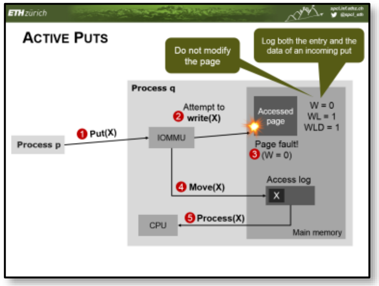

Active Messages

IOMMUs

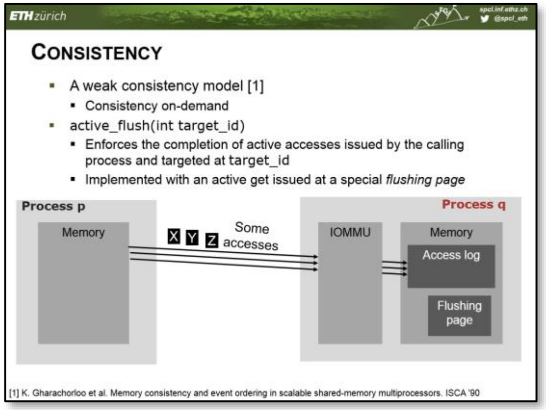



Active Puts/Gets



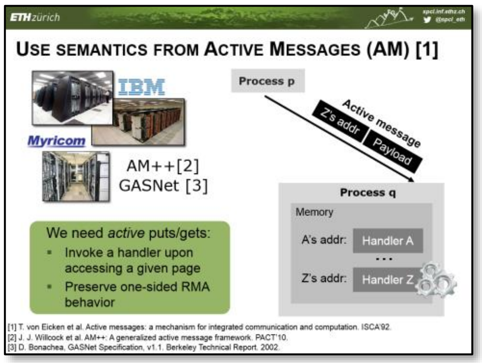



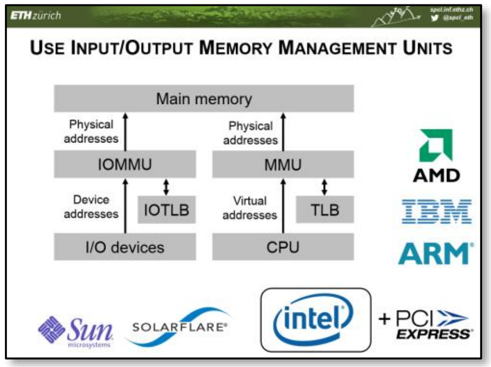
# Let's summarize…




Consistency

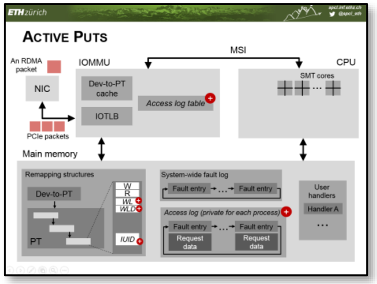
IOMMUs

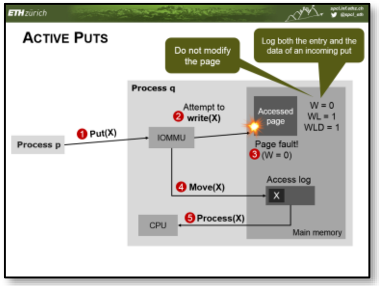


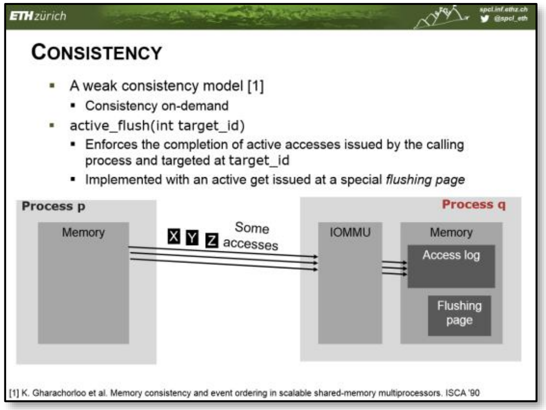

Active Messages

Active Puts/Gets

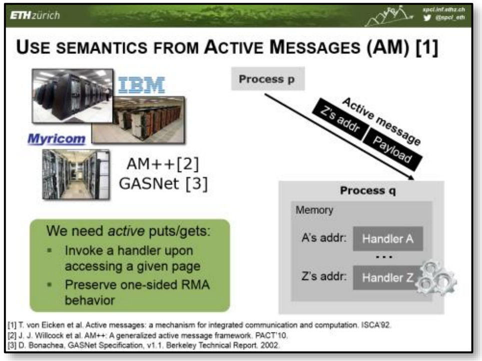


# Let's summarize…

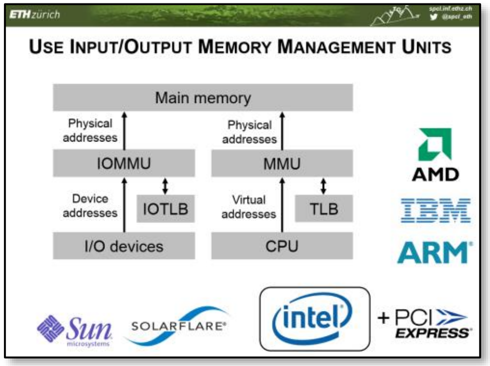



Consistency

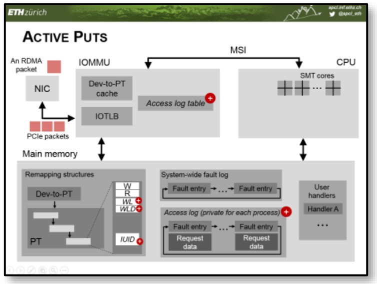


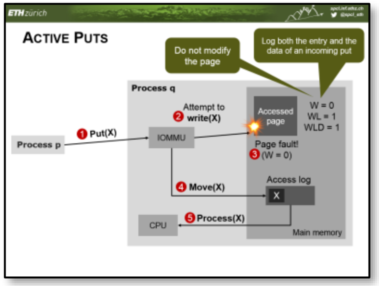

Active Messages

IOMMUs




Active Puts/Gets




# How can we use it?

# ACTIVE ACCESS USE-CASES

## DISTRIBUTED HASHTABLE

# ACTIVE ACCESS USE-CASES

## DISTRIBUTED HASHTABLE

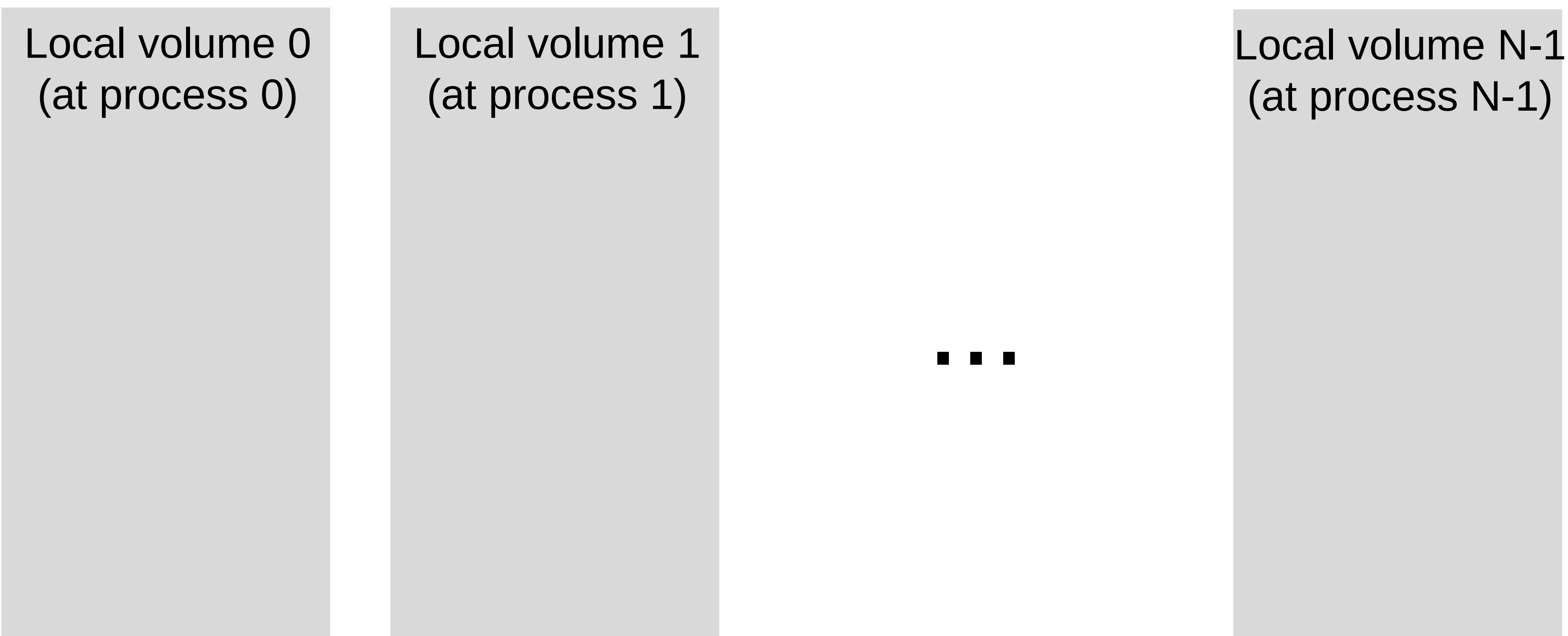
- Used to construct key-value stores (e.g., Memcached [1])

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

# Active Access Use-Cases

## Distributed Hashtable

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

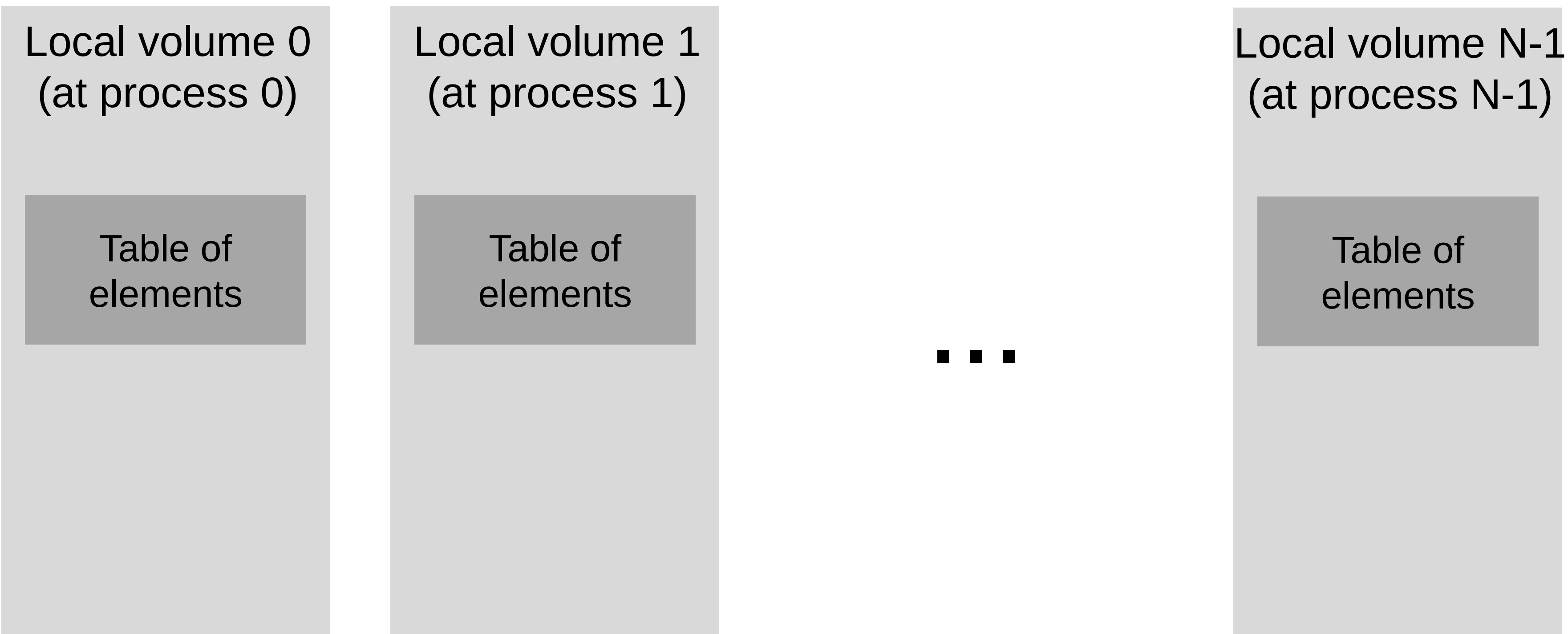



[1] B. Fitzpatrick. Distributed caching with memcached. Linux journal, 2004.

# Active Access Use-Cases

## Distributed Hashtable

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

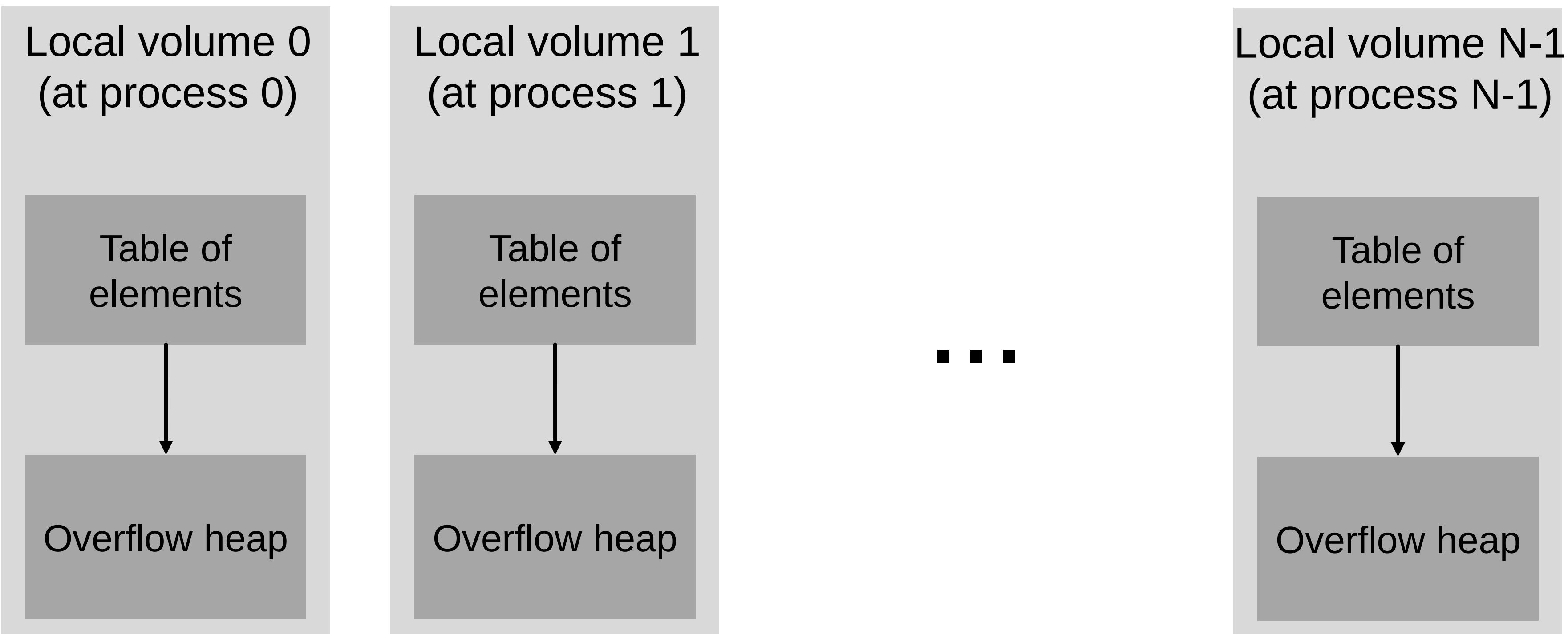



[1] B. Fitzpatrick. Distributed caching with memcached. Linux journal, 2004.

# Active Access Use-Cases

## Distributed Hashtable

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



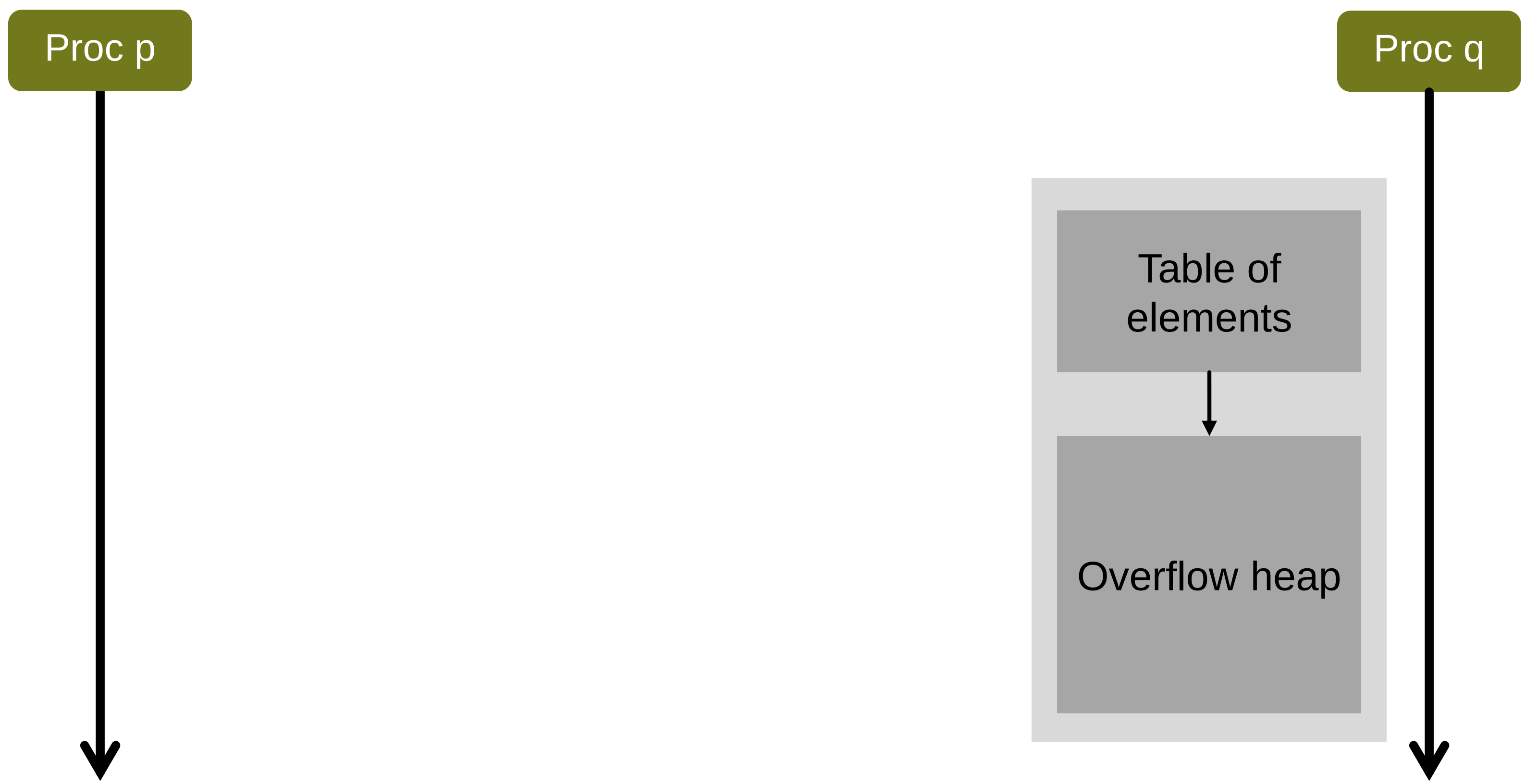

[1] B. Fitzpatrick. Distributed caching with memcached. Linux journal, 2004.

# Active Access Use-Cases

## Distributed Hashtable: Inserts (RMA)

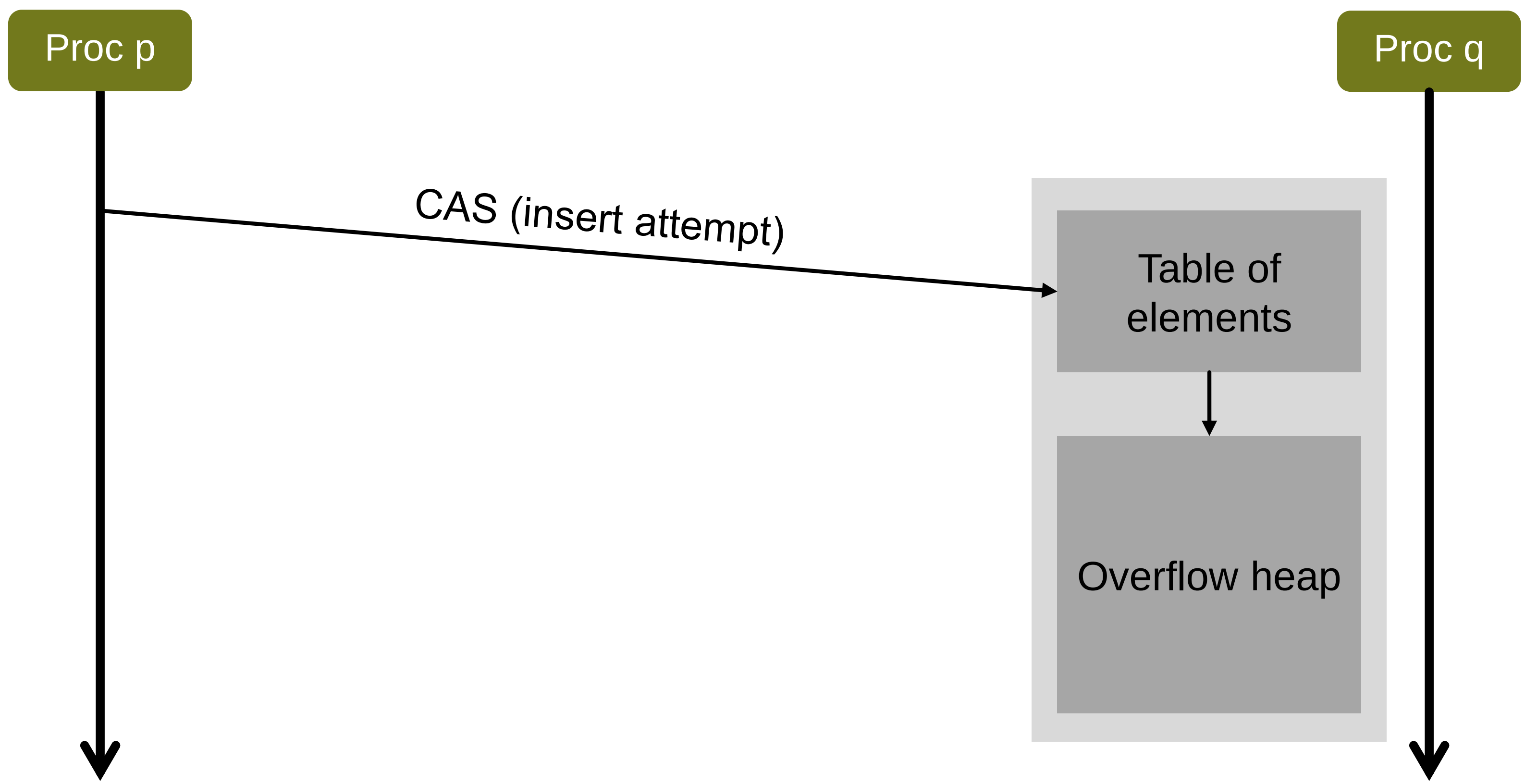
# Active Access Use-Cases

## Distributed Hashtable: Inserts (RMA)



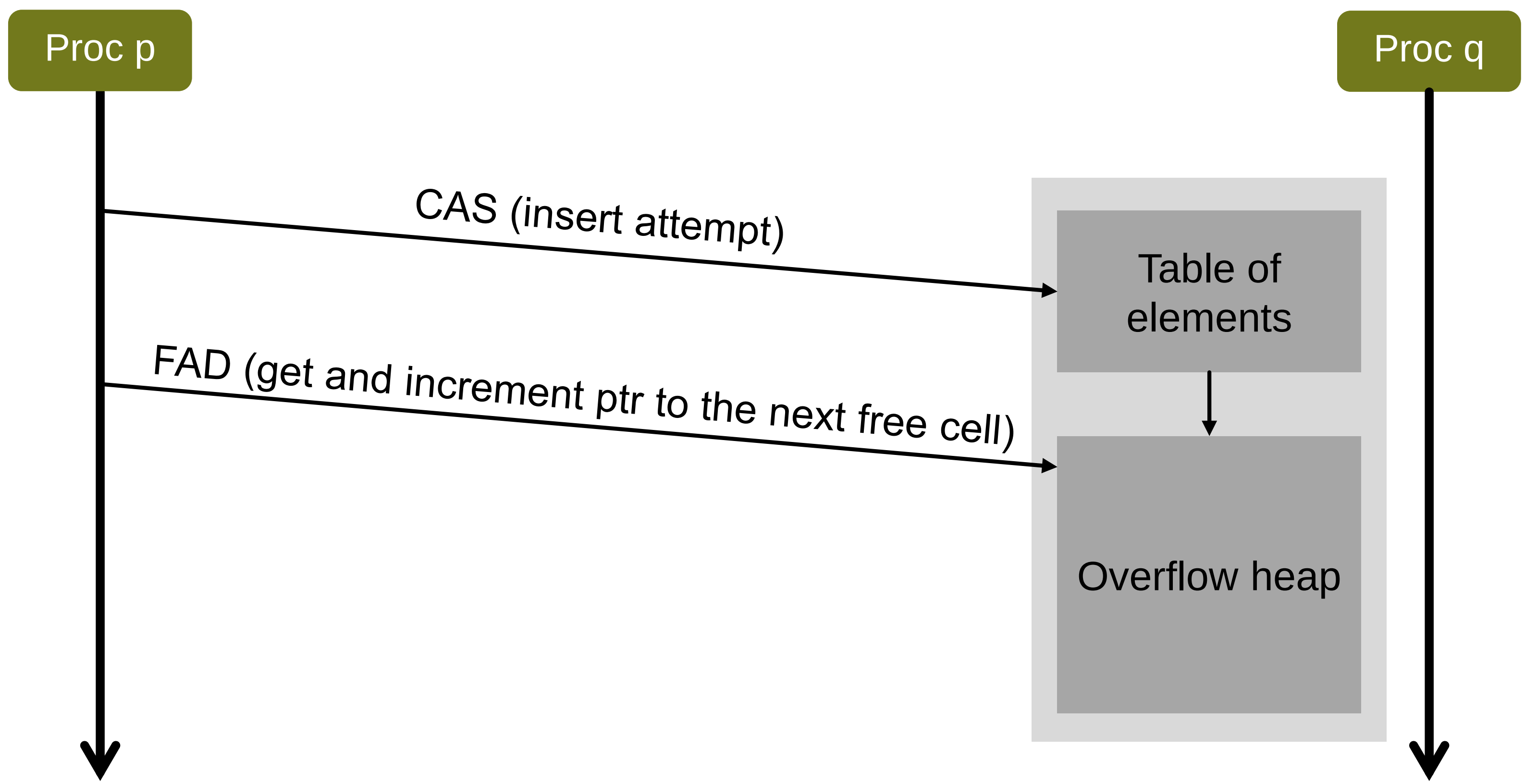
# ACTIVE ACCESS USE-CASES

## DISTRIBUTED HASHTABLE: INSERTS (RMA)



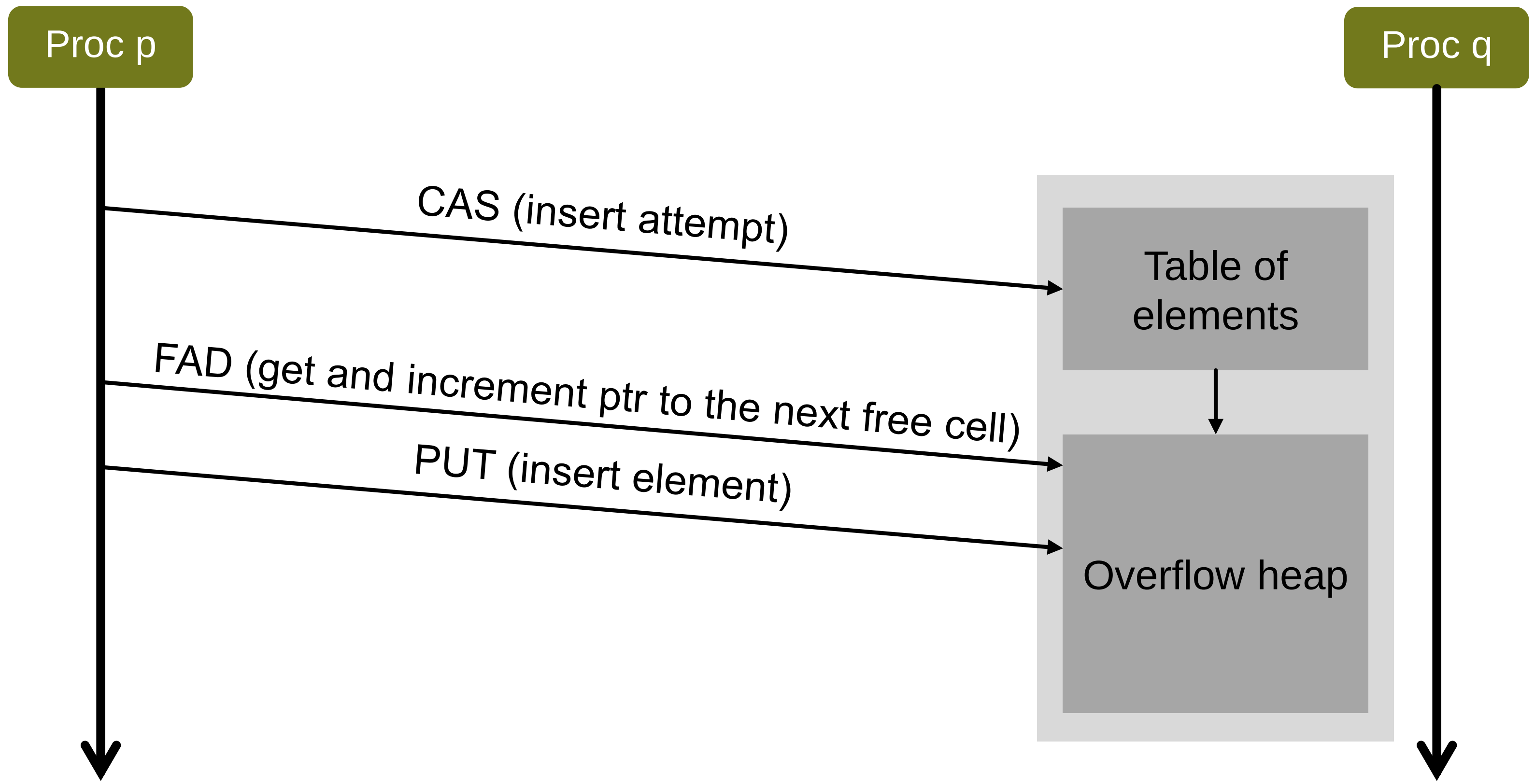
# Active Access Use-Cases

## Distributed Hashtable: Inserts (RMA)



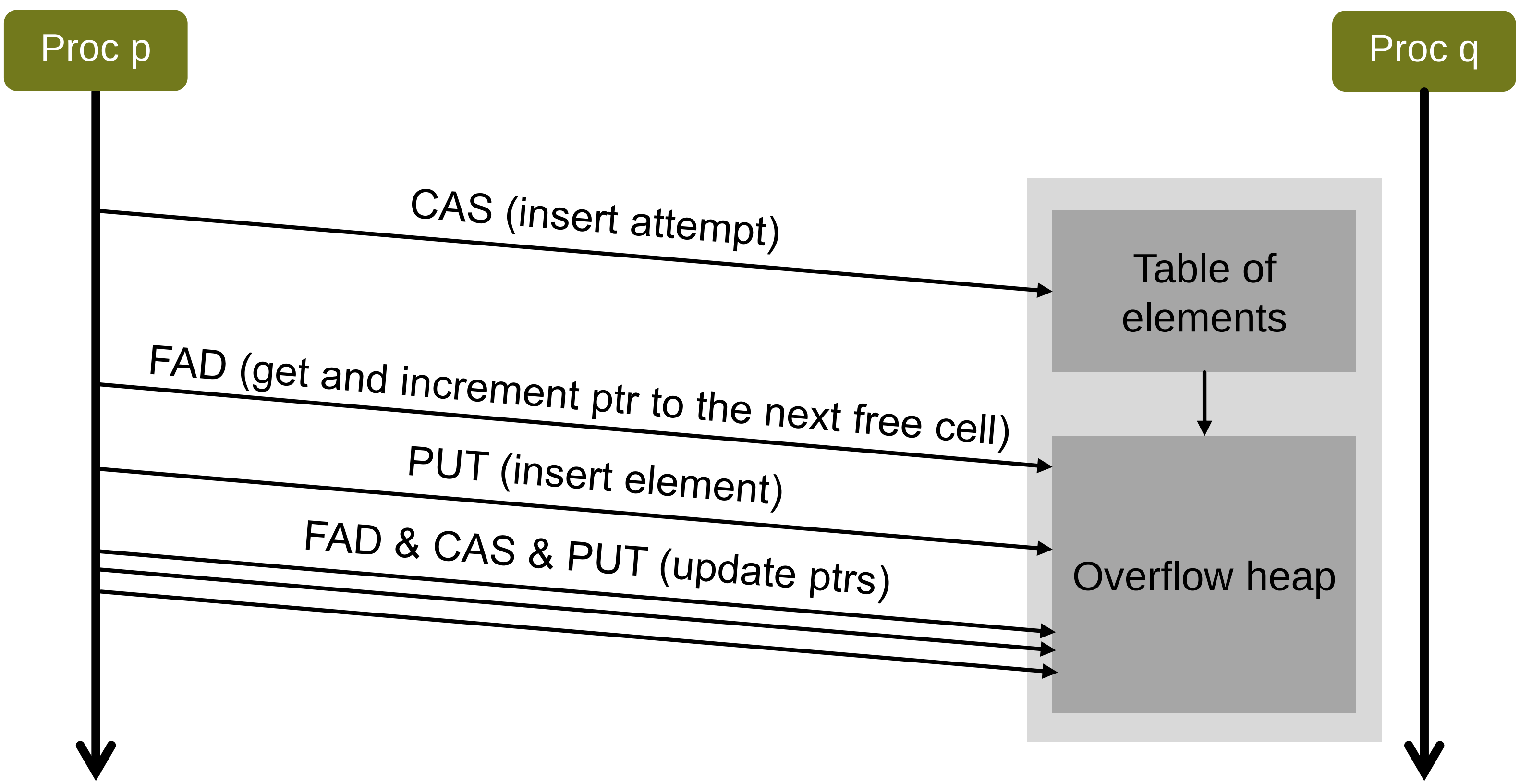
# Active Access Use-Cases

## Distributed Hashtable: Inserts (RMA)



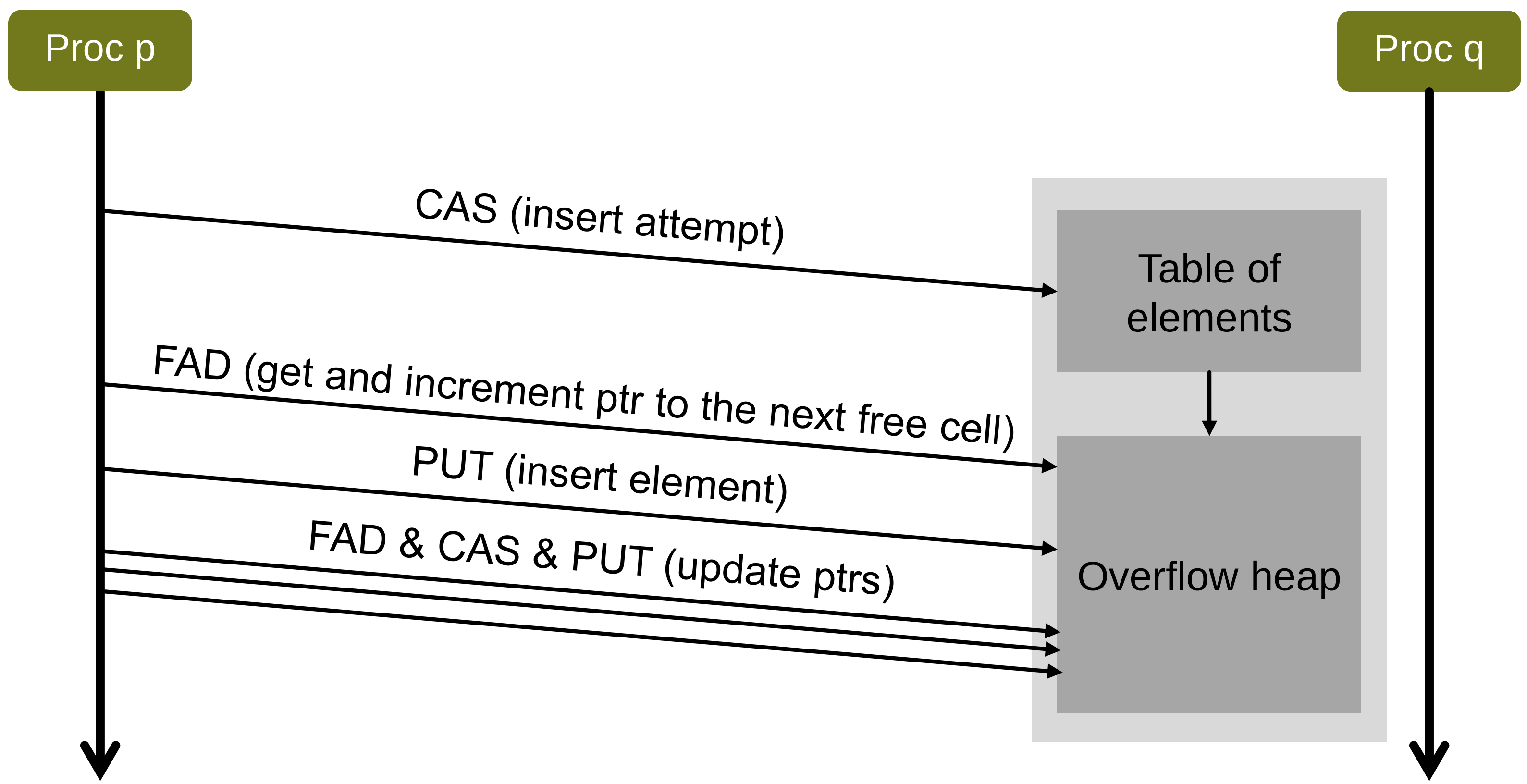
# ACTIVE ACCESS USE-CASES

## DISTRIBUTED HASHTABLE: INSERTS (RMA)



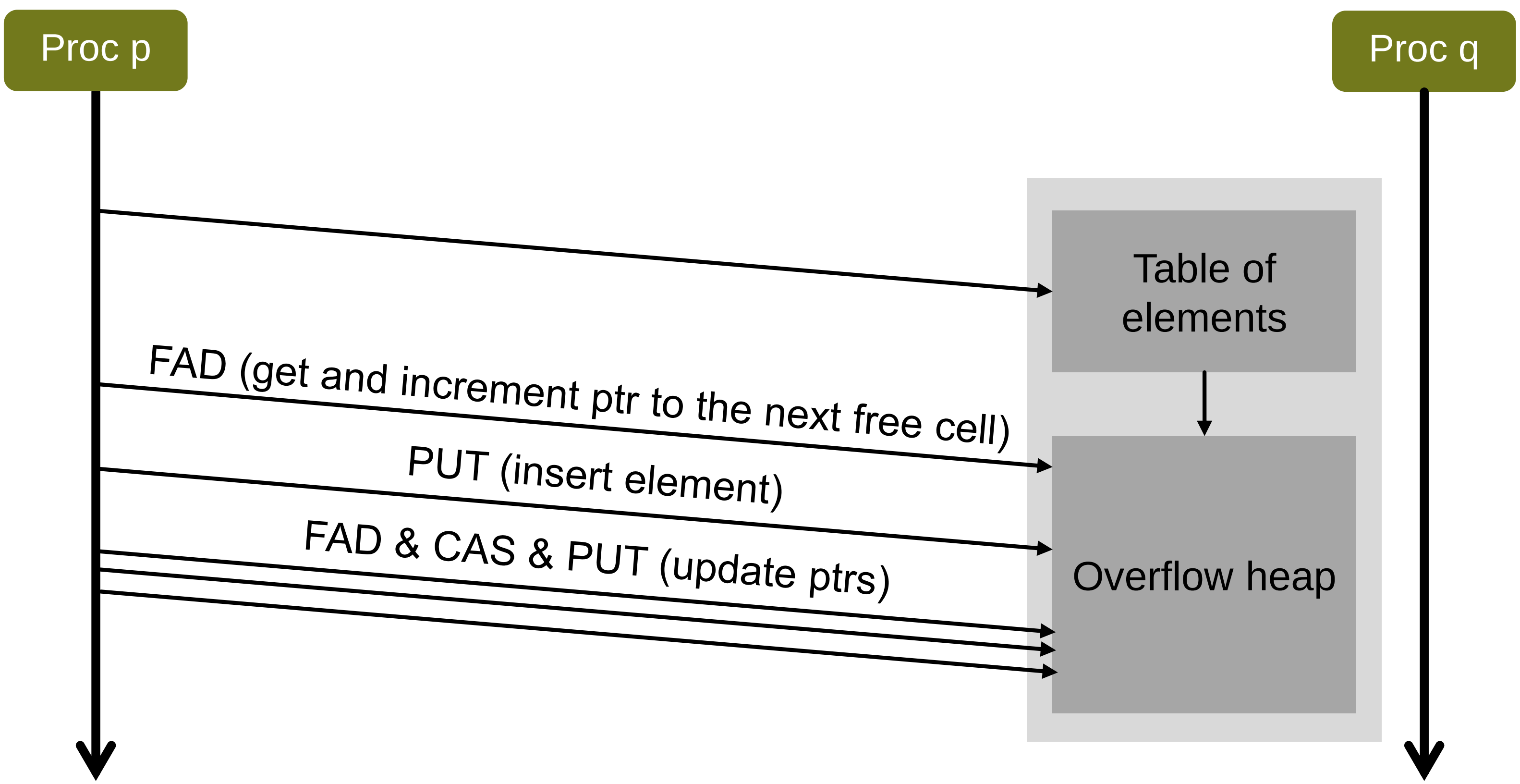
# Active Access Use-Cases

## Distributed Hashtable: Inserts (AA)



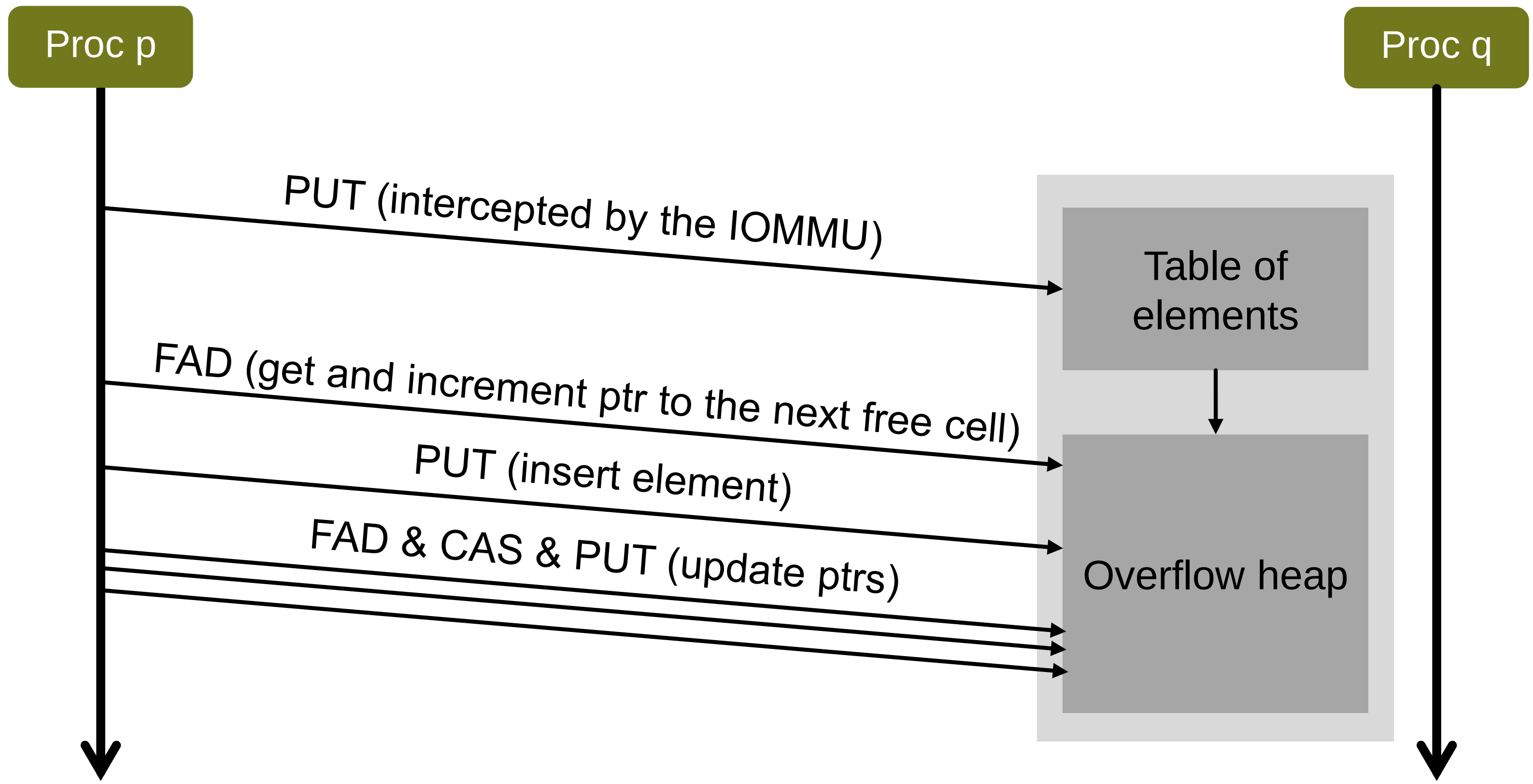
# Active Access Use-Cases

## Distributed Hashtable: Inserts (AA)



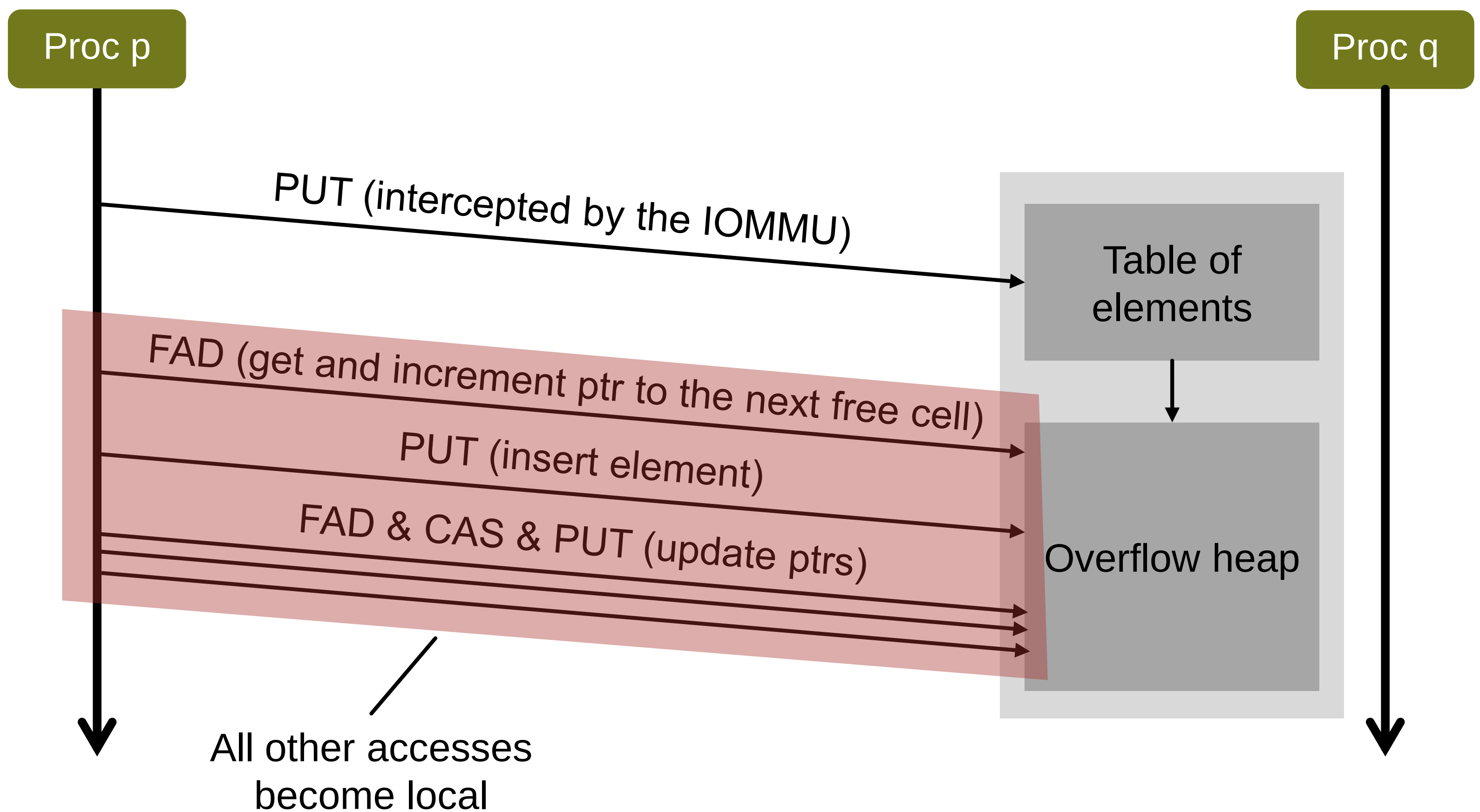
# Active Access Use-Cases

## Distributed Hashtable: Inserts (AA)



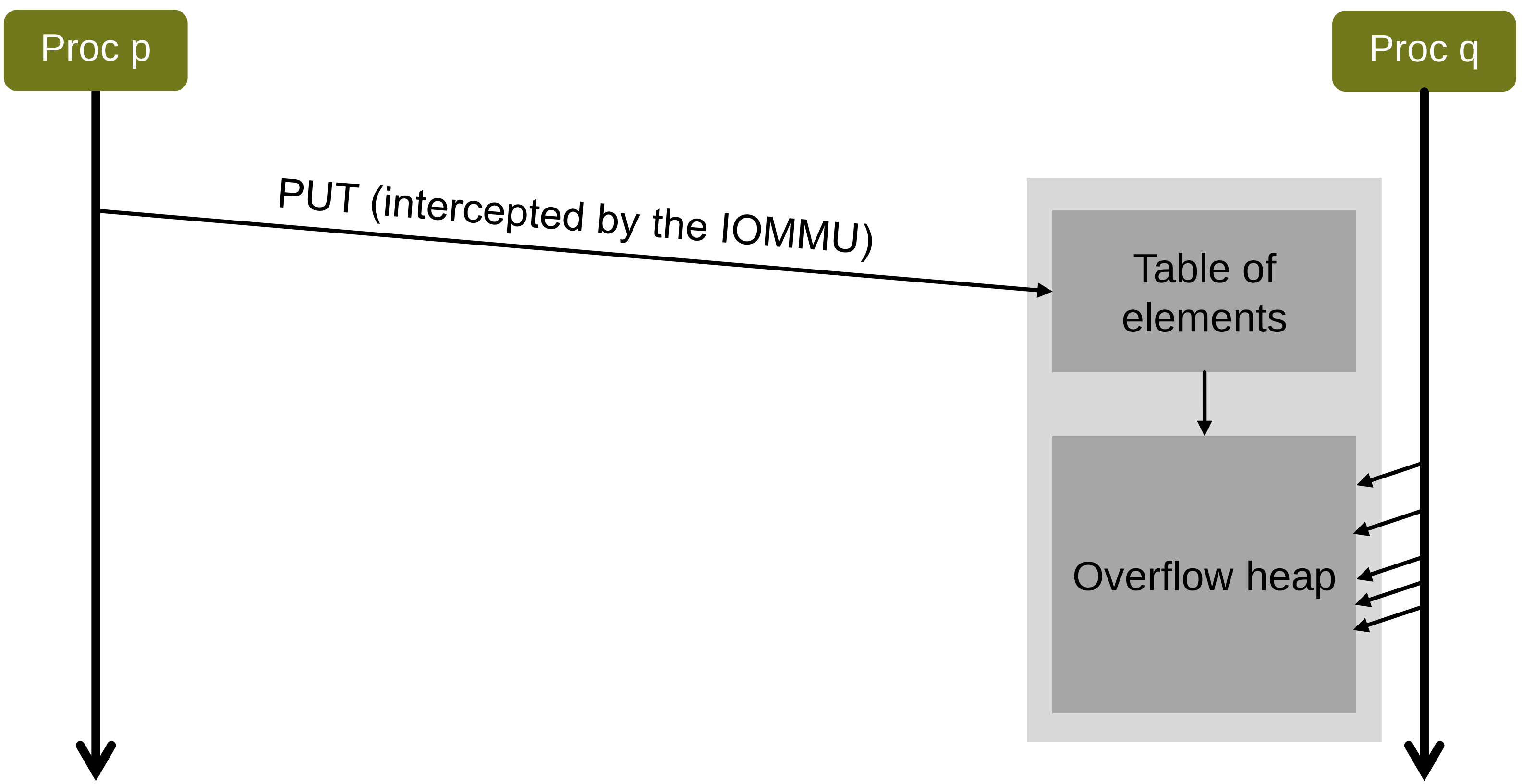
# Active Access Use-Cases

## Distributed Hashtable: Inserts (AA)



# Active Access Use-Cases

## Distributed Hashtable: Inserts (AA)

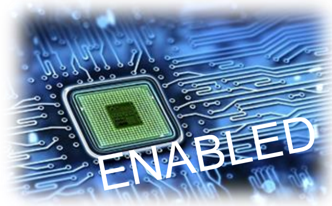


# Active Access Use-Cases

## Virtual Global Address Space (V-GAS)

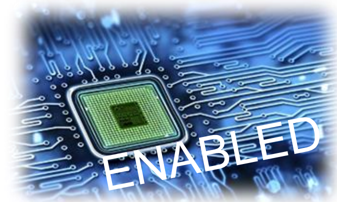
# ACTIVE ACCESS USE-CASES

## VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)



# Active Access Use-Cases

## Virtual Global Address Space (V-GAS)



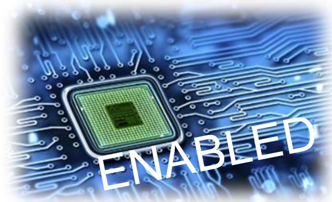

Machine 0

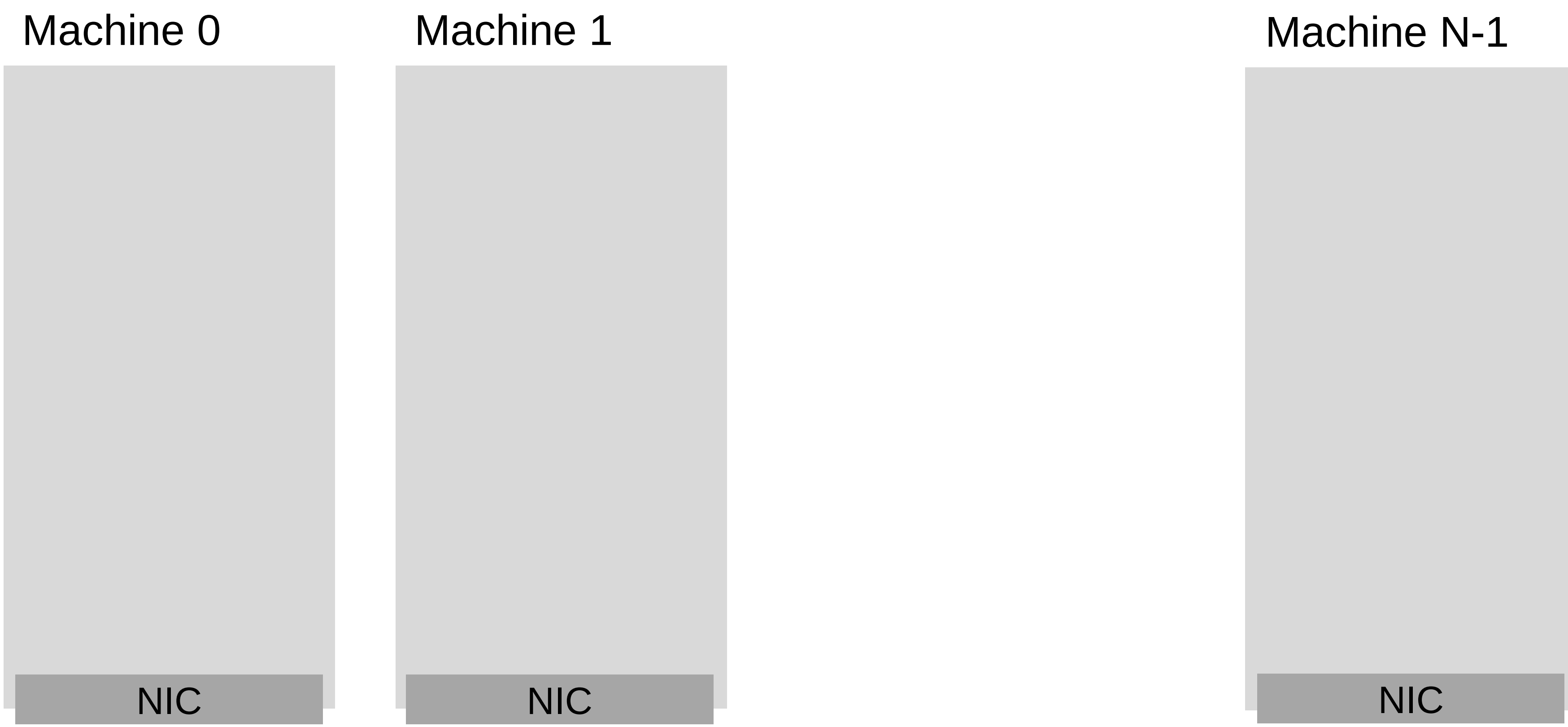
Machine 1

Machine N-1

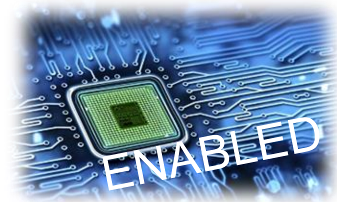
# Active Access Use-Cases

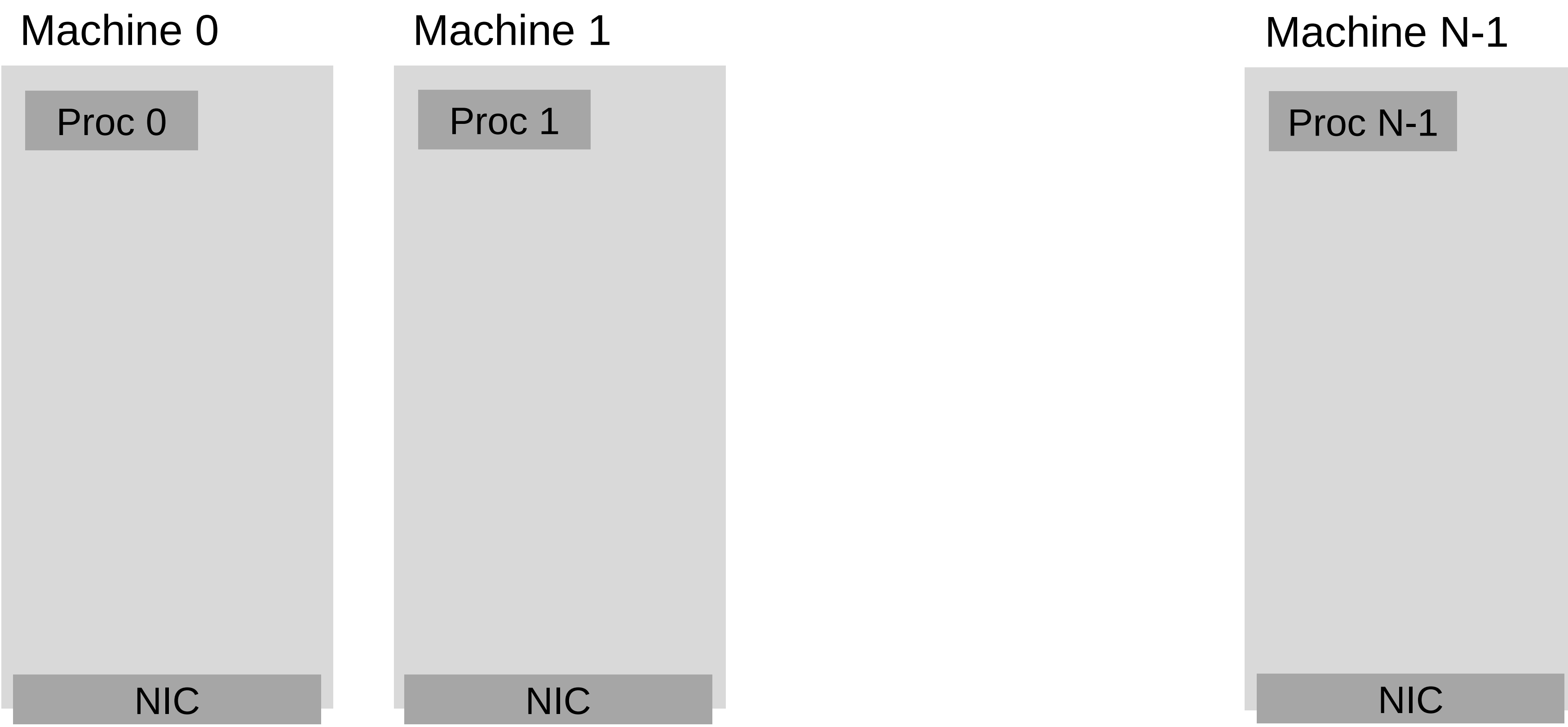
## Virtual Global Address Space (V-GAS)



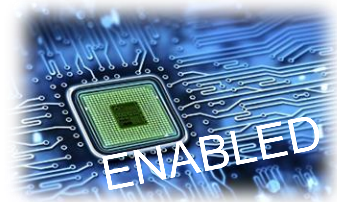
# Active Access Use-Cases
## Virtual Global Address Space (V-GAS)

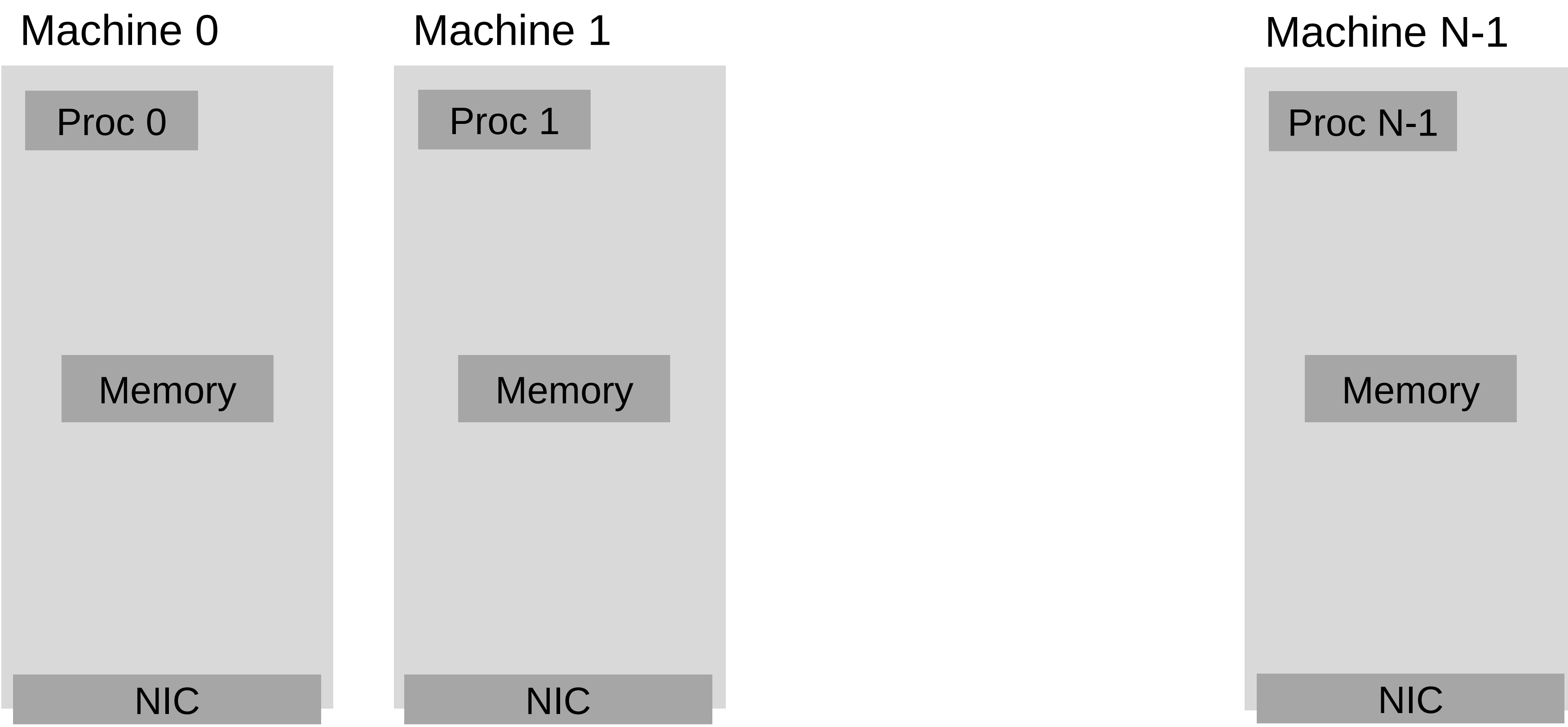


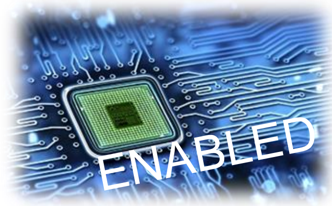
# Active Access Use-Cases

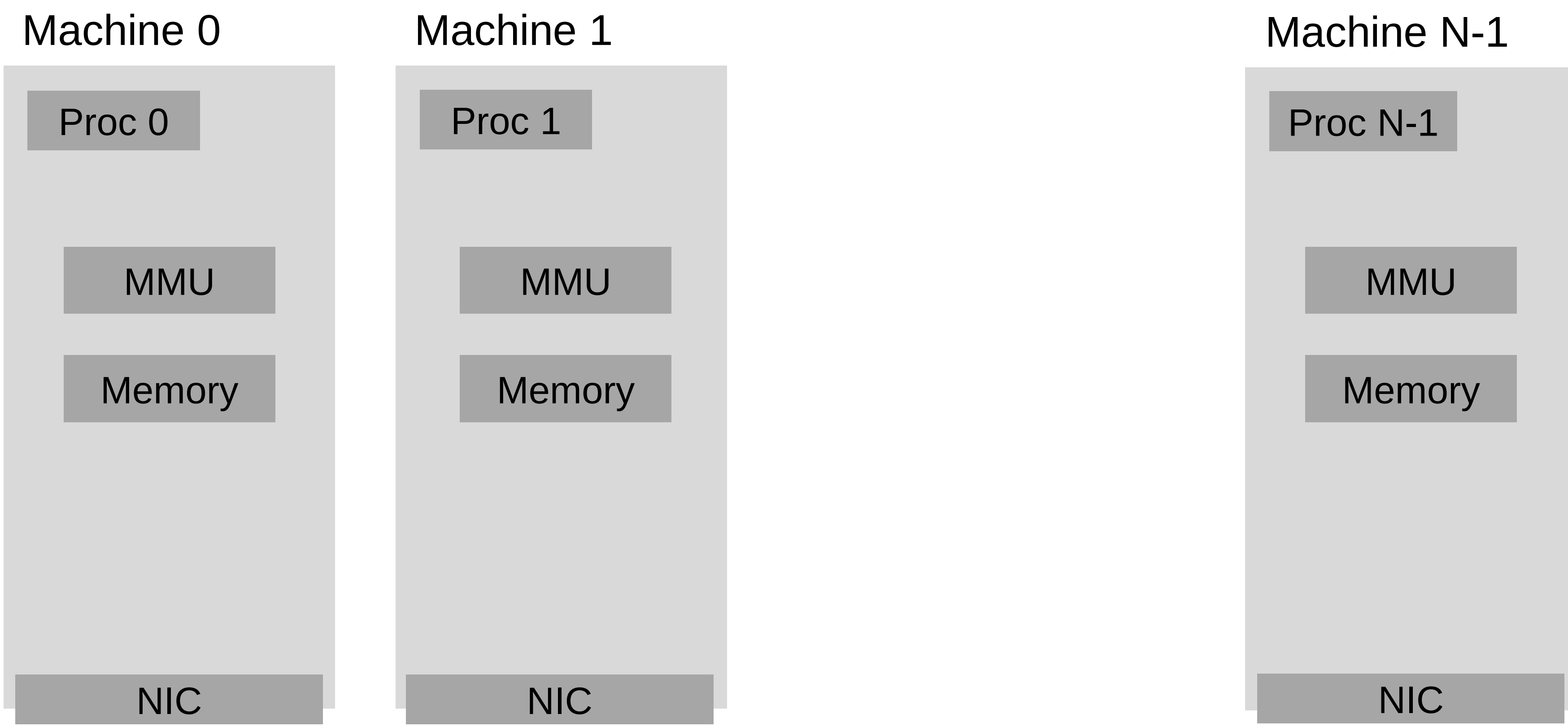
## Virtual Global Address Space (V-GAS)



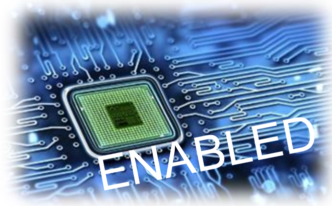
# Active Access Use-Cases
## Virtual Global Address Space (V-GAS)

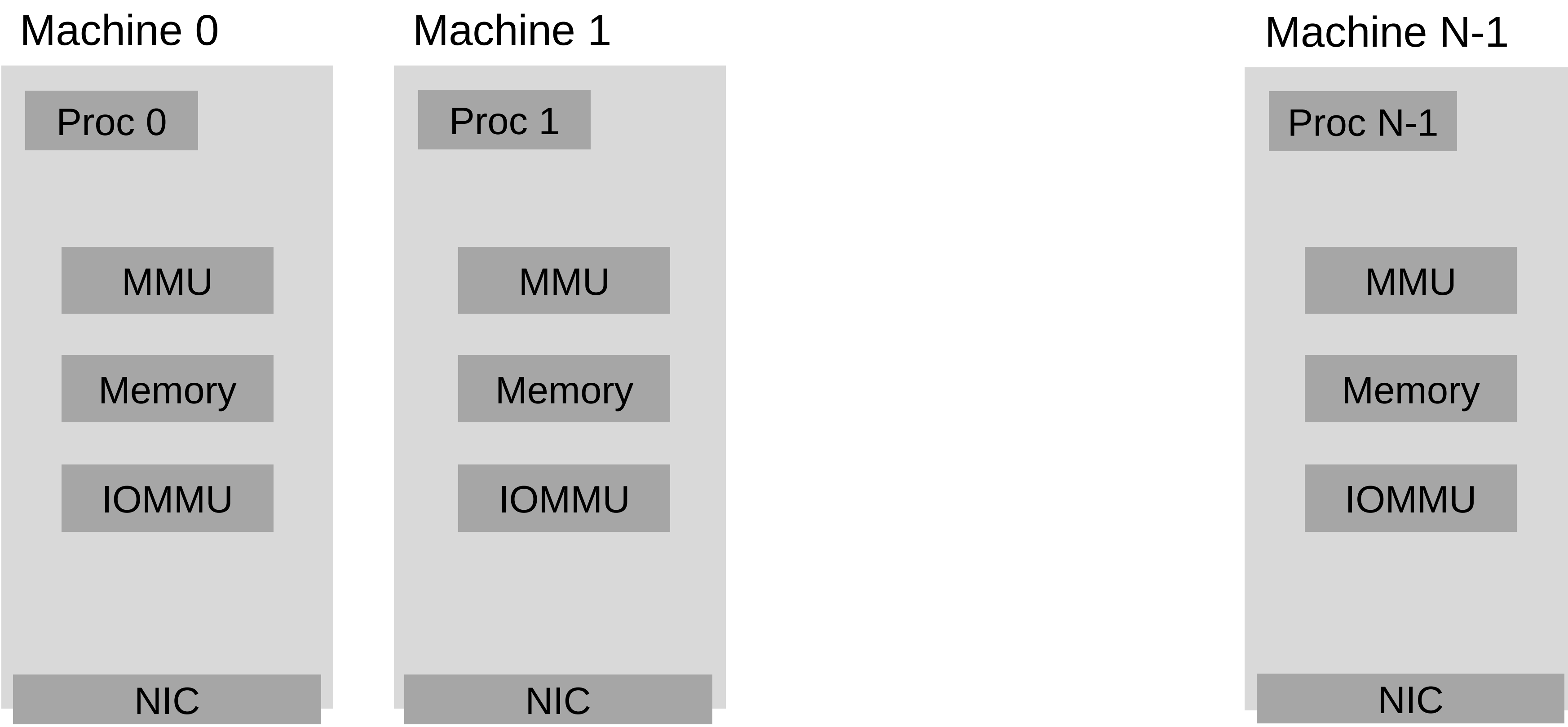


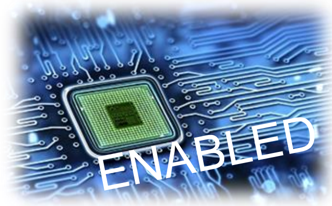
# Active Access Use-Cases

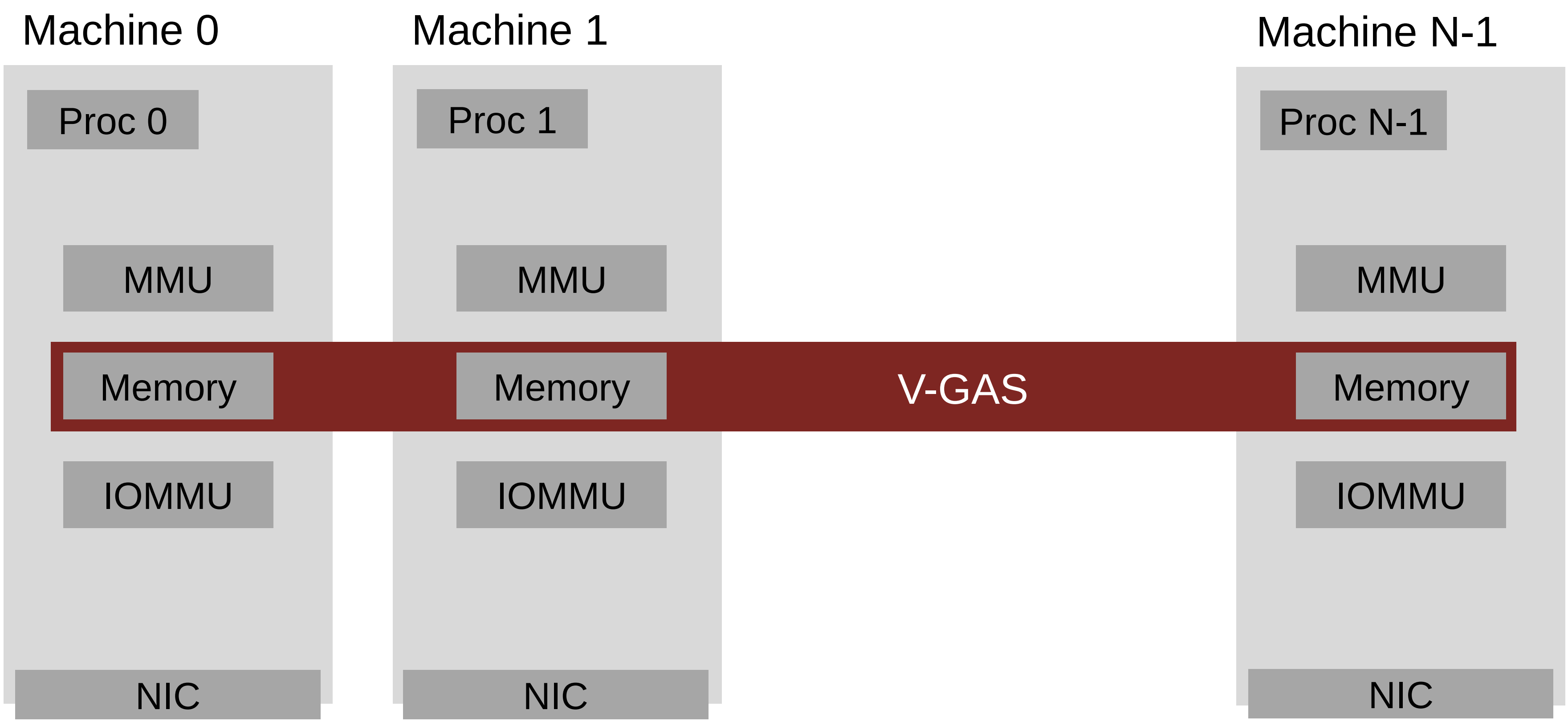
## Virtual Global Address Space (V-GAS)



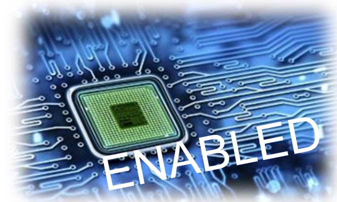
# Active Access Use-Cases
## Virtual Global Address Space (V-GAS)

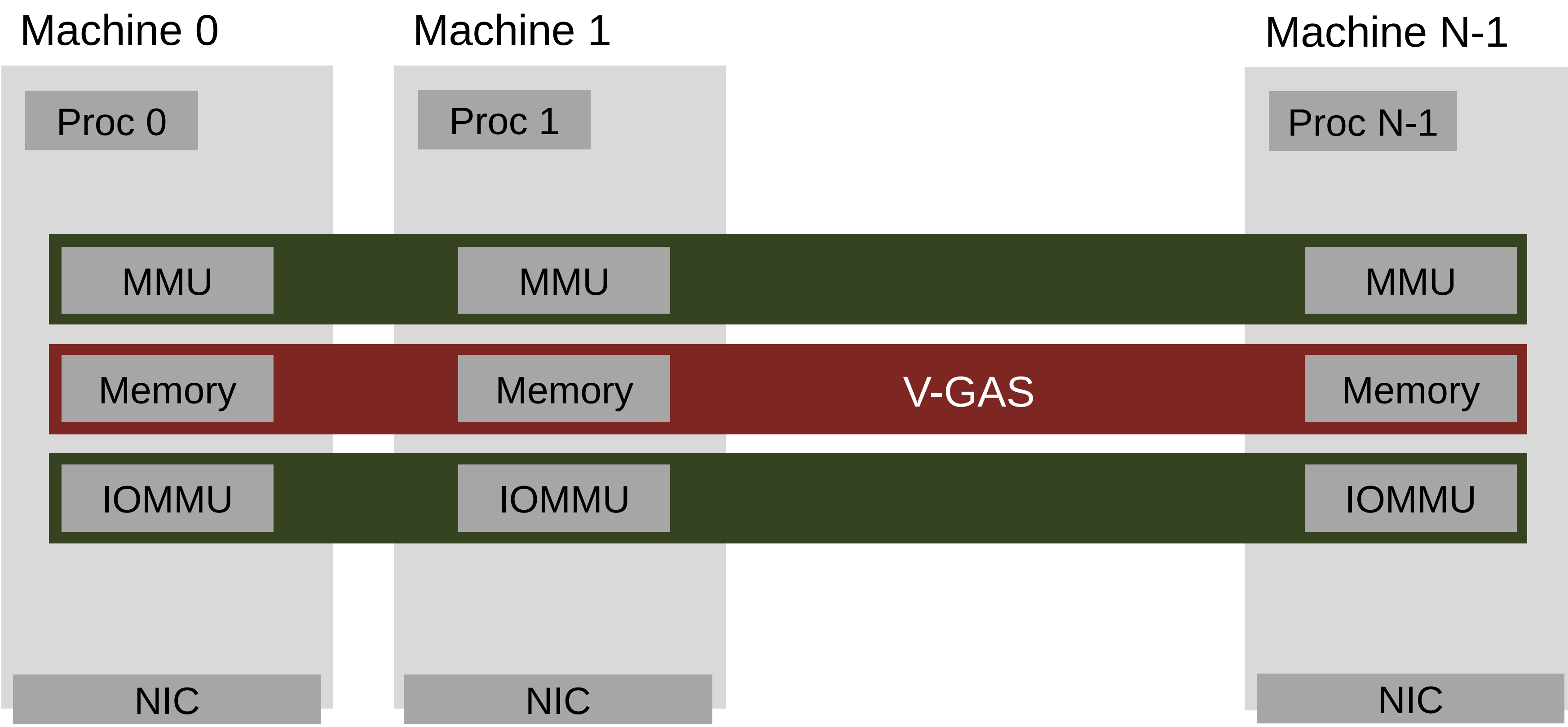


# Active Access Use-Cases

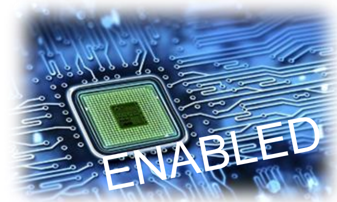
## Virtual Global Address Space (V-GAS)

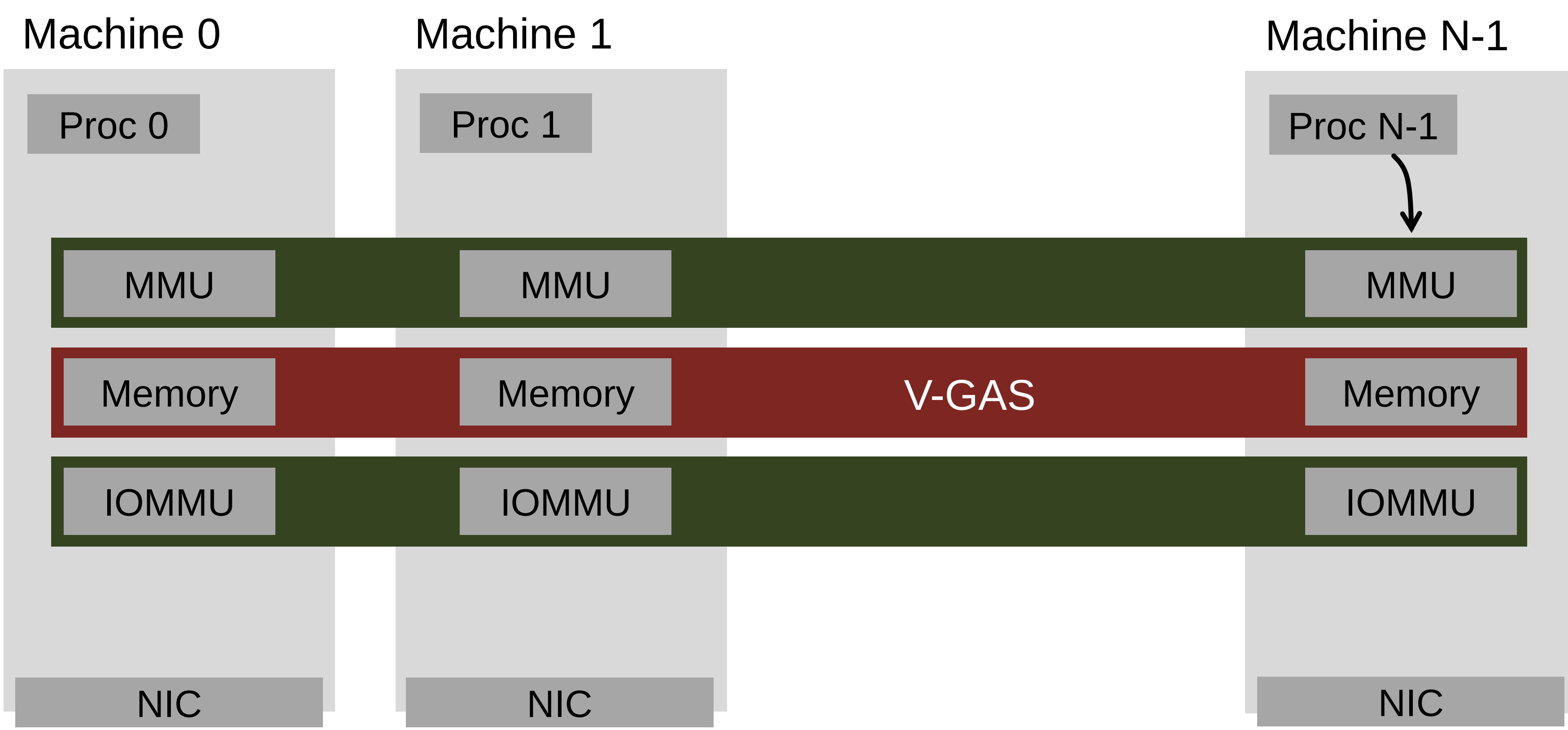


# Active Access Use-Cases

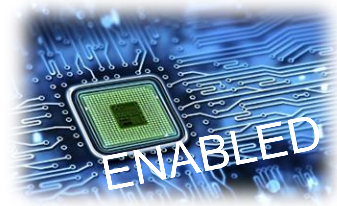
## Virtual Global Address Space (V-GAS)

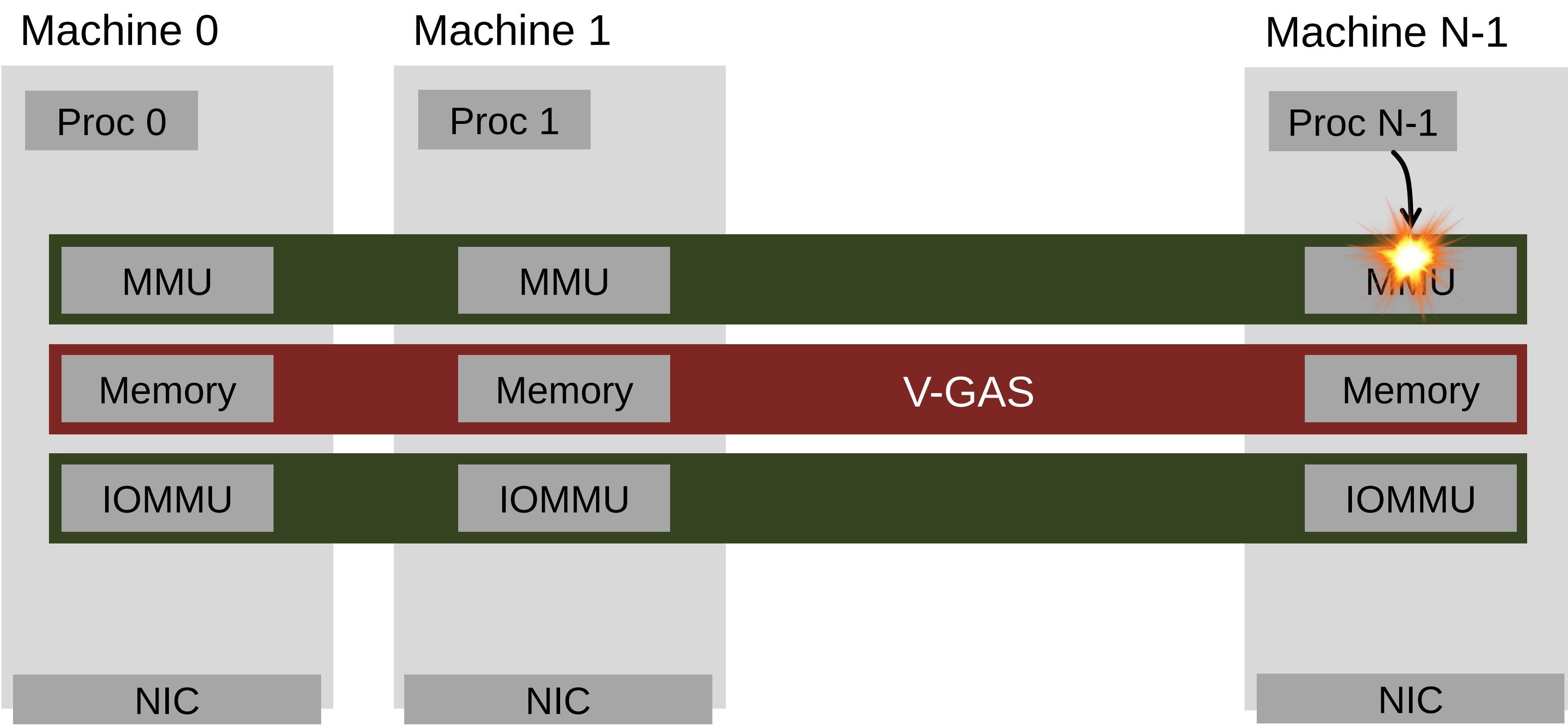


# Active Access Use-Cases

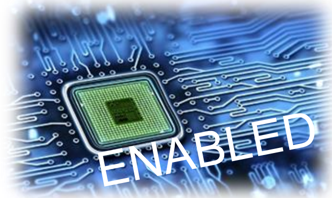
## Virtual Global Address Space (V-GAS)

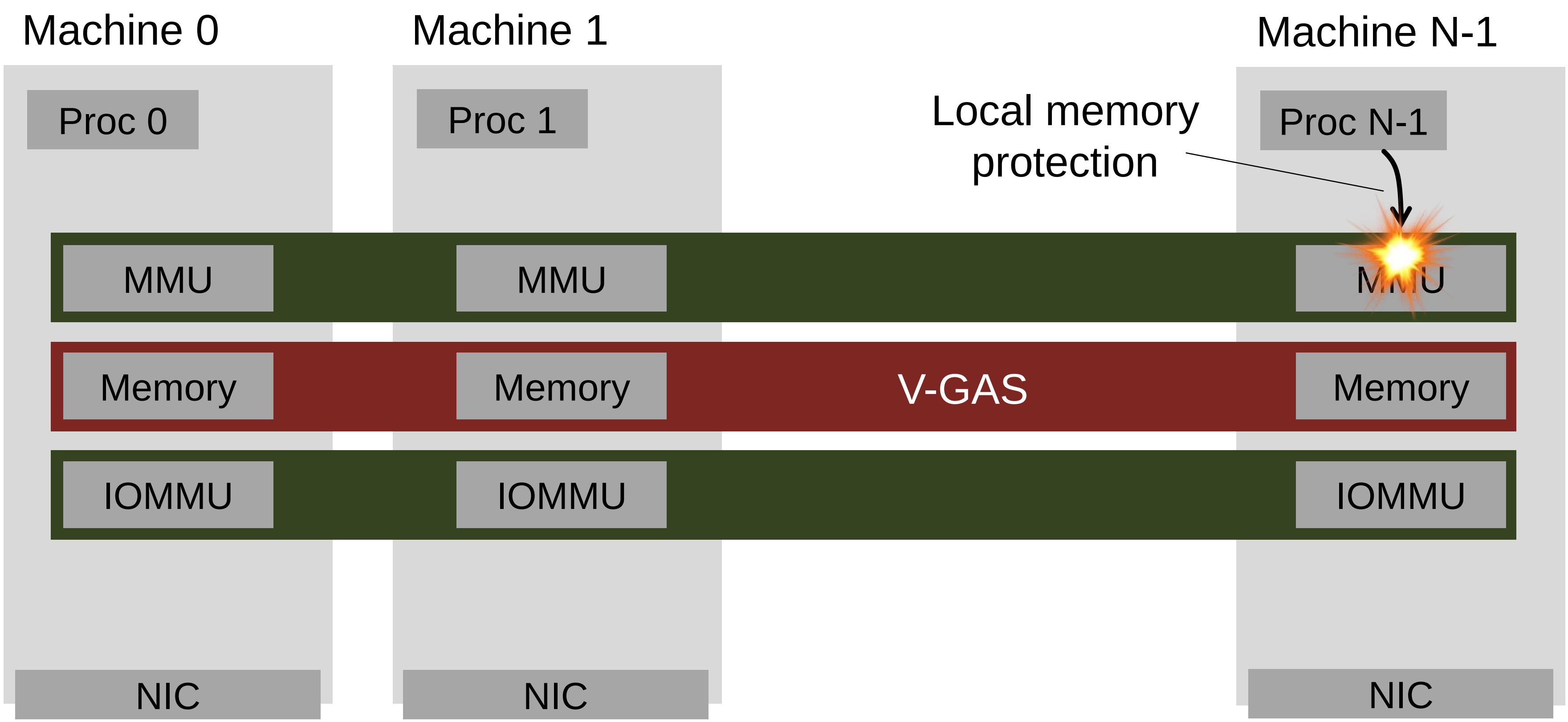


# Active Access Use-Cases

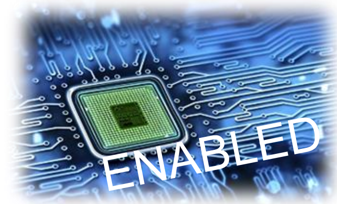
## Virtual Global Address Space (V-GAS)

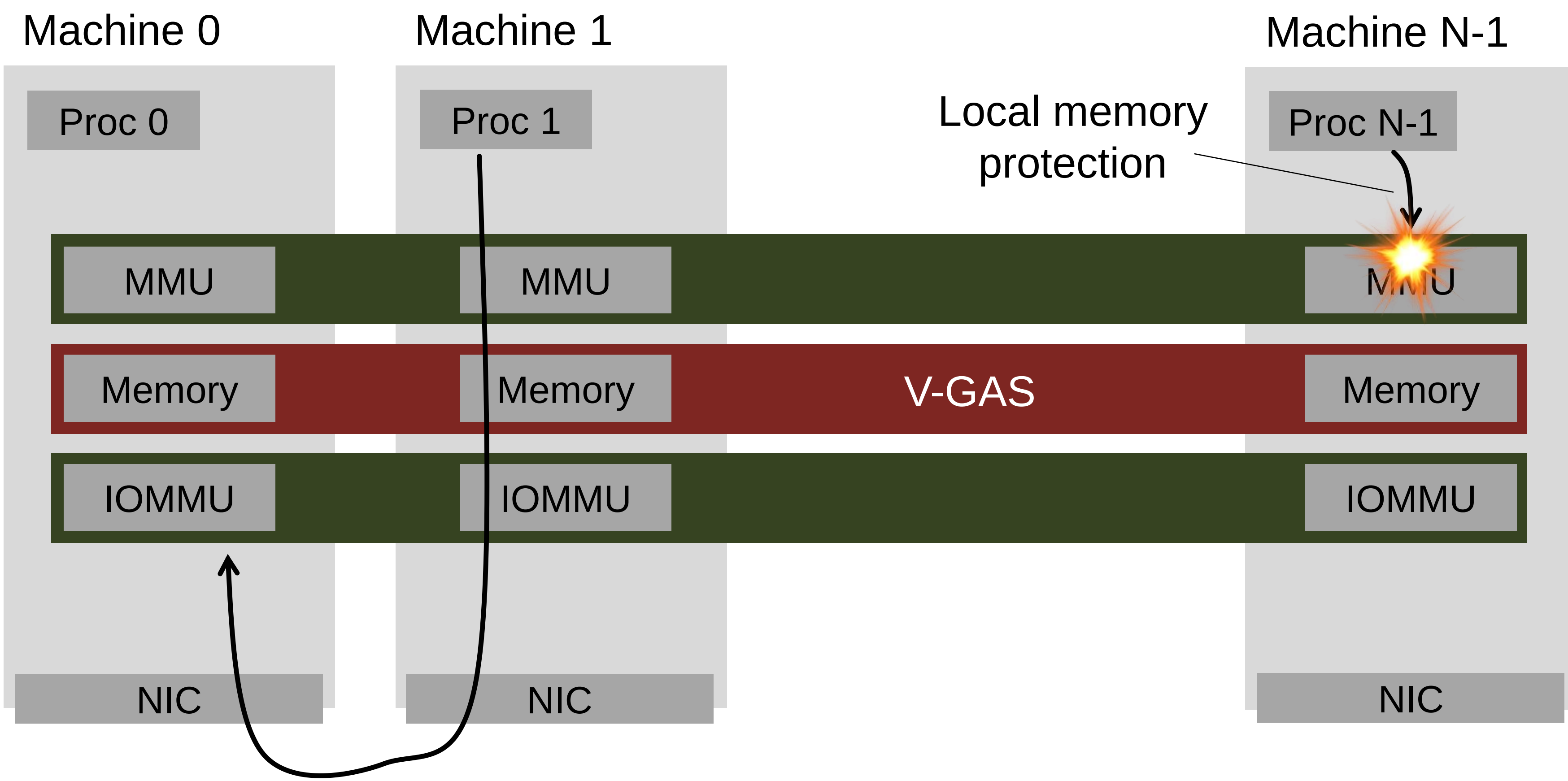


# Active Access Use-Cases

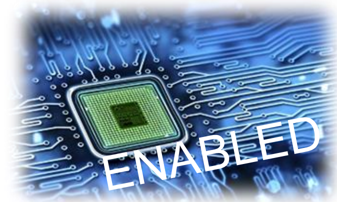
## Virtual Global Address Space (V-GAS)

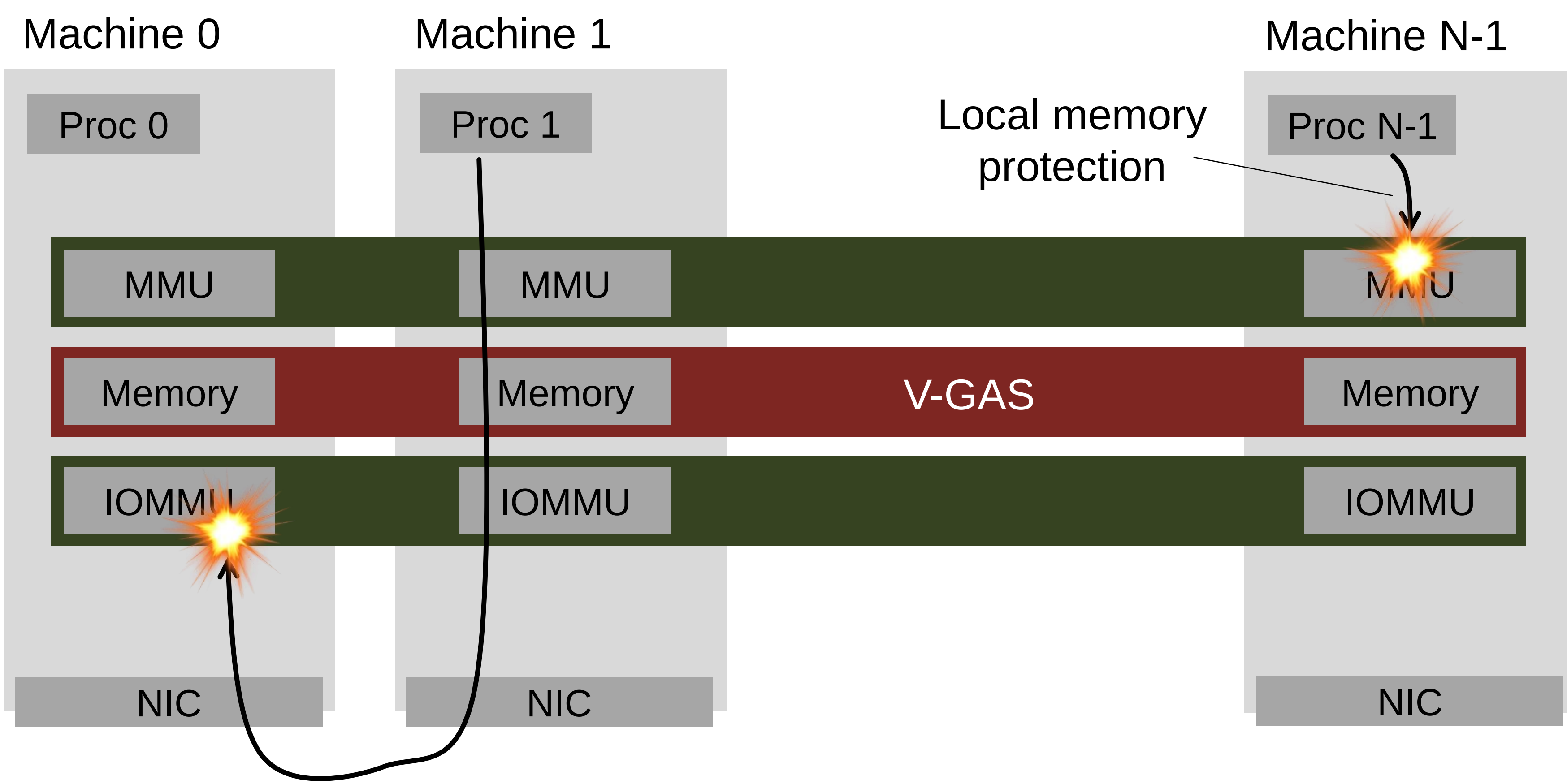


# Active Access Use-Cases

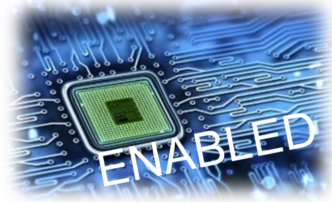
## Virtual Global Address Space (V-GAS)

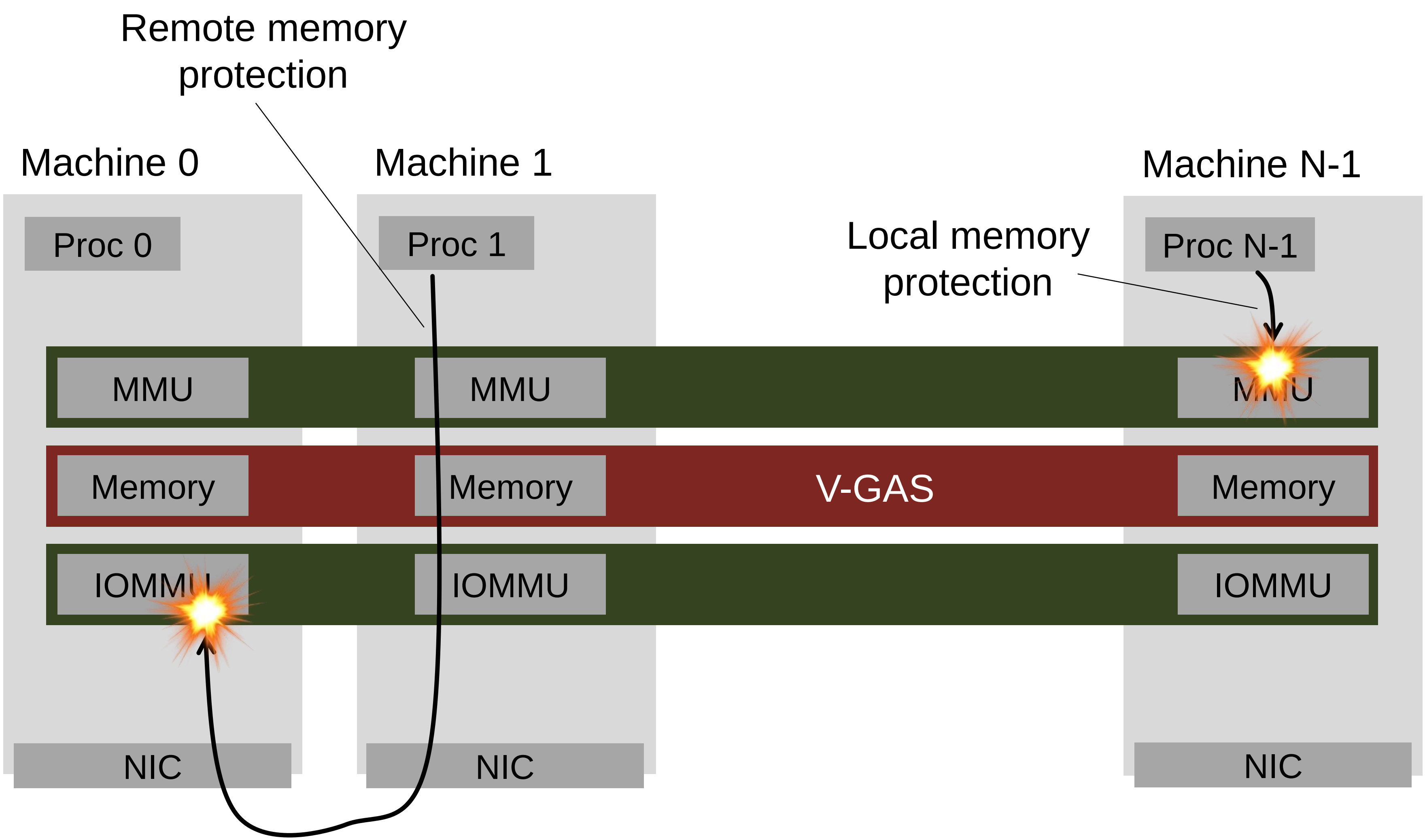


# Active Access Use-Cases

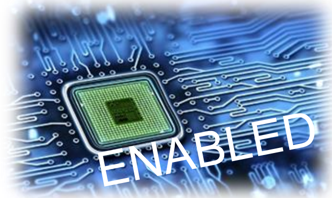
## Virtual Global Address Space (V-GAS)

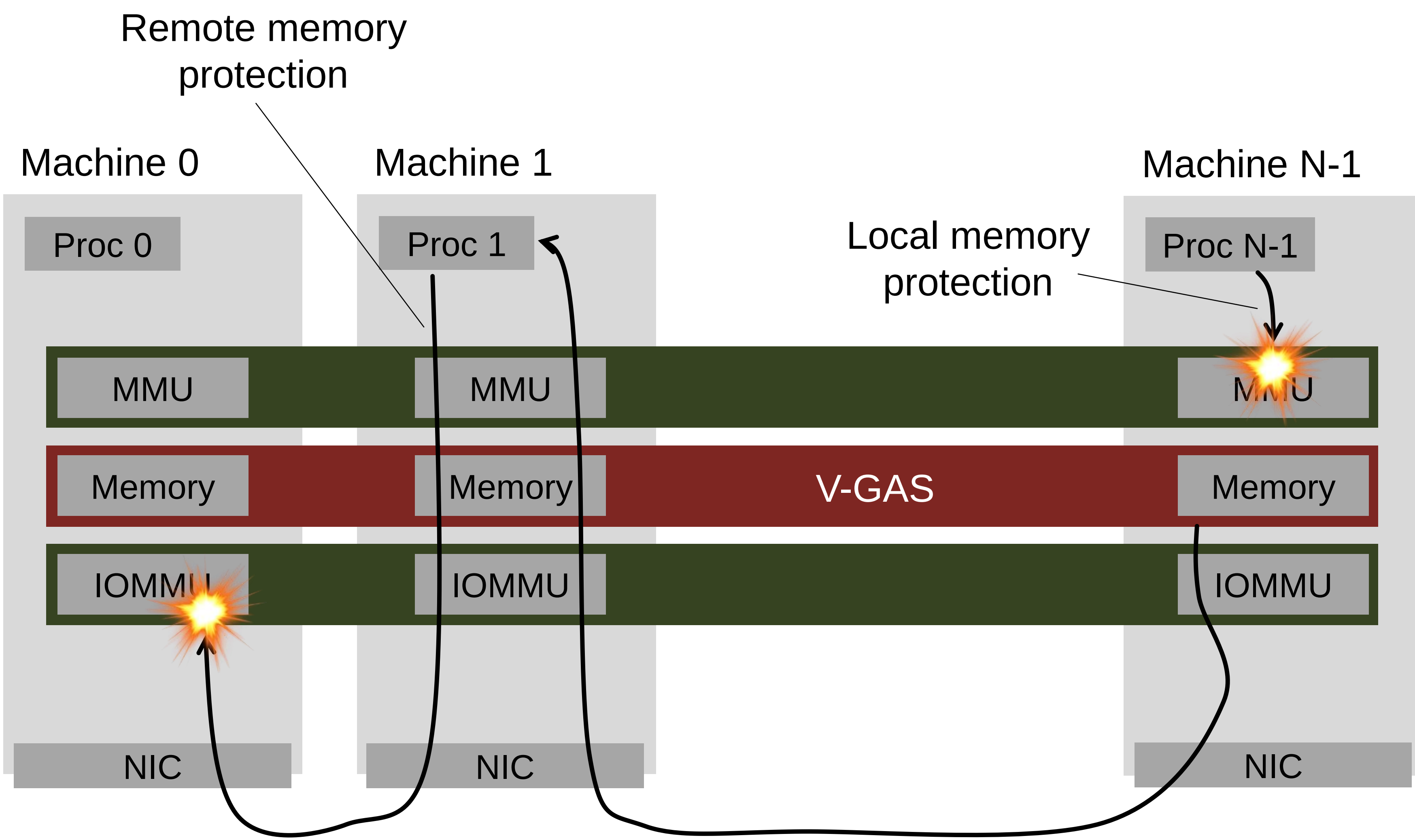


# Active Access Use-Cases

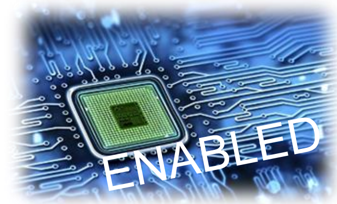
## Virtual Global Address Space (V-GAS)

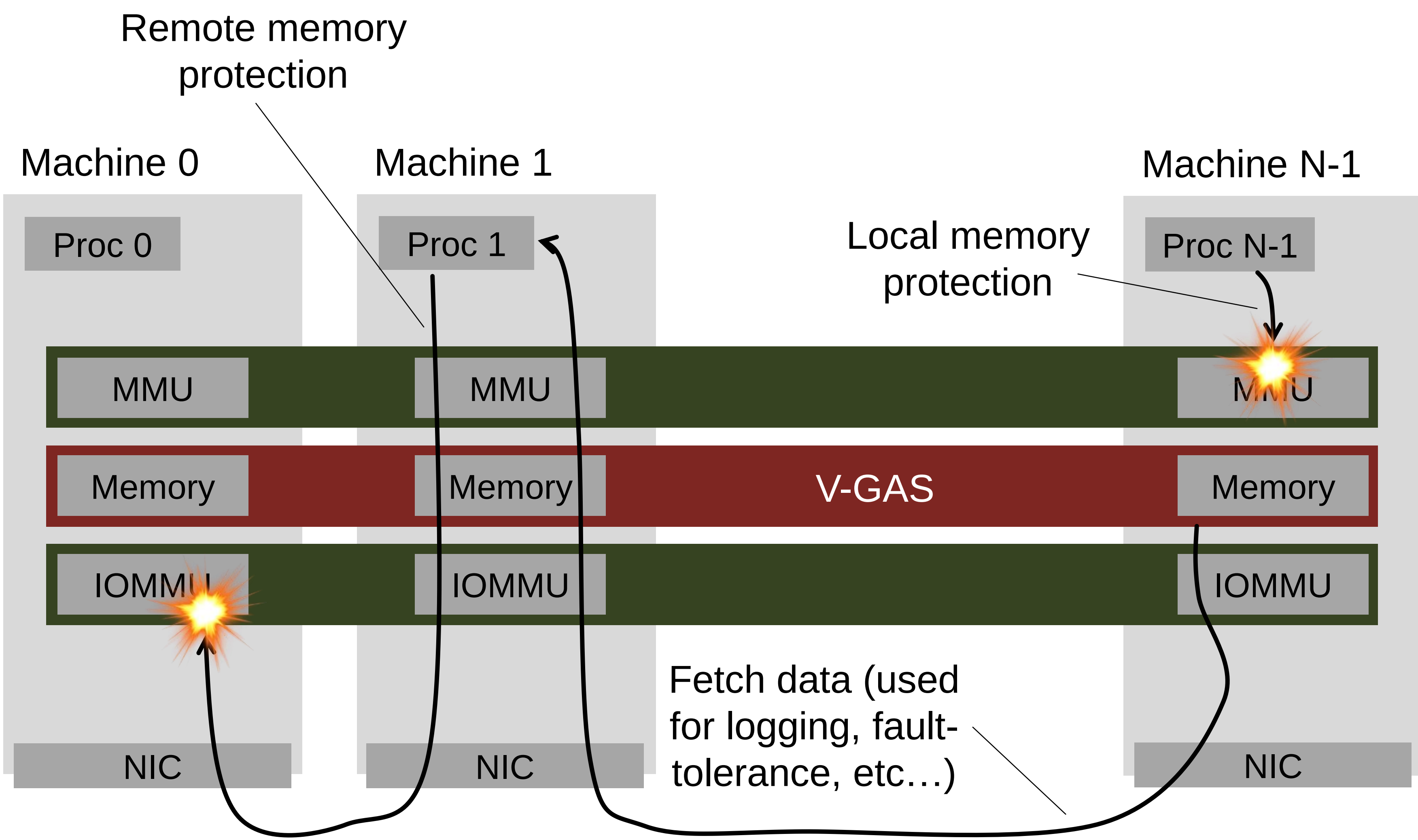


# Active Access Use-Cases

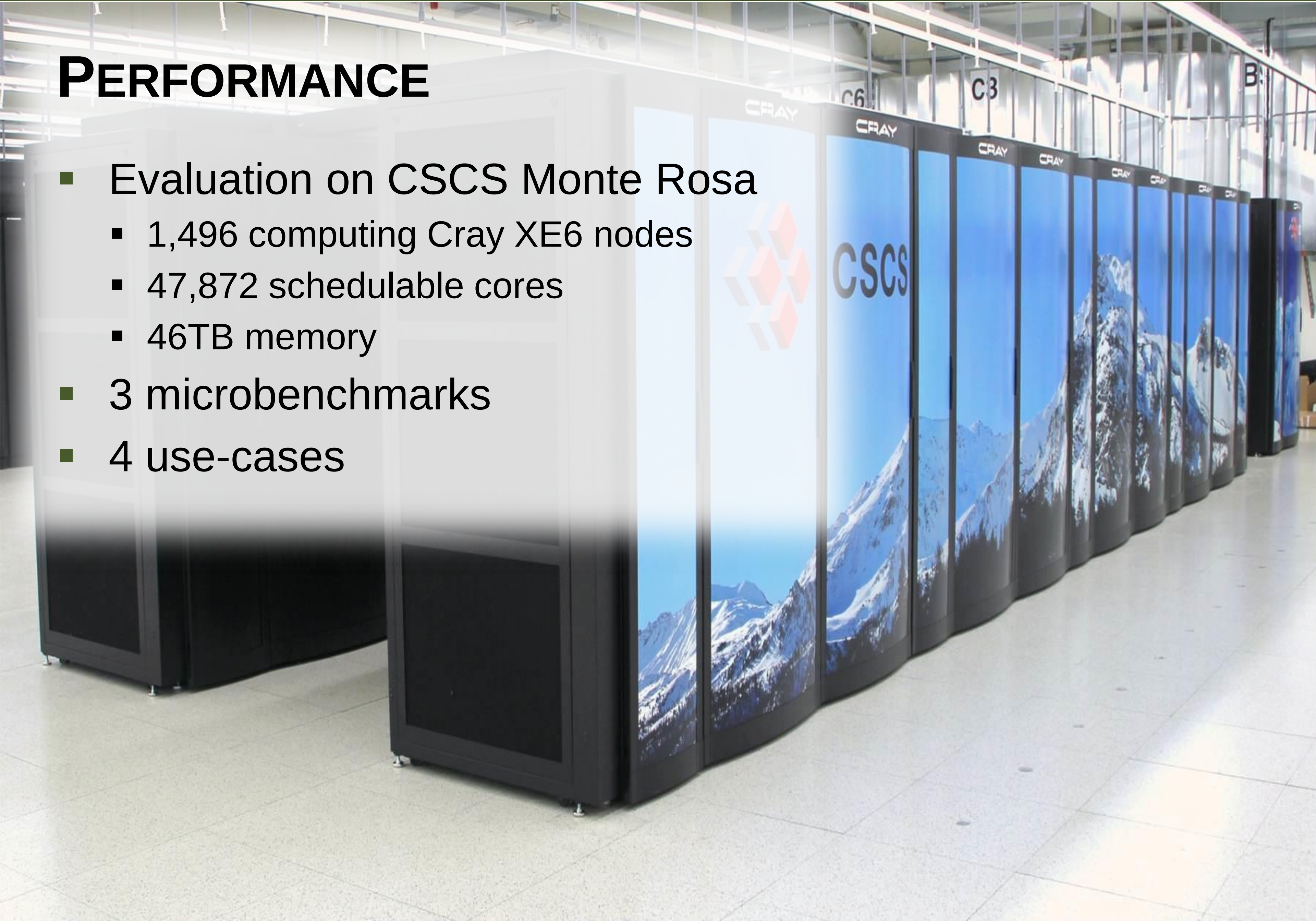
## Virtual Global Address Space (V-GAS)



# 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

# PERFORMANCE: MICROBENCHMARKS
## RAW DATA TRANSFER

- Workload simulated with [1]:

# PERFORMANCE: MICROBENCHMARKS
## RAW DATA TRANSFER

- Workload simulated with [1]:

[1] N. Binkert et al. The gem5 simulator. SIGARCH Comput. Archit. News. 2011

# Performance: Microbenchmarks
## Raw Data Transfer

- Workload simulated with [1]:

- Data generated with:

[1] N. Binkert et al. The gem5 simulator. SIGARCH Comput. Archit. News. 2011

# PERFORMANCE: MICROBENCHMARKS
## RAW DATA TRANSFER

- Workload simulated with [1]:
- Data generated with:
  - PktGen [2]

[1] N. Binkert et al. The gem5 simulator. SIGARCH Comput. Archit. News. 2011

[2] R. Olsson. PktGen the linux packet generator. Linux Symposium. 2005

# Performance: Microbenchmarks
## Raw Data Transfer

- Workload simulated with [1]:

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

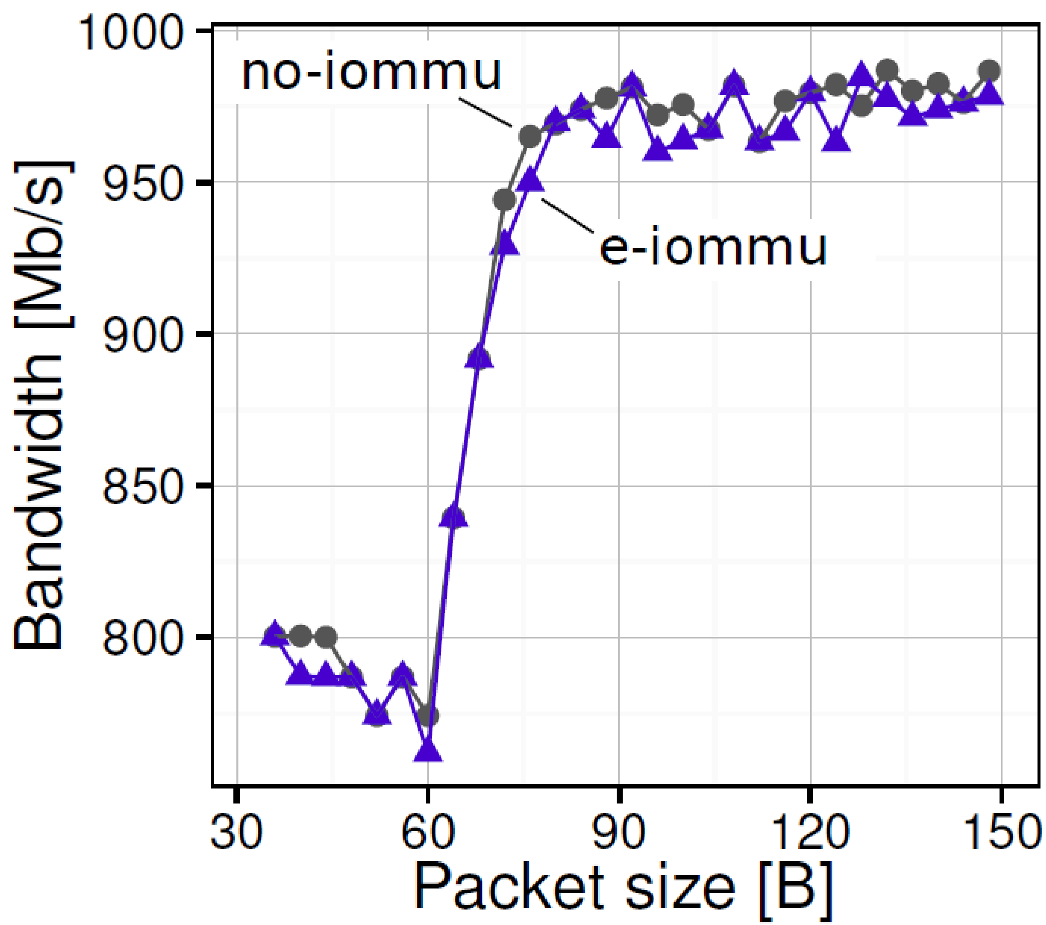
[1] N. Binkert et al. The gem5 simulator. SIGARCH Comput. Archit. News. 2011
[2] R. Olsson. PktGen the linux packet generator. Linux Symposium. 2005
[3] L. Rizzo. netmap: A novel framework for fast packet i/o. USENIX Annual Technical Conference. 2012

# Performance: 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

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Int

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

# Performance: Large-Scale Codes
## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

RMA

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

RMA

CRAY

DMAPP

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

RMA

CRAY

DMAPP

InfiniBand

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

RMA

IBM Cell

CRAY DMAPP

InfiniBand

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

RMA

IBM Cell

CRAY DMAPP

InfiniBand

Mellanox Technologies RoCE

# 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

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

Active Messages

AM

RMA

IBM

Cell

CRAY

DMAPP

InfiniBand

Mellanox Technologies

RoCE

# 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-Exp

# 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

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

Active Messages

AM AM-Onload

AM-Exp AM-Ints

RMA

Cell

DMAPP

RoCE

# Performance: Large-Scale Codes
## Comparison Targets

Active Access

AA-Poll
AA-Int
AA-SP

Active Messages

AM AM-Onload
AM-Exp AM-Ints

IBM
DCMF LAPI
PAMI

RMA

IBM
Cell

CRAY
DMAPP

InfiniBand

Mellanox Technologies
RoCE

# Performance: Large-Scale Codes

## Comparison Targets

Active Access

AA-Poll

AA-Int

AA-SP

Active Messages

AM AM-Onload

AM-Exp AM-Ints

IBM DCMF LAPI PAMI

Myricom MX

RMA

IBM Cell

Cray DMAPP

InfiniBand

Mellanox Technologies RoCE

# Performance: Large-Scale Codes

## Comparison Targets

**Active Access**

AA-Poll

AA-Int

AA-SP

**Active Messages**

AM

AM-Onload

AM-Exp

AM-Ints

DCMF

LAPI

PAMI

MX

AM++

GASNet

**RMA**

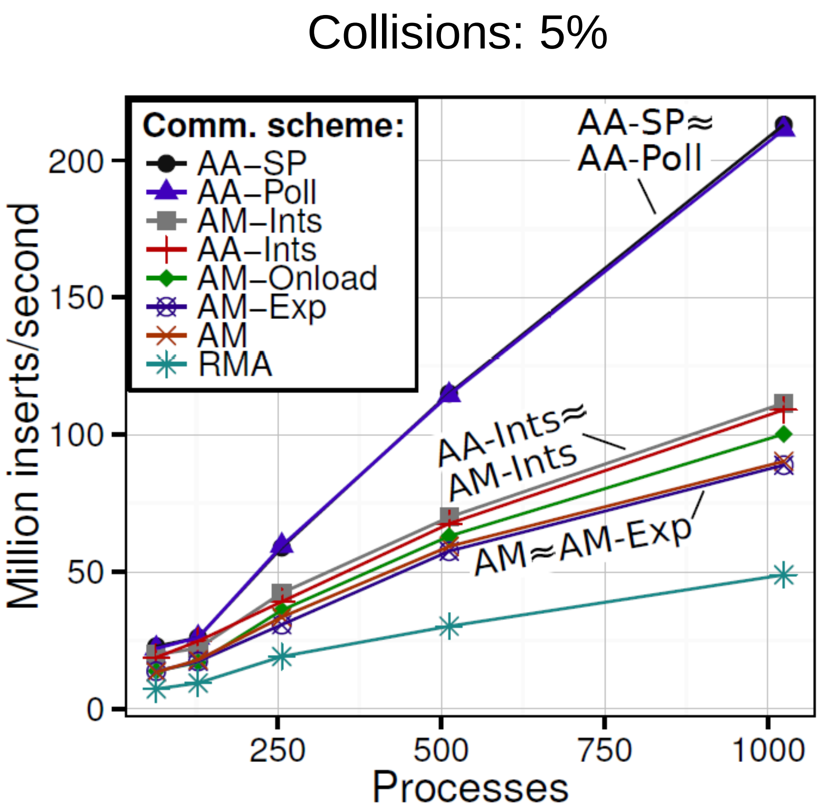
DMAPP

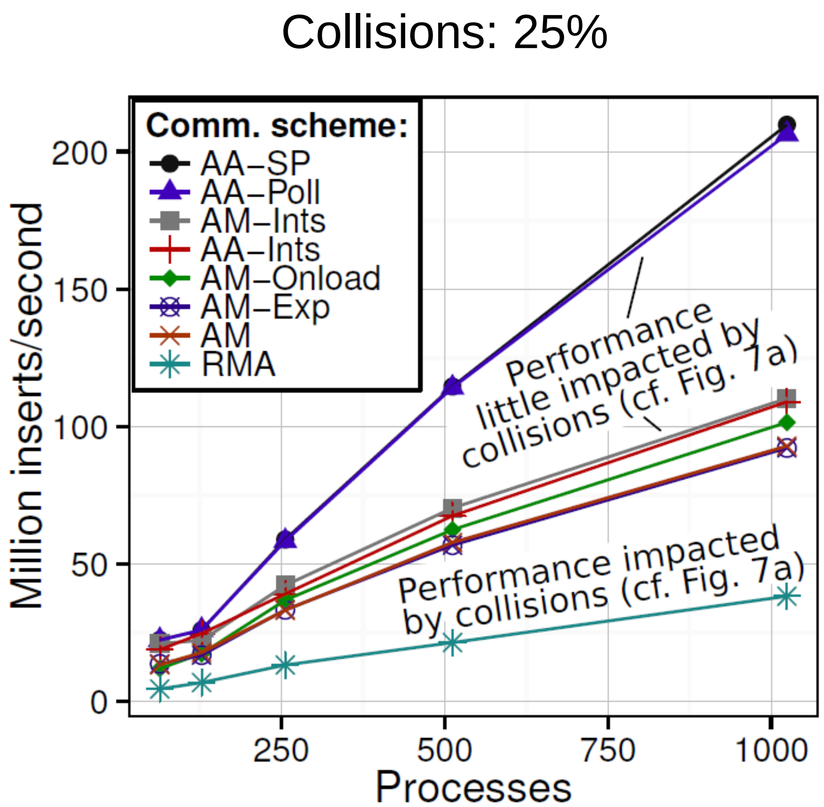
Cell

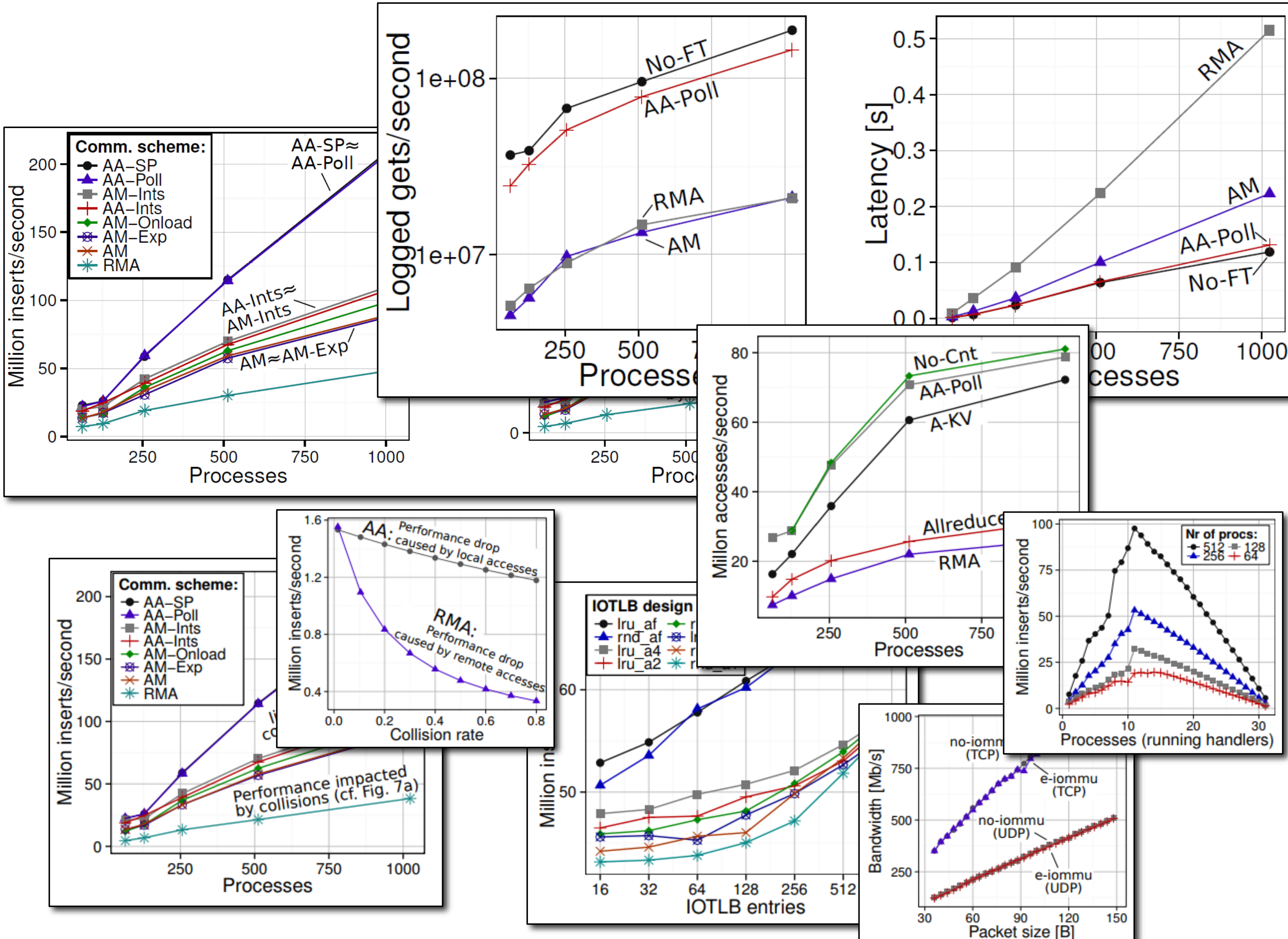
RoCE

# Performance: Large-Scale Codes

## Distributed Hashtable



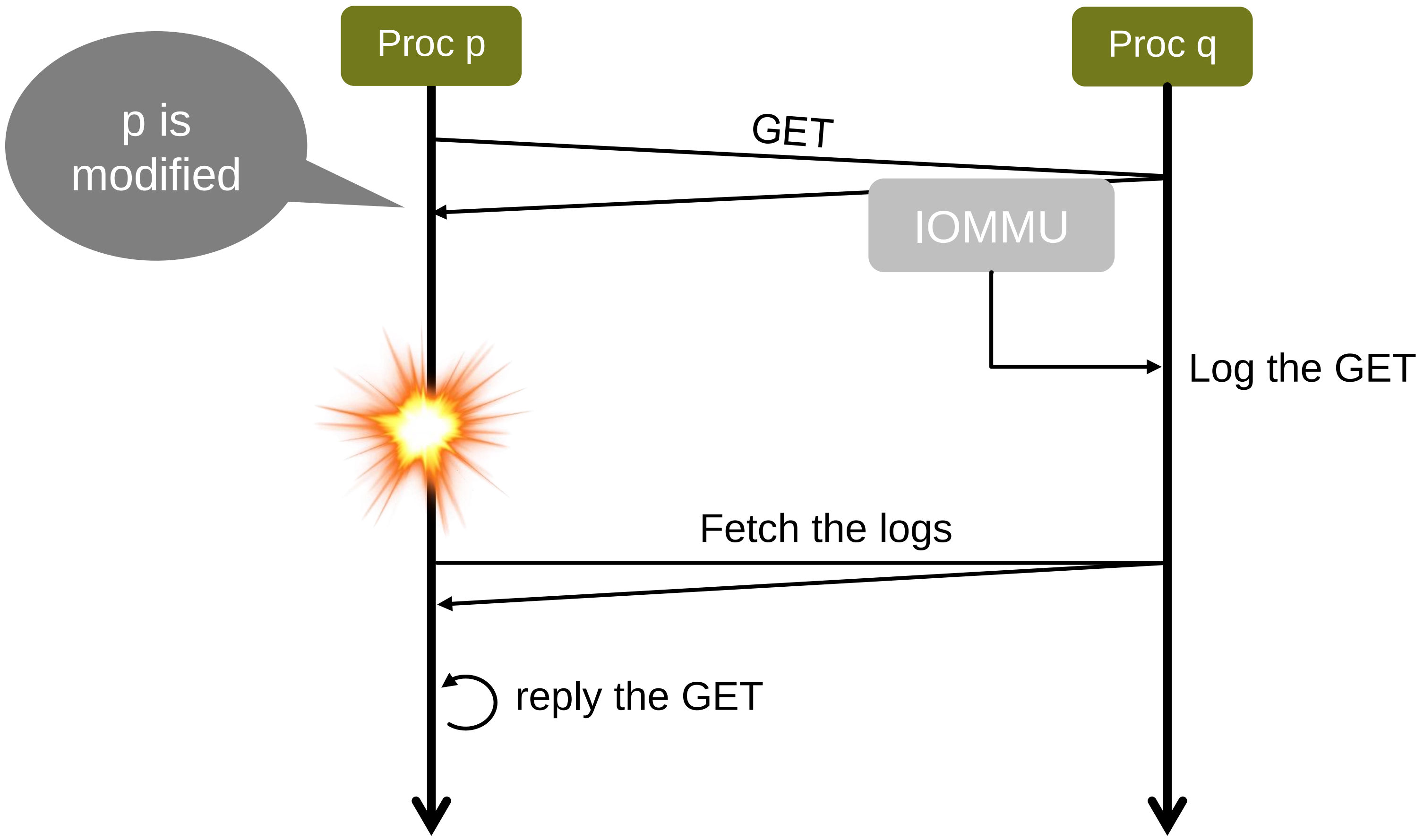




Comm. scheme:
AA–SP
AA–Poll
AM–Ints
AA–Ints
AM–Onload
AM–Exp
AM
RMA
AA-SP≈
AA-Poll
AA-Ints≈
AM-Ints
AM≈AM-Exp
Million inserts/second
Processes
Logged gets/second
No-FT
AA-Poll
RMA
AM
Latency [s]
Million accesses/second
No-Cnt
A-KV
Allreduce
AA: Performance drop caused by local accesses
RMA: Performance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

# ACTIVE ACCESS USE-CASES

## ACCELERATING LOGGING FOR RMA

- Logging gets (AA):



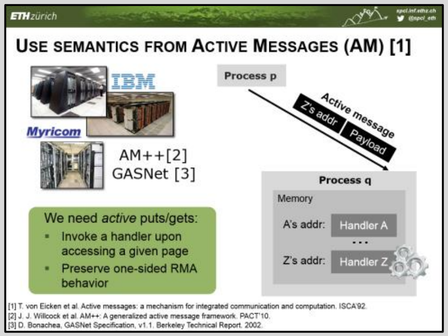
# CONCLUSIONS

# CONCLUSIONS

## Active Access

# CONCLUSIONS

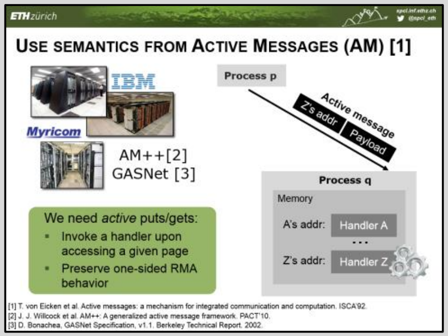
## Active Access




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

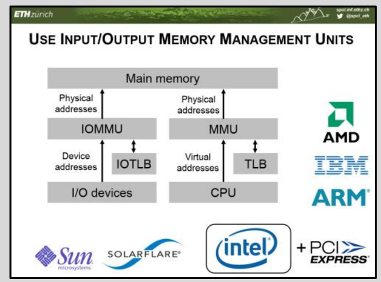
# Conclusions

## Active Access



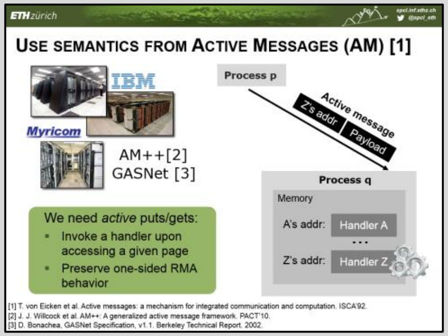

Alleviates RMA's problems with AMs while preserving one-sided semantics

Uses commodity & common IOMMUs



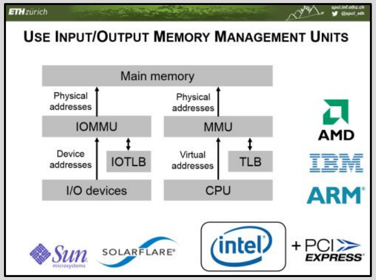
# Conclusions

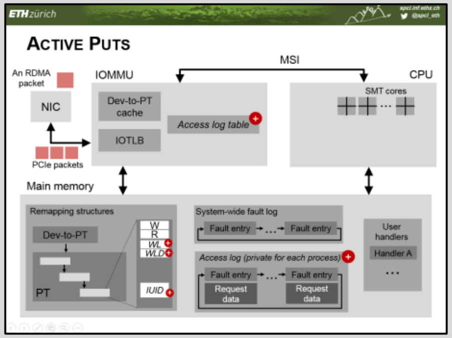
## Active Access




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

Uses commodity & common IOMMUs

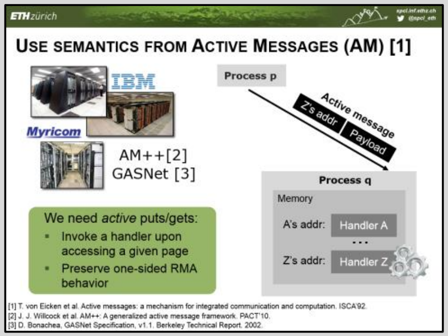



Extends paging capabilities in a distributed environment

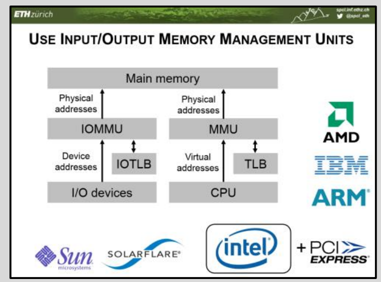
# Conclusions

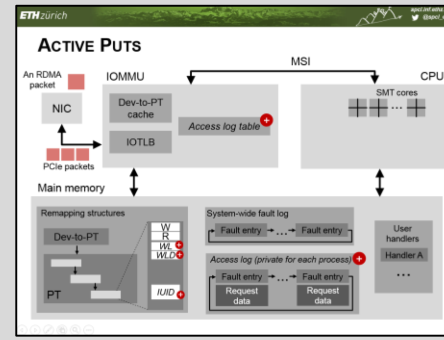
## Active Access




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

Uses commodity & common IOMMUs



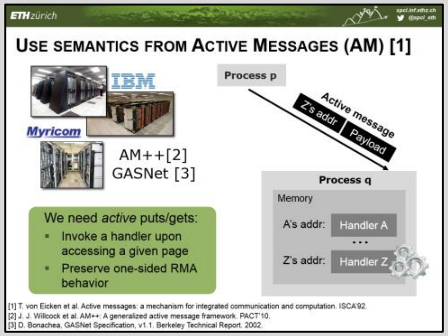




Extends paging capabilities in a distributed environment

## Data-centric programming

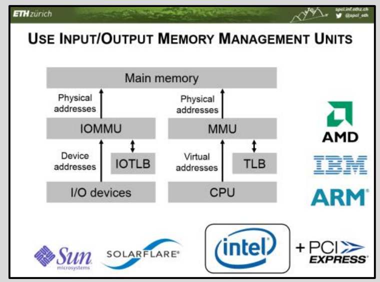
# Conclusions

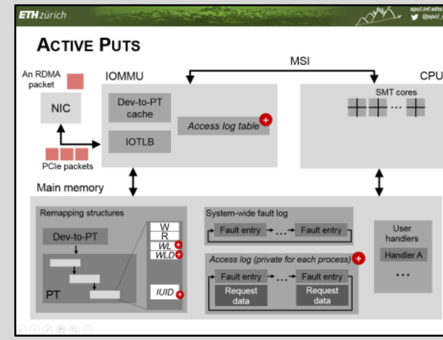
## Active Access




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

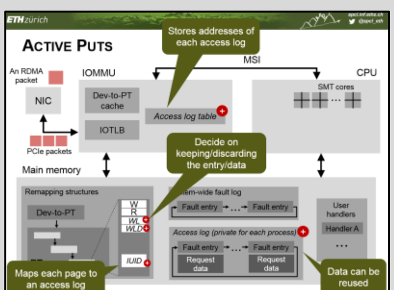
Uses commodity & common IOMMUs




Extends paging capabilities in a distributed environment

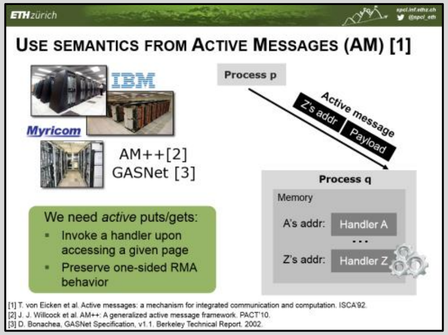
## Data-centric programming




Addresses of pages guide the execution of handlers

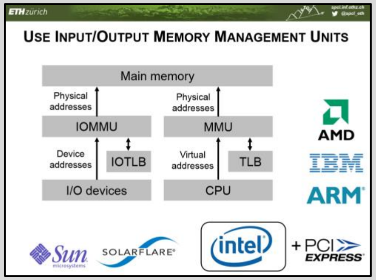
# Conclusions

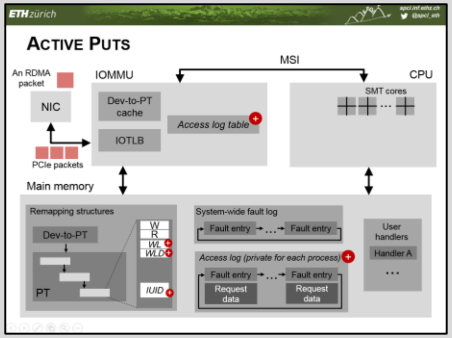
## Active Access




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

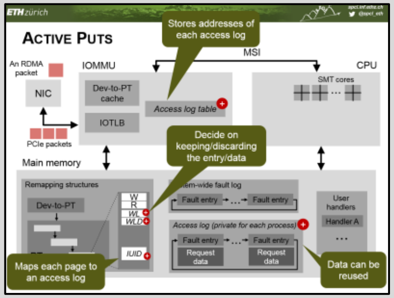
Uses commodity
& common IOMMUs



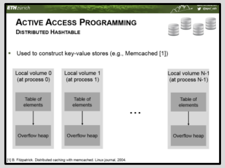



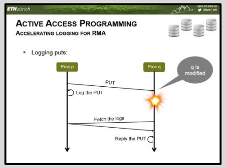

Extends paging capabilities in a distributed environment

## Data-centric programming




Addresses of pages guide the execution of handlers

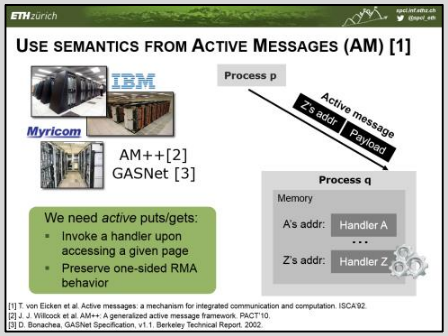



Hashtables, logging schemes, counters, V-GAS, checkpointing...

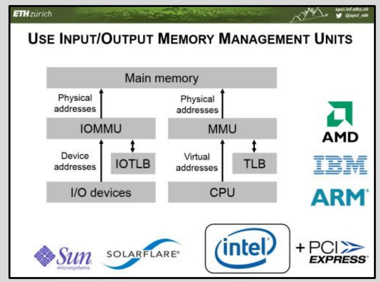
# Conclusions

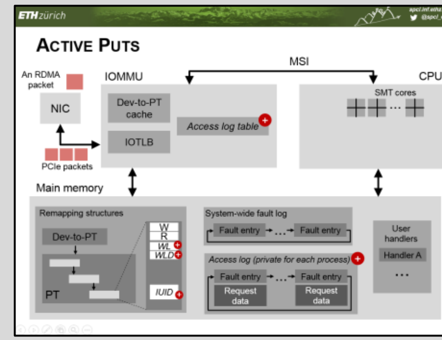
## Active Access




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

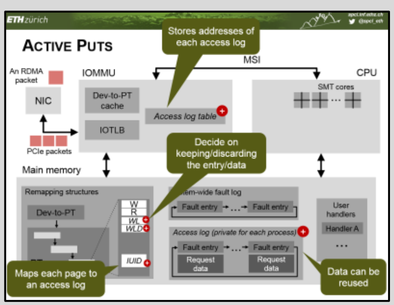
Uses commodity & common IOMMUs



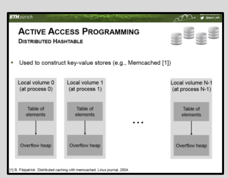



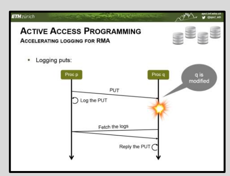

Extends paging capabilities in a distributed environment

## Data-centric programming




Addresses of pages guide the execution of handlers



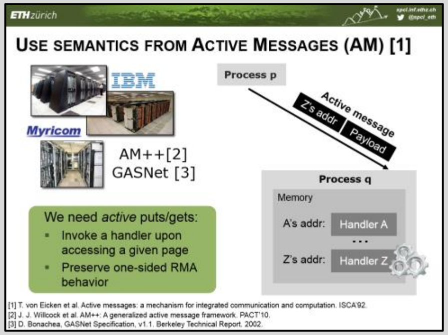




Hashtables, logging schemes, counters, V-GAS, checkpointing...

## Performance

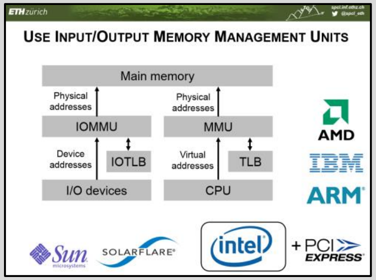
# Conclusions

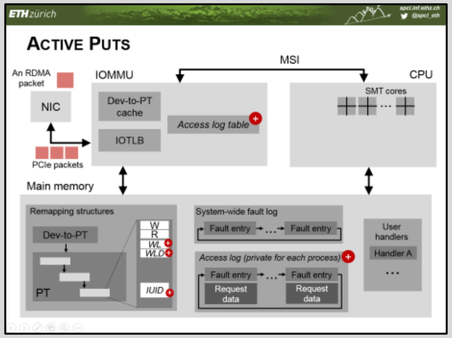
## Active Access




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

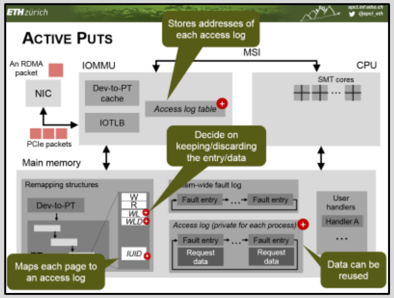
Uses commodity & common IOMMUs



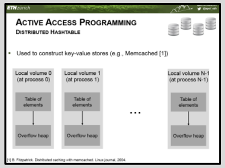



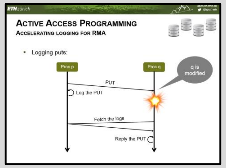

Extends paging capabilities in a distributed environment

## Data-centric programming



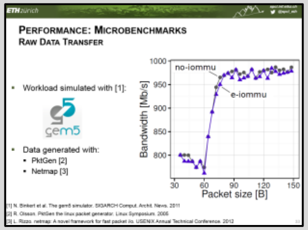

Addresses of pages guide the execution of handlers

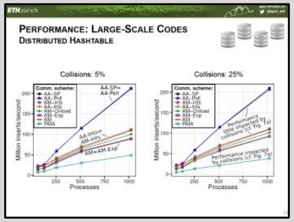


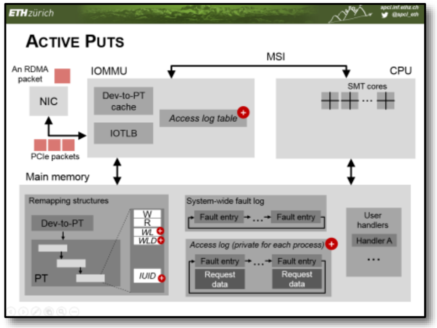

Hashtables, logging schemes, counters, V-GAS, checkpointing...

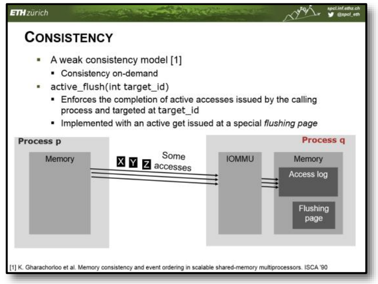
## Performance

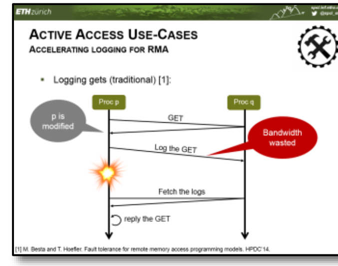


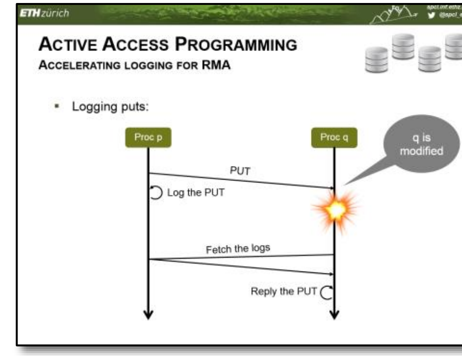



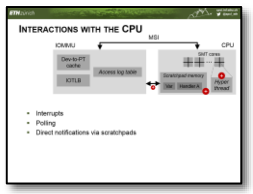

Accelerates various distributed codes

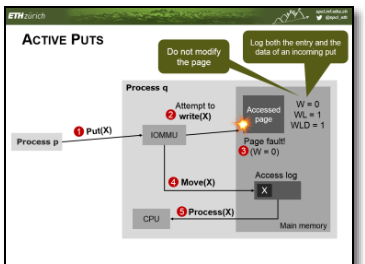


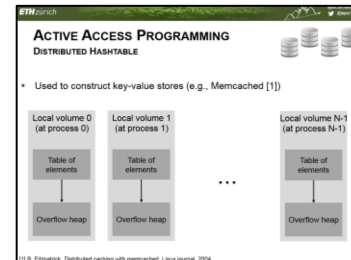



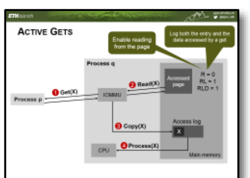



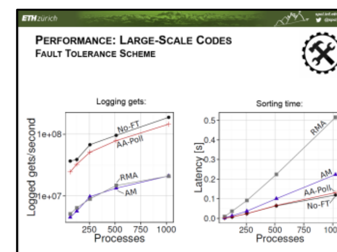



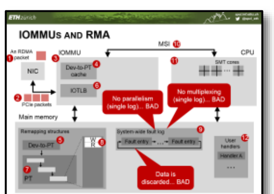




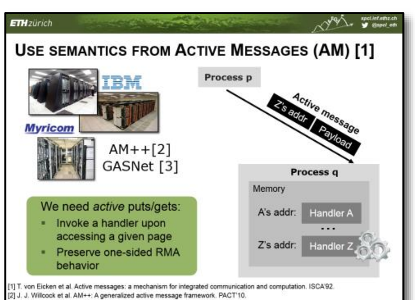

# Thank you for your attention

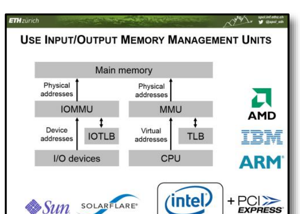


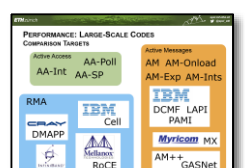



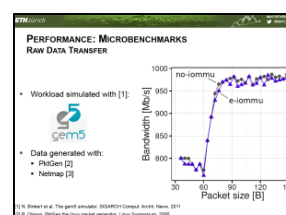



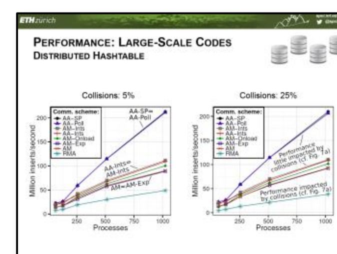




# Active Access Use-Cases

## Accelerating logging for RMA

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging – a popular mechanism for fault-tolerance.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging – a popular mechanism for fault-tolerance.
- Remote communication (puts/gets) is logged.

# 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.

# 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

# Active Access Use-Cases

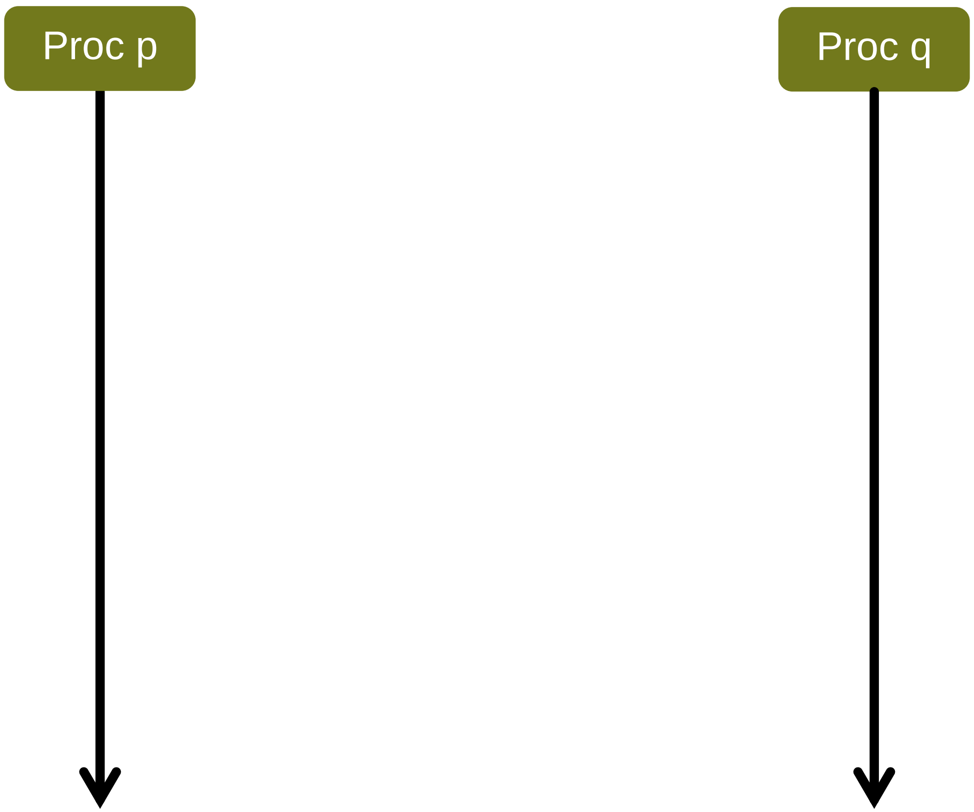
## Accelerating logging for RMA

- Logging puts:

# Active Access Use-Cases

## Accelerating logging for RMA

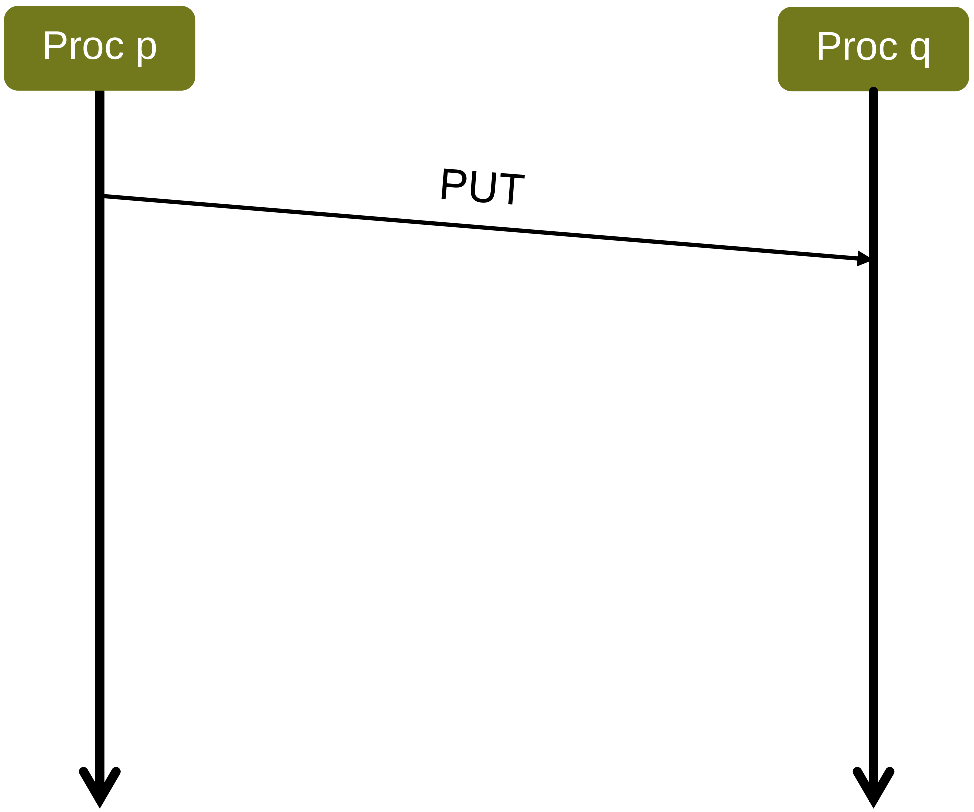
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

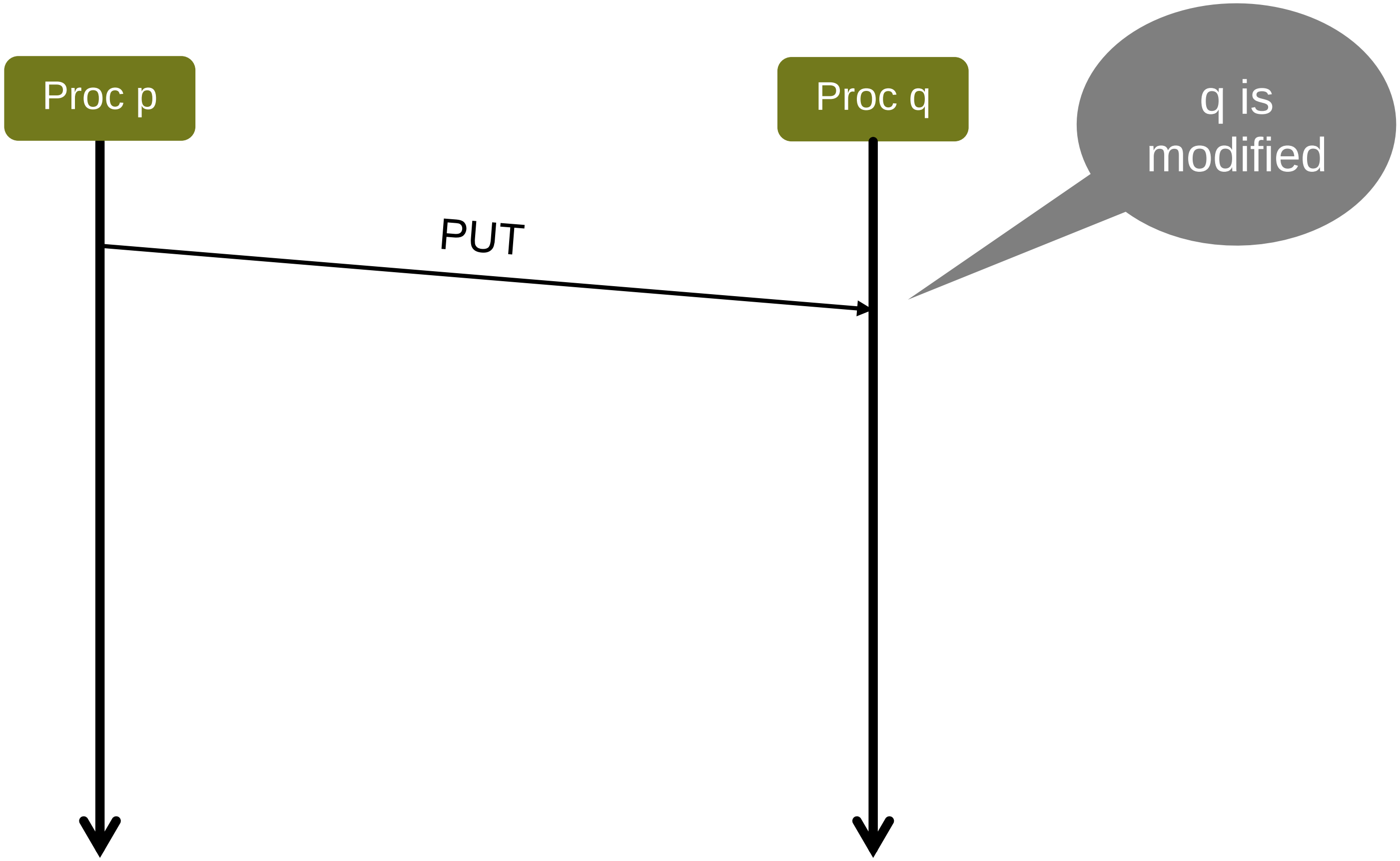
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

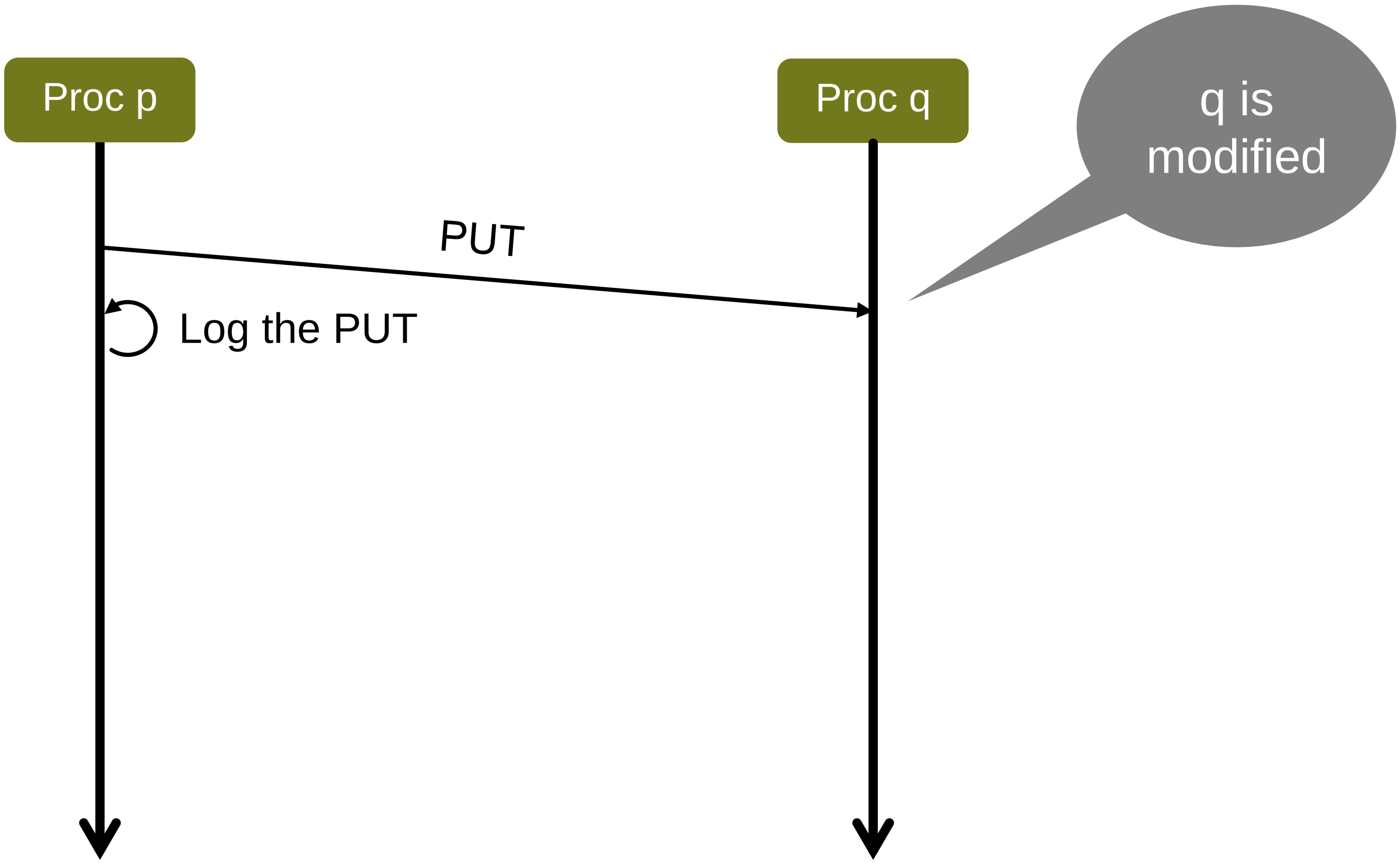
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

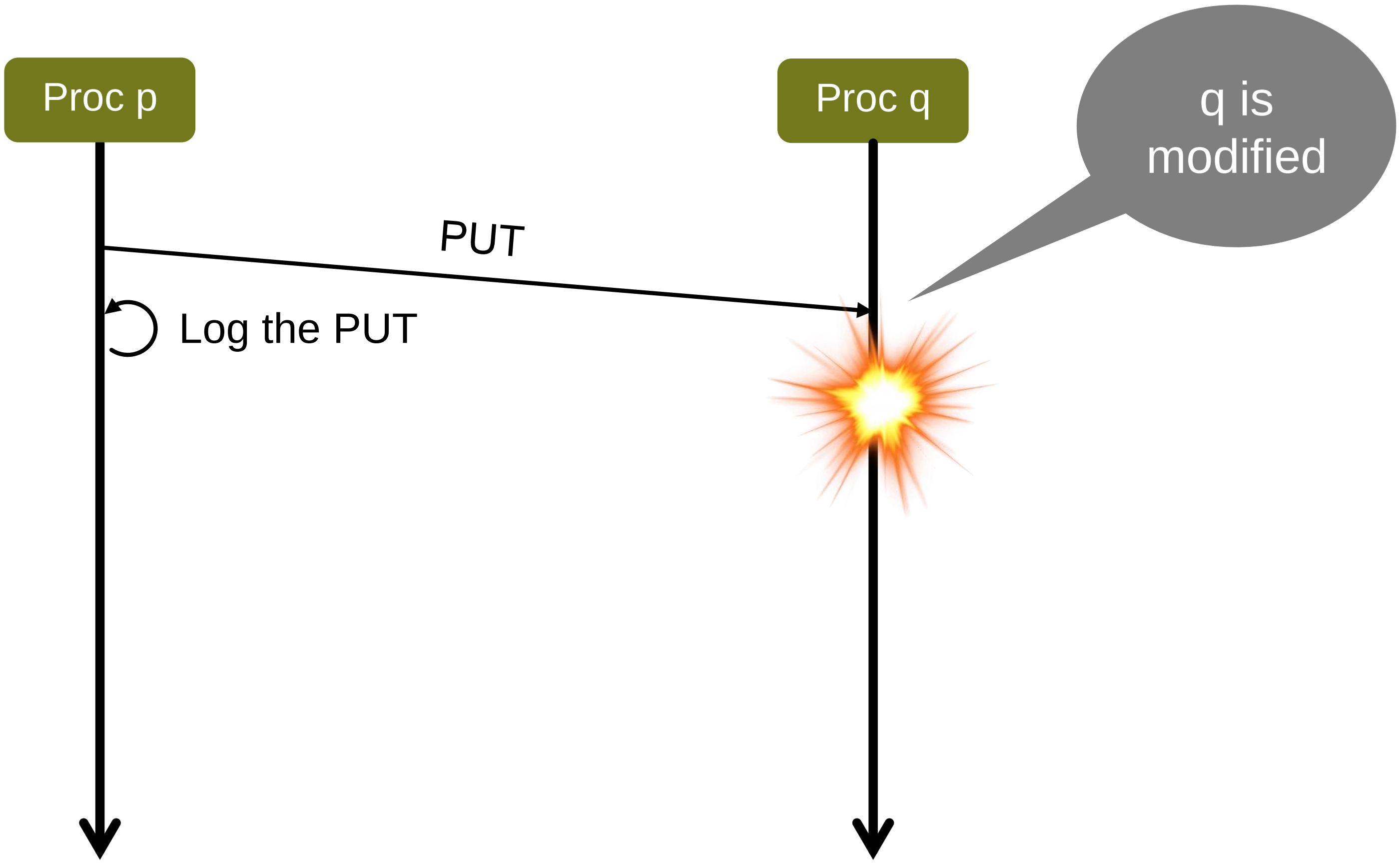
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

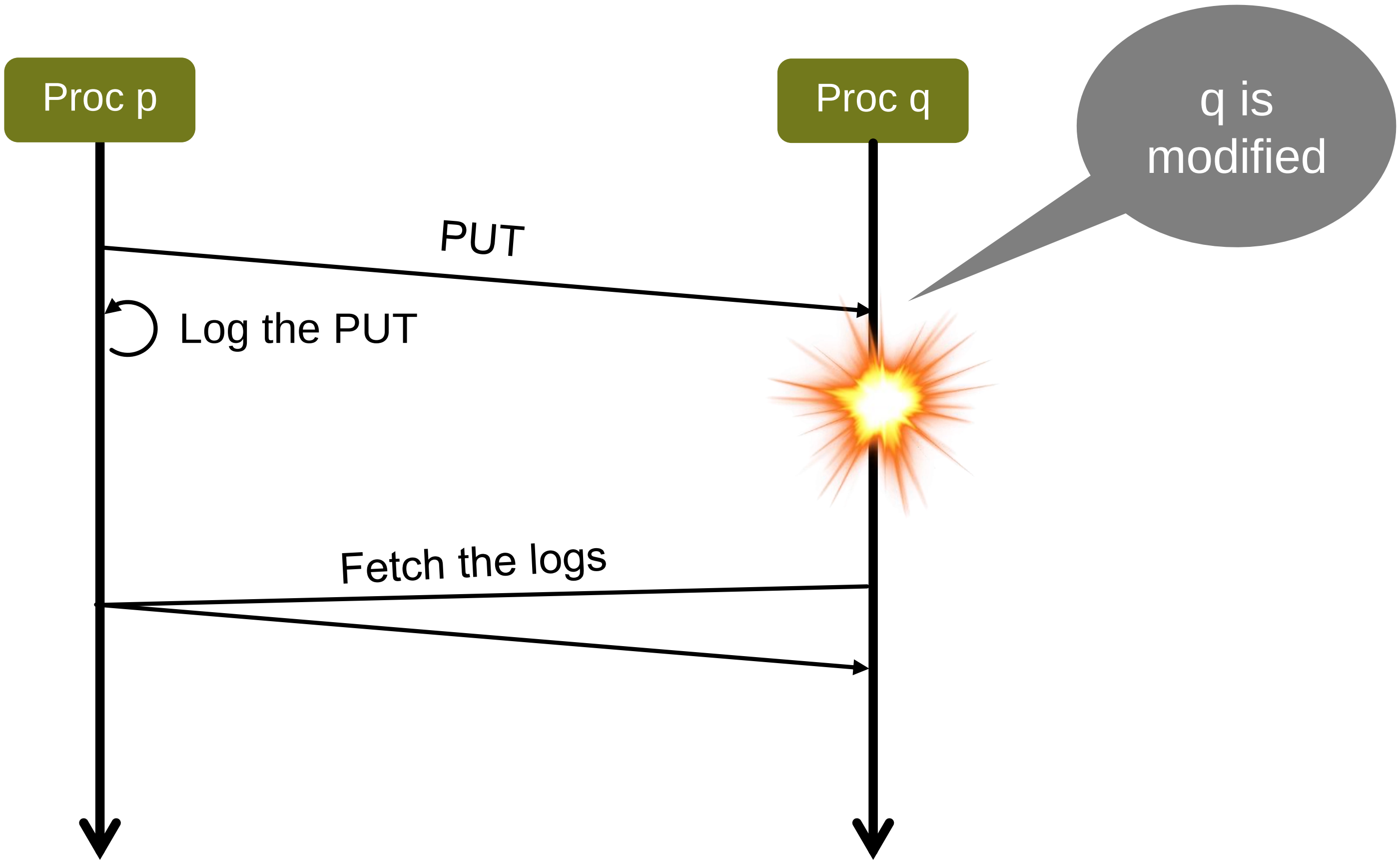
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

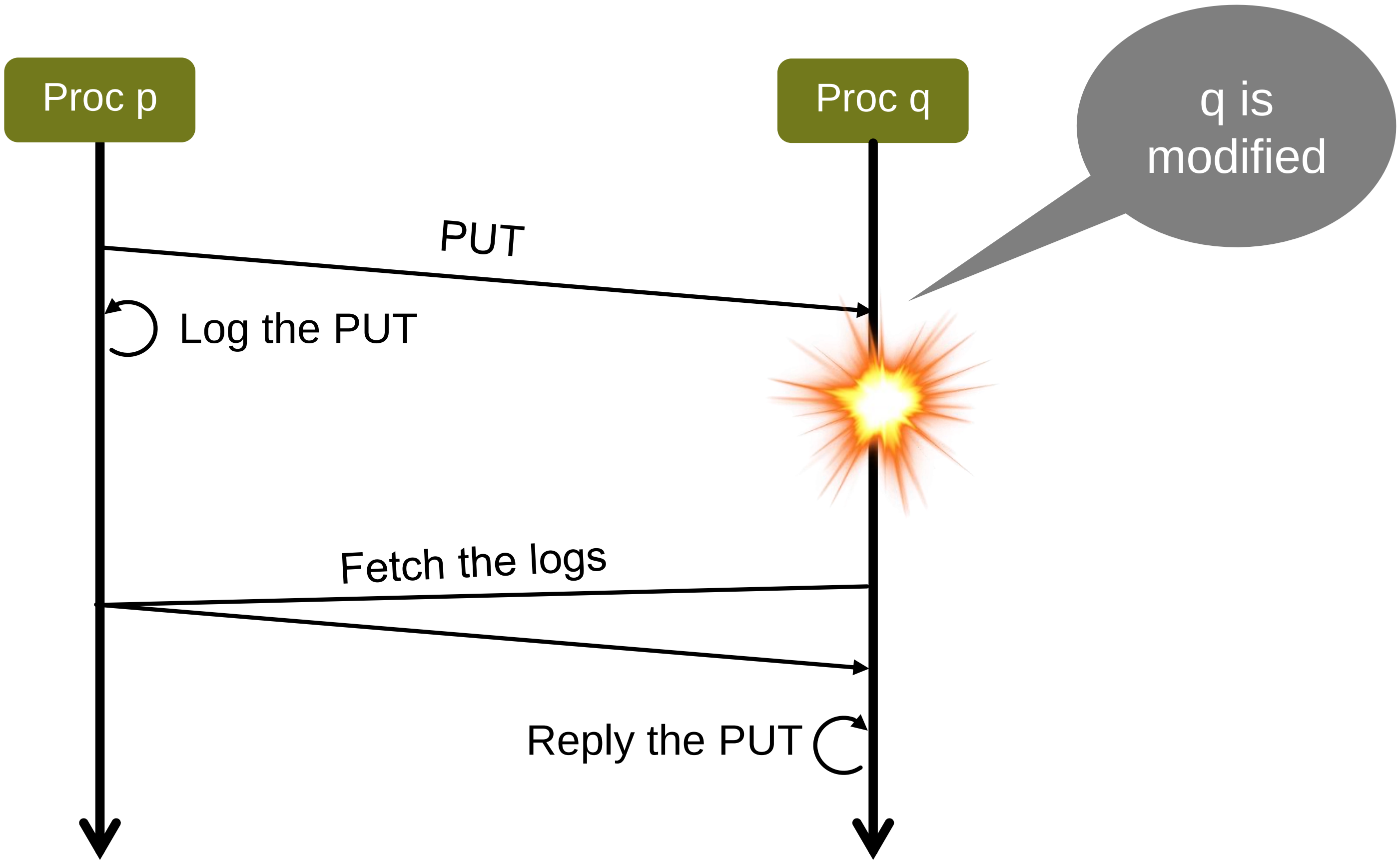
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

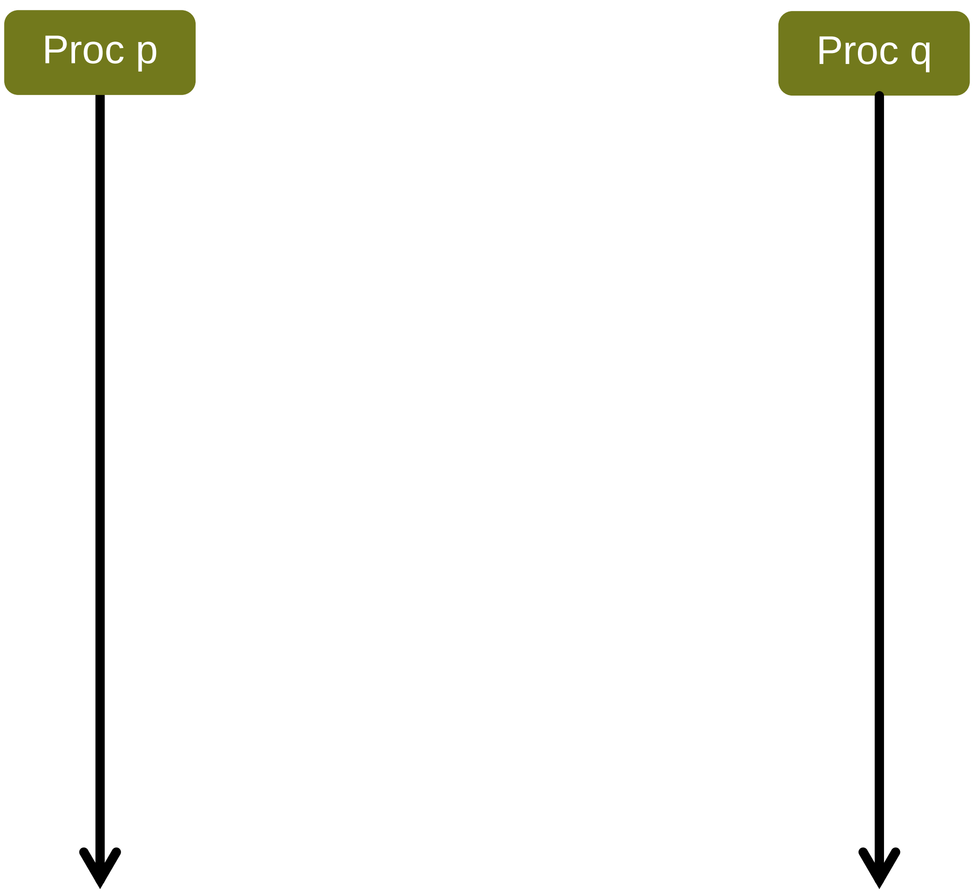
- Logging puts:



# Active Access Use-Cases

## Accelerating logging for RMA

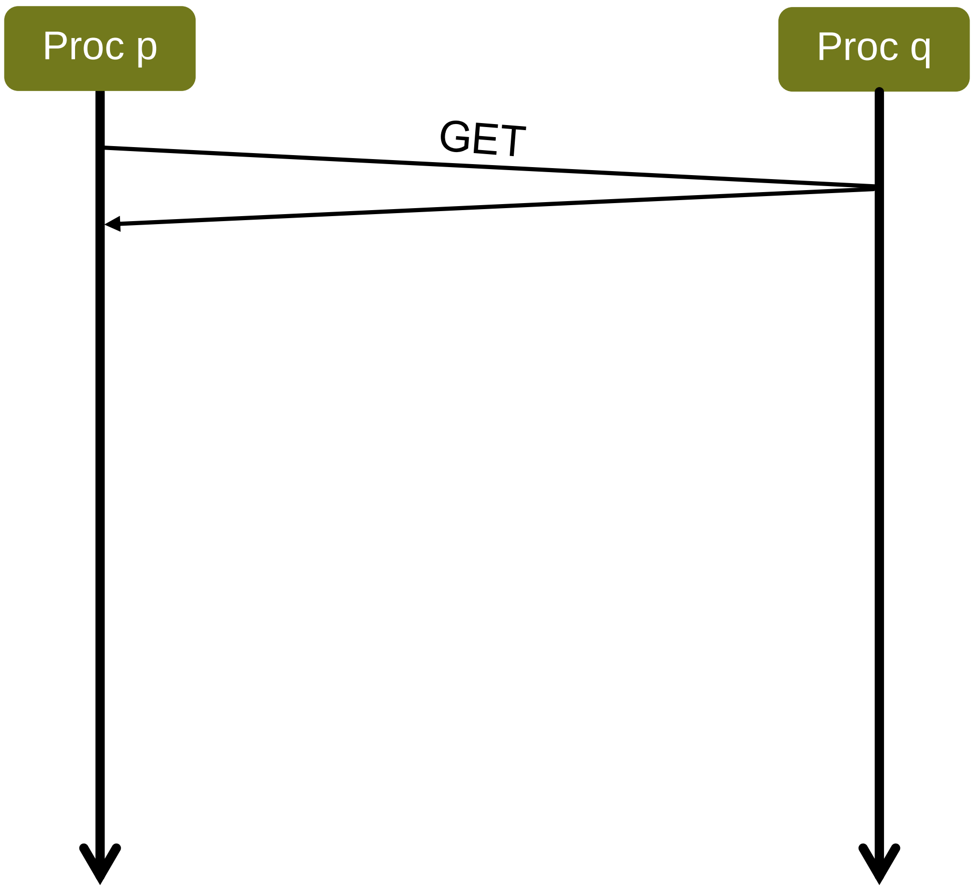
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

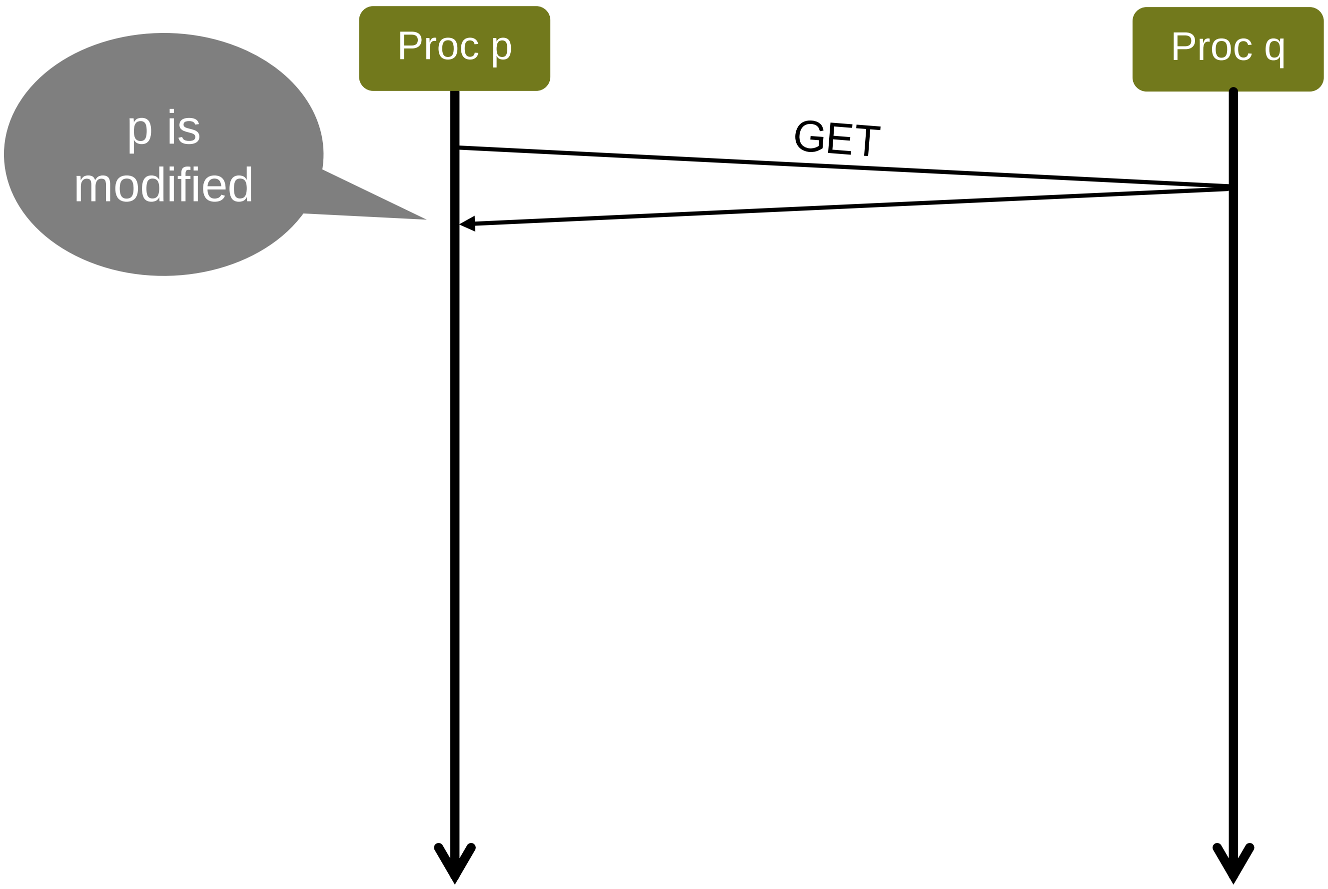
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

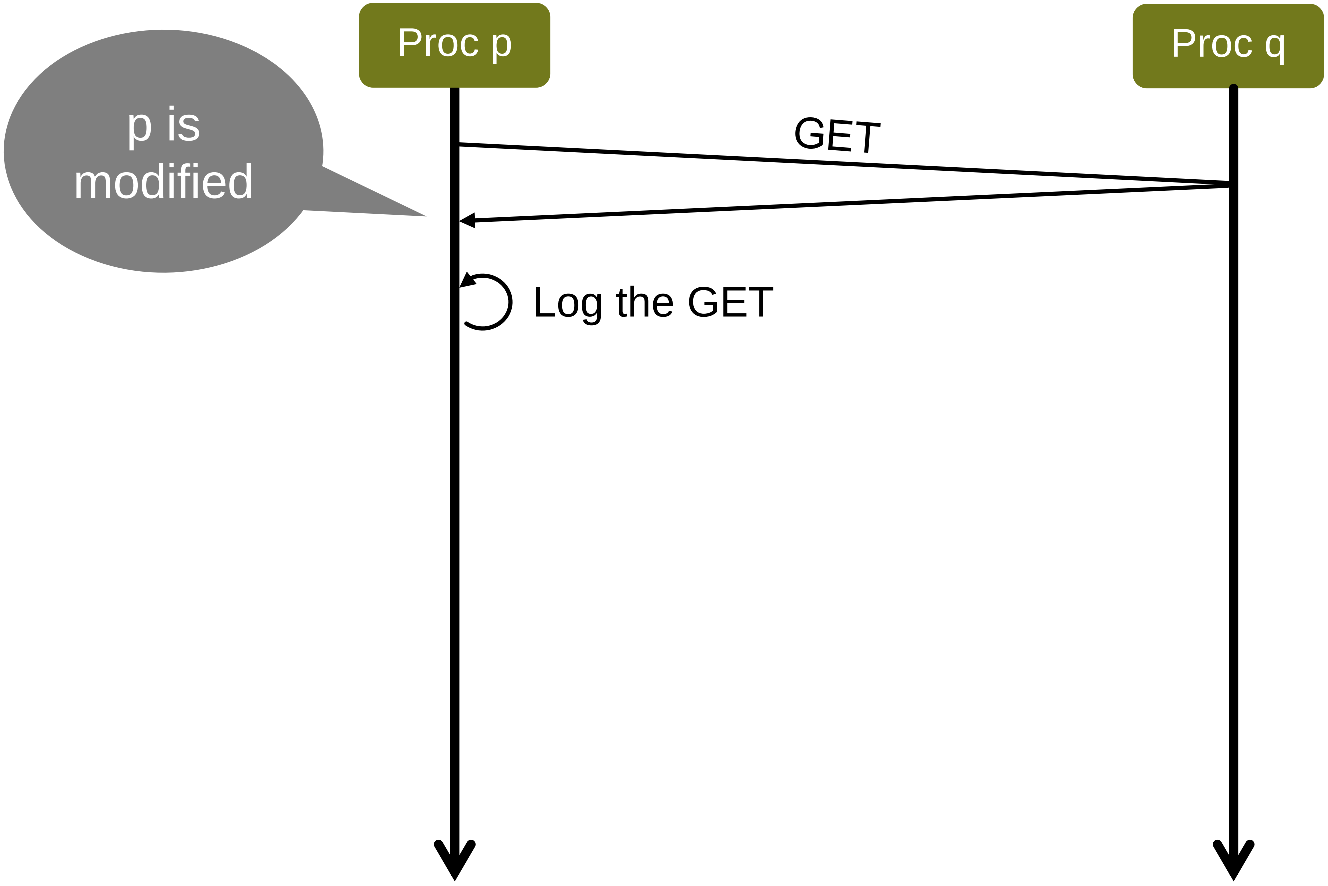
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

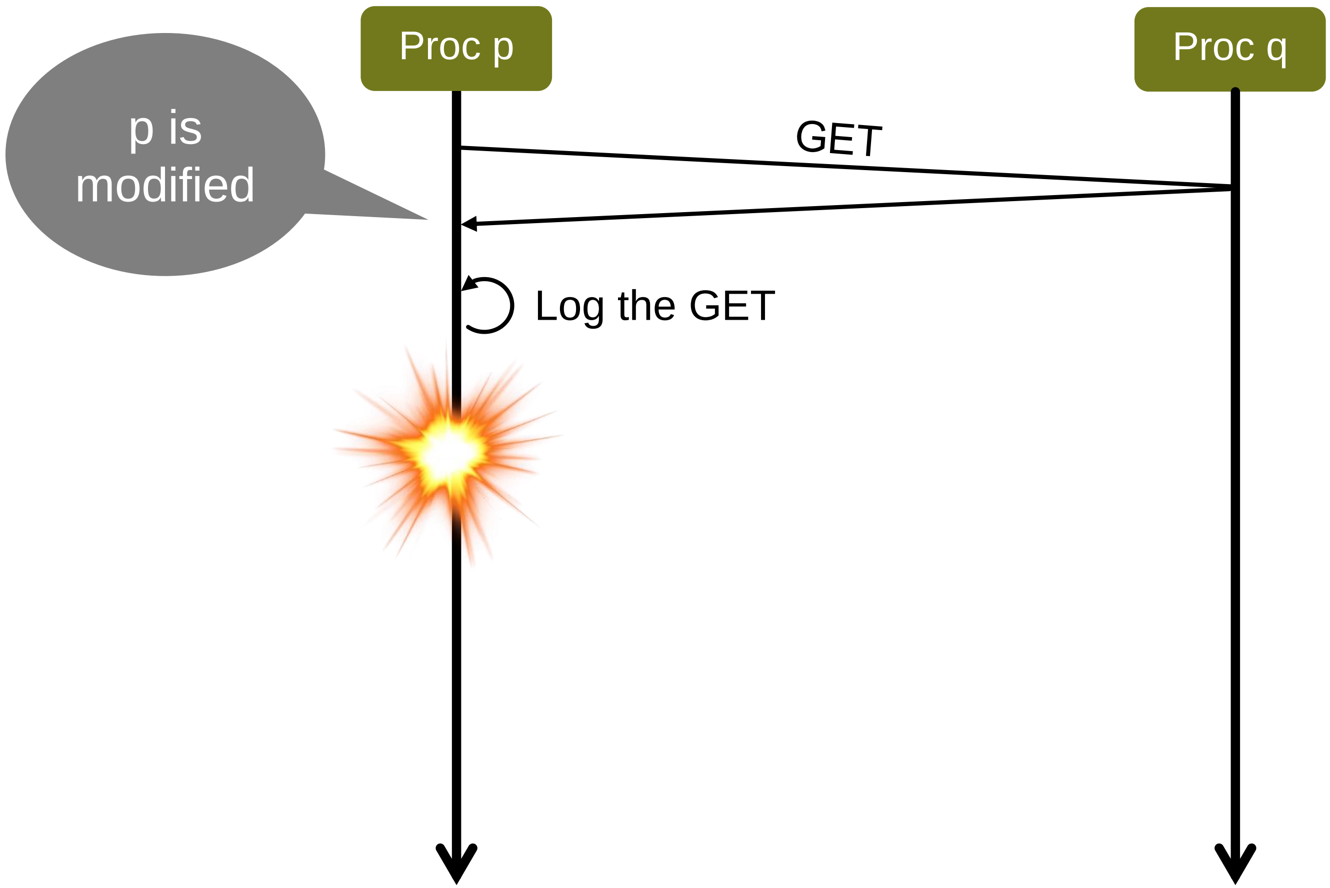
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

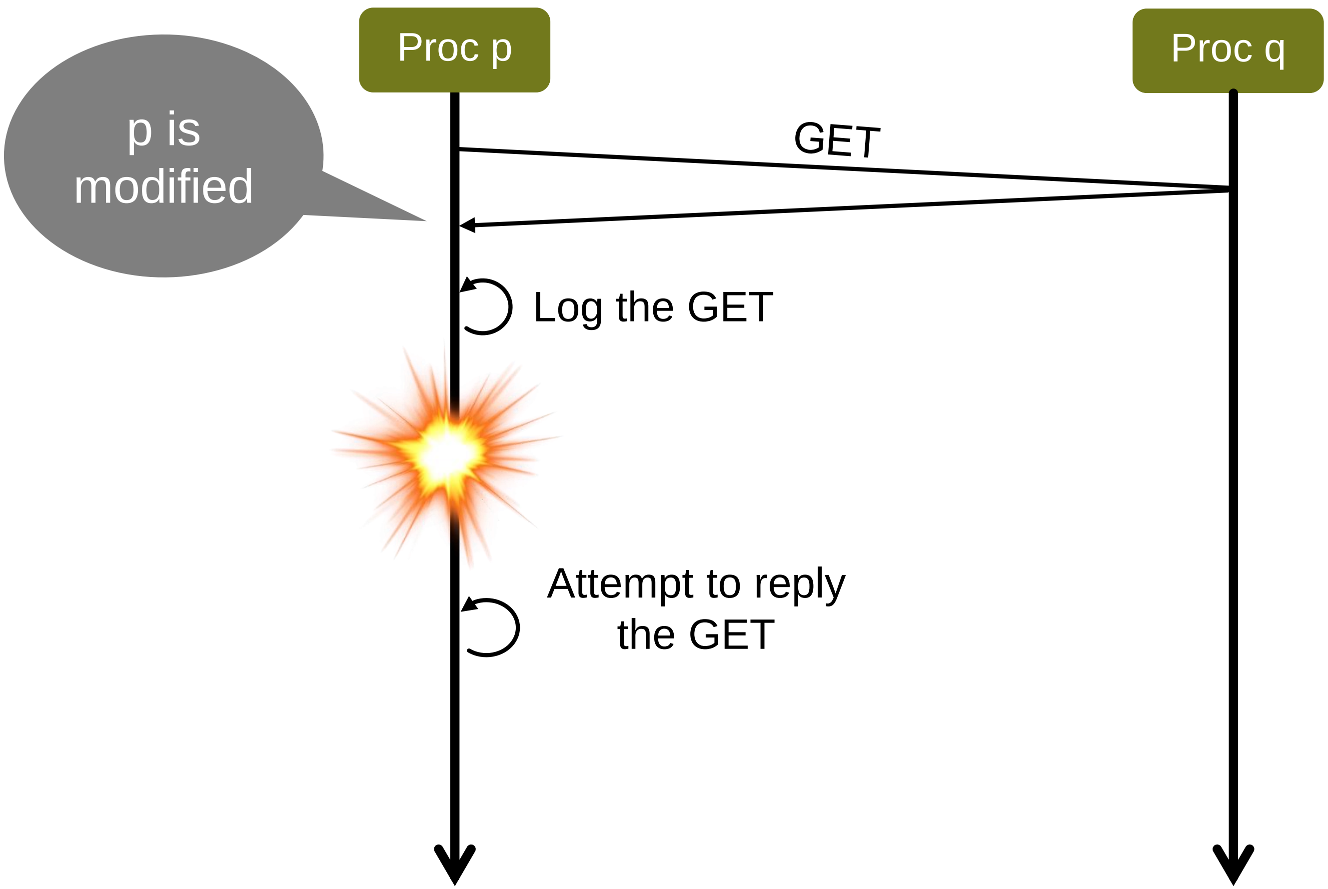
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

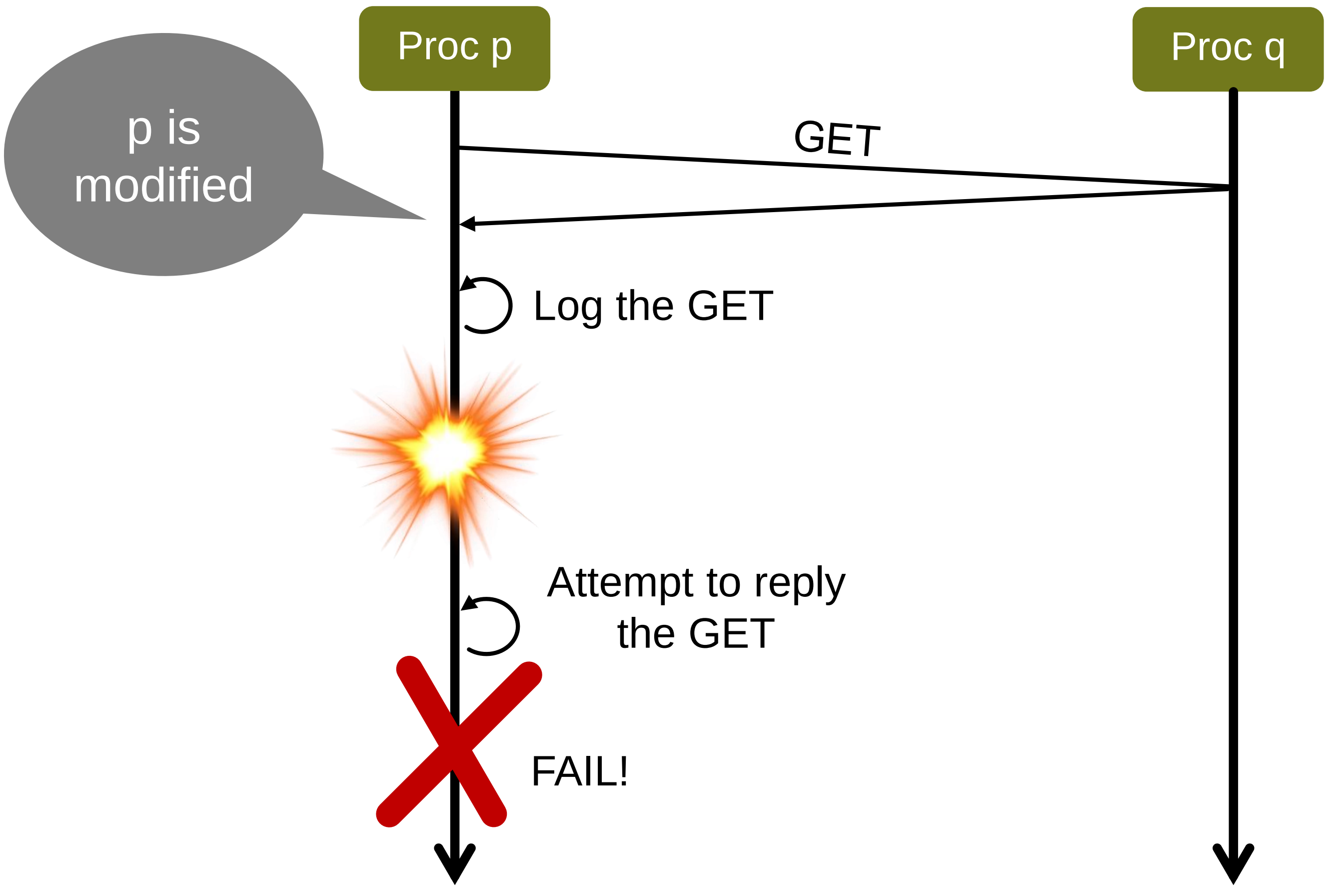
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

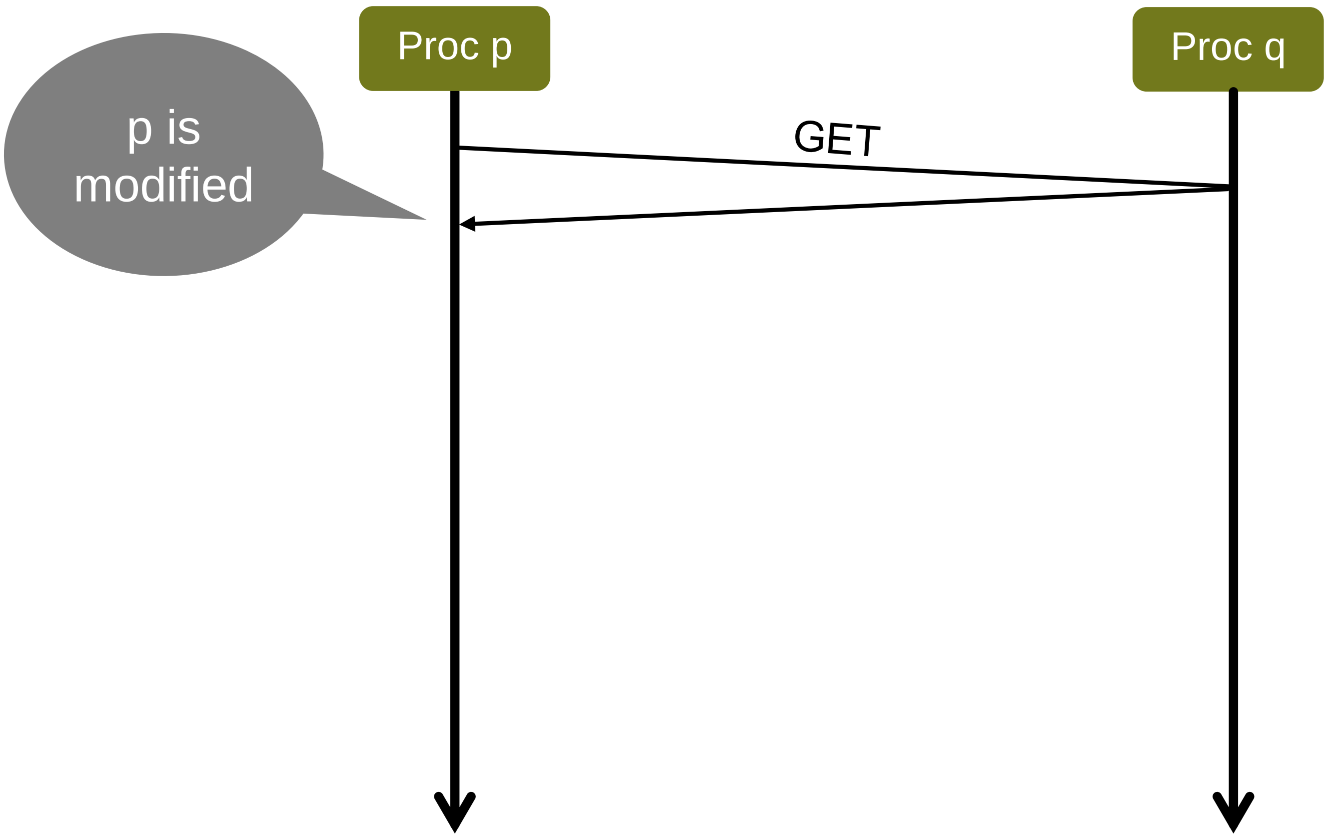
- Logging gets (naive):



# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

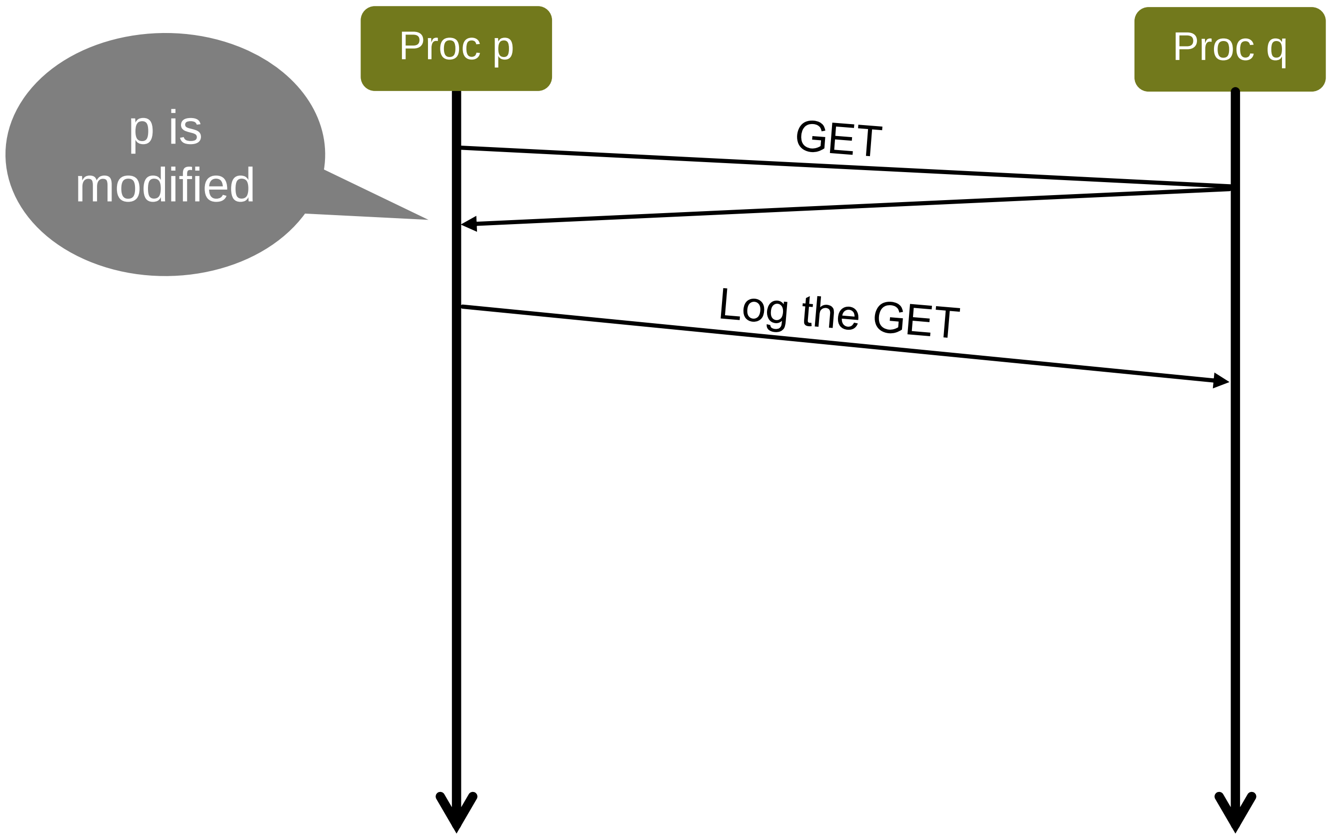



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

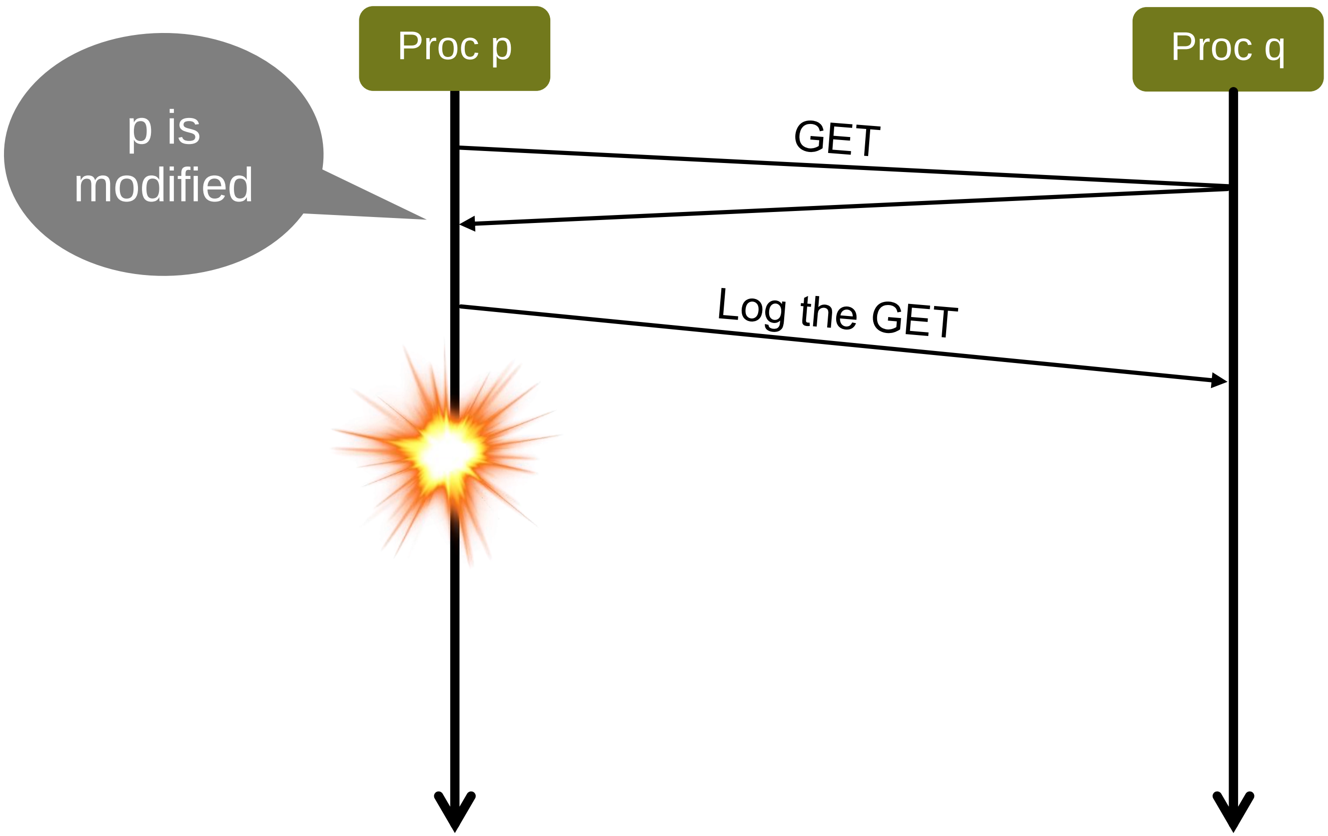



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

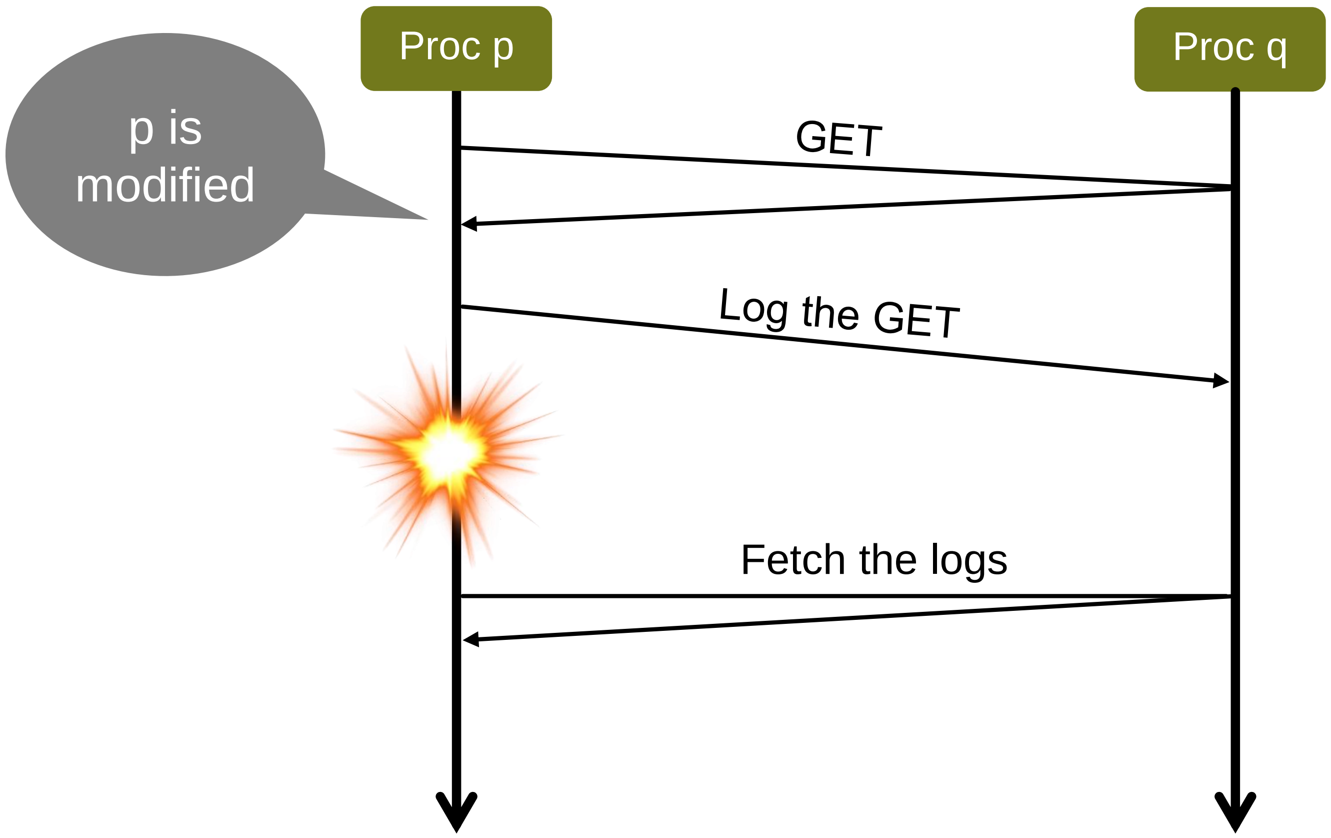



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

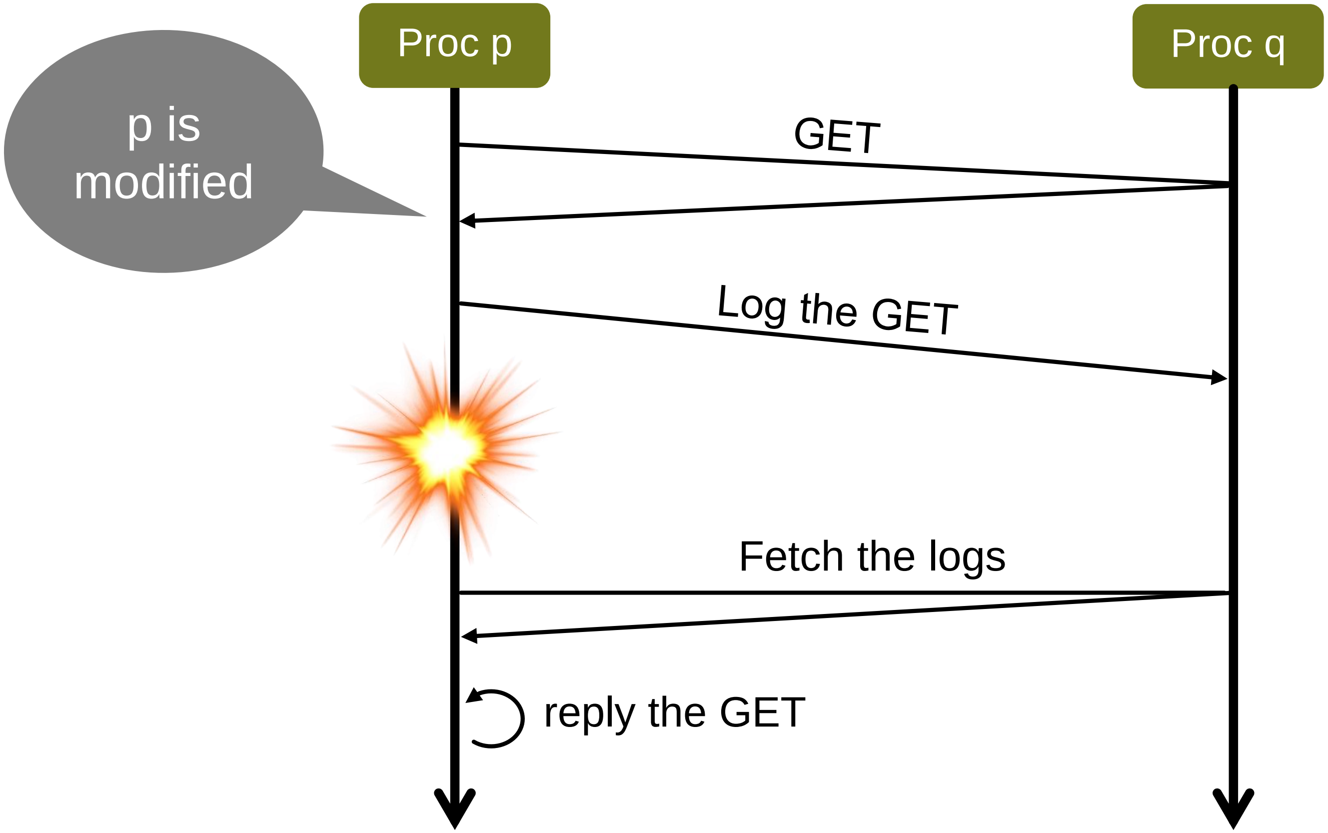



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

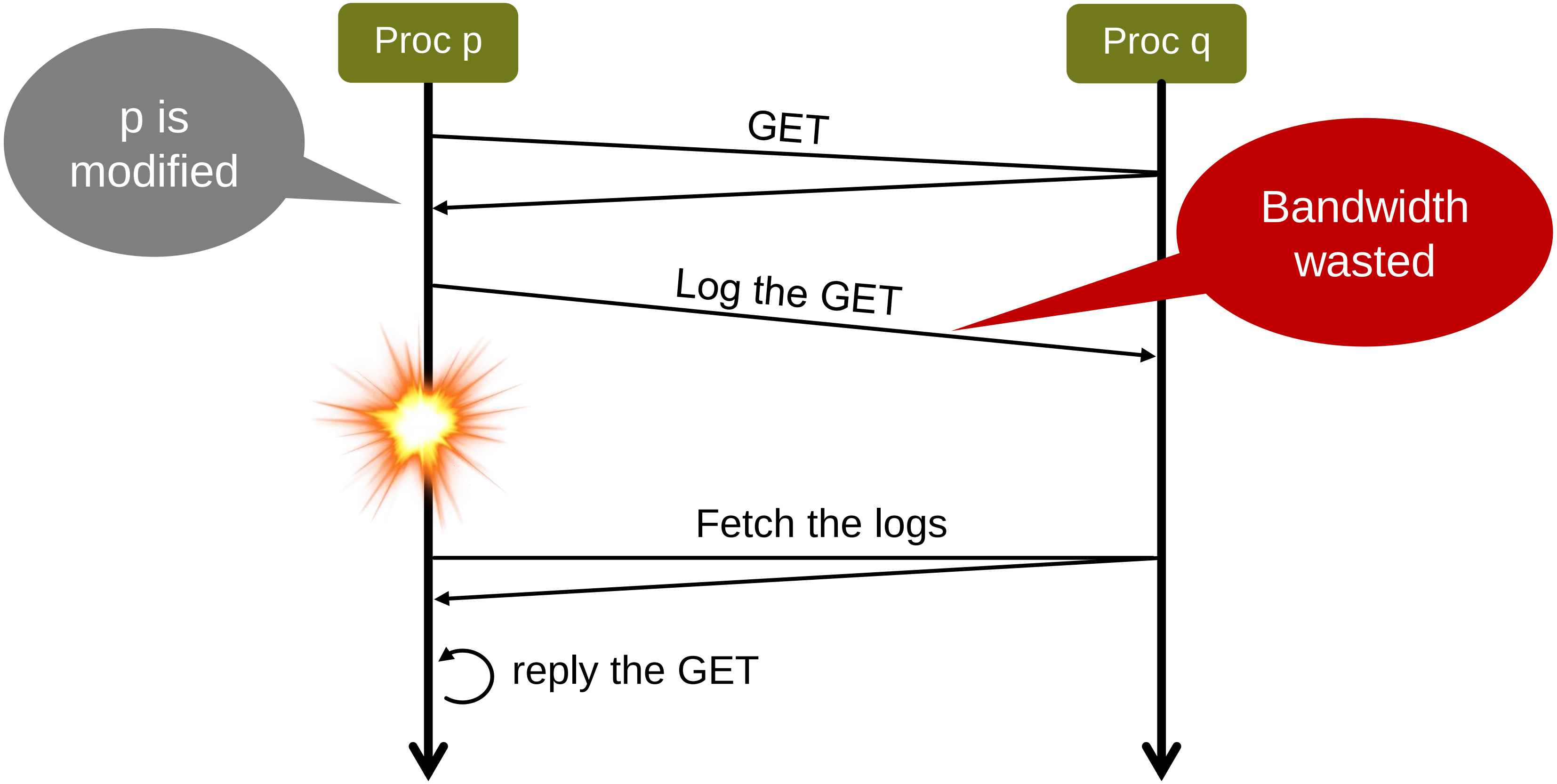



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

- Logging gets (traditional) [1]:

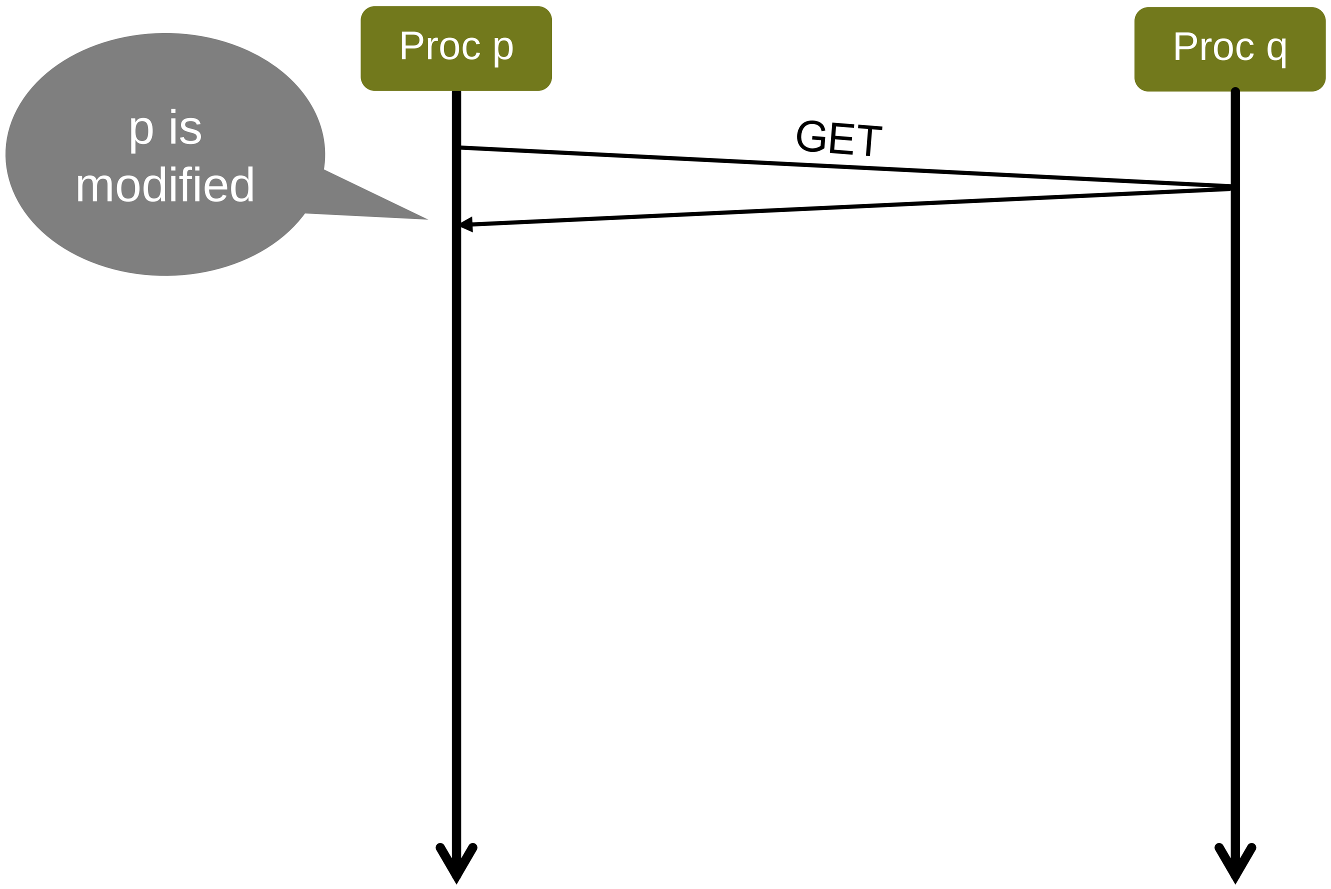



[1] M. Besta and T. Hoefler. Fault tolerance for remote memory access programming models. HPDC'14.

# Active Access Use-Cases

## Accelerating logging for RMA

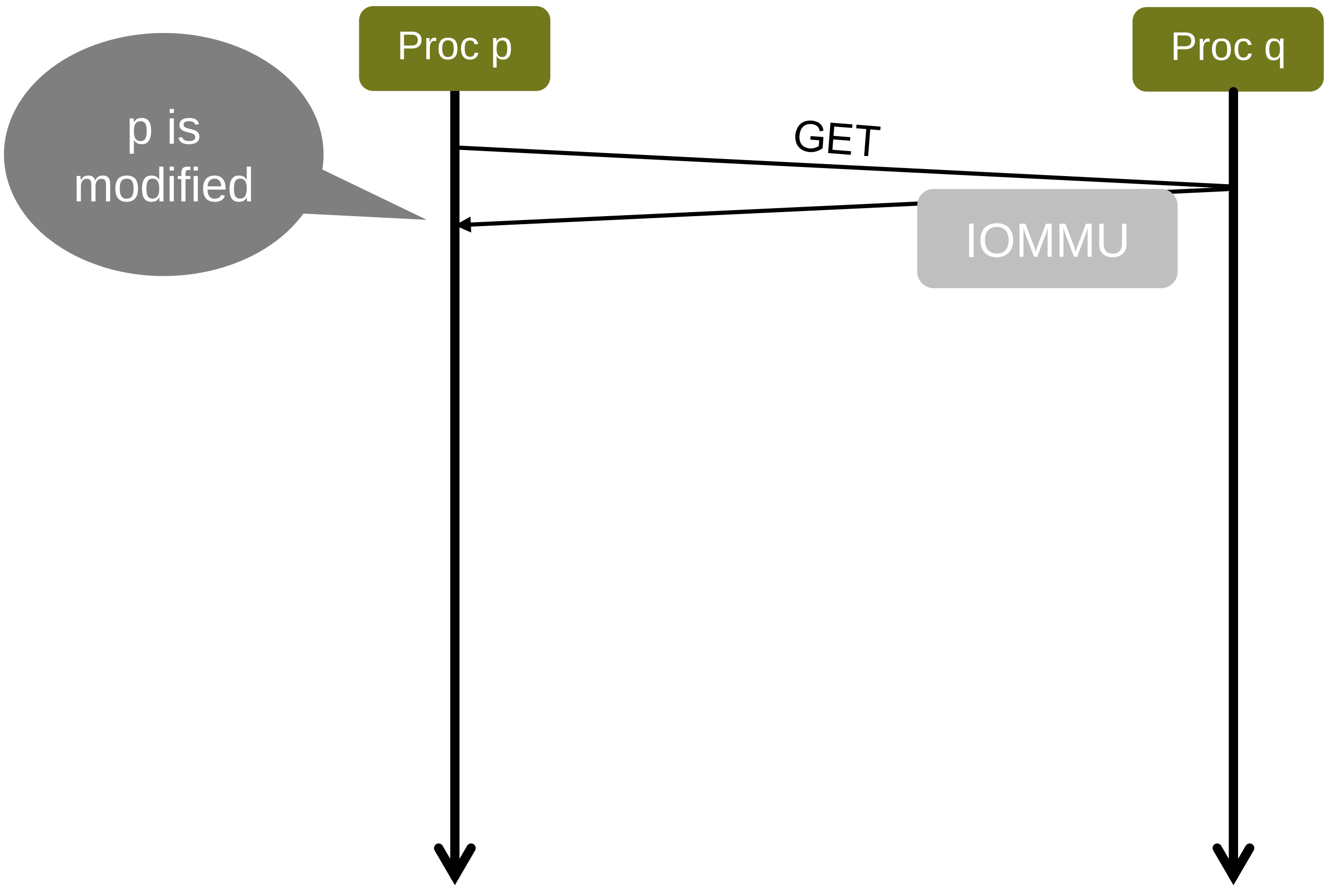
- Logging gets (AA):



# Active Access Use-Cases

## Accelerating logging for RMA

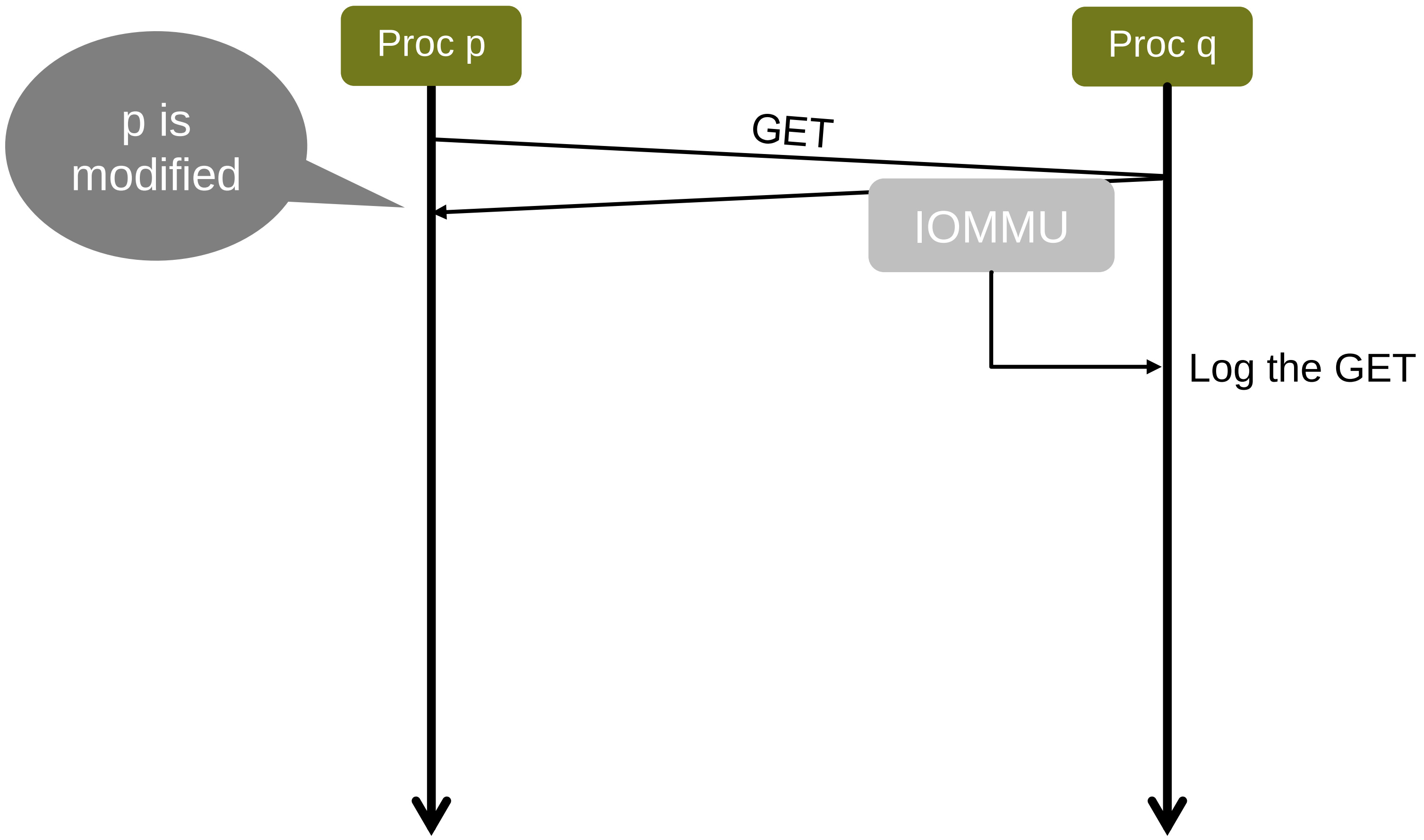
- Logging gets (AA):



# Active Access Use-Cases

## Accelerating logging for RMA

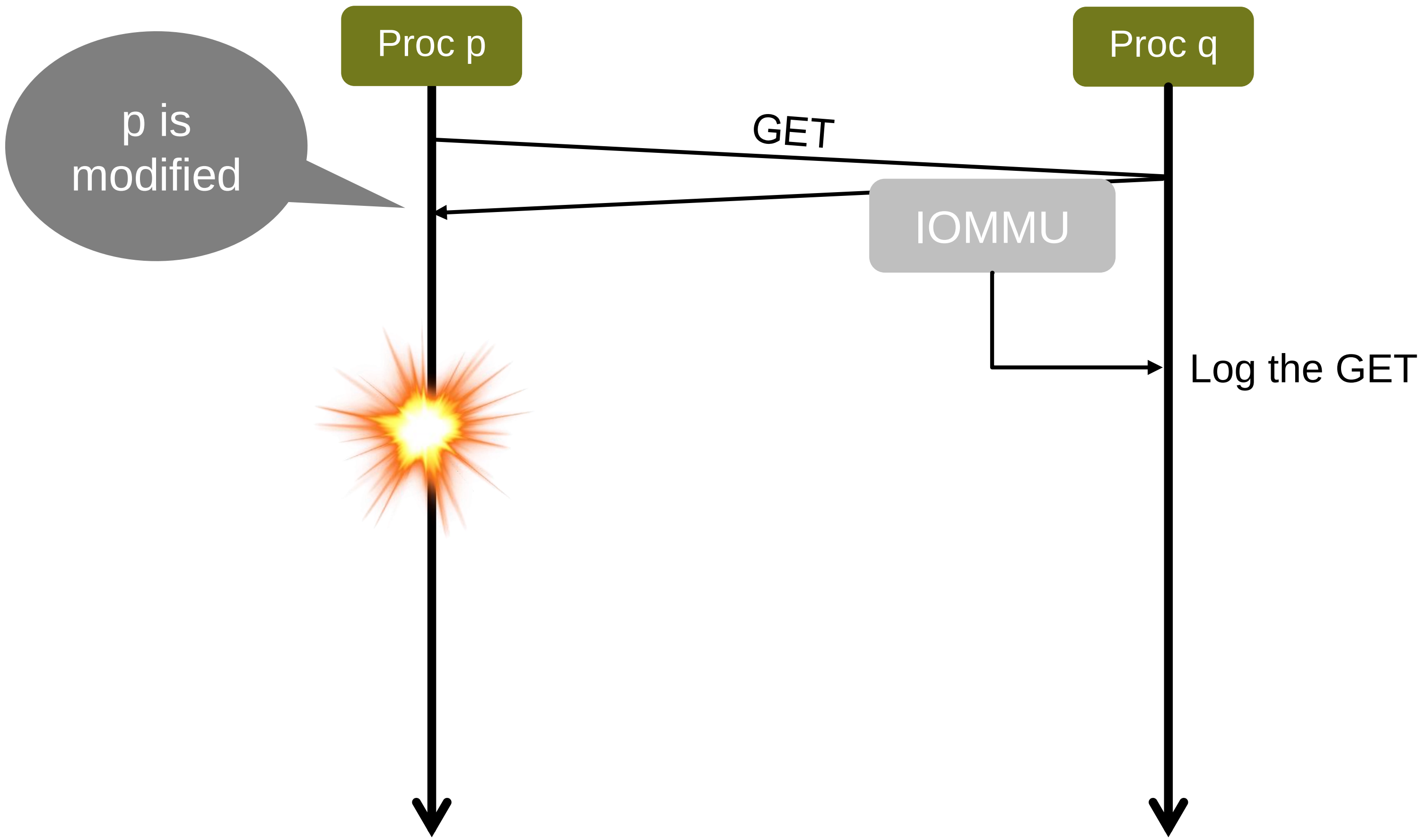
- Logging gets (AA):



# Active Access Use-Cases

## Accelerating logging for RMA

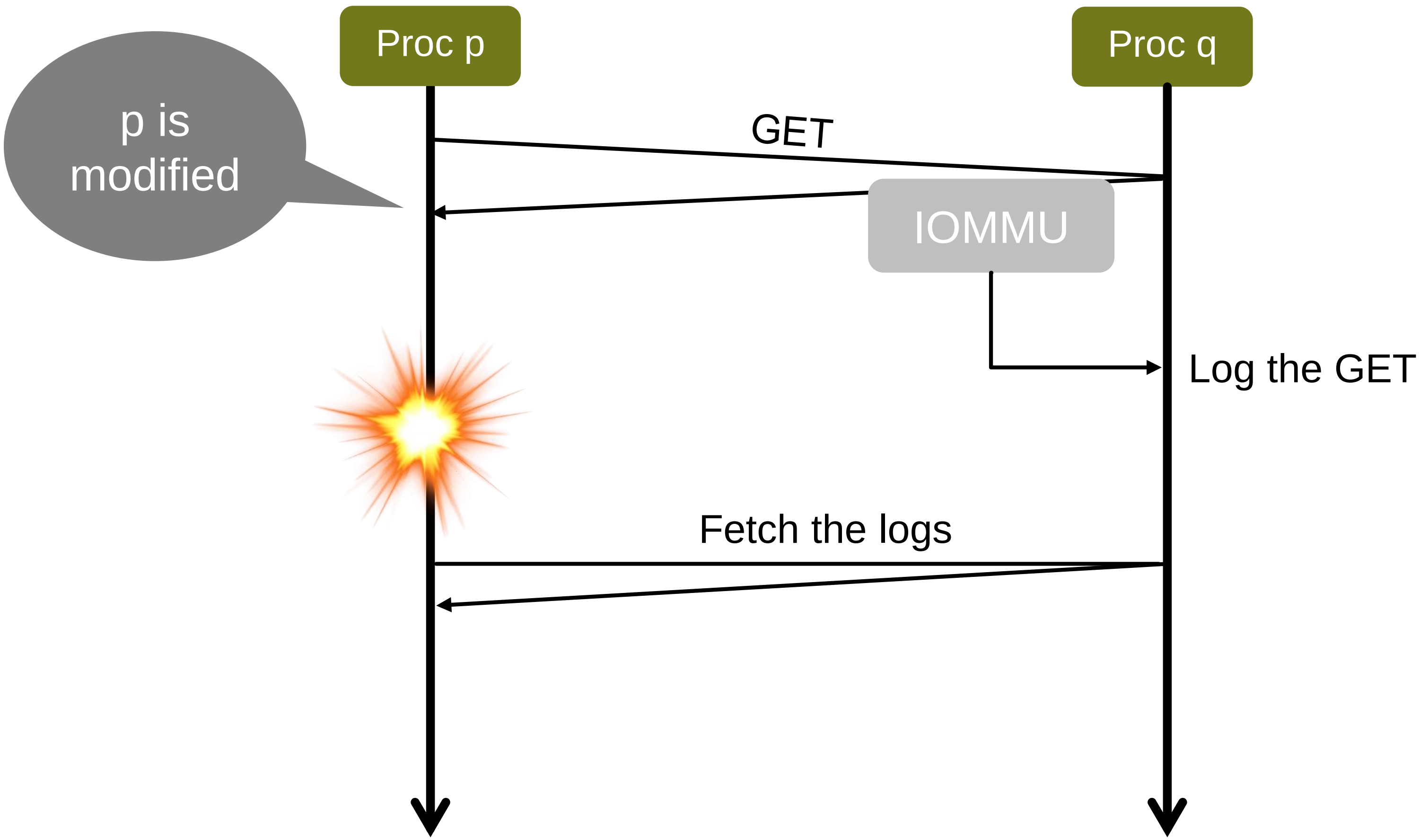
- Logging gets (AA):



# ACTIVE ACCESS USE-CASES

## ACCELERATING LOGGING FOR RMA

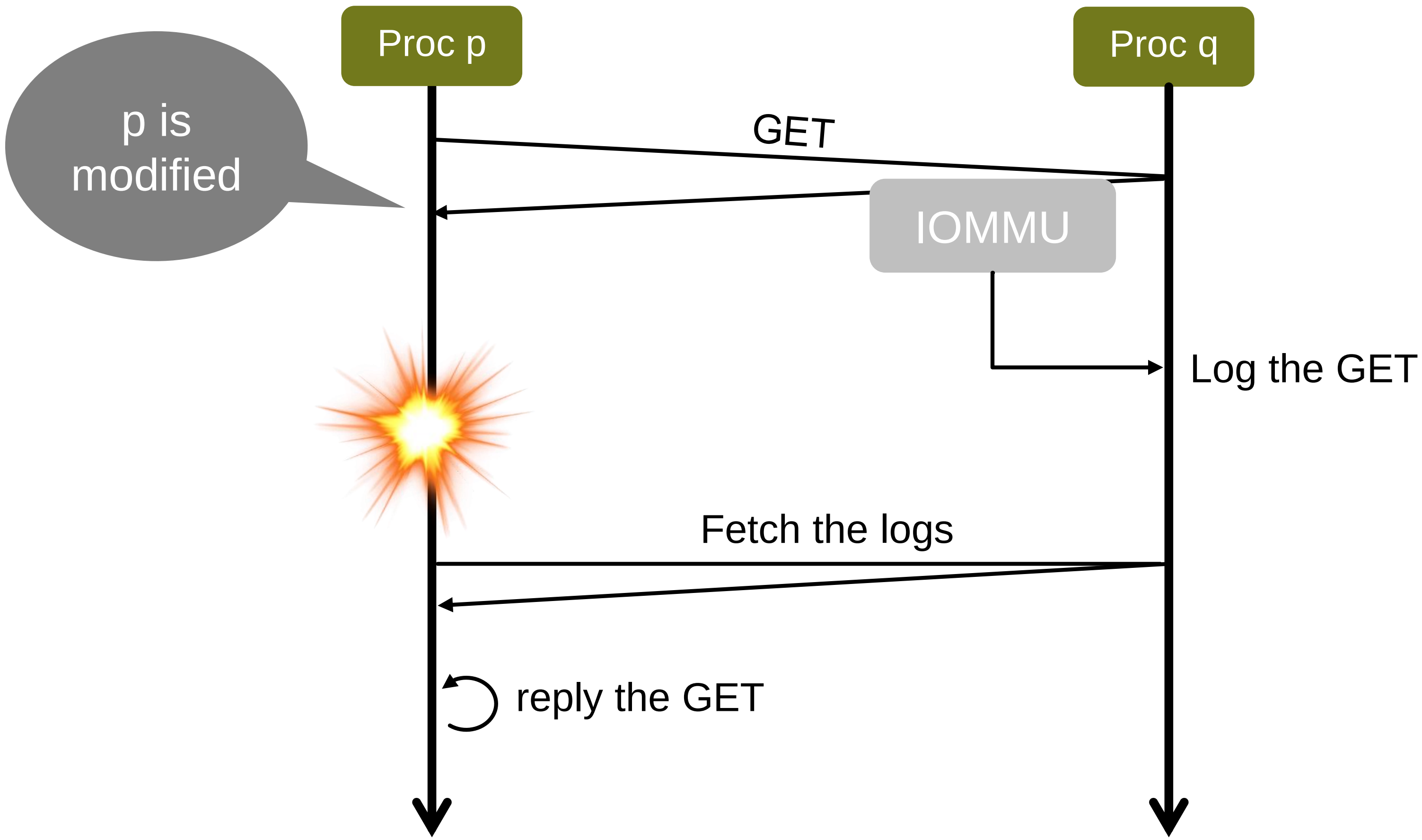
- Logging gets (AA):



# ACTIVE ACCESS USE-CASES

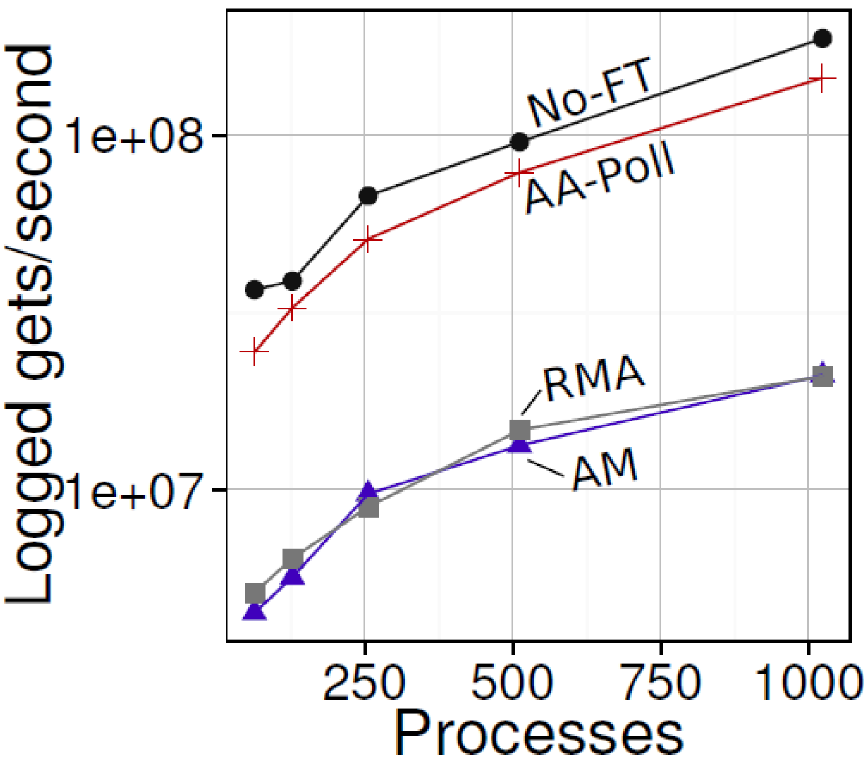
## ACCELERATING LOGGING FOR RMA

- Logging gets (AA):



# Performance: Large-Scale Codes
## Fault Tolerance Scheme

Logging gets:




Sorting time: