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

Process p

Memory

A

Process q

Memory

B
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

A

B

Cray BlueWaters
REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

B

get B

Process q

Memory

A

B

put A

Cray
BlueWaters
REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p
Memory
A
B

Process q
Memory
A
B

A put
get B
flush

Cray
BlueWaters
REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

B

Process q

Memory

A

B

put

get

flush

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.:

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

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

RMA vs. Message Passing

RMA:

Process p

Memory

A

A put

flush

Memory

Process q
**RMA vs. Message Passing**

**RMA:**
- Process p
- Memory
- A
- put
- flush

**Message Passing:**
- Process q
- Memory
- A
RMA vs. Message Passing

RMA:

- Process p
  - Memory
  - A put
  - flush

Message Passing:

- Process p
  - Memory
  - A message

- Process q
  - Memory
  - A
RMA vs. Message Passing

- Communication in RMA is one-sided

**RMA:**

- Process p
- Memory

**Process q**

- Memory

**Message Passing:**

- Process p
- Memory

**Process q**

- Memory

A put

flush

A message
RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:

```
Process p
Memory
```

```
Process q
Memory
```

Message Passing:

```
Process p
Memory
```

```
Process q
```

`put` `flush` `message`
RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:
Process p
- put
- Memory

Message Passing:
Process p
- Message
- Memory

Process q
- put
- Memory
- flush

no active participation, direct access to memory
RMA vs. Message Passing

- Communication in RMA is one-sided

RMA:

Process p
Memory

A

put

flush

A put

Process q
Memory

Message Passing:

Process p
Memory

A

send

A message

Process q
Memory
REMOTE MEMORY ACCESS PROGRAMMING

REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal?

REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal?  
  NO!

REMOTE MEMORY ACCESS PROGRAMMING

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

REMOTE MEMORY ACCESS PROGRAMMING

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

REMOTE MEMORY ACCESS PROGRAMMING

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

---

REMOTE MEMORY ACCESS PROGRAMMING

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

No hash collision:

⇒ 1 remote atomic

REMOTE MEMORY ACCESS PROGRAMMING

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

No hash collision:

⇒ 1 remote atomic

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]

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:

---
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

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

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

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

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

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

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

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

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

---

REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal?
- Consider an insert in a distributed hashtable...

How to enable it?

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

Proc p

Active access

Proc q

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

REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal?
- Consider an insert in a distributed hashtable...

How to enable it?

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

Use “active” semantics

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

REMOTE MEMORY ACCESS PROGRAMMING

- Is it ideal?
- Consider an insert in a distributed hashtable...

How to enable it?

- No hash collision:
  - 1 remote atomic
  - Up to 5x speedup over MP [1]

Use and extend I/O MMUs and their paging capabilities

Use “active” semantics

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

“ACTIVE” REMOTE MEMORY ACCESS PROGRAMMING
Local execution; triggered by an active access. In RMA?
Local execution; triggered by an active access. In RMA?

How to enable it?
“ACTIVE” REMOTE MEMORY ACCESS PROGRAMMING

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

How to enable it?

Use “active” semantics
“ACTIVE” REMOTE MEMORY ACCESS PROGRAMMING

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

How to enable it?

Use “active” semantics

Use and extend I/O MMUs and their paging capabilities
USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

USE SEMANTICS FROM ACTIVE MESSAGES (AM) [1]

AM++ [2]
GASNet [3]

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.

We need active puts/gets:
- Invoke a handler upon accessing a given page
- Preserve one-sided RMA behavior

USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

MMU

CPU
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

MMU

Virtual addresses

CPU
Use Input/Output Memory Management Units

Main memory

Physical addresses

MMU

Virtual addresses

CPU
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

- Physical addresses
  - MMU
  - TLB
  - CPU

Virtual addresses
Use Input/Output Memory Management Units

Main memory

IOMMU

MMU

I/O devices

Physical addresses

Virtual addresses

CPU

TLB
Use Input/Output Memory Management Units

Main memory

IOMMU

Device addresses

I/O devices

MMU

Physical addresses

CPU

Virtual addresses

TLB
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

Physical addresses

IOMMU

Device addresses

I/O devices

Physical addresses

MMU

Virtual addresses

CPU

TLB
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

Physical addresses

IOMMU

Device addresses

IOTLB

I/O devices

Physical addresses

MMU

Virtual addresses

TLB

CPU
**USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS**

- **Main memory**
  - Physical addresses
    - IOMMU
      - Device addresses
        - I/O devices
    - MMU
      - Virtual addresses
        - CPU
      - TLB
  - Physical addresses

- **CPU**
- **I/O devices**
- **MMU**
- **IOMMU**
- **IOTLB**
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

- Physical addresses
- IOMMU
  - Device addresses
  - IOTLB
  - I/O devices

- Physical addresses
- MMU
  - Virtual addresses
  - TLB
  - CPU

Virtual addresses

Physical addresses
**USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS**

- **Main memory**
  - Physical addresses
  - **IOMMU**
    - Device addresses
    - **IOTLB**
  - I/O devices
  - Physical addresses
  - **MMU**
    - Virtual addresses
    - **TLB**
  - CPU

- **CPU**
  - **TLB**
  - Virtual addresses
  - **MMU**
    - Physical addresses
    - **IOMMU**
      - Device addresses
      - **IOTLB**
    - Physical addresses
  - I/O devices
  - **IOMMU**
  - **IOTLB**

- **Logos**
  - AMD
  - IBM
  - ARM
  - Intel
  - Sun Microsystems
  - Solarflare
USE INPUT/OUTPUT MEMORY MANAGEMENT UNITS

Main memory

IOMMU
- Physical addresses
- Device addresses
- I/O devices

MMU
- Physical addresses
- Virtual addresses
- CPU

IOTLB

TLB

CPU

Virtual addresses

Device addresses

Physical addresses
We propose it as a way to implement the “active” behavior.
IOMMUs AND RMA
IOMMUs AND RMA

NIC
IOMMUs AND RMA

NIC

IOMMU
IOMMUs and RMA

NIC → IOMMU → MSI → CPU
IOMMUs and RMA
IOMMUs AND RMA

NIC

IOMMU

MSI

CPU

SMT cores

Main memory
IOMMU S AND RMA

IOMMU

NIC

Main memory

Remapping structures

MSI

SMT cores

CPU
IOMMUs AND RMA

NIC → IOMMU → MSI → CPU

Main memory

Remapping structures

Dev-to-PT
IOMMUs and RMA

NIC

Main memory

Remapping structures
- Dev-to-PT
- PT

IOMMU

MSI

CPU

SMT cores
IOMMUs AND RMA

An RDMA packet

Nic

1

IOMMU

MSI

CPU

SMT cores

Main memory

Remapping structures

Dev-to-PT

PT
IOMMUs and RMA

An RDMA packet

1

NIC

PCIe packets

2

Main memory

Remapping structures

Dev-to-PT

PT

IOMMU

MSI

CPU

SMT cores

MSI

1

2
IOMMU and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU

NIC

PCIe packets

Main memory

Remapping structures

- Dev-to-PT
- PT

IOMMU

MSI

SMT cores

CPU

Spcl.inf.ethz.ch
@spcl_eth
ETH Zürich
IOMMUS AND RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache

Main memory

Remapping structures
- Dev-to-PT
- PT

MSI

CPU
- SMT cores
An RDMA packet \[\rightarrow\] NIC \[\rightarrow\] PCIe packets \[\rightarrow\] Main memory \[\rightarrow\] IOMMU

IOMMU

Dev-to-PT cache \[\rightarrow\] MSI \[\rightarrow\] CPU

Remapping structures

Dev-to-PT \[\rightarrow\] PT

NIC

SMT cores

PCIe packets

Main memory

IOMMUs AND RMA
IOMMU and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB

Main memory

MSI

CPU

SMT cores
IOMMUs and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT

Main memory

MSI

CPU

SMT cores
**IOMMMUs AND RMA**

1. An RDMA packet
2. PCIe packets
3. NIC
4. IOMMU
5. Remapping structures
6. IOTLB
7. PT

**Main memory**

**CPU**

**IOMMU**

- Dev-to-PT cache
- Dev-to-PT
- SMT cores

**MSI**
IOMMRUS AND RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT

Main memory

MSI

IOMMU

CPU

SMT cores
IOMMUs and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. W R

Main memory

MSI

CPU

SMT cores
IOMMUS AND RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. IOMMU
5. Remapping structures
6. IOTLB
7. PT
8. System-wide fault log

MSI

CPU

SMT cores

Main memory

Dev-to-PT cache

PCIe packets

Dev-to-PT

Remapping structures

System-wide fault log

W
R
IOMMUS AND RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. System-wide fault log

Main memory

CPU

SMT cores

MSI
IOMMUs and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. Dev-to-PT
9. System-wide fault log
IOMMUs and RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. IOMMU
5. Dev-to-PT
6. IOTLB
7. PT
8. Remapping structures
9. System-wide fault log
10. MSI
IOMMUS AND RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. W
9. System-wide fault log
10. MSI
11. SMT cores

Main memory

Fault entry → ... → Fault entry
IOMMU and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. W R
9. System-wide fault log
10. MSI
11. CPU

Main memory

Remapping structures
- Dev-to-PT
- PT

System-wide fault log
- Fault entry → ... → Fault entry

User handlers
- Handler A
- ...
IOMMU and RMA

1. An RDMA packet
2. PCIe packets
3. IOMMU
4. Dev-to-PT cache
5. Remapping structures
6. IOTLB
7. PT
8. W R
9. System-wide fault log
10. MSI
11. CPU
12. User handlers

NIC
Main memory

Dev-to-PT
Fault entry → ... → Fault entry

Handler A
...
IOMMU AND RMA

An RDMA packet

NIC

PCIe packets

Main memory

Remapping structures

Dev-to-PT

PT

System-wide fault log

Fault entry

User handlers

Handler A

We could use it somehow. But…
IOMMU and RMA

An RDMA packet

1. NIC

2. PCIe packets

3. IOMMU

4. Dev-to-PT cache

5. Remapping structures

6. IOTLB

7. PT

8. Dev-to-PT

9. System-wide fault log

10. MSI

11. CPU

12. User handlers

No parallelism (single log)... BAD

We could use it somehow. But...
We could use it somehow. But…

IOMMUs and RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. IOMMU
5. Remapping structures
6. IOTLB
7. PT
8. User handlers
9. System-wide fault log
10. MSI
11. CPU
12. Handler A

- No parallelism (single log)... BAD
- No multiplexing (single log)... BAD

Main memory
IOMMU S AND RMA

1. An RDMA packet
2. PCIe packets
3. NIC
4. IOMMU
5. Remapping structures
6. IOTLB
7. PT
8. Dev-to-PT
9. System-wide fault log
10. MSI
11. No multiplexing (single log)... BAD
12. User handlers
   - Handler A
   - ...

Data is discarded... Extremely BAD
No parallelism (single log)... BAD

Main memory

We could use it somehow. But…
**ACTIVE PUTS**

An RDMA packet

- NIC
- PCIe packets
- Main memory

IOMMU

- Dev-to-PT cache
- IOTLB

MSI

CPU

- SMT cores

System-wide fault log

Remapping structures

- Dev-to-PT
- PT

User handlers

Handler A

...
**ACTIVE PUTS**

- An RDMA packet
  - NIC
  - IOMMU
    - Dev-to-PT cache
    - IOTLB
  - Main memory
  - PCIe packets
  - Remapping structures
    - Dev-to-PT
    - PT

- MSI
  - CPU
    - SMT cores
      - User handlers
        - Handler A

- Fault entry → ... → Fault entry
- Access log (private for each process)
ACTIVE PUTS

- An RDMA packet
- PCIe packets
- NIC
- IOMMU
- Dev-to-PT cache
- IOTLB
- Main memory
- MSI
- System-wide fault log
- User handlers
- Fault entry → ... → Fault entry
- Request data → ... → Request data
- Access log (private for each process)
- SMT cores
- CPU
Active Puts

An RDMA packet

NIC

PCIe packets

IOMMU

Dev-to-PT cache

IOTLB

Main memory

MSI

CPU

SMT cores

System-wide fault log

Fault entry → ... → Fault entry

Access log (private for each process)

Fault entry → ... → Fault entry

Request data

Request data

Remapping structures

Dev-to-PT

PT

Data can be reused

User handlers

Handler A

...
**Active PUTS**

- An RDMA packet
- NIC
- PCIe packets
- NIC
- IOMMU
- Dev-to-PT cache
- IOTLB
- Access log table
- MSI
- System-wide fault log
- Sense-Error
- Fault entry
- Fault entry
- Request data
- Access log table
- Data can be reused

**Remapping structures**
- Dev-to-PT
- PT

**CPU**
- SMT cores
**ACTIVE PUTS**

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

An RDMA packet

NIC

PCIe packets

Main memory

IOMMU

Dev-to-PT cache

IOTLB

Access log table

Stores addresses of each access log

MSI

CPU

SMT cores

Remapping structures

Dev-to-PT

PT

Access log table

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

An RDMA packet

NIC

PCIe packets

IOMMU

Dev-to-PT cache

IOTLB

Access log table

Main memory

Remapping structures

Dev-to-PT

PT

Stores addresses of each access log

MSI

CPU

SMT cores

User handlers

Handler A

Fault entry

Fault entry

Fault entry

Fault entry

Fault entry

Data can be reused

Access log (private for each process)

Request data

Request data

System-wide fault log

Access log table

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD

Stores addresses of each access log

W

R

WL

WLD
**ACTIVE PUTS**

- An RDMA packet
- PCIe packets
- NIC
- Main memory
- Access log table
- Remapping structures
  - Dev-to-PT
  - PT
  - Access log (private for each process)
  - System-wide fault log
    - Fault entry
    - Request data
  - User handlers
    - Handler A

**Stores addresses of each access log**

**Decide on keeping/discarding the entry/data**

**Data can be reused**
**ACTIVE PUTS**

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

**Stores addresses of each access log**

**Decide on keeping/discarding the entry/data**

**Data can be reused**
ACTIVE PUTS

Stores addresses of each access log

Dev-to-PT cache

IOMMU

MSI

CPU

SMT cores

Dev-to-PT

Remapping structures

Access log table

Decide on keeping/discarding the entry/data

Main memory

An RDMA packet

NIC

PCIe packets

IOTLB

Maps each page to an access log

System-wide fault log

User handlers

Handler A

Data can be reused

Access log (private for each process)

Access log table

Fault entry → ... → Fault entry

Fault entry → ... → Fault entry

Fault entry

Request data

Request data

W
R
WL
WLD

IUID

+ + + + + +

Data can be reused
**ACTIVE PUTS**

- An RDMA packet
- PCIe packets
- NIC
- IOMMU
  - Dev-to-PT cache
  - IOTLB
- Main memory
- Access log table
- Remapping structures
  - Dev-to-PT
  - PT
- User handlers
  - Handler A
- SMT cores

Stores addresses of each access log

Decide on keeping/discarding the entry/data

Enables data-centric programming

Maps each page to an access log

Data can be reused

- Access log (private for each process)
- System-wide fault log
- Fault entry
- Fault entry
- Request data
ACTIVE PUTS
ACTIVE PUTS

Process p

Process q

IOMMU

CPU

Main memory
ACTIVE PUTS

Process p

Process q

IOMMU

Accessed page

Access log

CPU

Main memory
ACTIVE PUTS

Process q

IOMMU

Accessed page

W = 0
WL = 1
WLD = 1

Access log

CPU

Main memory
 ACTIVE PUTS

Process p

Process q

IOMMU

Accessed page

W = 0
WL = 1
WLD = 1

Access log

CPU

Main memory

Do not modify the page
**ACTIVE PUTS**

Do not modify the page

Log both the entry and the data of an incoming put

Process q

Accessed page

Access log

W = 0
WL = 1
WLD = 1

Main memory

Process p

IOMMU

CPU
ACTIVE PUTS

Process p

1 Put(X)

IOMMU

Process q

W = 0
WL = 1
WLD = 1

Accessed page

Access log

CPU

Main memory

Do not modify the page

Log both the entry and the data of an incoming put
ACTIVE PUTS

Process p

1. Put(X)

Process q

2. Attempt to write(X)

IOMMU

Accessed page

W = 0
WL = 1
WLD = 1

Access log

CPU

Main memory

Do not modify the page

Log both the entry and the data of an incoming put
**ACTIVE PUTS**

Process $p$ sends a `Put(X)` request to Process $q$.

1. The request reaches the IOMMU.
2. Process $q$ attempts to write `X`.
3. A page fault occurs because the page is accessed with $W = 0$, $WL = 1$, and $WLD = 1$.

Log both the entry and the data of an incoming put.

Do not modify the page.

Access log:

- $W = 0$
- $WL = 1$
- $WLD = 1$
ACTIVE PUTS

Process p

1. Put(X)

Process q

2. Attempt to write(X)

Page fault!

3. (W = 0)

W = 0
WL = 1
WLD = 1

Accessed page

Access log

Main memory

Do not modify the page

Log both the entry and the data of an incoming put

CPU

IOMMU

Accessed page

Attempt to write(X)

Move(X)
**Active Puts**

- **Process q**
  - Attempt to **write**(*X*)
  - Page fault! (**W = 0**)  
  - Access log
  - Move(*X*)
  - Accessed page
    - **W** = 0
    - **WL** = 1
    - **WLD** = 1

- **Process p**
  - **Put**(*X*)
  - IOMMU

- **CPU**
  - **Main memory**

- **Log both the entry and the data of an incoming put**

- **Do not modify the page**
**Active Puts**

1. Process p

2. Attempt to write(X)

3. Page fault! (W = 0)

4. Move(X)

5. Process(X)

- 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

Main memory
ACTIVE GETS

An RDMA packet
NIC
PCIe packets
Main memory

IOMMU
Dev-to-PT cache
IOTLB

Access log table

MSI
CPU
SMT cores

Remapping structures
Dev-to-PT
PT

Access log table

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**

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

Process p

Process q

IOMMU

CPU

Main memory
**ACTIVE GETS**

Process p

Process q

IOMMU

Accessed page

Access log

CPU

Main memory
ACTIVE GETS

Process p

Process q

IOMMU

Accessed page

Access log

Main memory

CPU

R = 1
RL = 1
RLD = 1
ACTIVE GETS

Enable reading from the page

Process q

IOMMU

Accessed page

R = 1
RL = 1
RLD = 1

Access log

CPU

Main memory

Process p
ACTIVE GETS

Enable reading from the page

Log both the entry and the data accessed by a get

Process q

IOMMU

Accessed page

R = 1
RL = 1
RLD = 1

Access log

Main memory
Active Gets

Enable reading from the page

Log both the entry and the data accessed by a get

1. Get(X)

Process p

IOMMU

Accessed page

R = 1
RL = 1
RLD = 1

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

Access log

Main memory
ACTIVE GETS

Enable reading from the page

Log both the entry and the data accessed by a get

Process p

1 Get(X)

Process q

IOMMU

2 Read(X)

Accessed page

Access log

CPU

R = 1
RL = 1
RLD = 1

Main memory
ACTIVE GETS

Process p

1. Get(X)

IOMMU

2. Read(X)

Accessed page

Process q

Enable reading from the page

Log both the entry and the data accessed by a get

Access log

Main memory

CPU

Accessed page

R = 1
RL = 1
RLD = 1

Log both the entry and the data accessed by a get
**ACTIVE GETS**

Enable reading from the page

Log both the entry and the data accessed by a get

- **Process p**
  - 1. Get(X)
  - 2. Read(X)
  - 3. Copy(X)

- **IOMMU**
  - **Accessed page**
    - $R = 1$
    - $RL = 1$
    - $RLD = 1$

- **CPU**
  - **Main memory**
  - **Access log**
    - X
**ACTIVE GETS**

Enable reading from the page

Log both the entry and the data accessed by a get

---

1. **Get(X)**

   Process p

2. **Read(X)**

   IOMMU

3. **Copy(X)**

   Access log

4. **Process(X)**

   Main memory

---

Accessed page

R = 1
RL = 1
RLD = 1

Copy(X)

Process(X)

- CPU

---

1. Process p

2. IOMMU

3. Access log

4. Main memory
**ACTIVE GETS**

Sounds like we can reuse most of the existing stuff!
INTERACTIONS WITH THE CPU

An RDMA packet → NIC → IOMMU → Dev-to-PT cache → IOTLB → Main memory

PCIe packets

MSI

CPU

SMT cores

Access log table

Remapping structures

Dev-to-PT

PT

Access log (private for each process)

Request data

System-wide fault log

Fault entry → … → Fault entry

User handlers

Handler A

…
INTERACTIONS WITH THE CPU

IOMMU

- Dev-to-PT cache
- IOTLB

Access log table

MSI

CPU

SMT cores
INTERACTIONS WITH THE CPU

IOMMU

Dev-to-PT cache

IOTLB

Access log table

MSI

CPU

SMT cores
INTERACTIONS WITH THE CPU

- Interrupts

IOMMU

- Dev-to-PT cache
- IOTLB

Access log table

MSI

CPU

SMT cores

...
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

![Diagram showing interactions between IOMMU, MSI, CPU, and various memory and cache components.](image-url)
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

Are we done?
INTERACTIONS WITH THE CPU

- Interrupts
- Polling
- Direct notifications via scratchpads

Are we done?

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*

**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

IOMMU

Dev-to-PT cache

IOTLB

Access log table

MSI

CPU

SMT cores

Scratchpad memory

Handler A

Hyper thread

+ + +
CONSISTENCY

IOMMU

Dev-to-PT cache

IOTLB

Access log table

Flushing buffer

MSI

CPU

Scratchpad memory

Handler A

Hyper thread

SMT cores

Hyper thread
**CONSISTENCY**

- **IOMMU**
  - Dev-to-PT cache
  - IOTLB
  - Access log table
  - Flushing buffer

- **MSI**
  - Scratches memory

- **CPU**
  - SMT cores
  - Handler A
  - Hyper thread

Contains the addresses of flushing pages
**CONSISTENCY**

**IOMMU**
- Dev-to-PT cache
- IOTLB
- Access log table
- Flushing buffer

**CPU**
- SMT cores
- Scratchpad memory
- Handler A
- Hyper thread

**MSI**

Contains the addresses of flushing pages

Maps flushing pages to IUIDs and access logs
CONSISTENCY

IOMMU

- Dev-to-PT cache
- Packet tag buffer
- Access log table
- Flushing buffer

IOTLB

Maps flushing pages to IUIDs and access logs

Contains the addresses of flushing pages

MSI

CPU

SMT cores

Scratchpad memory

Handler A

Hyper thread

Maps flushing pages to IUIDs and access logs

Contains the addresses of flushing pages
Let’s summarize…
Let’s summarize…

Active Messages
Let's summarize...

**Active Messages**

We need active put/gets:
- Initiate a handler upon accessing a given page
- Preserve one-sided RMA behavior

**IOMMUs**

<table>
<thead>
<tr>
<th>Use Input/Output Memory Management Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physical addresses</strong></td>
</tr>
<tr>
<td>IOMMU</td>
</tr>
<tr>
<td>MMU</td>
</tr>
<tr>
<td>Device addresses</td>
</tr>
<tr>
<td>IOTLB</td>
</tr>
<tr>
<td>Virtual addresses</td>
</tr>
<tr>
<td>TLB</td>
</tr>
<tr>
<td>I/O devices</td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
</tr>
</tbody>
</table>

**USE SEMANTICS FROM ACTIVE MESSAGES (AM)**

- AM+ [2]
- GASNet [3]

- IBM
- Myricom

Process p

Memory
- A's addr
- Handler A
- Z's addr
- Handler Z

Process q

- AMD
- IBM
- ARM

- Sun
- SOLARFLARE
- Intel
- PCI EXPRESS
Let’s summarize…

Active Messages

IOMMUs

Active Puts/Gets
Let’s summarize...

**Active Messages**

**IOMMUs**

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

**Active Puts/Gets**
Let’s summarize…

Active Messages

IOMMUs

Consistency

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

Active Puts/Getts

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])

ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE

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

Local volume 0 (at process 0)
Local volume 1 (at process 1)
Local volume N-1 (at process N-1)

ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE

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

Local volume 0 (at process 0)
Table of elements

Local volume 1 (at process 1)
Table of elements

Local volume N-1 (at process N-1)
Table of elements

ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE

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

---

ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (RMA)
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (RMA)
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASH TABLE: INSERTS (RMA)
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (RMA)

Proc p

CAS (insert attempt)

Proc q

FAD (get and increment ptr to the next free cell)

Table of elements

Overflow heap
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (RMA)
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (RMA)

Proc p

CAS (insert attempt)
FAD (get and increment ptr to the next free cell)
PUT (insert element)
FAD & CAS & PUT (update ptrs)

Table of elements
Overflow heap

Proc q
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (AA)

Proc p

CAS (insert attempt)
FAD (get and increment ptr to the next free cell)
PUT (insert element)
FAD & CAS & PUT (update ptrs)

Table of elements

Overflow heap

Proc q
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (AA)

Proc p

FAD (get and increment ptr to the next free cell)
PUT (insert element)
FAD & CAS & PUT (update ptrs)

Proc q

Table of elements
Overflow heap
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (AA)

Proc p

PUT (intercepted by the IOMMU)
FAD (get and increment ptr to the next free cell)
PUT (insert element)
FAD & CAS & PUT (update ptrs)

Table of elements
Overflow heap

Proc q
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (AA)

Proc p

PUT (intercepted by the IOMMU)

FAD (get and increment ptr to the next free cell)
PUT (insert element)
FAD & CAS & PUT (update ptrs)

Table of elements
Overflow heap

Proc q

All other accesses become local
ACTIVE ACCESS USE-CASES
DISTRIBUTED HASHTABLE: INSERTS (AA)

Proc p

PUT (intercepted by the IOMMU)

Proc q

Table of elements

Overflow heap
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)

Machine 0
- Proc 0
- MMU
- Memory
- NIC

Machine 1
- Proc 1
- MMU
- Memory
- NIC

Machine N-1
- Proc N-1
- MMU
- Memory
- NIC
ACTIVE ACCESS USE-CASES
VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)

Machine 0
- Proc 0
  - MMU
  - Memory
  - IOMMU
  - NIC

Machine 1
- Proc 1
  - MMU
  - Memory
  - IOMMU
  - NIC

Machine N-1
-Proc N-1
  - MMU
  - Memory
  - IOMMU
  - NIC
**ACTIVE ACCESS USE-CASES**

**VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)**
ACTIVE ACCESS USE-CASES
VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)

Machine 0
Proc 0
MMU
Memory
IOMMU
NIC

Machine 1
Proc 1
MMU
Memory
IOMMU
NIC

Machine N-1
Proc N-1
MMU
Memory
IOMMU
NIC
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

- Proc 0
- MMU
- Memory
- IOMMU
- NIC

Machine 1

- Proc 1
- MMU
- Memory
- IOMMU
- NIC

Machine N-1

- Proc N-1
- MMU
- Memory
- IOMMU
- NIC

**Local memory protection**
ACTIVE ACCESS USE-CASES
VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)

Machine 0
Proc 0
MMU
Memory
IOMMU
NIC

Machine 1
Proc 1
MMU
Memory
IOMMU
NIC

Machine N-1
Proc N-1
MMU
Memory
IOMMU
NIC

Local memory protection

V-GAS

Local memory protection

Memory

IOMMU

NIC

NIC

NIC
ACTIVE ACCESS USE-CASES
VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)

Machine 0
Proc 0
MMU
Memory
IOMMU
NIC

Machine 1
Proc 1
MMU
Memory
IOMMU
NIC

Machine N-1
Proc N-1
MMU
Memory
IOMMU
NIC

V-GAS
Local memory protection
**ACTIVE ACCESS USE-CASES**

**VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)**

Remote memory protection

Local memory protection

Machine 0

- Proc 0
- MMU
- Memory
- IOMMU
- NIC

Machine 1

- Proc 1
- MMU
- Memory
- IOMMU
- NIC

Machine N-1

- Proc N-1
- MMU
- Memory
- IOMMU
- NIC
ACTIVE ACCESS USE-CASES
VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)

Remote memory protection

Machine 0
Proc 0

Machine 1
Proc 1

Machine N-1
Proc N-1

MMU

Memory

IOMMU

NIC

MMU

Memory

IOMMU

NIC

V-GAS

Local memory protection

Remote memory protection
**ACTIVE ACCESS USE-CASES**

**VIRTUAL GLOBAL ADDRESS SPACE (V-GAS)**

Remote memory protection

Machine 0

- Proc 0
- MMU
- Memory
- IOMMU
- NIC

Machine 1

- Proc 1
- MMU
- Memory
- IOMMU
- NIC

Machine N-1

- Proc N-1
- MMU
- Memory
- IOMMU
- NIC

Local memory protection

V-GAS

Fetch data (used for logging, fault-tolerance, etc...)

Remote memory protection
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]:

  ![gem5 logo]

**Performance: Microbenchmarks**

**Raw Data Transfer**

- Workload simulated with [1]:

  ![gem5 logo](image)

- Data generated with:

  ![gem5 logo](image)

Performance: Microbenchmarks
Raw Data Transfer

- Workload simulated with [1]:

- Data generated with:
  - PktGen [2]

PERFORMANCE: MICROBENCHMARKS
RAW DATA TRANSFER

- Workload simulated with [1]:

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

**Performance: Microbenchmarks**

**Raw Data Transfer**

- Workload simulated with [1]:
  ![gem5](image)

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

---

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-Int

AA-Poll
PERFORMANCE: LARGE-SCALE CODES
COMPARISON TARGETS

Active Access
AA-Int
AA-Poll
AA-SP
## Performance: Large-Scale Codes

### Comparison Targets

<table>
<thead>
<tr>
<th>Active Access</th>
<th>AA-Poll</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA-Int</td>
<td>AA-SP</td>
</tr>
</tbody>
</table>

RMA
PERFORMANCE: LARGE-SCALE CODES
COMPARISON TARGETS

Active Access

AA-Int

AA-Poll

AA-SP

RMA

DMAPP
PERFORMANCE: LARGE-SCALE CODES
COMPARISON TARGETS

Active Access
AA-Int
AA-Poll
AA-SP

RMA
DMAPP
PERFORMANCE: LARGE-SCALE CODES
COMPARISON TARGETS

Active Access
AA-Int
AA-Poll
AA-SP

RMA
DMAPP
IBM Cell
**Performance: Large-Scale Codes Comparison Targets**

- Active Access
  - AA-Poll
  - AA-Int
  - AA-SP

- RMA
- DMAPP
- Cell
- InfiniBand
- Mellanox Technologies
- RoCE
Performance: Large-Scale Codes
Comparison Targets

Active Access
  AA-Int
  AA-Poll
  AA-SP

RMA
  DMAPP
  IBM Cell
  Mellanox Technologies
  InfiniBand
  RoCE

Active Messages
**Performance: Large-Scale Codes**

**Comparison Targets**

- Active Access
  - AA-Int
  - AA-Poll
  - AA-SP

- Active Messages
  - AM

- RMA
- IBM Cell
- DMAPP
- Mellanox Technologies
- RoCE

- InfiniBand
**Performance: Large-Scale Codes Comparison Targets**

Active Access
- AA-Int
- AA-Poll
- AA-SP

Active Messages
- AM
- AM-Exp

RMA
- IBM
- Cell
- DMAPP
- RoCE
- Mellanox Technologies
- InfiniBand
- CRAY

RoCE
**Performance: Large-Scale Codes Comparison Targets**

**Active Access**
- AA-Int
- AA-Poll
- AA-SP

**Active Messages**
- AM
- AM-Onload
- AM-Exp

**RMA**
- DMAPP
- Cell
- RoCE
**Performance: Large-Scale Codes**

**Comparison Targets**

**Active Access**
- AA-Poll
- AA-Int
- AA-SP

**Active Messages**
- AM
- AM-Onload
- AM-Exp
- AM-Ints

**RMA**
- IBM
- Cell

**DMAPP**
- CRAY

**RoCE**
- Mellanox Technologies
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

DMAPP

Cray

Cell

IBM

INFINIBAND

Mellanox

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

Cell

RoCE

DMAPP

InfiniBand

Mellanox

Myricom

MX
PERFORMANCE: LARGE-SCALE CODES
COMPARISON TARGETS

Active Access
  AA-Int
  AA-Poll
  AA-SP

RMA
  DMAPP
  Cell
  RoCE

Active Messages
  AM
  AM-Ints
  AM-Onload
  AM-Exp

IBM
  DCMF
  LAPI
  PAMI

Myricom
  MX

AM++GASNet

InfiniBand
Mellanox Technologies
RoCE
PERFORMANCE: LARGE-SCALE CODES DISTRIBUTED HASHTABLE

Collisions: 5%

Collisions: 25%
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

---

spcl.inf.ethz.ch
@spcl_eth
ETH Zürich
**CONCLUSIONS**

Active Access

**Uses commodity & common IOMMMUs**

- Extends paging capabilities in a distributed environment
- Alleviates RMA’s problems with AMs while preserving one-sided semantics

Uses commodity & common IOMMMUs

- Extends paging capabilities in a distributed environment
- Alleviates RMA’s problems with AMs while preserving one-sided semantics

Active Access
**CONCLUSIONS**

**Active Access**
- Alleviates RMA’s problems with AMs while preserving one-sided semantics

**Data-centric programming**
- Extends paging capabilities in a distributed environment

**Uses commodity & common IOMMMUs**
CONCLUSIONS

Active Access

- Uses commodity & common IOMMUs
- Extends paging capabilities in a distributed environment

Data-centric programming

- Alleviates RMA’s problems with AMs while preserving one-sided semantics
- Addresses of pages guide the execution of handlers
CONCLUSIONS

Active Access

- Alleviates RMA’s problems with AMs while preserving one-sided semantics
- Extends paging capabilities in a distributed environment

Data-centric programming

- Uses commodity & common IOMMUs
- 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

Performance

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

Active Access

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

Data-centric programming

- Addresses of pages guide the execution of handlers

Uses commodity & common IOMMUs

- Extends paging capabilities in a distributed environment

Performance

- Accelerates various distributed codes

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

  ![Diagram showing a connection between Proc p and Proc q]
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging puts:

```
Proc p  PUT  Proc q
```
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:

```
Proc p

PUT

Log the PUT

Proc q
```

q is modified
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging puts:

  ![Diagram showing logging process between two processes](Diagram of two processes, Proc p and Proc q, with PUT message and log actions)
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging puts:

  ![Diagram showing the interaction between Proc p and Proc q in the context of RMA.]
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):

  Proc p

  Proc q
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):

```
Proc p

p is modified

GET

Proc q
```
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):

  p is modified

  Proc p
  GET
  Log the GET

  Proc q
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):

  ![Diagram showing the process of logging](image)

  - Proc p
  - Proc q
  - GET
  - Log the GET
  - p is modified

- Proc p modifies p
- Proc q issues a GET request
- Proc p logs the GET request

- **Important:** Logging the GET operation ensures that modifications are traced accurately, even in complex RMA scenarios.
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):
  
  - Proc p
  - GET
  - Log the GET
  - Attempt to reply the GET
  - Proc q
  - p is modified
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (naive):

p is modified

Proc p

Log the GET

GET

Attempt to reply the GET

FAIL!
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

[p is modified]

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (traditional) [1]:

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

```plaintext
Proc p

p is modified

GET

Proc q
```
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

  p is modified

  Proc p

  GET

  IOMMU

  Proc q
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

```plaintext
p is modified
```

![Diagram showing the process flow involving Proc p, Proc q, GET, IOMMU, and logging the GET.]

ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

```
Proc p

GET

IOMMU

Log the GET
```

p is modified
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

```
<table>
<thead>
<tr>
<th>Proc p</th>
<th>GET</th>
<th>Proc q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p is modified</td>
<td>IOMMU</td>
<td>Log the GET</td>
</tr>
<tr>
<td></td>
<td>Fetch the logs</td>
<td></td>
</tr>
</tbody>
</table>
```

Log the GET
ACTIVE ACCESS USE-CASES
ACCELERATING LOGGING FOR RMA

- Logging gets (AA):

  Proc p
  p is modified
  GET
  Log the GET
  Fetch the logs
  reply the GET

  Proc q
  IOMMU
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

compute  compute  compute  compute
compute  compute  compute  compute
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

barrier

compute  compute  compute  compute

compute  compute  compute  compute
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

barrier
compute
compute
compute
compute

barrier
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

barrier  compute  compute  ...  compute  compute

barrier  compute  compute  ...  compute  compute
ACTIVE ACCESS USE-CASES

INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k

Proc 1  ...  Proc k

barrier

compute

compute

compute

compute

barrier
ACTIVE ACCESS USE-CASES

INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

barrier

compute

compute

compute

compute

global rollback
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k  ...  Proc 1  ...  Proc k

barrier  compute  compute  compute  compute

barrier  compute  compute  compute  compute
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES

INCREMENTAL CHECKPOINTING FOR RMA

Proc 1  ...  Proc k
  
  barrier

...  ...  ...

Proc 1  ...  Proc k

barrier

compute

compute

compute

compute
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES
INCREMENTAL CHECKPOINTING FOR RMA
ACTIVE ACCESS USE-CASES

INCREMENTAL CHECKPOINTING FOR RMA
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1 ➔ compute ➔ Proc k

Node N

Proc 1 ➔ compute ➔ Proc k
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1

...  

Proc k

Node N

Proc 1

...  

Proc k

barrier

compute

compute

compute

compute

compute

compute
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1 → ... → Proc k

Node N

Proc 1 → ... → Proc k

barrier

compute

barrier

compute

compute
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1  ...  Proc k

Node N

Proc 1  ...  Proc k

barrier

compute

barrier

compute

compute

compute

compute

compute

compute
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1  ...  Proc k

Node N

Proc 1  ...  Proc k

barrier  compute  compute  compute

barrier  compute  compute  compute
COORDINATED CHECKPOINTING (MP)

Node 1

- Proc 1
- Proc k

Node N

- Proc 1
- Proc k

barrier

compute

global rollback

compute

compute

compute

compute
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1

...  

Proc k

Node N

Proc 1

...  

Proc k

barrier

compute

barrier

compute

compute

compute
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)

Node 1

Proc 1

...  

Proc k

Node N

Proc 1

...  

Proc k

barrier

compute

compute

compute

compute

barrier
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)

Diagram showing the process of coordinated checkpointing (MP) involving multiple processes (Proc 1, Proc k). The diagram illustrates the compute and barrier phases across these processes, indicating how data is exchanged and synchronized.
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
COORDINATED CHECKPOINTING (MP)
PERFORMANCE: LARGE-SCALE CODES
FAULT TOLERANCE SCHEME

Logging gets:

Sorting time: