FAST'21

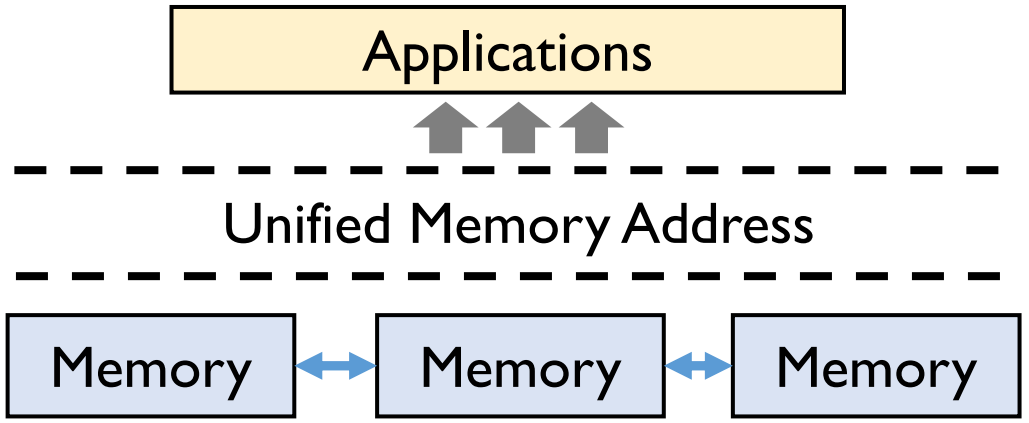
Concordia: Distributed Shared Memory with In-Network Cache Coherence

Qing Wang, Youyou Lu, Erci Xu, Junru Li, Youmin Chen, Jiwu Shu

Tsinghua University

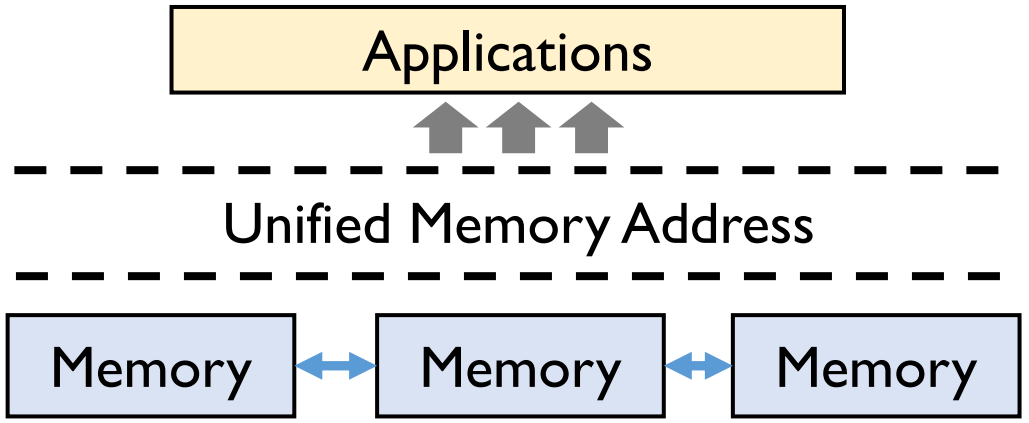
Distributed Shared Memory (DSM)

- **Classic DSM (early 1990s)**
  - Unified memory space
  - Ease programming for distributed apps



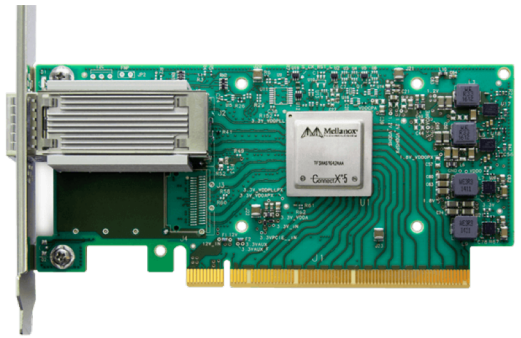
Distributed Shared Memory (DSM)

- **Classic DSM (early 1990s)**
  - Unified memory space
  - Ease programming for distributed apps
  - Failed due to unsatisfying performance

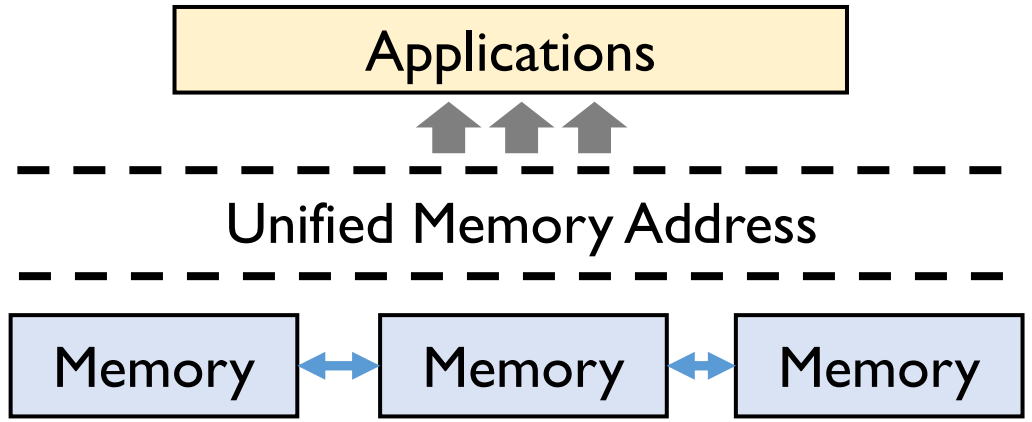


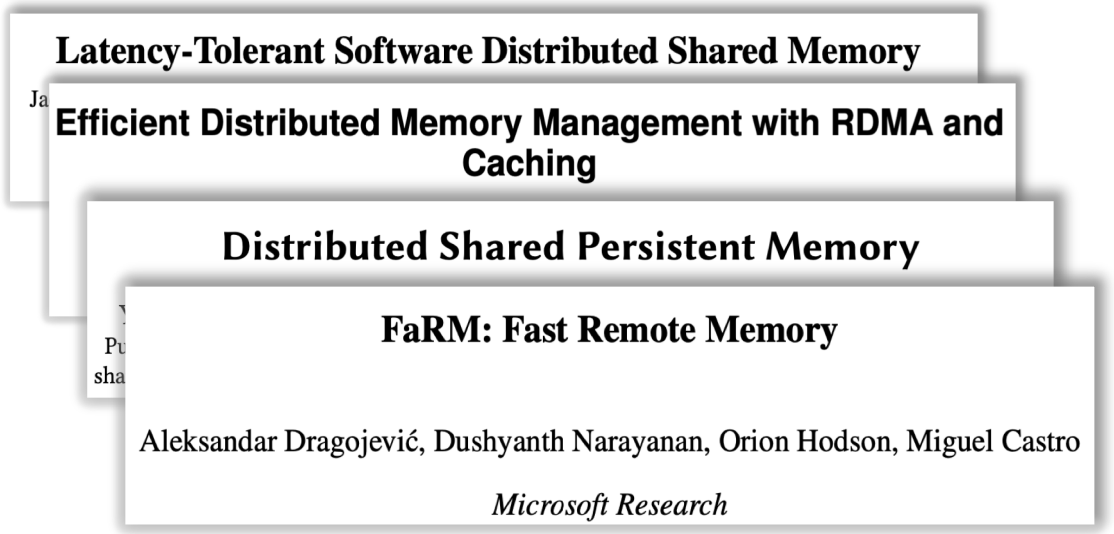
Distributed Shared Memory (DSM)

- **Classic DSM (early 1990s)**
  - Unified memory space
  - Ease programming for distributed apps
  - Failed due to unsatisfying performance
- **Modern DSM (2010 - now)**
  - Sparked by fast network (e.g., RDMA)
  - Support modern apps (e.g., OLTP, Graph)
  - Example: Microsoft's FaRM



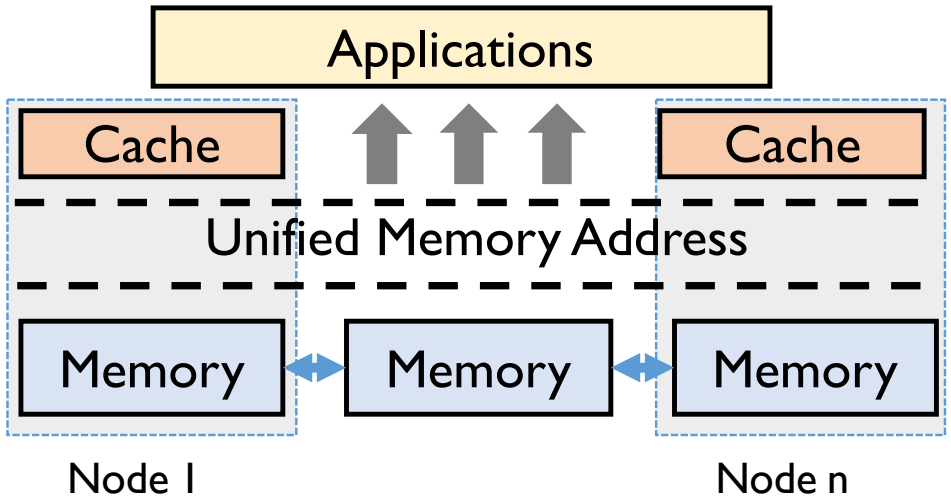
Mellanox Connectx-5
100Gbps, RTT < 2µs





Caching & Coherence

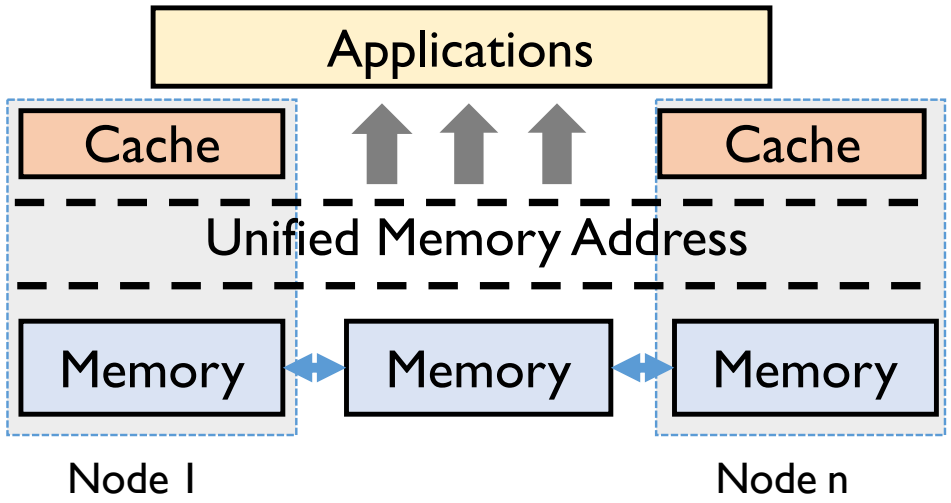
- **Still, caching is a must**
  - Boost performance
  - Bandwidth: local mem vs. network → 2-4X
  - Latency: local mem vs. network → 12X



Data of memory bandwidth and latency is from "https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf", OSDI'18

Caching & Coherence

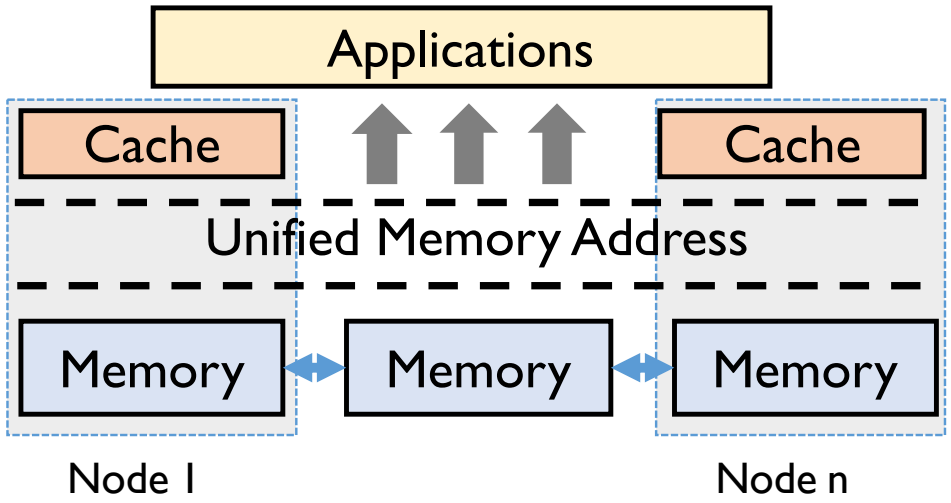
- **Still, caching is a must**
  - Boost performance
  - Bandwidth: local mem vs. network → 2-4X
  - Latency: local mem vs. network → 12X
- **Coherence is a necessary evil**
  - ✓ Offer strong consistency

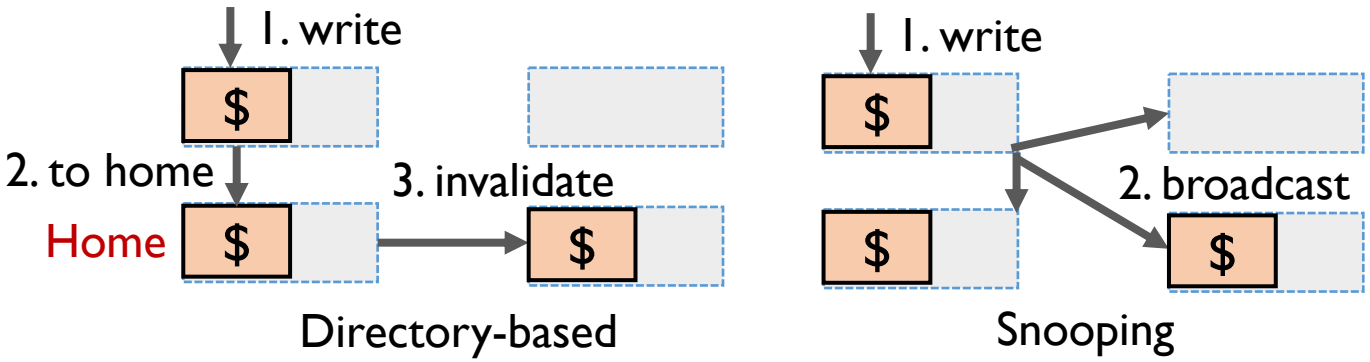


Data of memory bandwidth and latency is from "https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf", OSDI'18

Caching & Coherence

- **Still, caching is a must**
  - Boost performance
  - Bandwidth: local mem vs. network → 2-4X
  - Latency: local mem vs. network → 12X
- **Coherence is a necessary evil**
  - ✓ Offer strong consistency
  - ✗ But, need slow distributed coordination

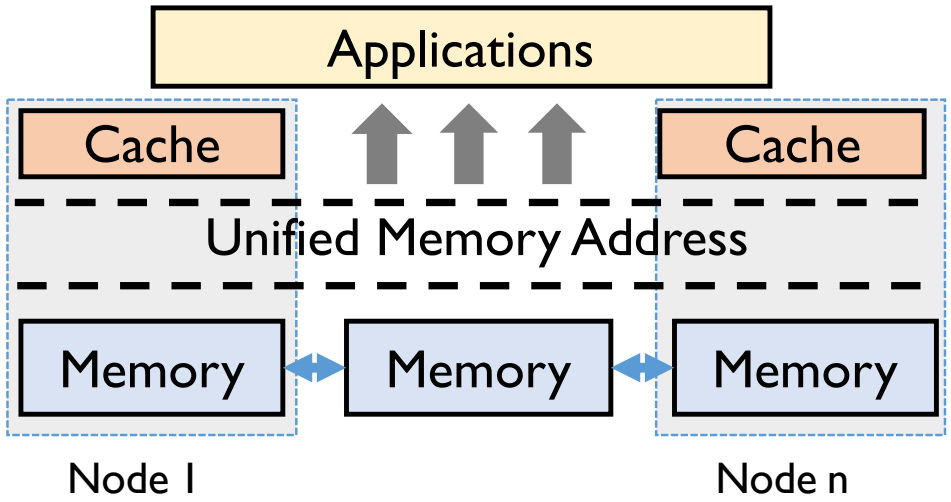




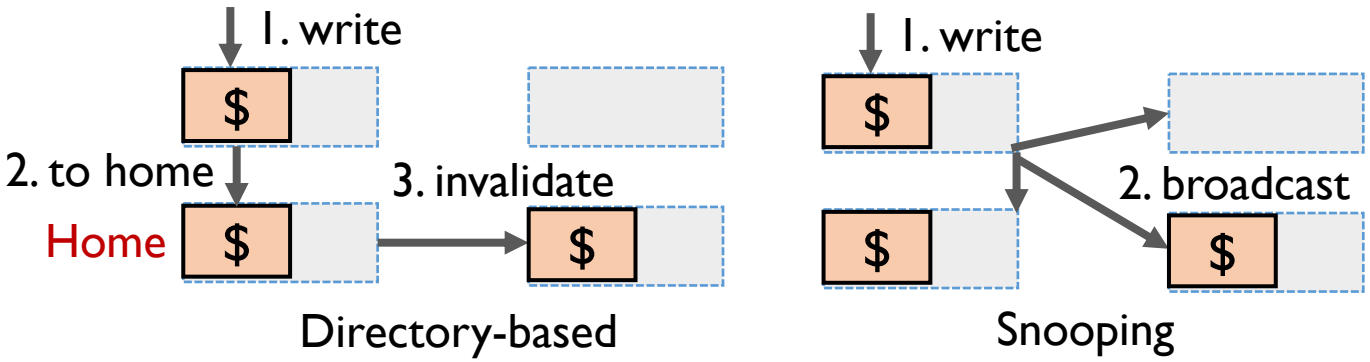
Data of memory bandwidth and latency is from "https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf", OSDI'18

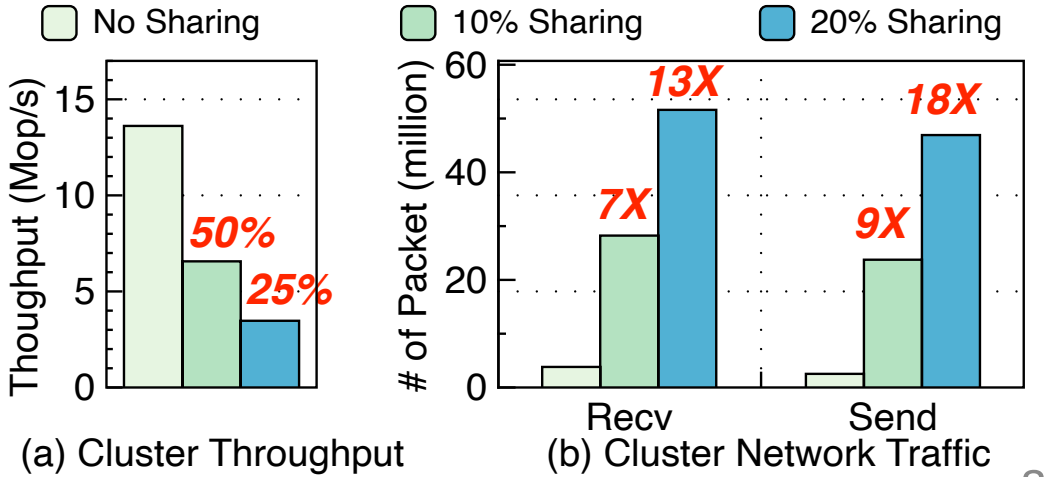
Caching & Coherence

- **Still, caching is a must**
  - Boost performance
  - Bandwidth: local mem vs. network → 2-4X
  - Latency: local mem vs. network → 12X
- **Coherence is a necessary evil**
  - ✓ Offer strong consistency
  - ✗ But, need slow distributed coordination



GAM [VLDB'18], directory coherence using RDMA





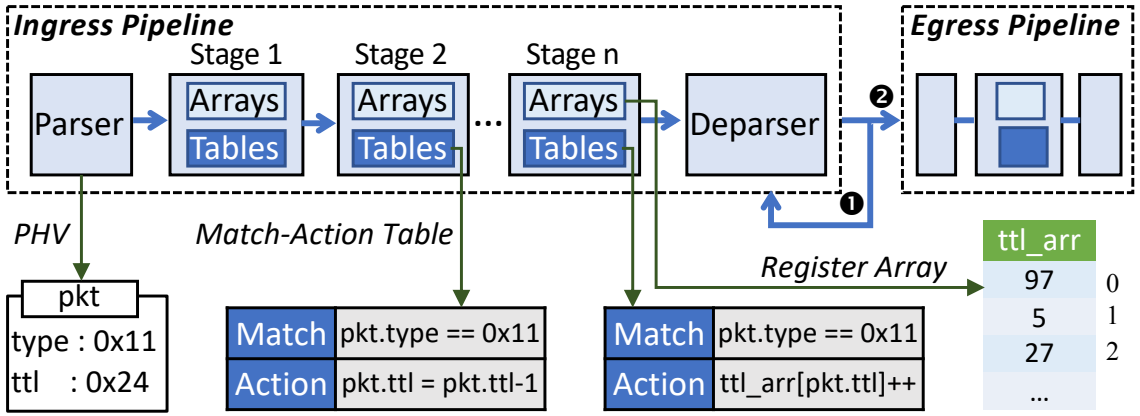
Data of memory bandwidth and latency is from "https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf", OSDI'18

Opportunities from Programmable Switches

BAREFOOT NETWORKS | an Intel company

BROADCOM® connecting everything®

user-defined line-rate network protocols



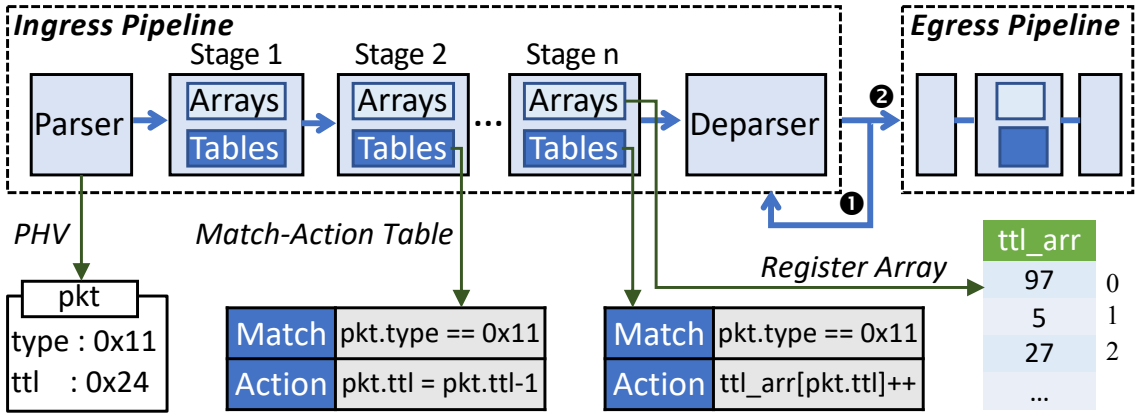
match-action pipeline

Opportunities from Programmable Switches

BAREFOOT NETWORKS | an Intel company

BROADCOM connecting everything®

user-defined line-rate network protocols



match-action pipeline

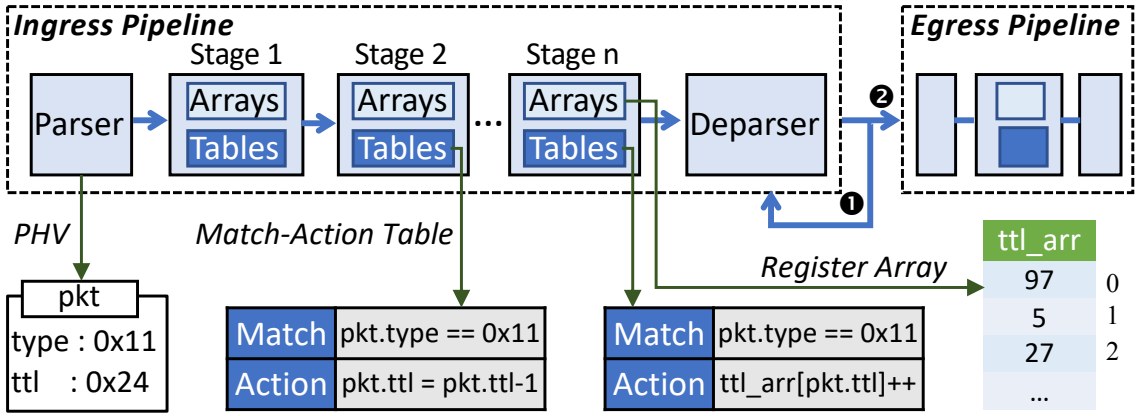
Using programmable switches to handle cache coherence is promising …

Opportunities from Programmable Switches

BAREFOOT NETWORKS | an Intel company

BROADCOM® connecting everything®

user-defined line-rate network protocols



match-action pipeline

Using programmable switches to handle cache coherence is promising …

Programmable Switches

- Centralized hub & line-rate forwarding

Coherence Protocols

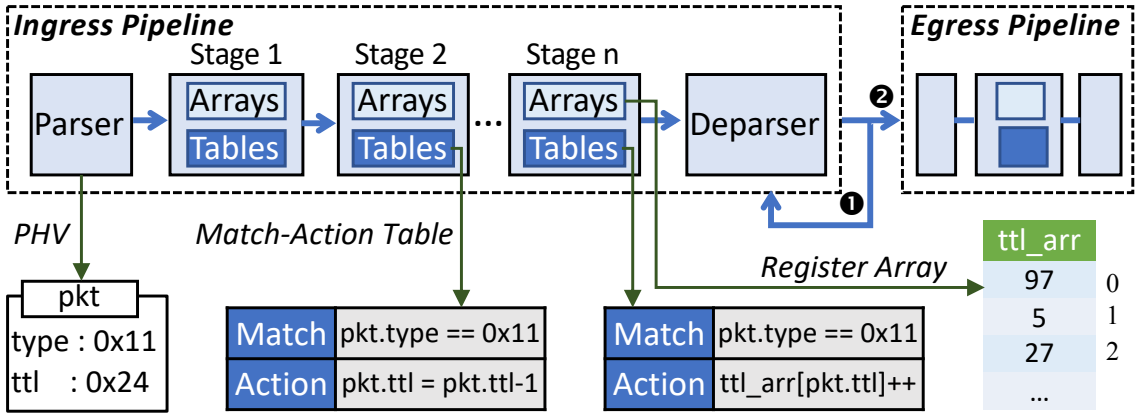
- Multicast invalidation

Opportunities from Programmable Switches

BAREFOOT NETWORKS | an Intel company

BROADCOM connecting everything®

user-defined line-rate network protocols



match-action pipeline

Using programmable switches to handle cache coherence is promising …

Programmable Switches

- Centralized hub & line-rate forwarding
- On-chip memory (i.e., register arrays)

Coherence Protocols

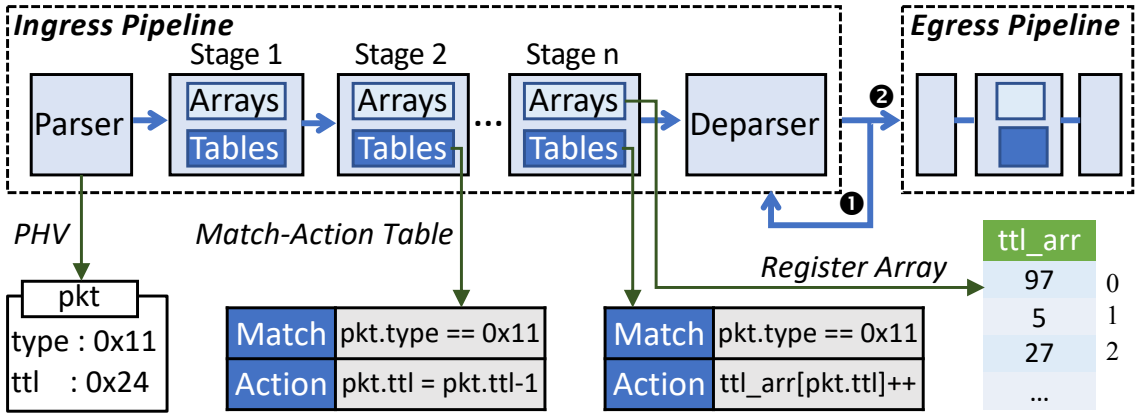
- Multicast invalidation
- Track cache copies' metadata

Opportunities from Programmable Switches

BAREFOOT NETWORKS | an Intel company

BROADCOM connecting everything®

user-defined line-rate network protocols



match-action pipeline

Using programmable switches to handle cache coherence is promising …

Programmable Switches

- Centralized hub & line-rate forwarding
- On-chip memory (i.e., register arrays)
- Atomic read-modify-write

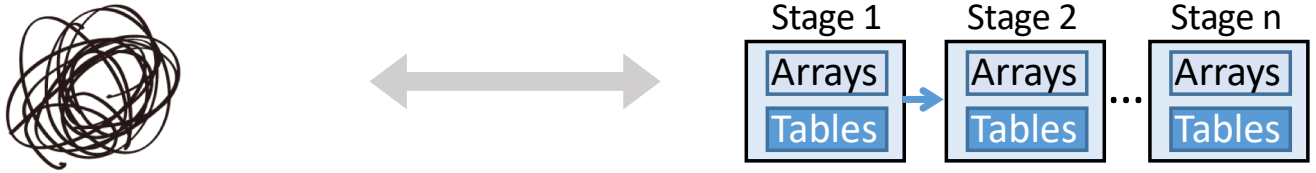
Coherence Protocols

- Multicast invalidation
- Track cache copies' metadata
- Serialize conflicting requests

Challenges

1. Restricted expressive power

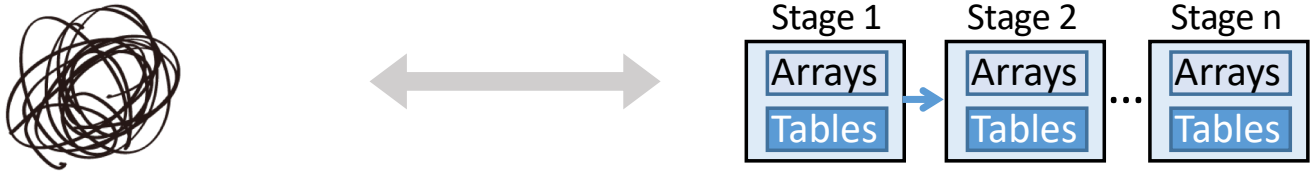
How to map coherence logic into switch pipelines ?



Challenges

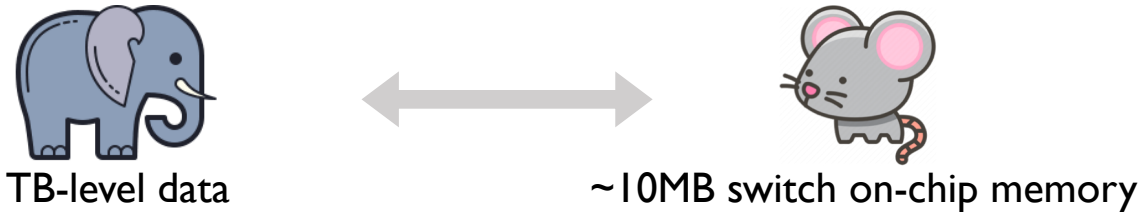
1. Restricted expressive power

How to map coherence logic into switch pipelines ?



2. Limited switch memory

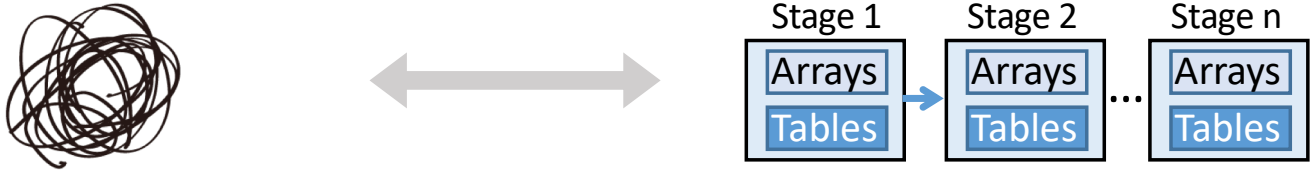
How to manage coherence of large amounts of data ?



Challenges

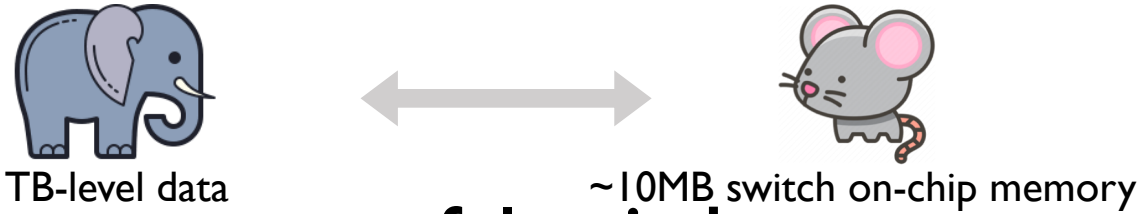
1. Restricted expressive power

How to map coherence logic into switch pipelines ?



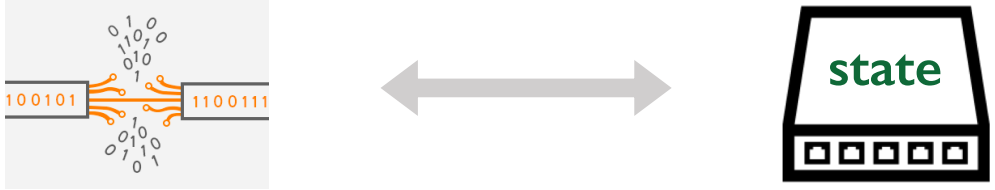
2. Limited switch memory

How to manage coherence of large amounts of data ?



3. Packet loss upon stateful switch

How to enforce exactly-once semantics ?

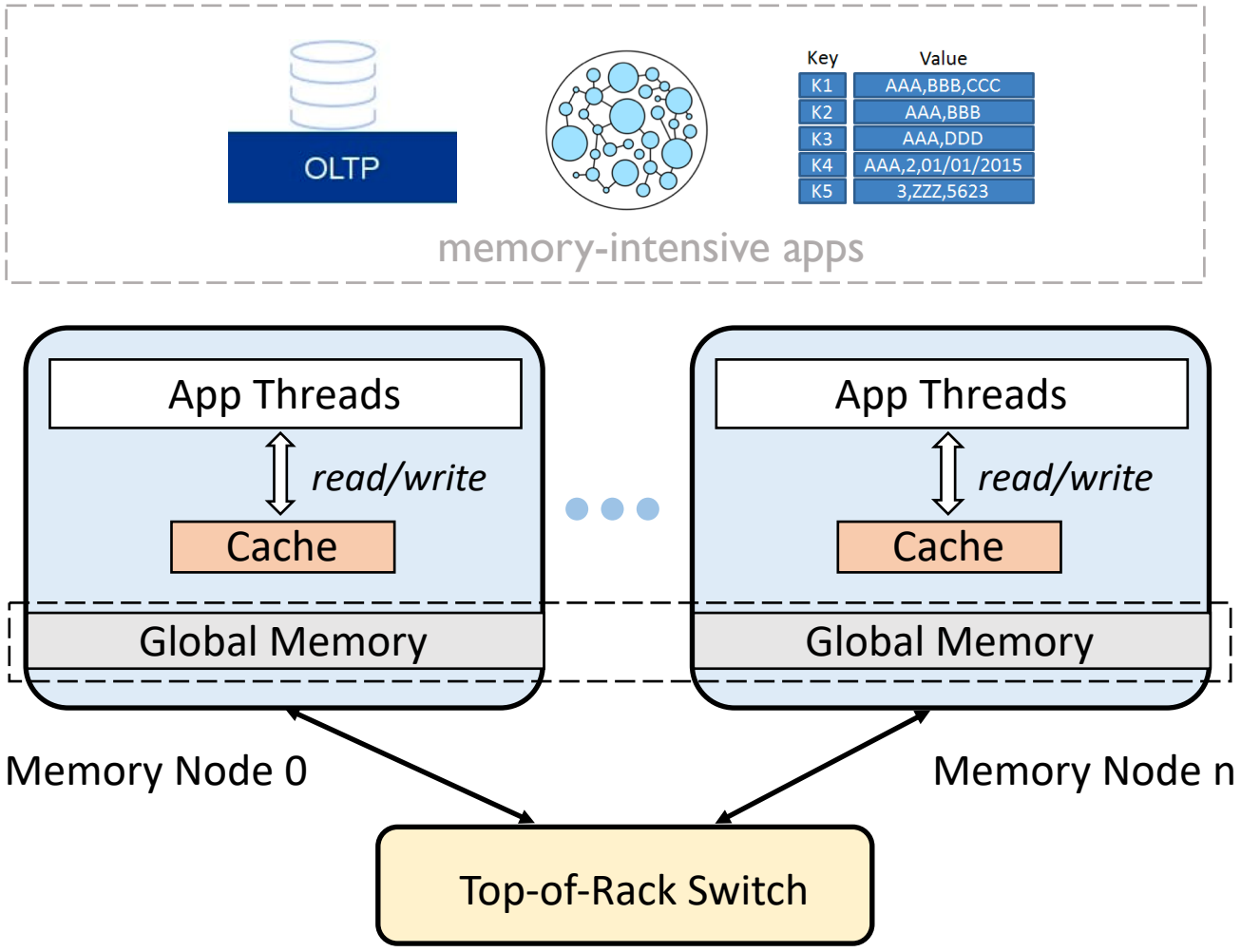


Outline

- Background & Motivation
- Concordia: a DSM with In-Network Coherence
- Results
- Summary & Conclusion

Concordia Architecture

A user-level rack-scale DSM



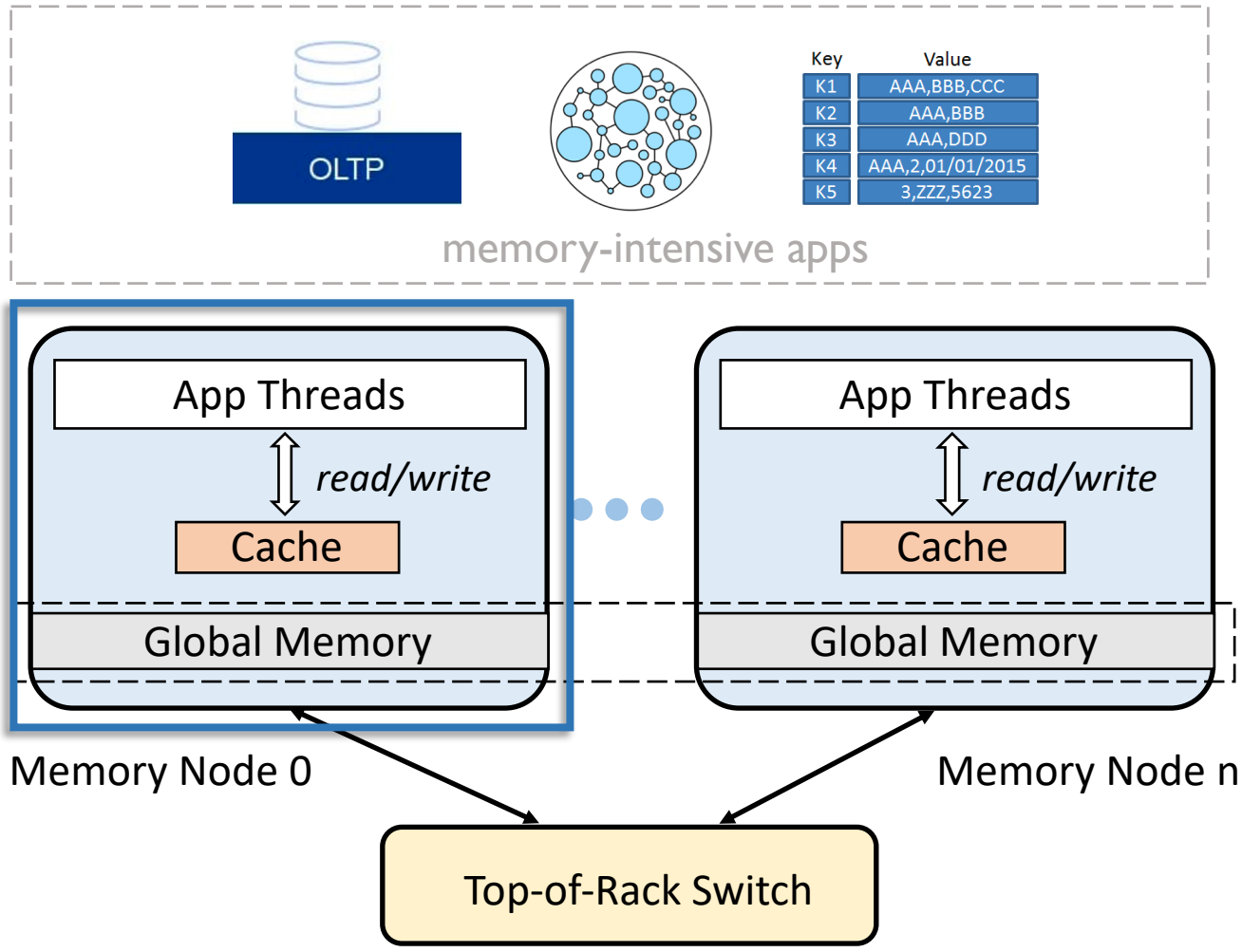
Concordia Architecture

Concordia Architecture

A user-level rack-scale DSM

Memory nodes (servers)

- Global memory
- Local cache
- Linearizable write/read interface



Concordia Architecture

Concordia Architecture

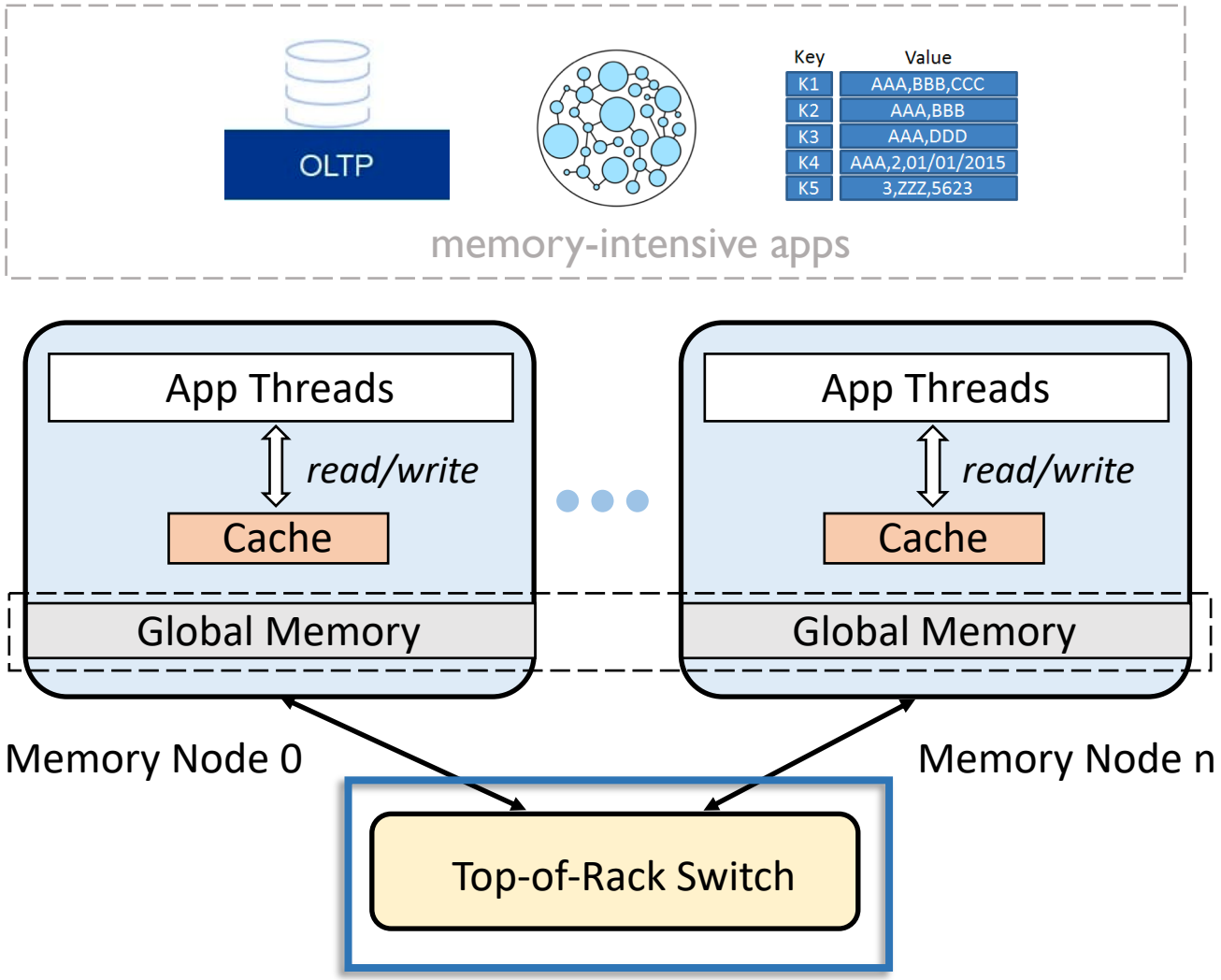
A user-level rack-scale DSM

Memory nodes (servers)

- Global memory
- Local cache
- Linearizable write/read interface

Top-of-rack switch

- Route normal packets
- Accelerate coherence



Concordia Architecture

Key Designs

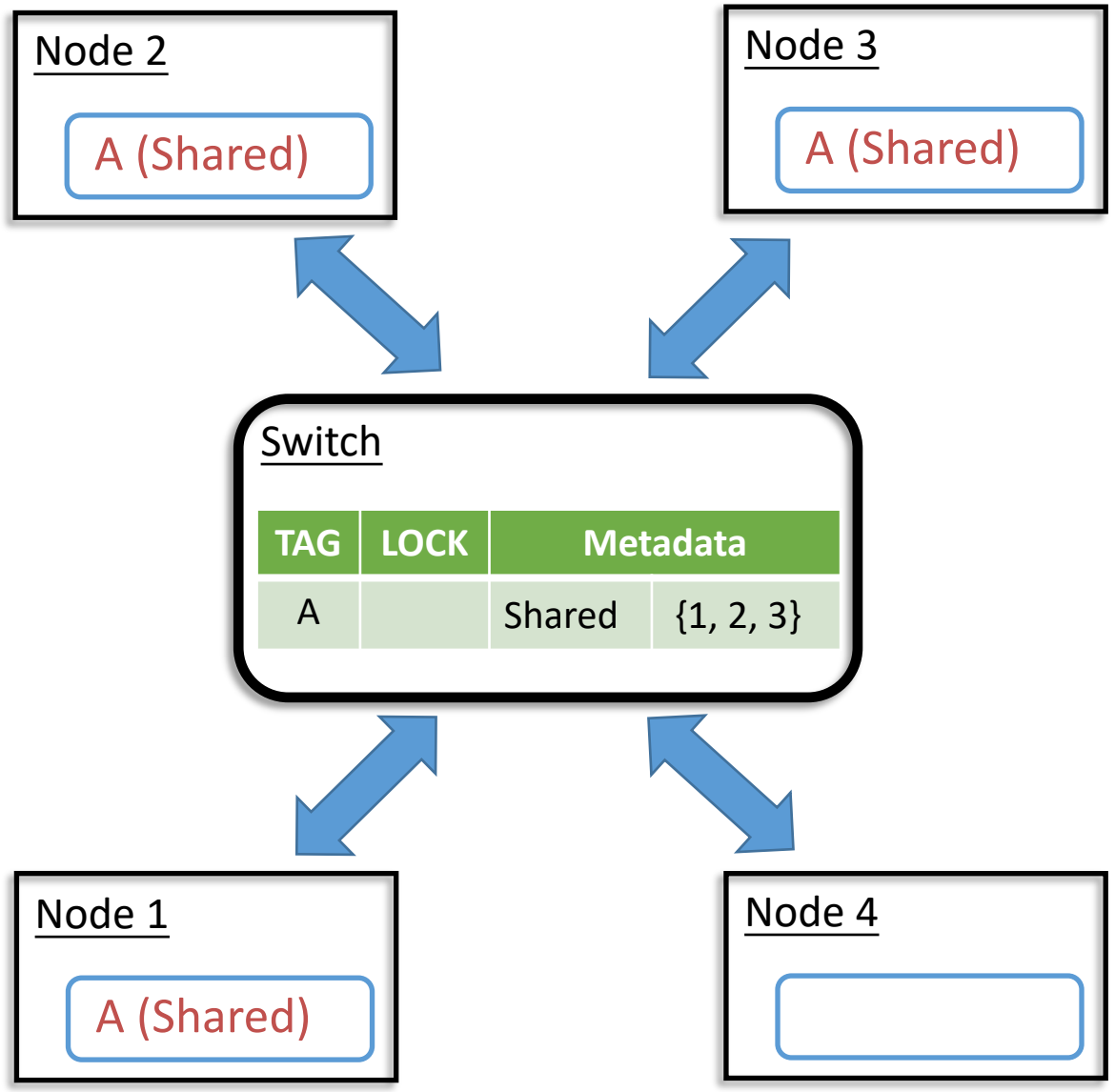
- 🌧️ Restricted expressive power
- ☀️ In-Network Coherence Protocol

- 🌧️ Limited switch memory
- ☀️ Ownership Migration

- 🌧️ Packet loss upon stateful switch
- ☀️ Idempotent Operations

In-Network Coherence Protocol

Write-invalidate protocol

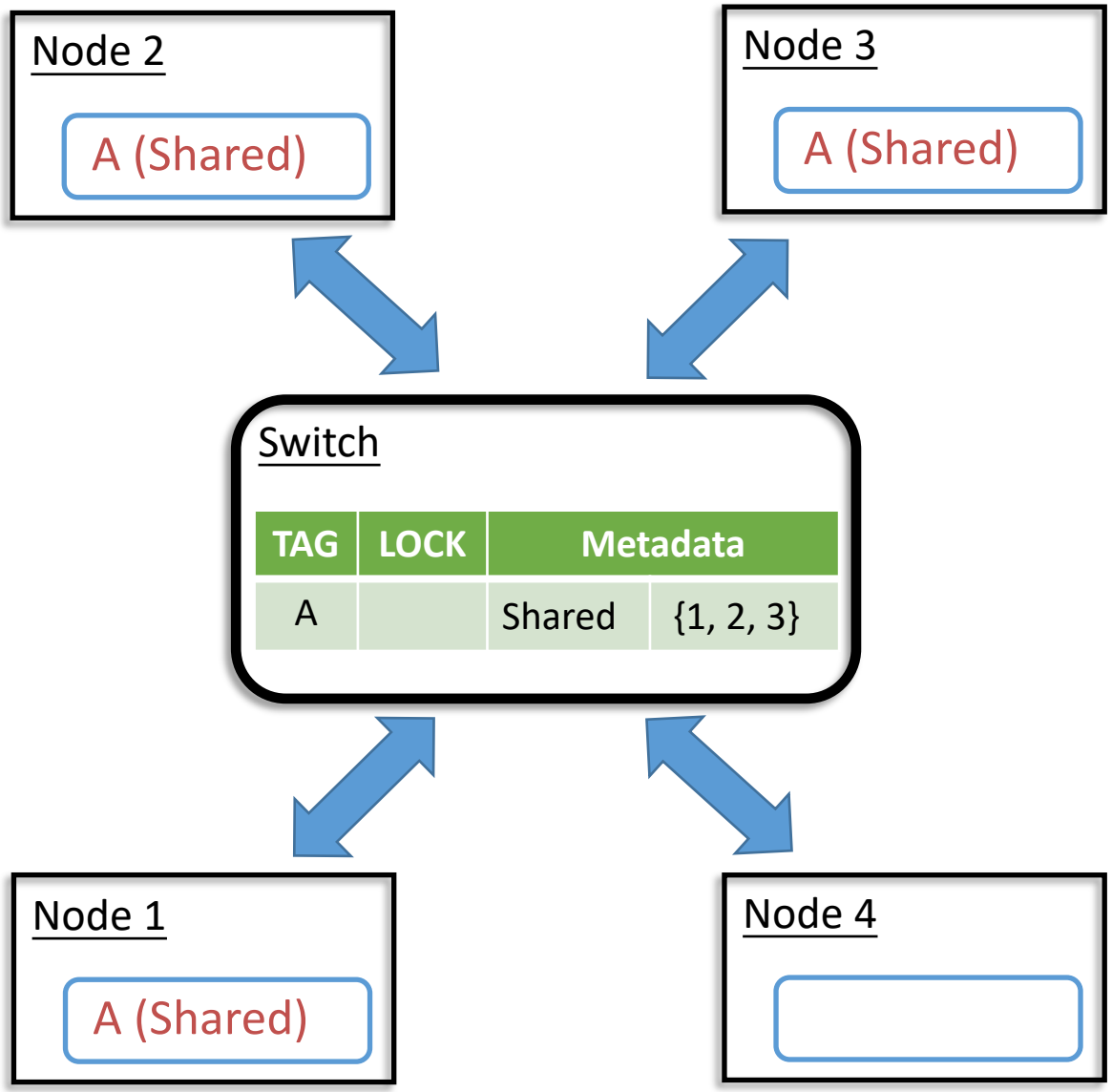


In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

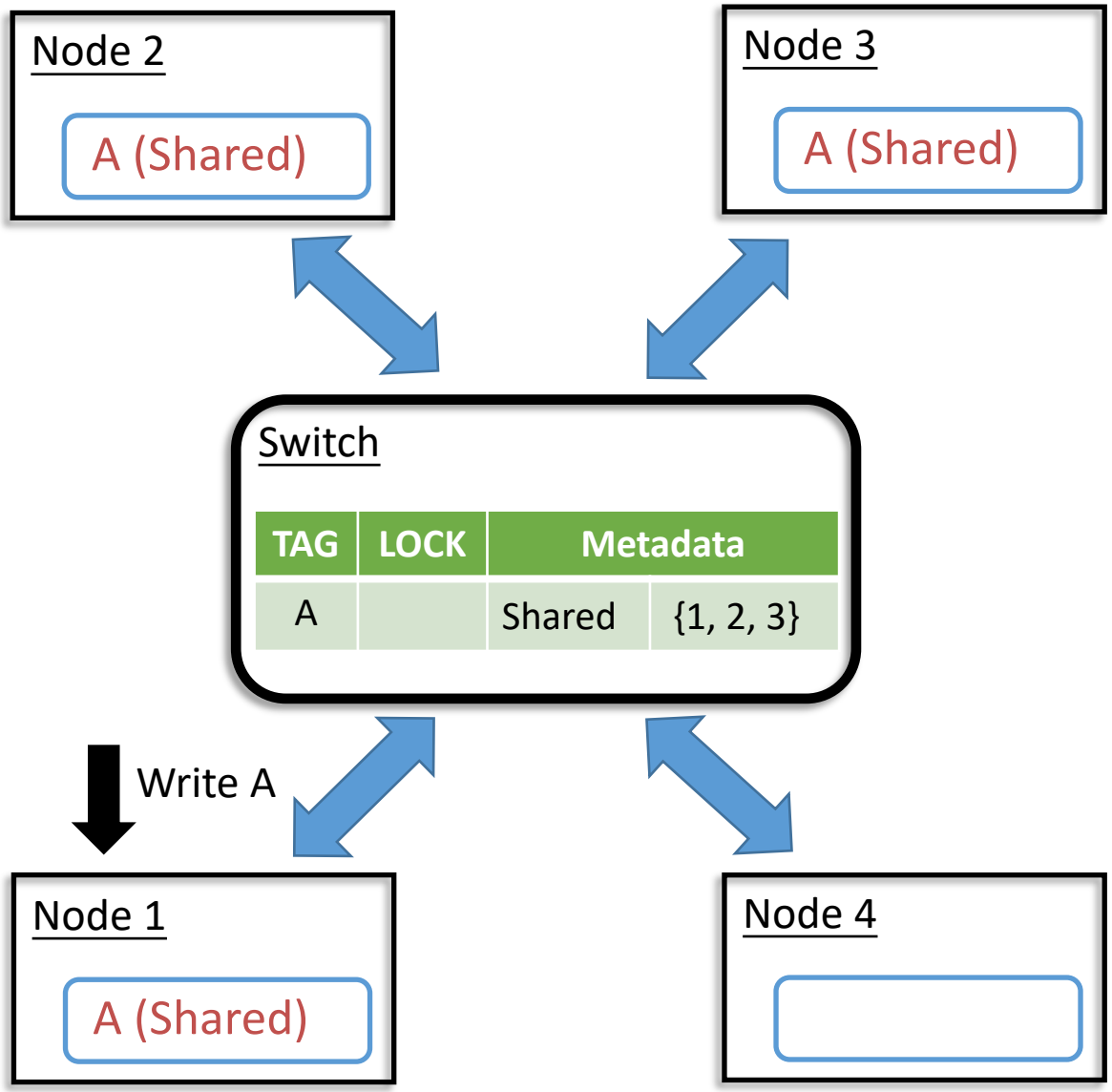


In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests



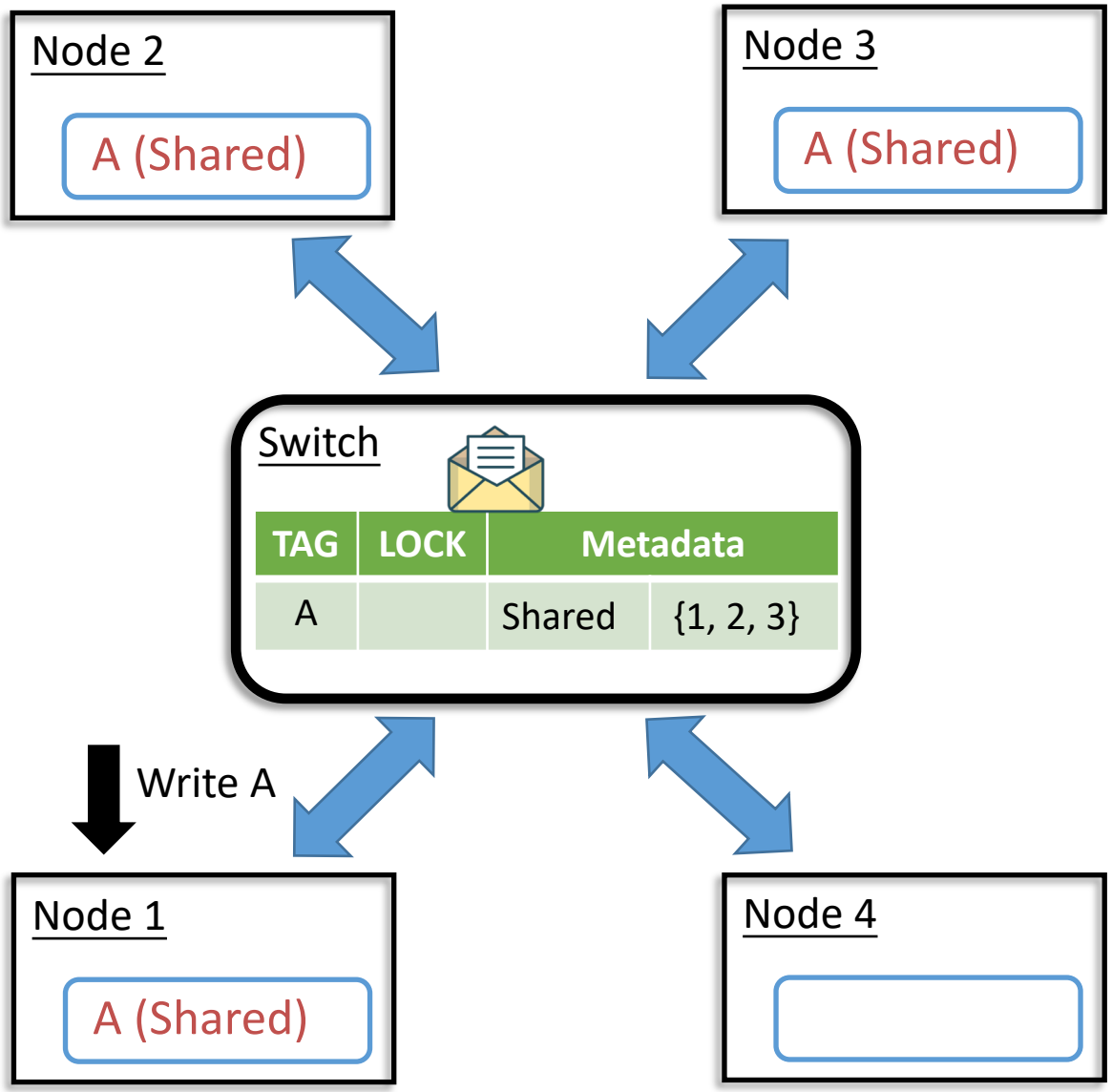
In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request



In-Network Coherence Protocol

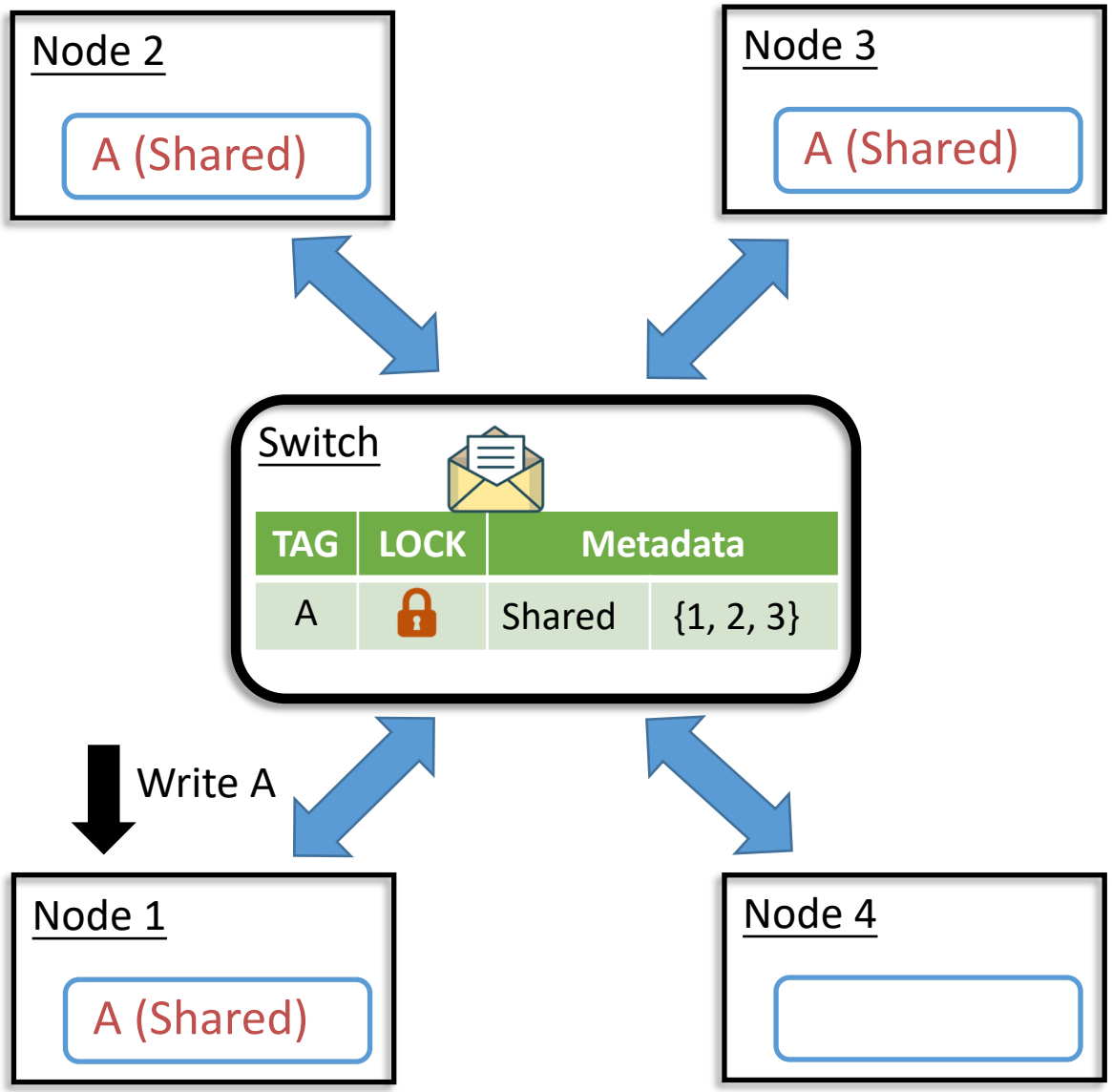
Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast



In-Network Coherence Protocol

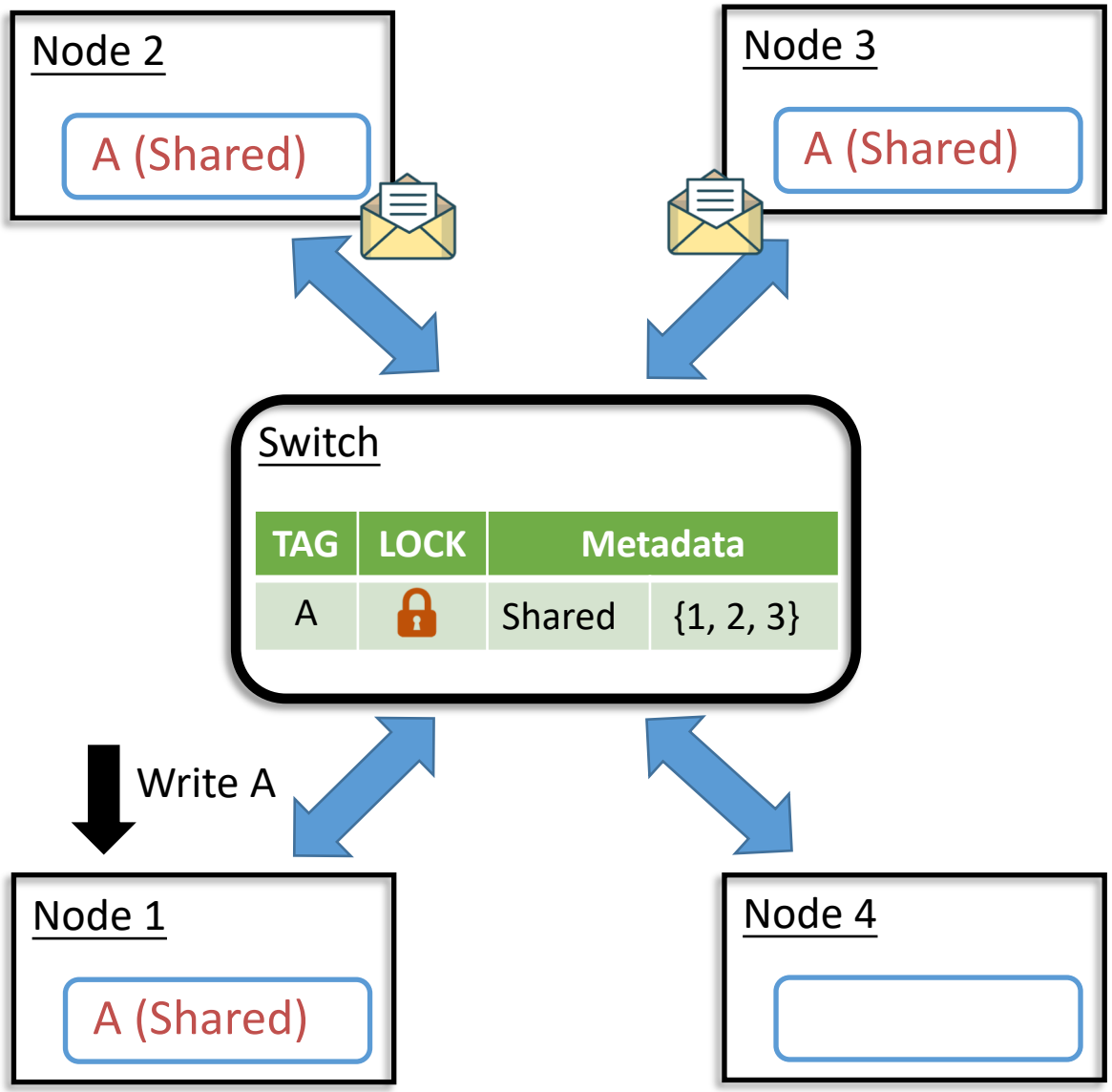
Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast



In-Network Coherence Protocol

Write-invalidate protocol

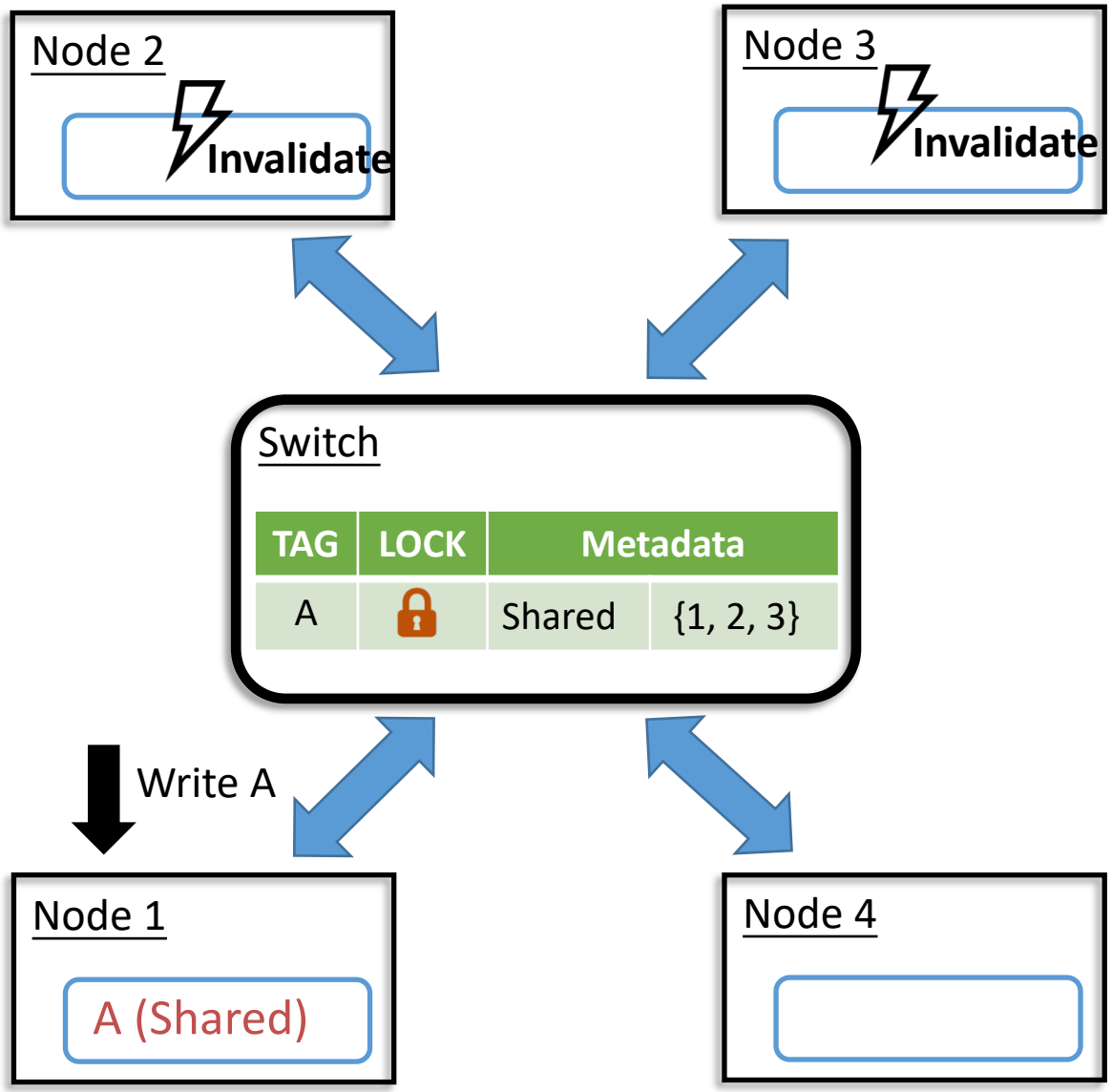
Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast

Step 3. invalidation



In-Network Coherence Protocol

Write-invalidate protocol

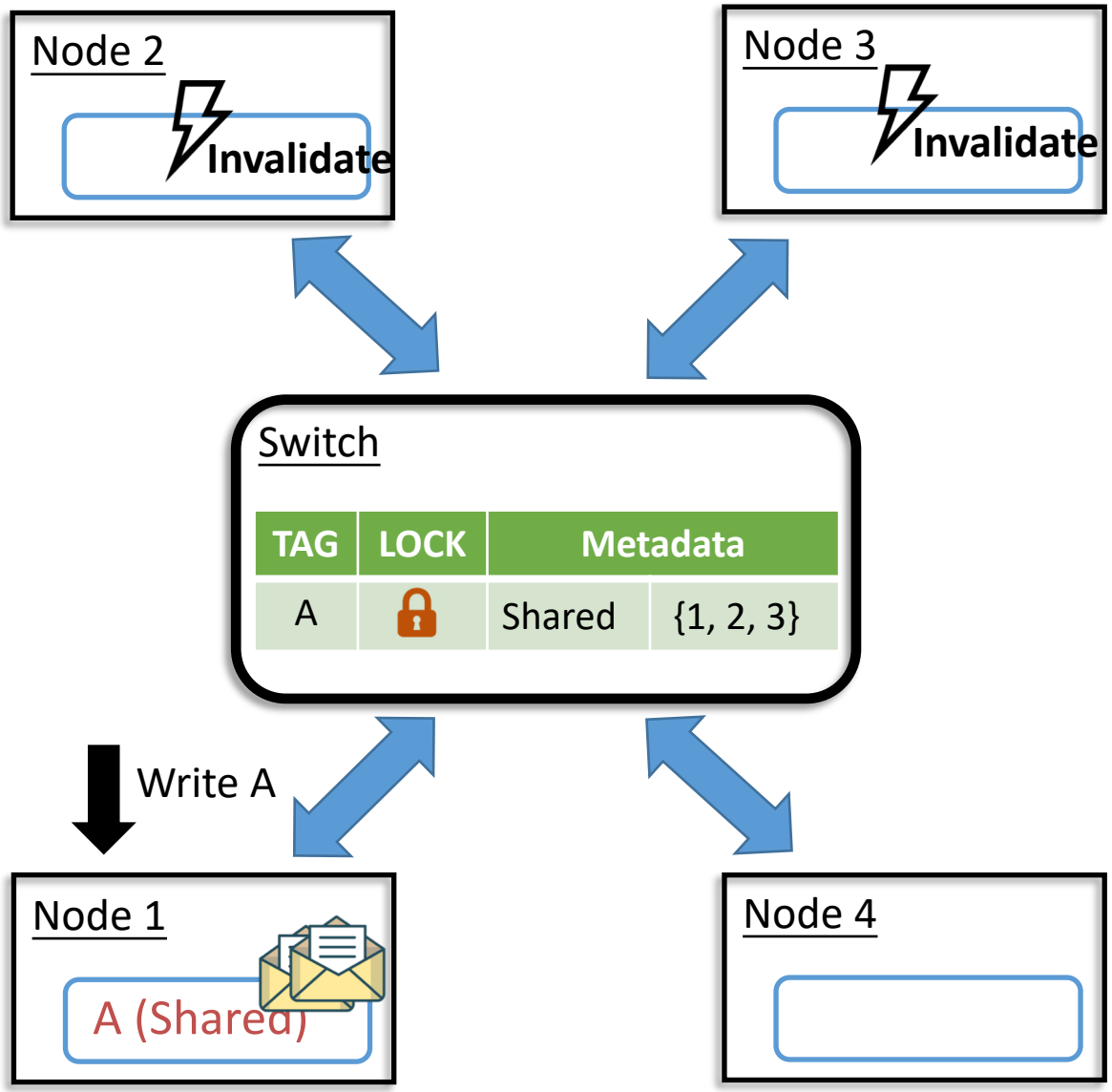
Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast

Step 3. invalidation



In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

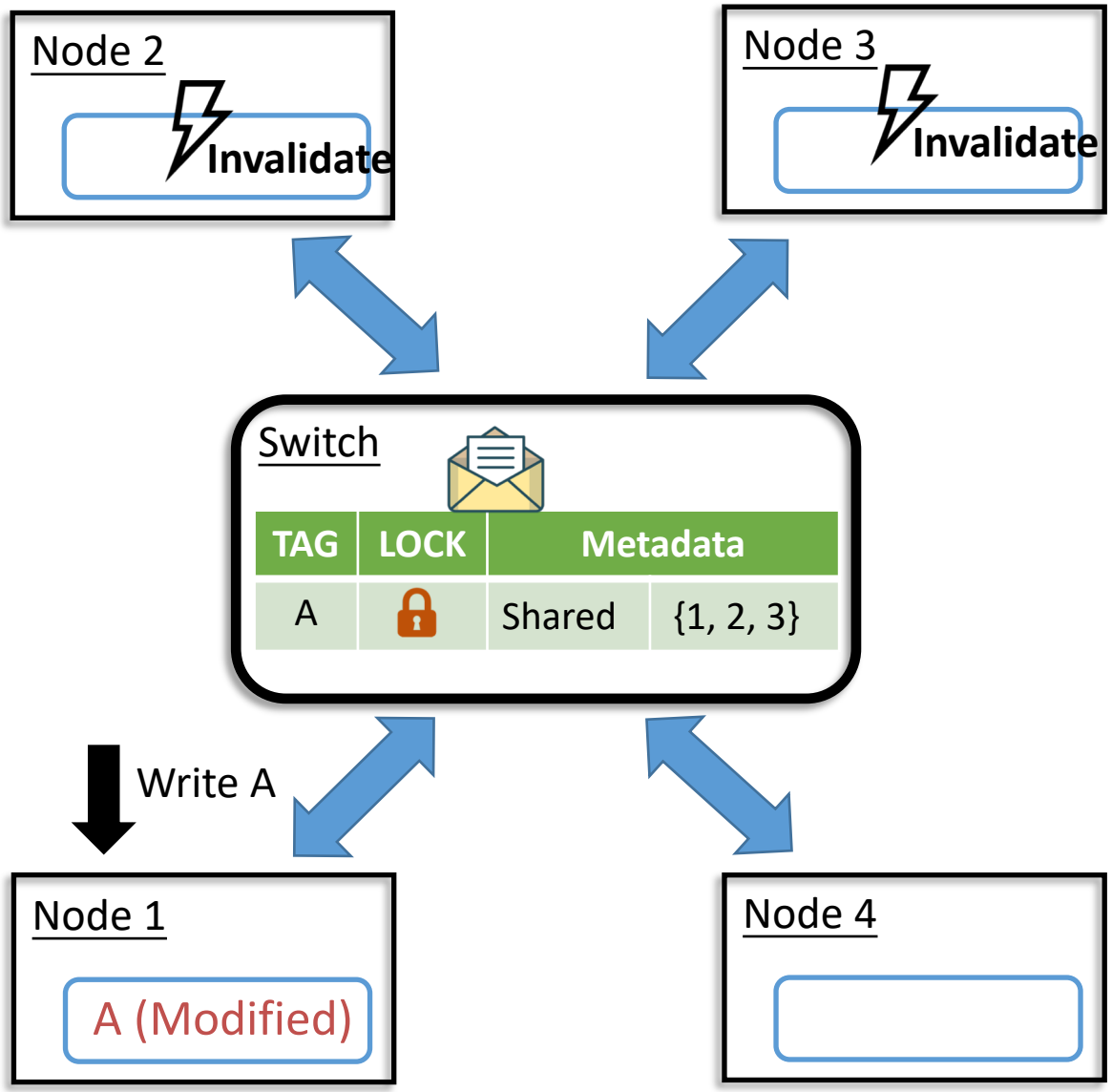
- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast

Step 3. invalidation

Step 4. unlock and install new metadata



In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

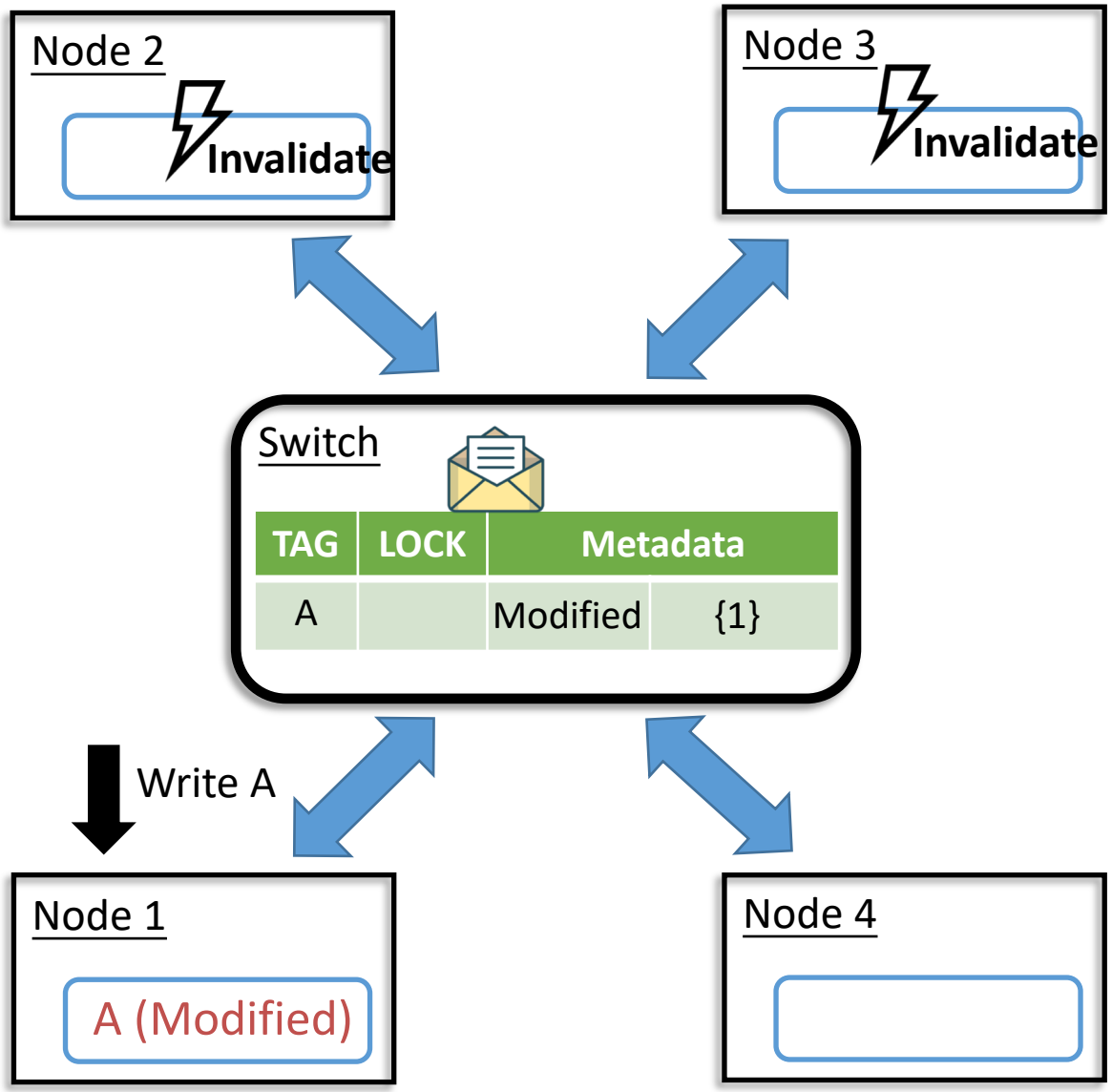
- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

Step 1. generate the coherence request

Step 2. lock and multicast

Step 3. invalidation

Step 4. unlock and install new metadata



In-Network Coherence Protocol

Write-invalidate protocol

Programmable Switch

- Cache block metadata (status, sharers)
- Reader-writer locks
- Multicast coherence requests

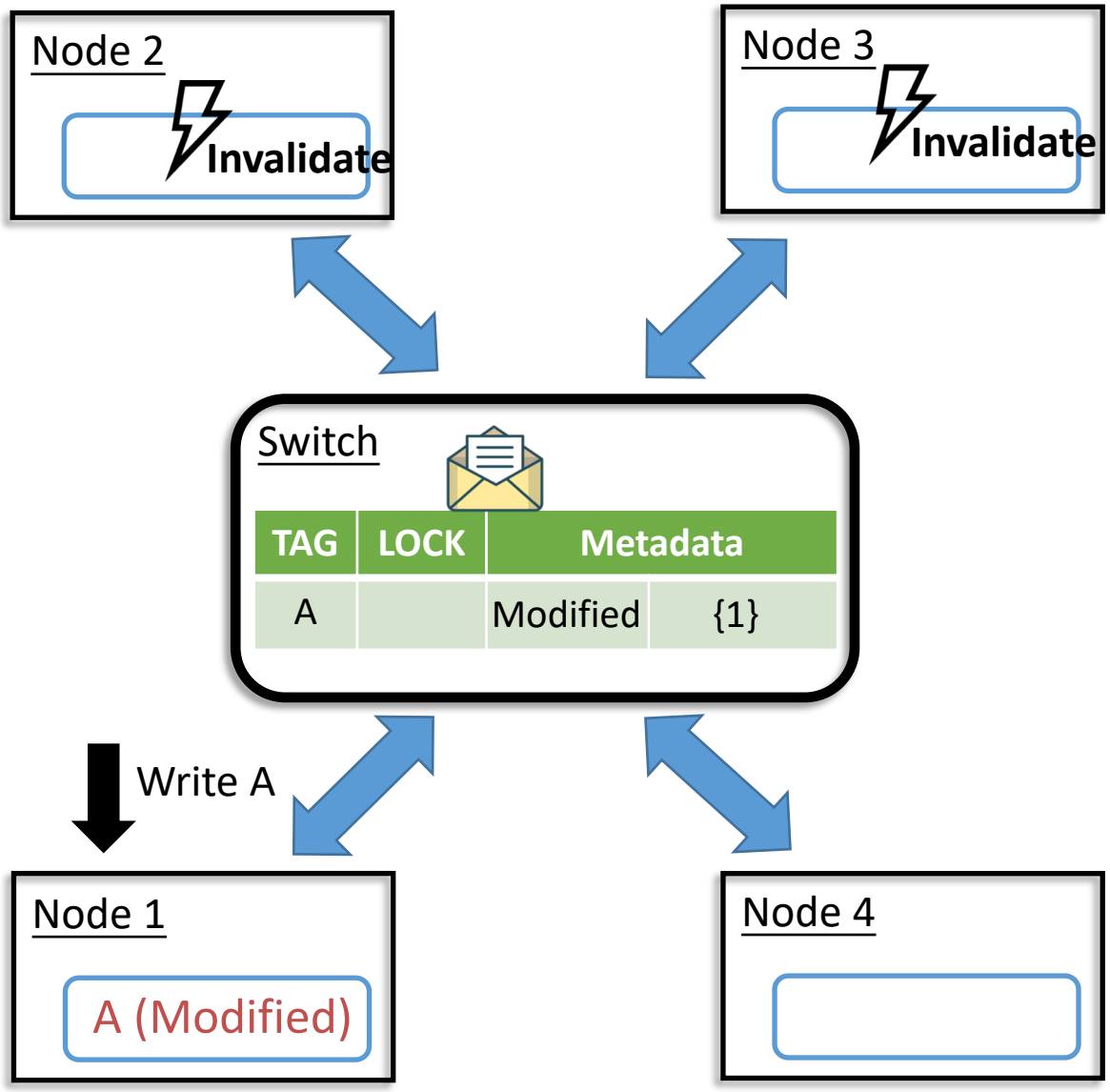
Step 1. generate the coherence request

Step 2. lock and multicast

Step 3. invalidation

Step 4. unlock and install new metadata

(1) 1 RTT (unlock is async)
(2) Minimized network traffic



Ownership Migration

Limited switch memory cannot accommodate metadata of all cache blocks

Ownership Migration

Limited switch memory cannot accommodate metadata of all cache blocks

The switch manages coherence of active cache blocks

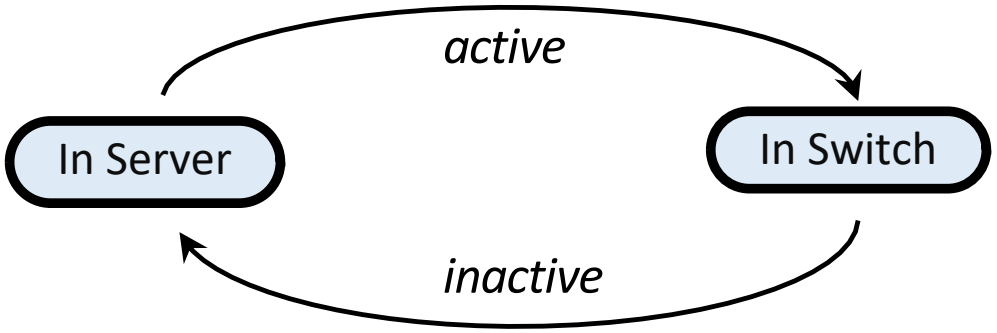
Ownership Migration

Limited switch memory cannot accommodate metadata of all cache blocks

The switch manages coherence of active cache blocks

Ownership Migration

- If a cache block triggers lots of invalidations, migrate its ownership: Servers → Switch



* For a cache block, its ownership refers to the right to manage its coherence.

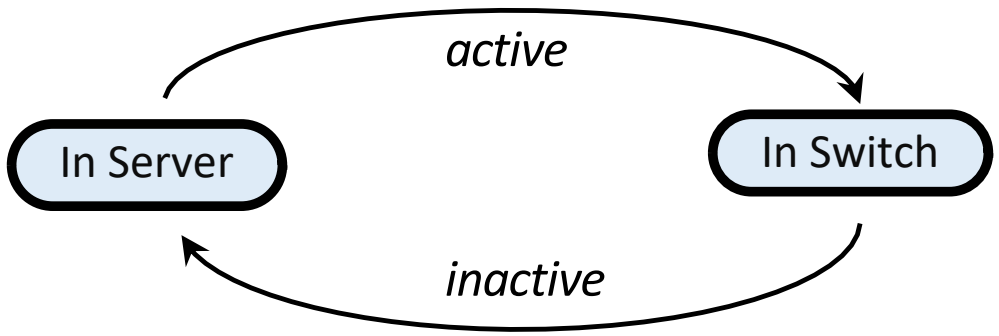
Ownership Migration

Limited switch memory cannot accommodate metadata of all cache blocks

The switch manages coherence of active cache blocks

Ownership Migration

- If a cache block triggers lots of invalidations, migrate its ownership: Servers → Switch
- If a cache block is inactive, migrate its ownership: Switch → Servers



* For a cache block, its ownership refers to the right to manage its coherence.

Idempotent Operations

Packet loss due to switch buffer overflow

Idempotent Operations

Packet loss due to switch buffer overflow

Make all operations both in servers and switches idempotent

Idempotent Operations

Packet loss due to switch buffer overflow

Make all operations both in servers and switches idempotent

On suspecting a lost request via timeout, the app thread retransmits it …

Idempotent Operations

Packet loss due to switch buffer overflow

Make all operations both in servers and switches idempotent

On suspecting a lost request via timeout, the app thread retransmits it …

Server Idempotence

- Requests contain unique SNs (sequence number)
- Servers record SNs of executed requests,
  to filter out duplicated requests

Idempotent Operations

Packet loss due to switch buffer overflow

Make all operations both in servers and switches idempotent

On suspecting a lost request via timeout, the app thread retransmits it …

Server Idempotence

- Requests contain unique SNs (sequence number)
- Servers record SNs of executed requests, to filter out duplicated requests

Switch Idempotence

- For each app thread, the switch records whether it holds lock, to avoid deadlock and repeated unlock

More details: checkout our paper

- Design details
  - Switch data plane design
  - Cases of different requests (e.g., cache eviction)
  - Handling failures of different components
  - Correctness proof
  - Corner cases of ownership migration
- Implementation details
  - Network stack (RDMA)
  - Cache organization (set-associative)
  - Global memory allocation (two-level allocation)
  - Synchronization primitive (rwlock)

Outline

- Background & Motivation
- Concordia: a DSM with In-Network Coherence
- Results
- Summary & Conclusion

Experimental Setup

Hardware Platform

Server * 8

CPU	2 Intel Xeon E5-2650 CPUs
NIC	100Gbps Mellanox ConnectX-5
OS	CentOS 7.4 , Linux kernel 3.10.0

Programmable Switch * 1

Type	Barefoot Tofino switch
Port	32 * 100Gbps
SW	bf-sde-8.2.0

Experimental Setup

Hardware Platform

Server * 8

CPU	2 Intel Xeon E5-2650 CPUs
NIC	100Gbps Mellanox ConnectX-5
OS	CentOS 7.4 , Linux kernel 3.10.0

Programmable Switch * 1

Type	Barefoot Tofino switch
Port	32 * 100Gbps
SW	bf-sde-8.2.0

Compared DSMs

Grappa [ATC'15]	No caching, ship computation, RDMA-based
GAM [VLDB'18]	Caching, directory-based coherence, RDMA-based
Concordia-base	Servers manage coherence of all cache blocks

Experimental Setup

Hardware Platform

Server * 8

CPU	2 Intel Xeon E5-2650 CPUs
NIC	100Gbps Mellanox ConnectX-5
OS	CentOS 7.4 , Linux kernel 3.10.0

Programmable Switch * 1

Type	Barefoot Tofino switch
Port	32 * 100Gbps
SW	bf-sde-8.2.0

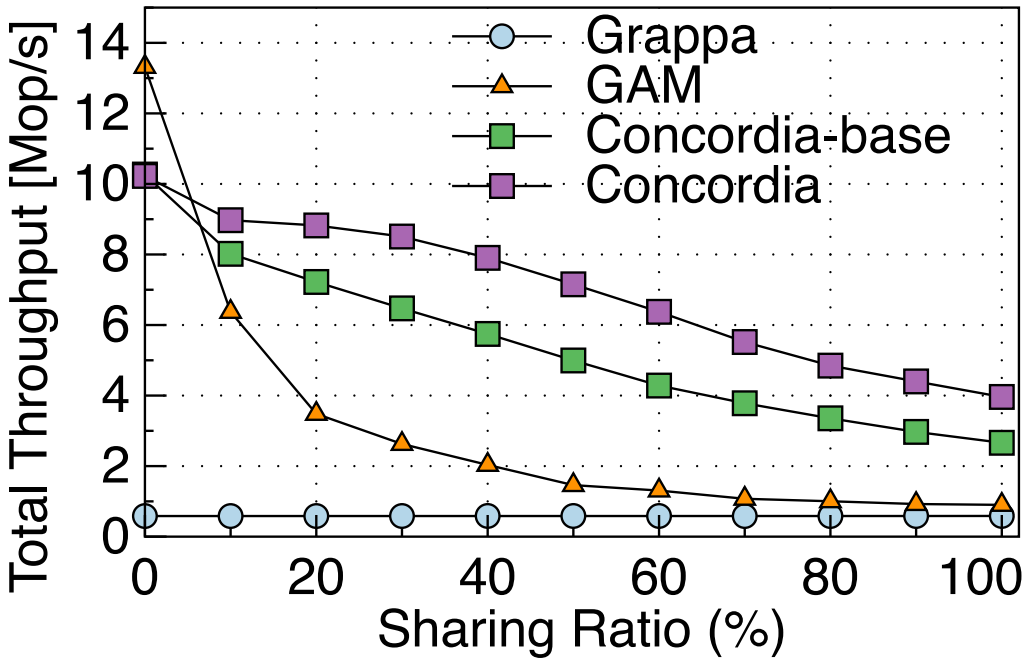
Compared DSMs

Grappa [ATC'15]	No caching, ship computation, RDMA-based
GAM [VLDB'18]	Caching, directory-based coherence, RDMA-based
Concordia-base	Servers manage coherence of all cache blocks

Benchmark

- Micro benchmarks (sharing ratio, locality ratio, reading ratio…)
- Applications (key-value store, transaction, graph computation)

Micro Benchmarks



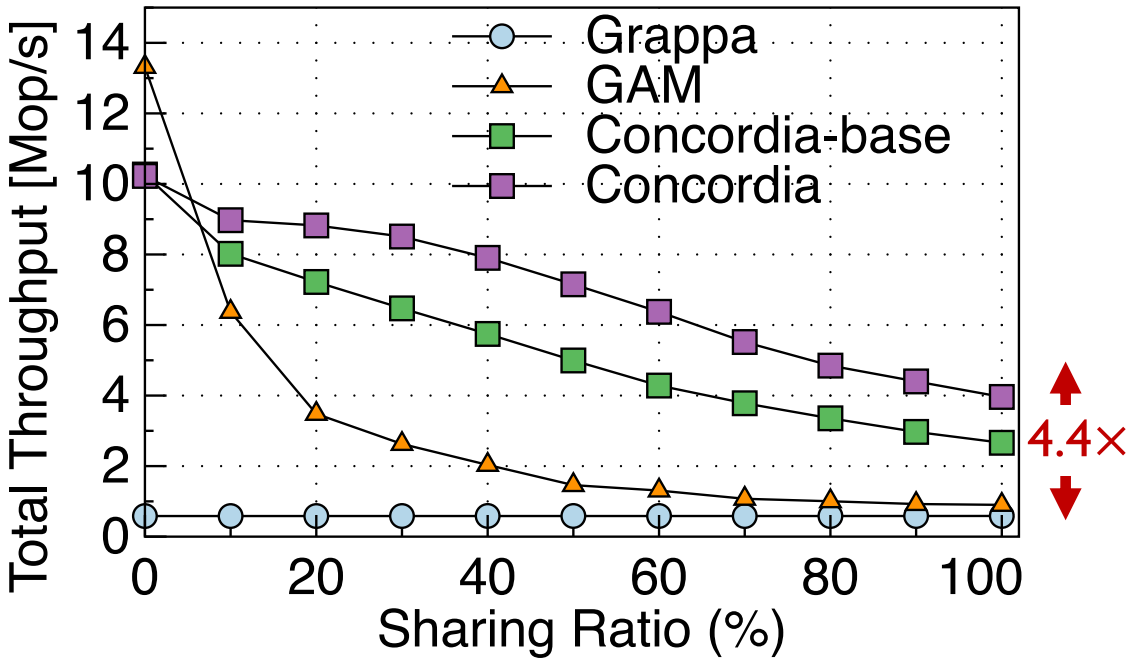
(A) Throughput

4 app threads per node,

8-bytes writes/reads (1:1), No locality

8GB working set, 1GB cache, 250MB shared areas

Micro Benchmarks



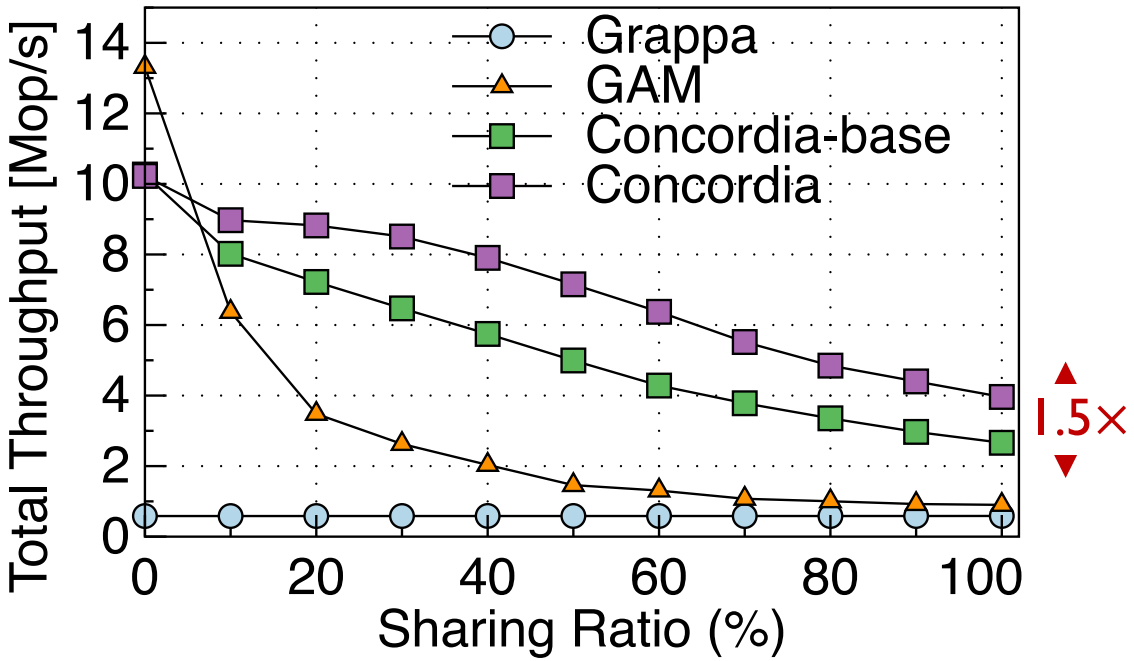
(A) Throughput

4 app threads per node,

8-bytes writes/reads (1:1), No locality

8GB working set, 1GB cache, 250MB shared areas

Micro Benchmarks



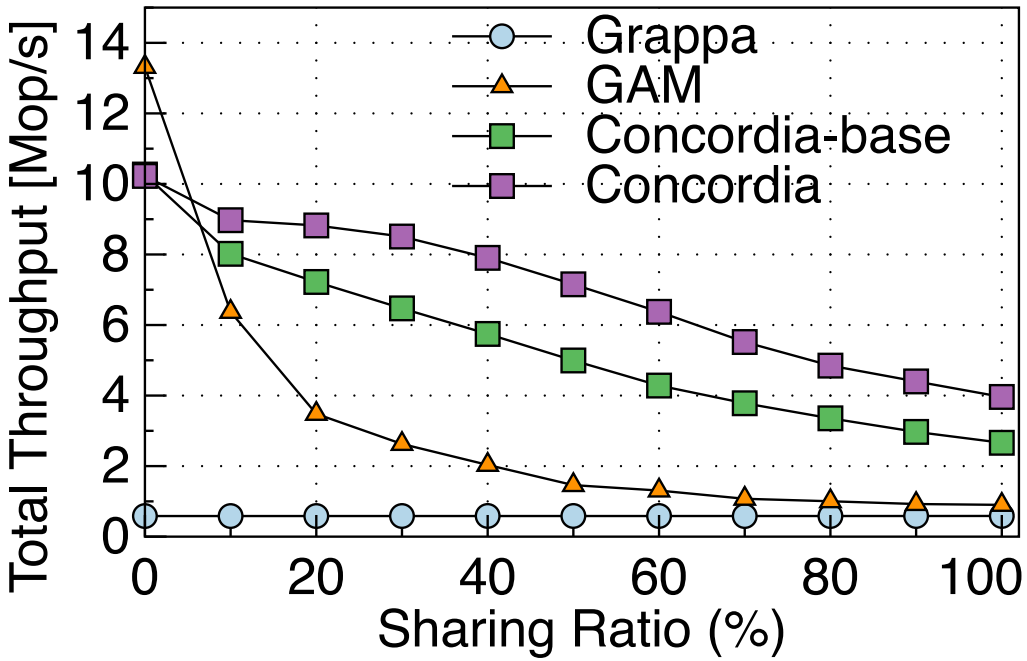
(A) Throughput

4 app threads per node,

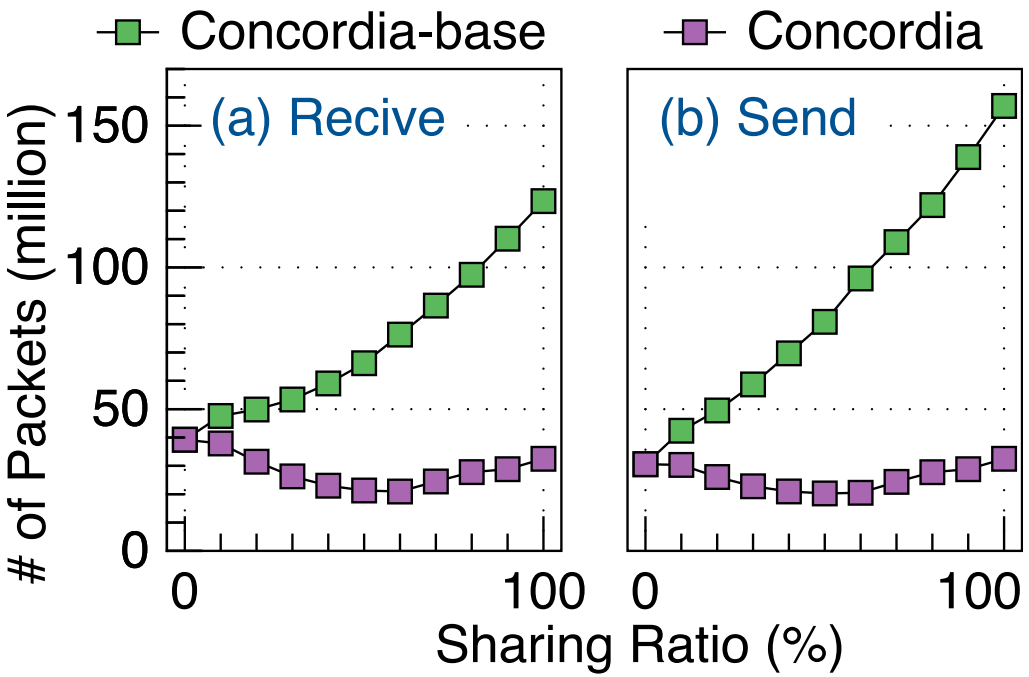
8-bytes writes/reads (1:1), No locality

8GB working set, 1GB cache, 250MB shared areas

Micro Benchmarks



(A) Throughput



(B) Packets received/sent by servers

4 app threads per node,

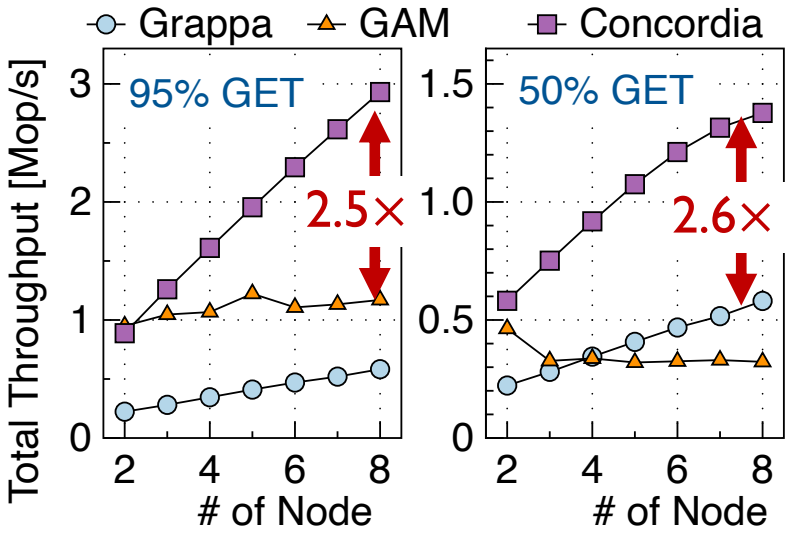
8-bytes writes/reads (1:1), No locality

8GB working set, 1GB cache, 250MB shared areas

Concordia effectively reduces network traffic

Applications

- Zipfan 0.99
- 8-byte key, 128-byte value
- Hash table index

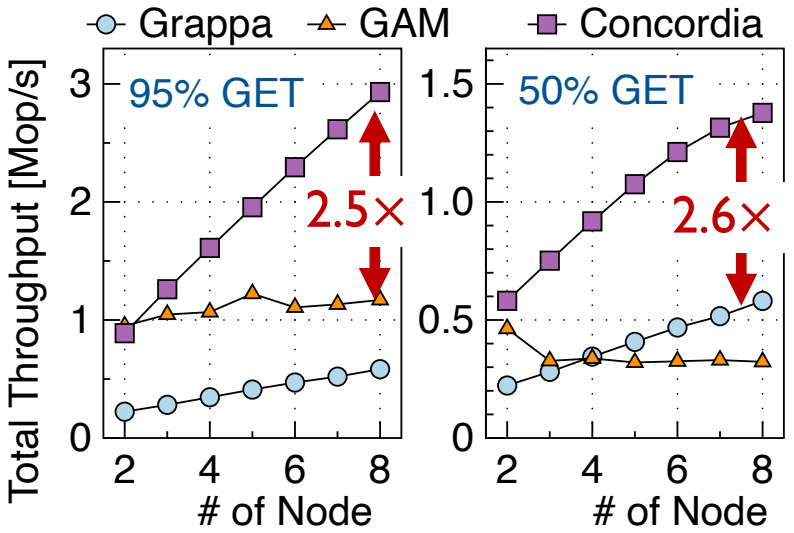


(A) Key-value Store (YCSB-like)

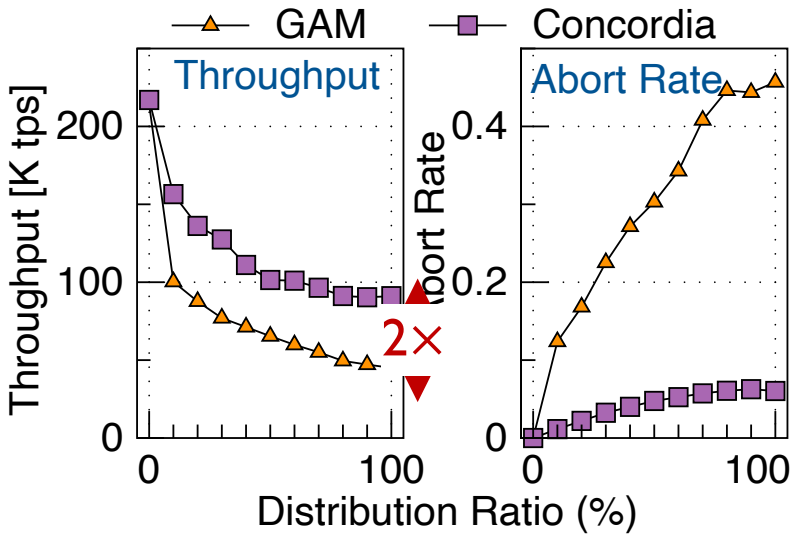
Applications

- Zipfan 0.99
- 8-byte key, 128-byte value
- Hash table index

- Ported from GAM
- Share everything
- 32 warehouses



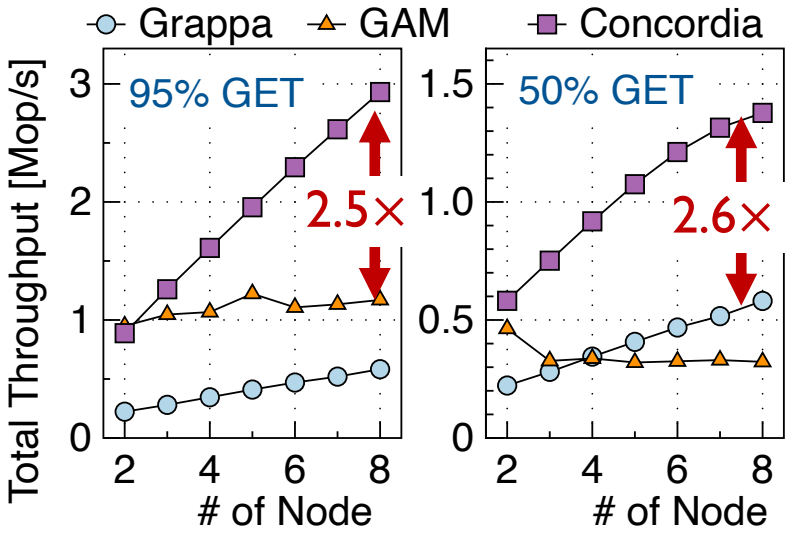
(A) Key-value Store (YCSB-like)



(B) Transaction (TPCC)

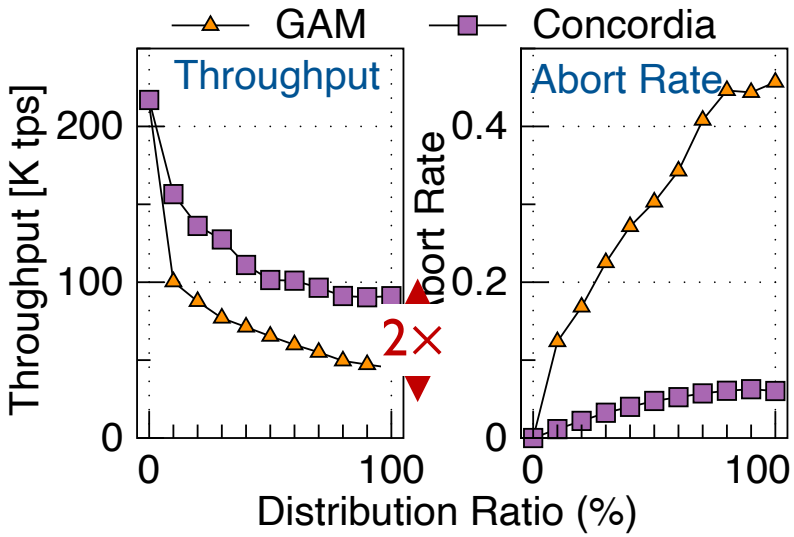
Applications

- Zipfan 0.99
- 8-byte key, 128-byte value
- Hash table index



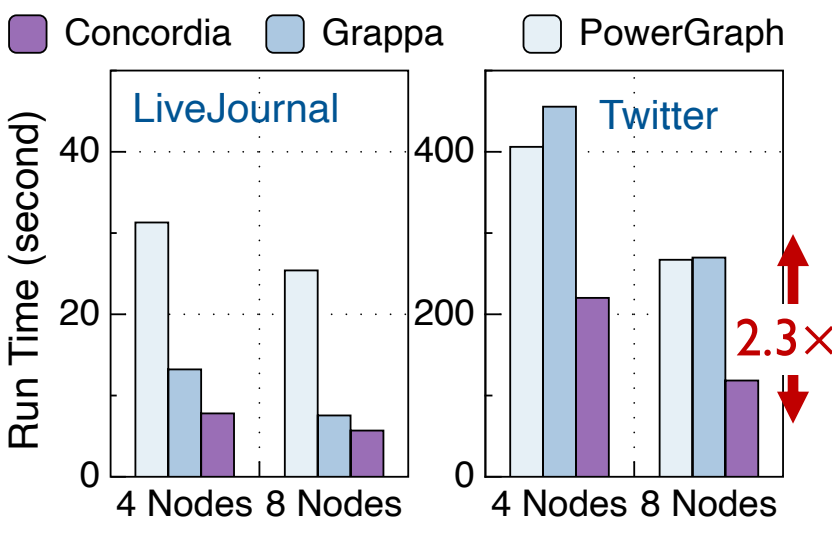
(A) Key-value Store (YCSB-like)

- Ported from GAM
- Share everything
- 32 warehouses



(B) Transaction (TPCC)

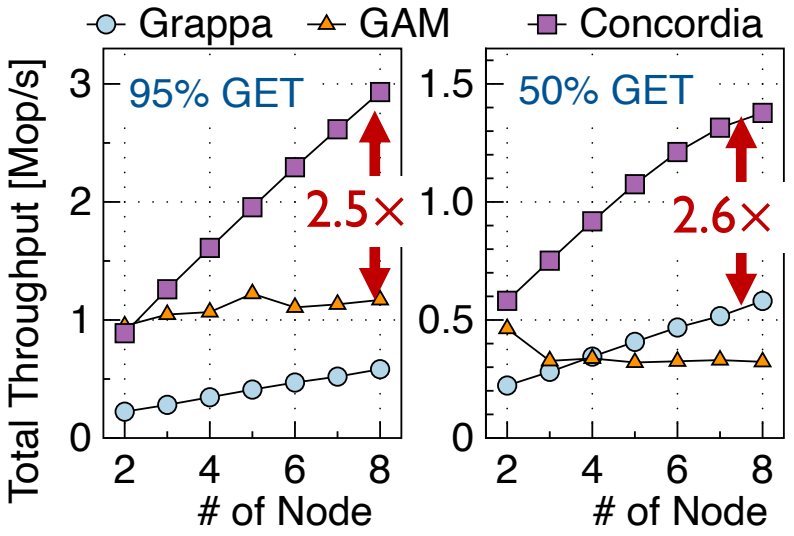
- Vertex-centric
- Gather, apply and scatter
- Two dataset



(C) Graph Comp (PageRank)

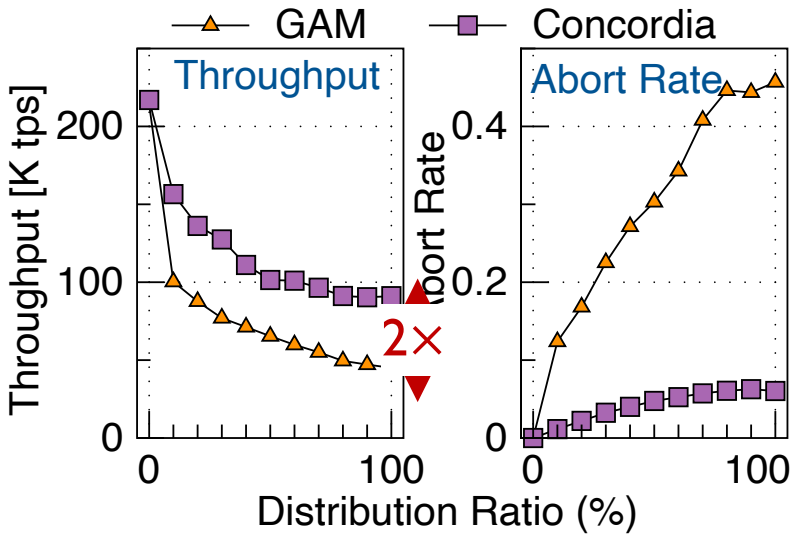
Applications

- Zipfan 0.99
- 8-byte key, 128-byte value
- Hash table index



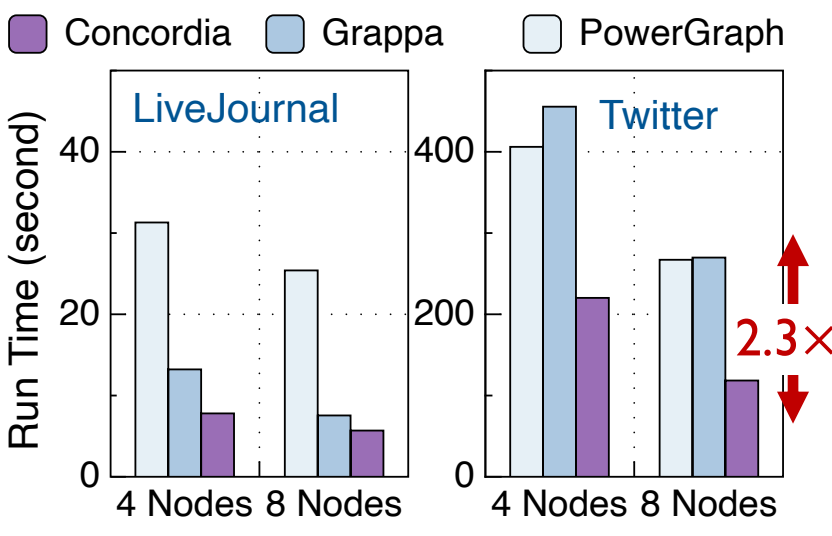
(A) Key-value Store (YCSB-like)

- Ported from GAM
- Share everything
- 32 warehouses



(B) Transaction (TPCC)

- Vertex-centric
- Gather, apply and scatter
- Two dataset



(C) Graph Comp (PageRank)

Concordia supports various applications, and boosts their performance

Outline

- Background & Motivation
- Concordia: a DSM with In-Network Coherence
- Results
- Summary & Conclusion

Summary & Conclusion

- Problem
  - Cache coherence is slow in modern DSM

Summary & Conclusion

- Problem
  - Cache coherence is slow in modern DSM
- Key Idea
  - Using programmable switches, with switch-server co-design

Summary & Conclusion

- Problem
  - Cache coherence is slow in modern DSM
- Key Idea
  - Using programmable switches, with switch-server co-design
- Techniques in Concordia
  - In-network coherence protocol
  - Ownership migration
  - Idempotent operations

Summary & Conclusion

- Problem
  - Cache coherence is slow in modern DSM
- Key Idea
  - Using programmable switches, with switch-server co-design
- Techniques in Concordia
  - In-network coherence protocol
  - Ownership migration
  - Idempotent operations
- Results
  - Concordia outperforms state-of-the-arts
  - Concordia underpins various applications

Thanks & QA

Concordia: Distributed Shared Memory with In-Network Cache Coherence

Contact Information: q-wang18@mails.tsinghua.edu.cn