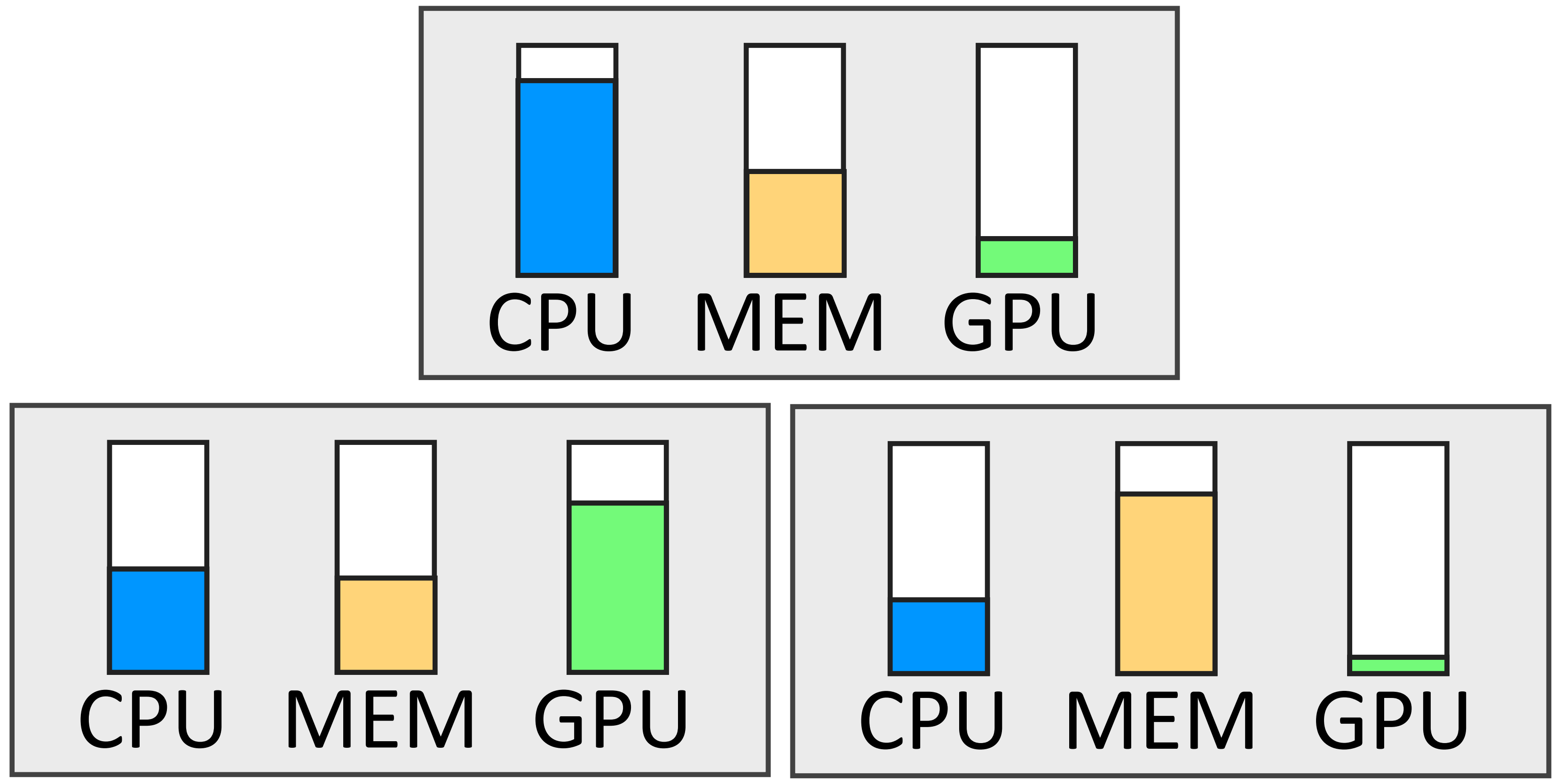
MIND: In-Network Memory Management for Disaggregated Data Centers

Seung-seob Lee, Yanpeng Yu, Yupeng Tang,
Anurag Khandelwal, Lin Zhong, Abhishek Bhattacharjee

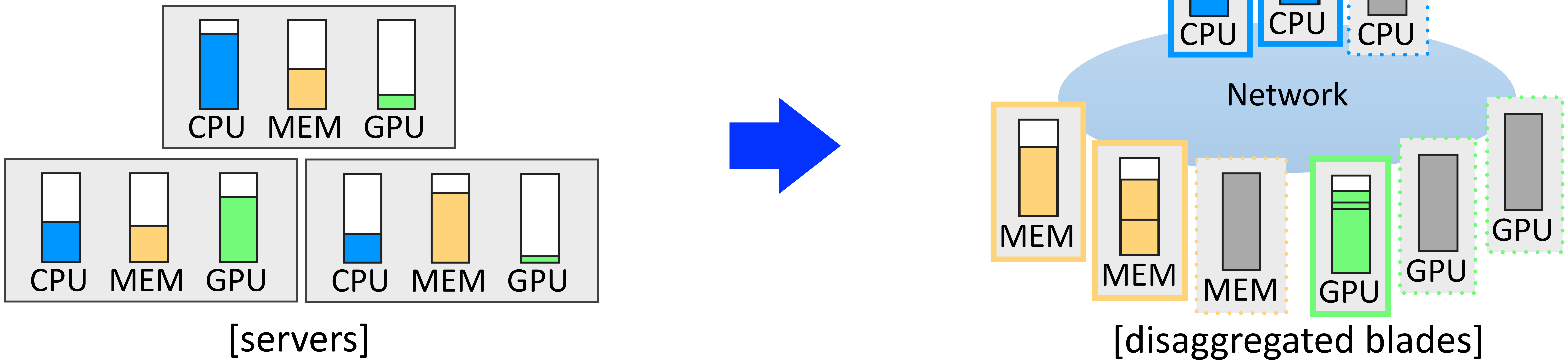
Yale University

Resource Disaggregation

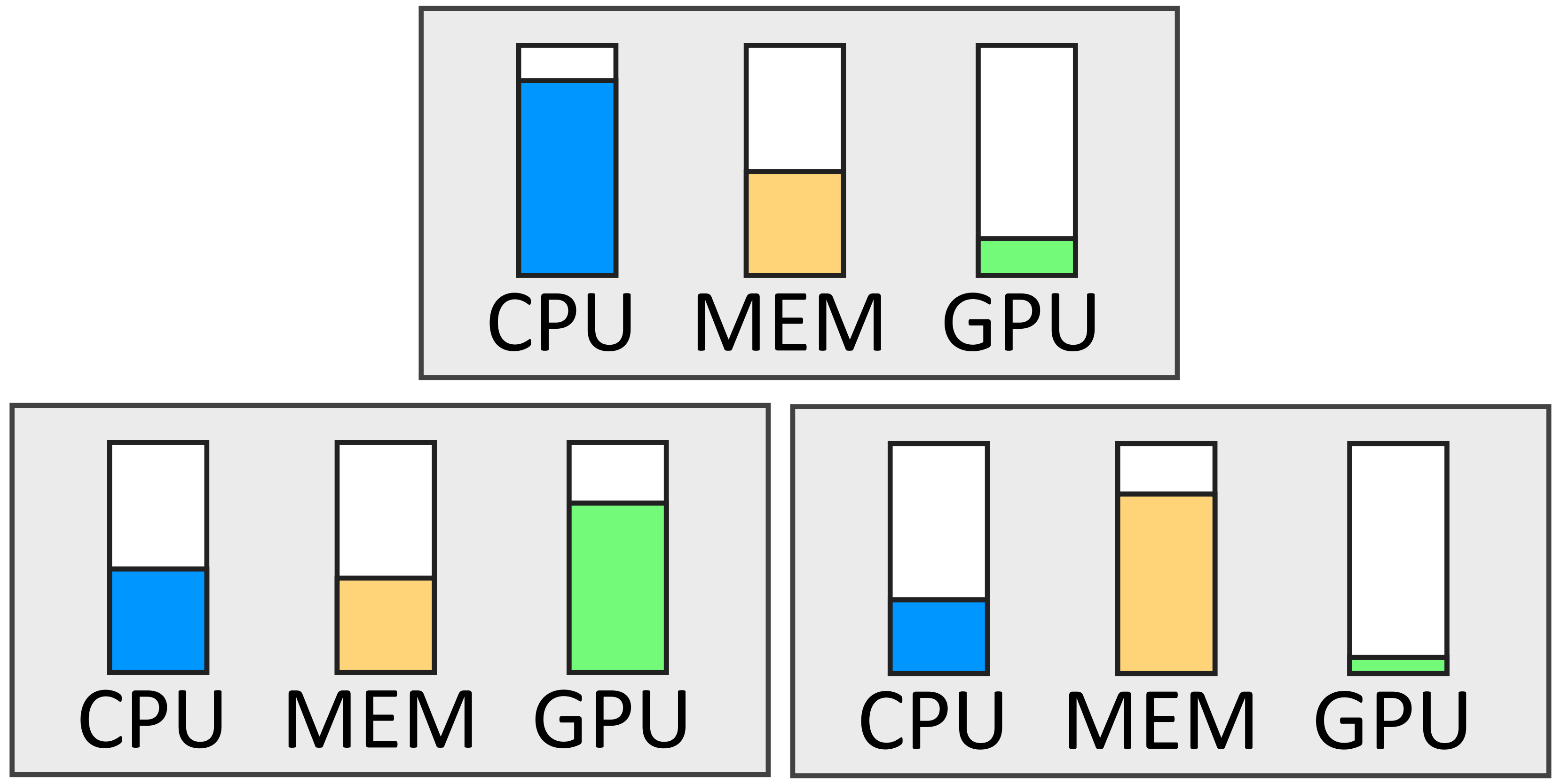


[servers]

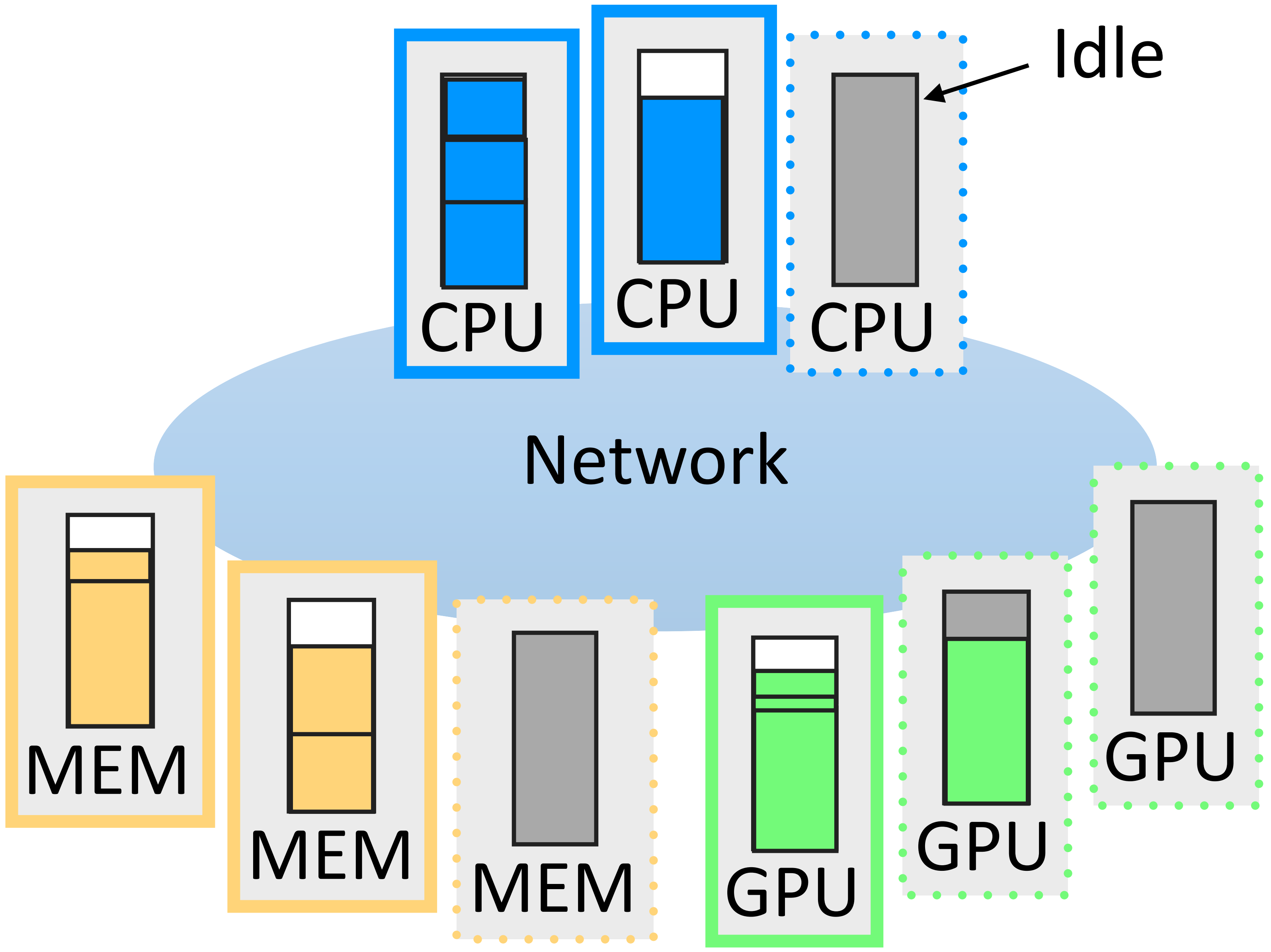
Resource Disaggregation



Resource Disaggregation

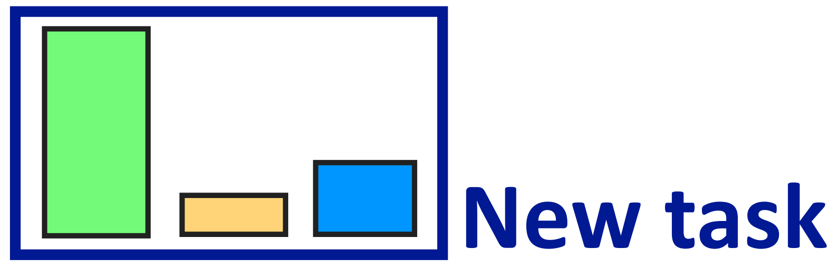


[servers]



[disaggregated blades]

- **Benefits of resource disaggregation**
  - High resource utilization
  - Easy to manage (modularity)
  - Elastic scalability



Memory Disaggregation

- **Need for memory disaggregation**

Low utilization
(as low as 30%)

Unbalanced usage
(> 70% of the time in clusters)

Energy consumption
(up to 46% of avg. system)

Memory Disaggregation

- **Need for memory disaggregation**

Low utilization
(as low as 30%)

Unbalanced usage
(> 70% of the time in clusters)

Energy consumption
(up to 46% of avg. system)

- **Timely problem**

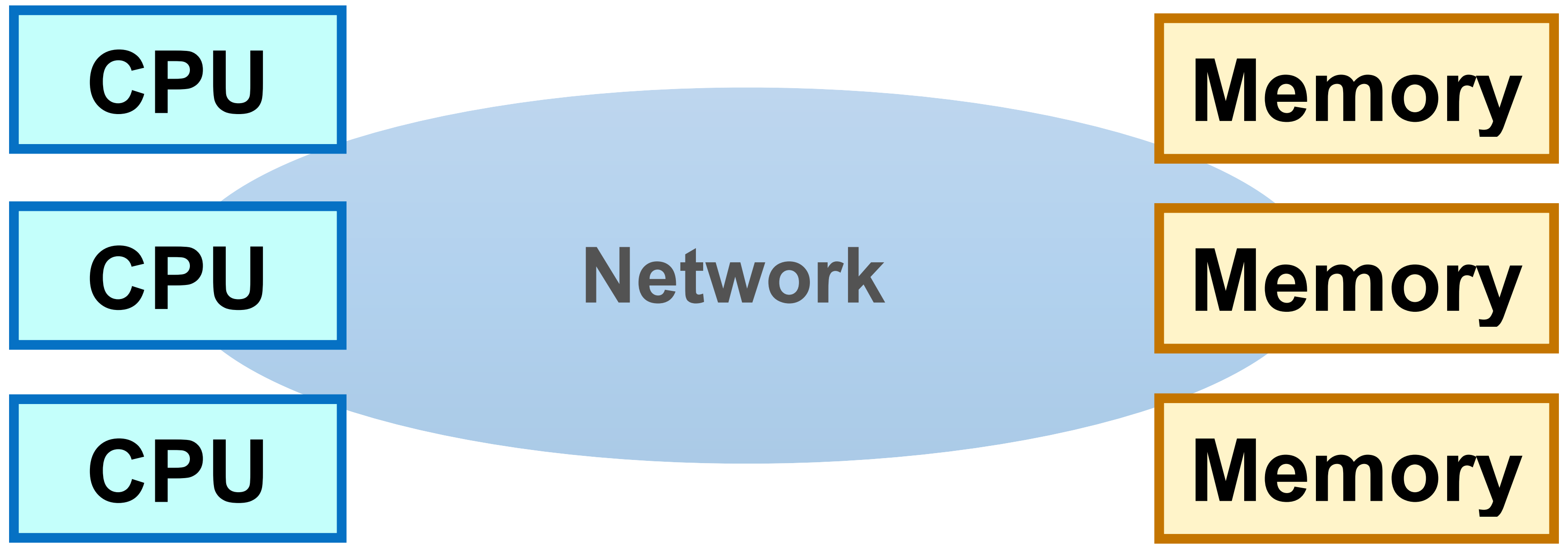
snowflake
amazon S3

Faster & harder to disaggregate →

	Storage (NVMe SSD)	**Network**	**Memory (DDR4)**	**L3 cache**
Data rate	5 GB/s (PCIe 4.0)	25 GB/s (PCIe 4.0)	25.6 GB/s (per channel)	370 GB/s
Latency	45 / 450 us (R/W)	~5 us (4KB)	<50 ns	11 ns

Storage disaggregation is already popular

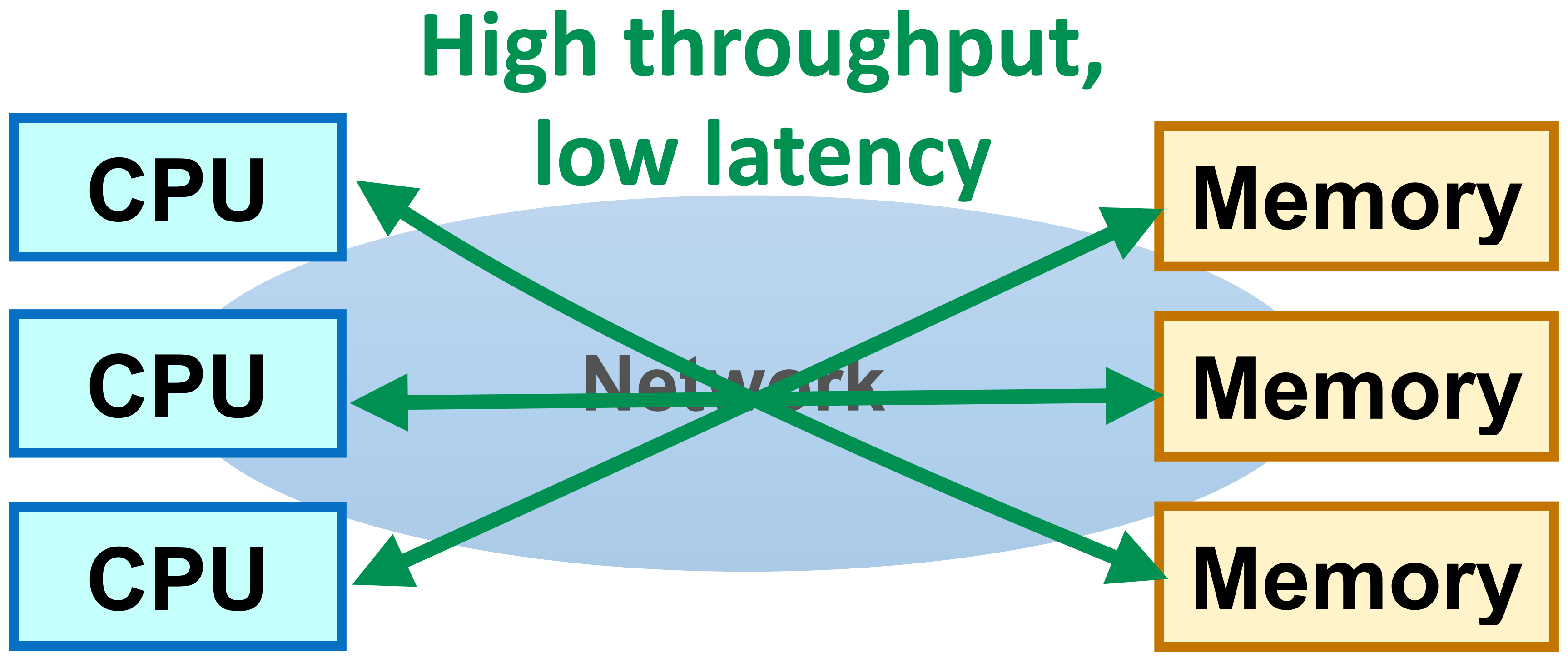
Memory Disaggregation Goals



[disaggregated blades]

Memory Disaggregation Goals

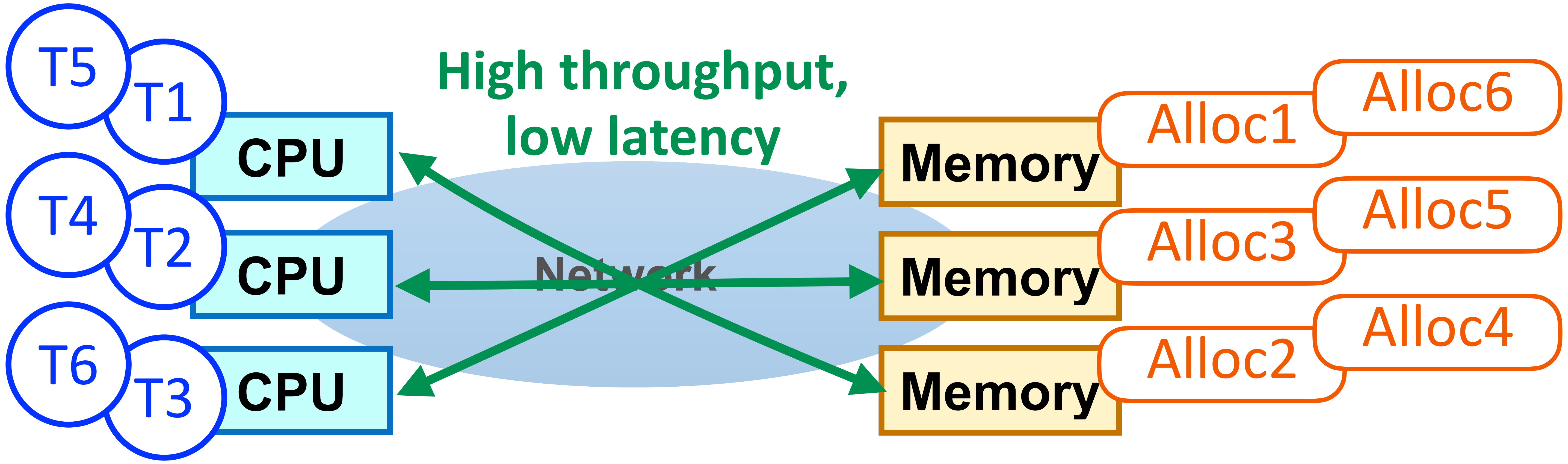
- **Performance (CPU ↔ memory)**: high throughput & low latency



[disaggregated blades]

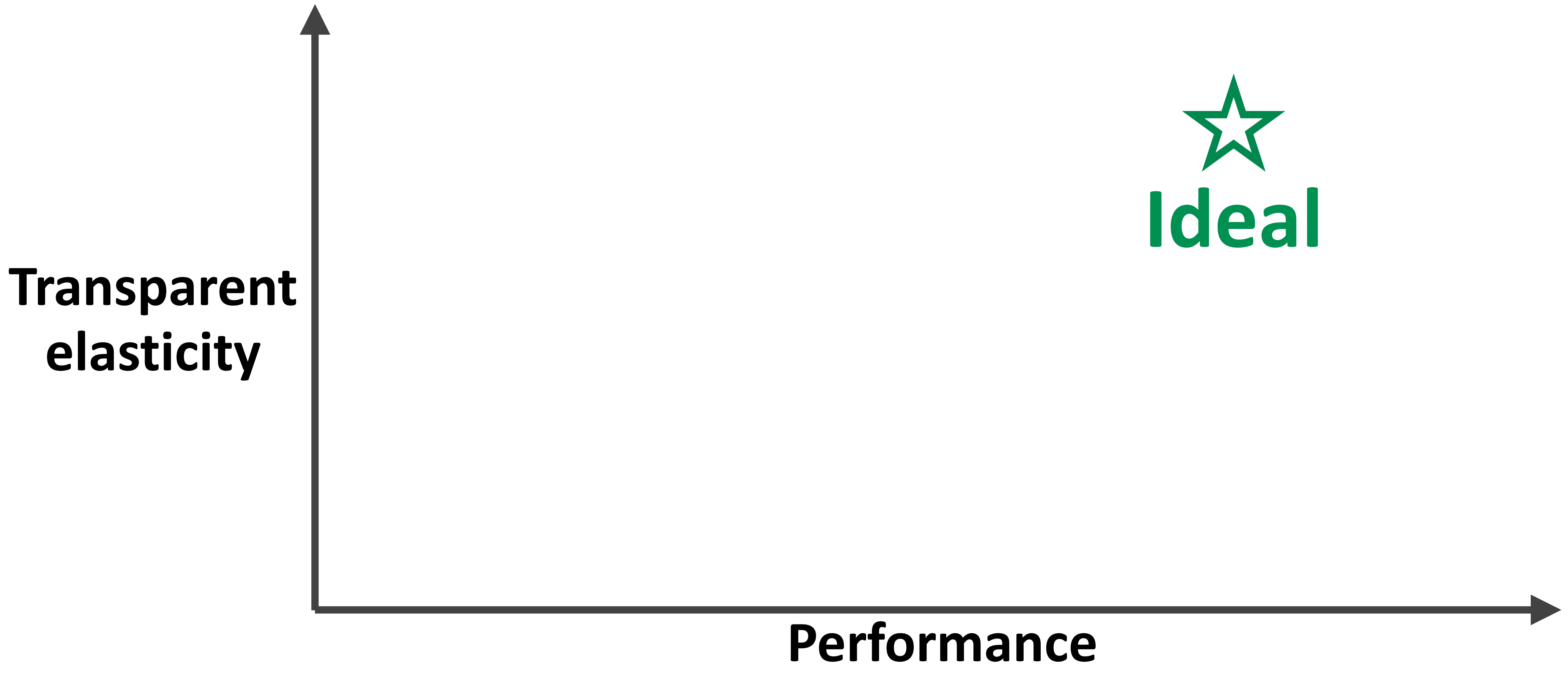
Memory Disaggregation Goals

- **Performance (CPU ↔ memory)**: high throughput & low latency
- **Transparent elasticity**: flexible resource allocation

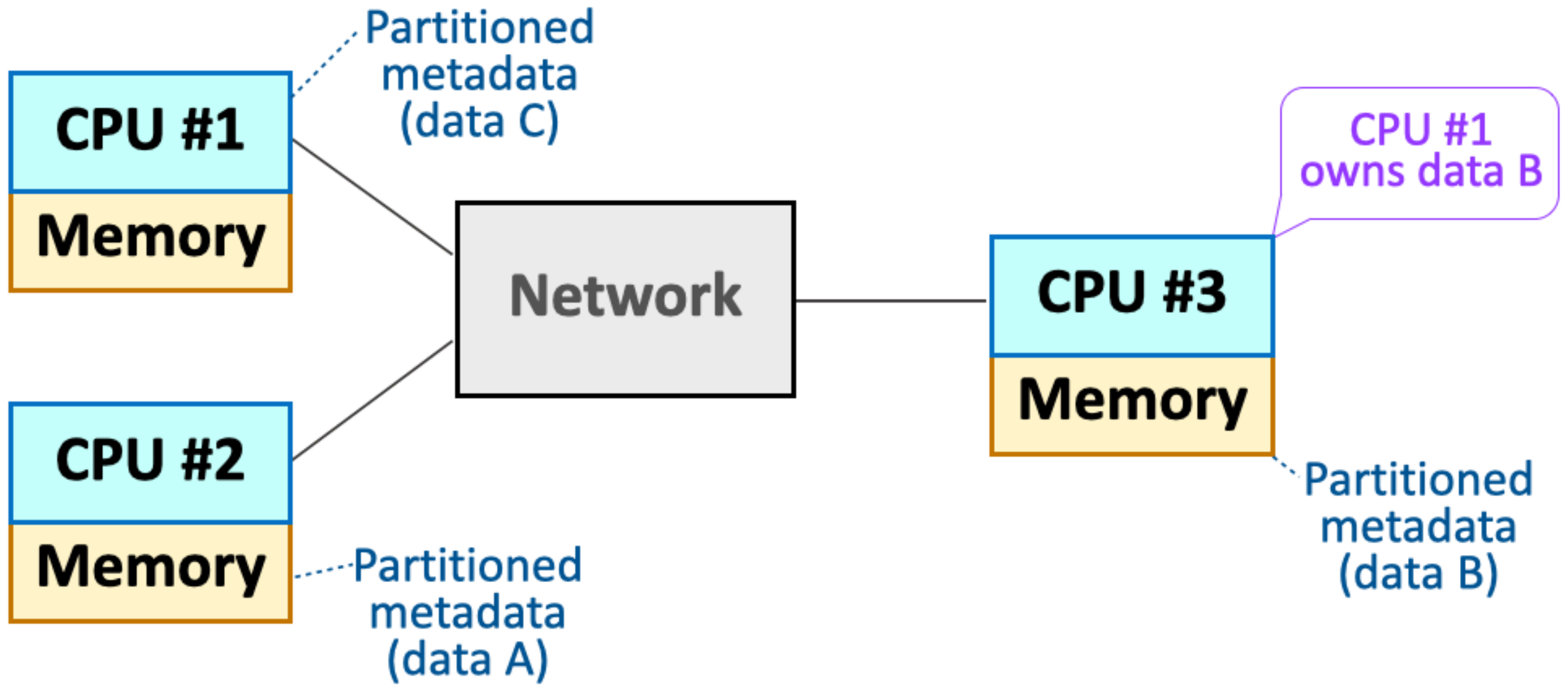


[disaggregated blades]

Existing Approaches

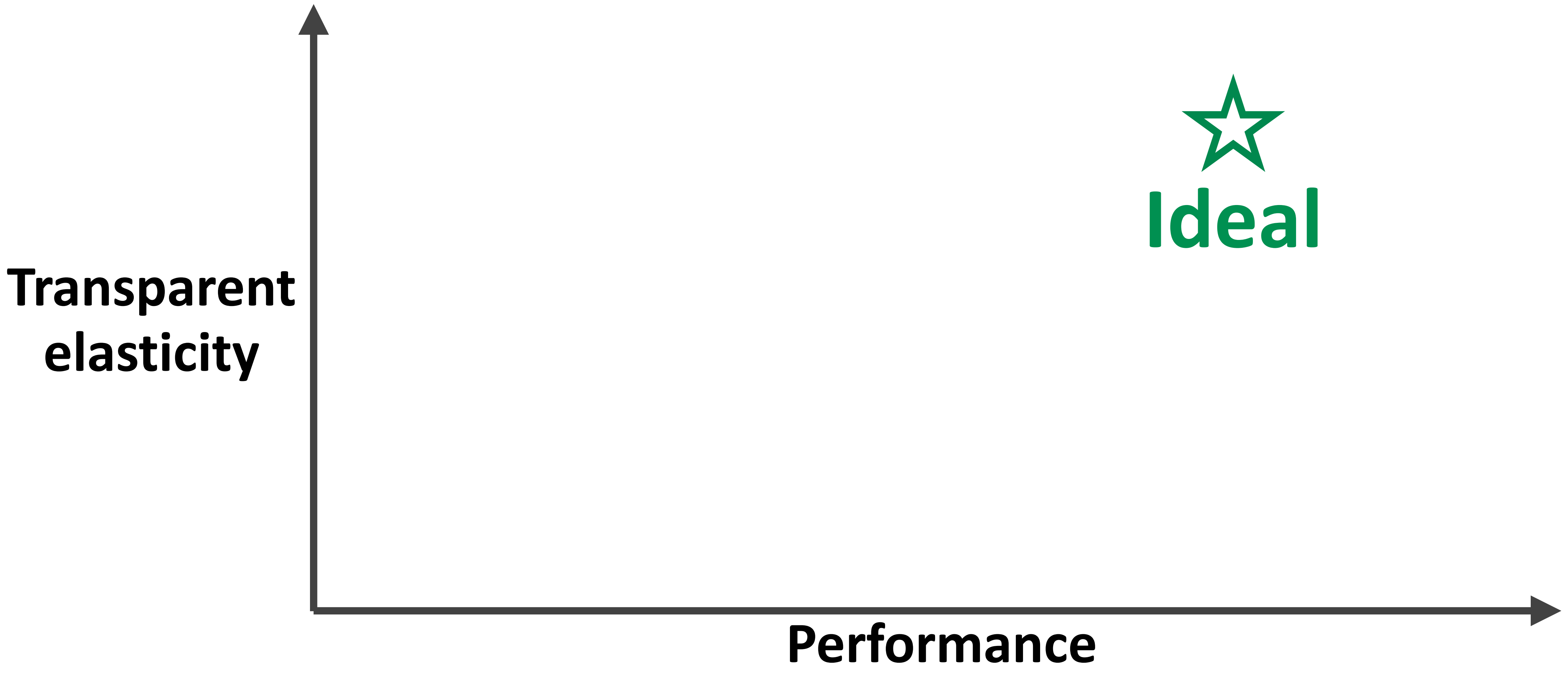


Existing Approaches

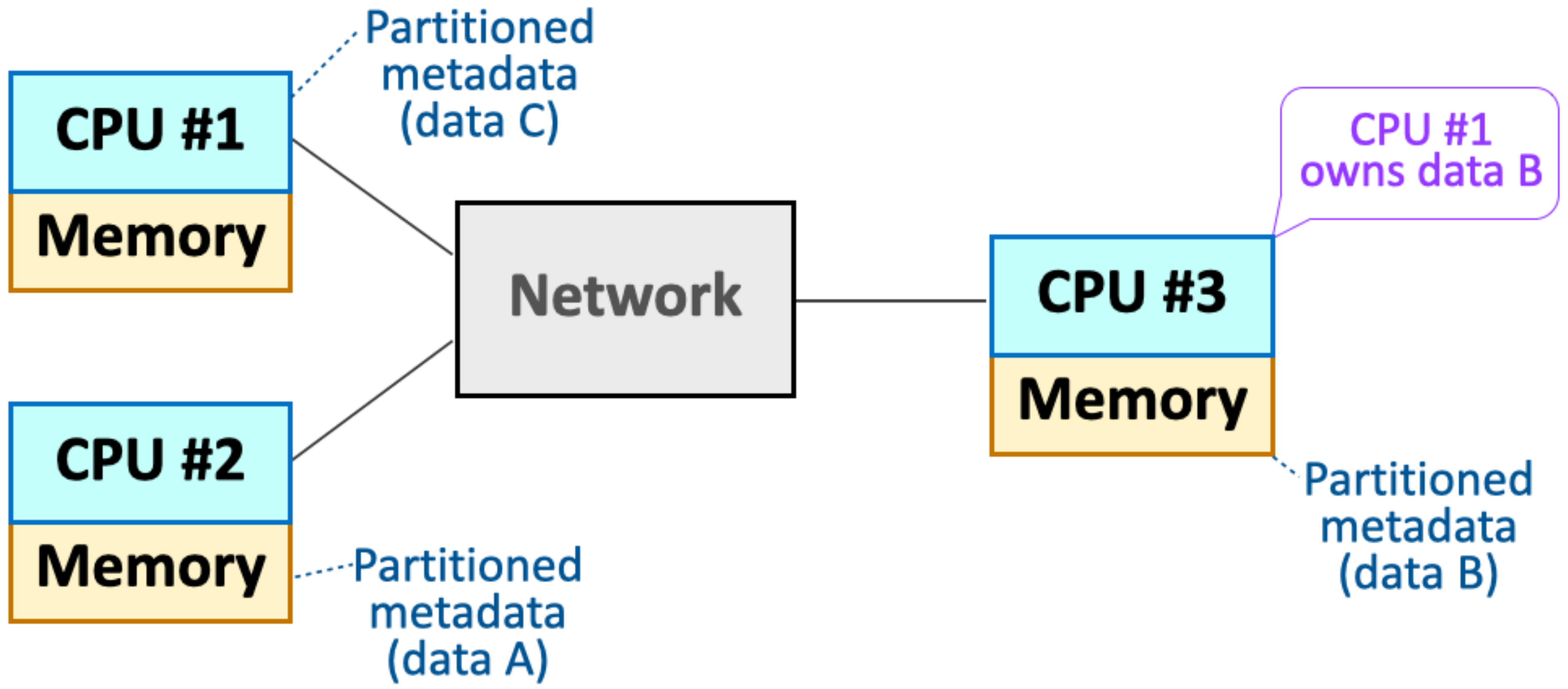


Partitioned metadata (data C)
CPU #1
Memory
Network
CPU #1 owns data B
CPU #3
Memory
CPU #2
Memory
Partitioned metadata (data B)
Partitioned metadata (data A)

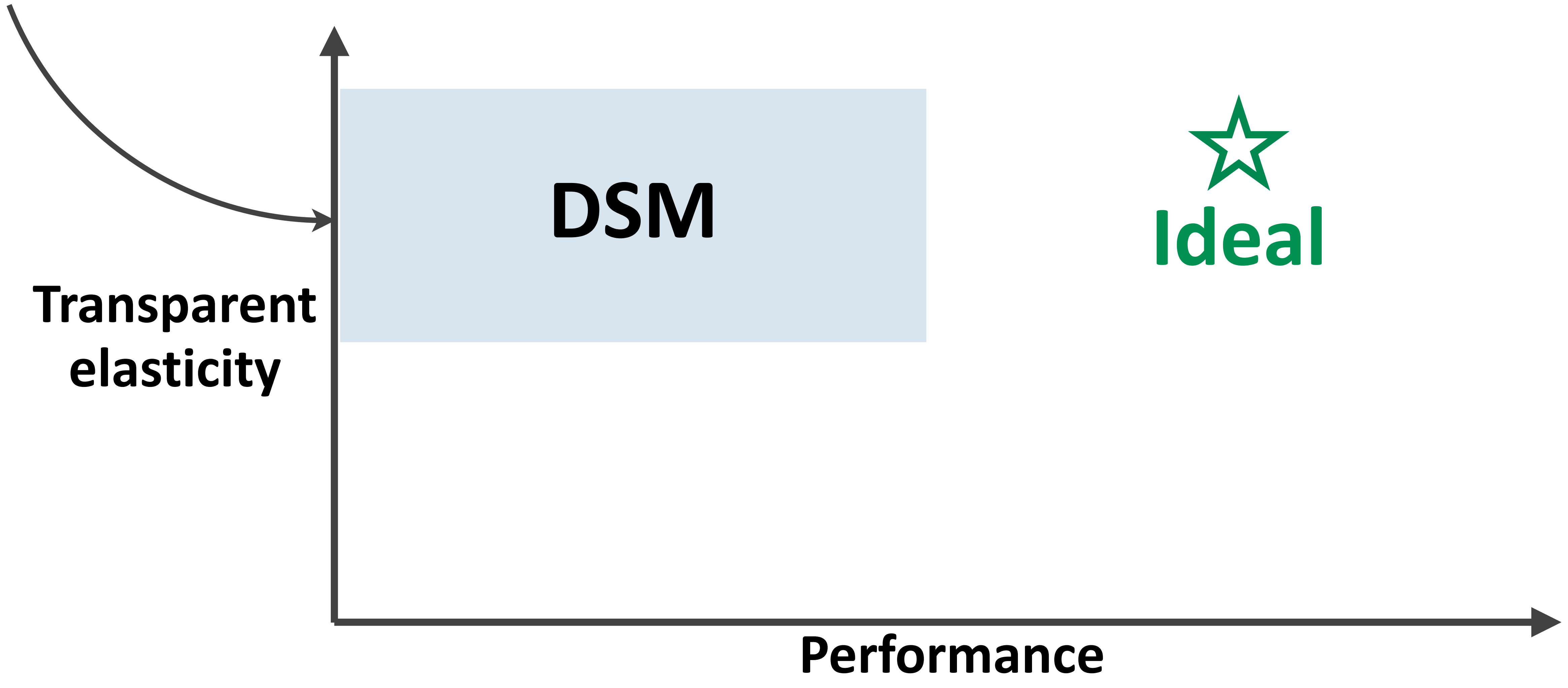
Distributed shared memory (DSM)



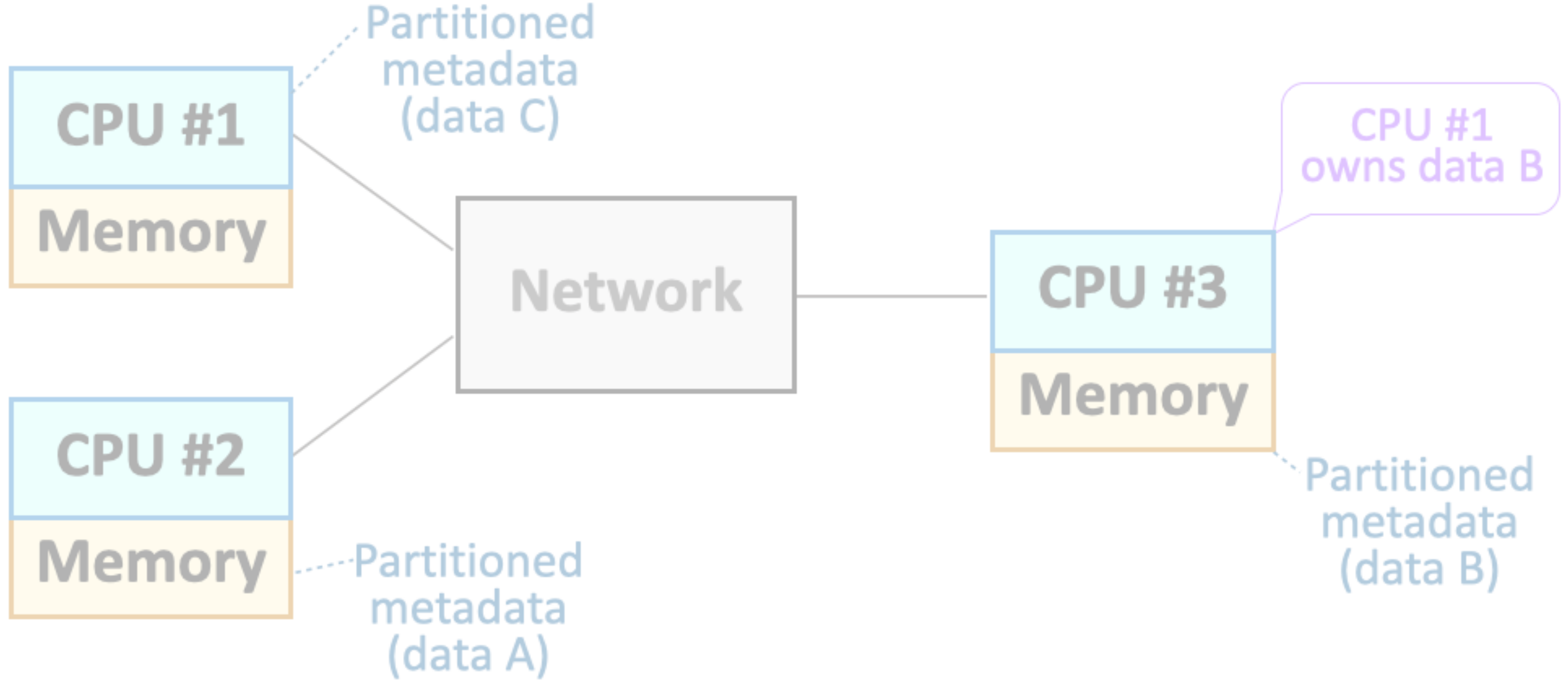
Existing Approaches


Partitioned metadata (data C)
CPU #1
Memory
Network
CPU #1 owns data B
CPU #3
Memory
CPU #2
Memory
Partitioned metadata (data B)
Partitioned metadata (data A)

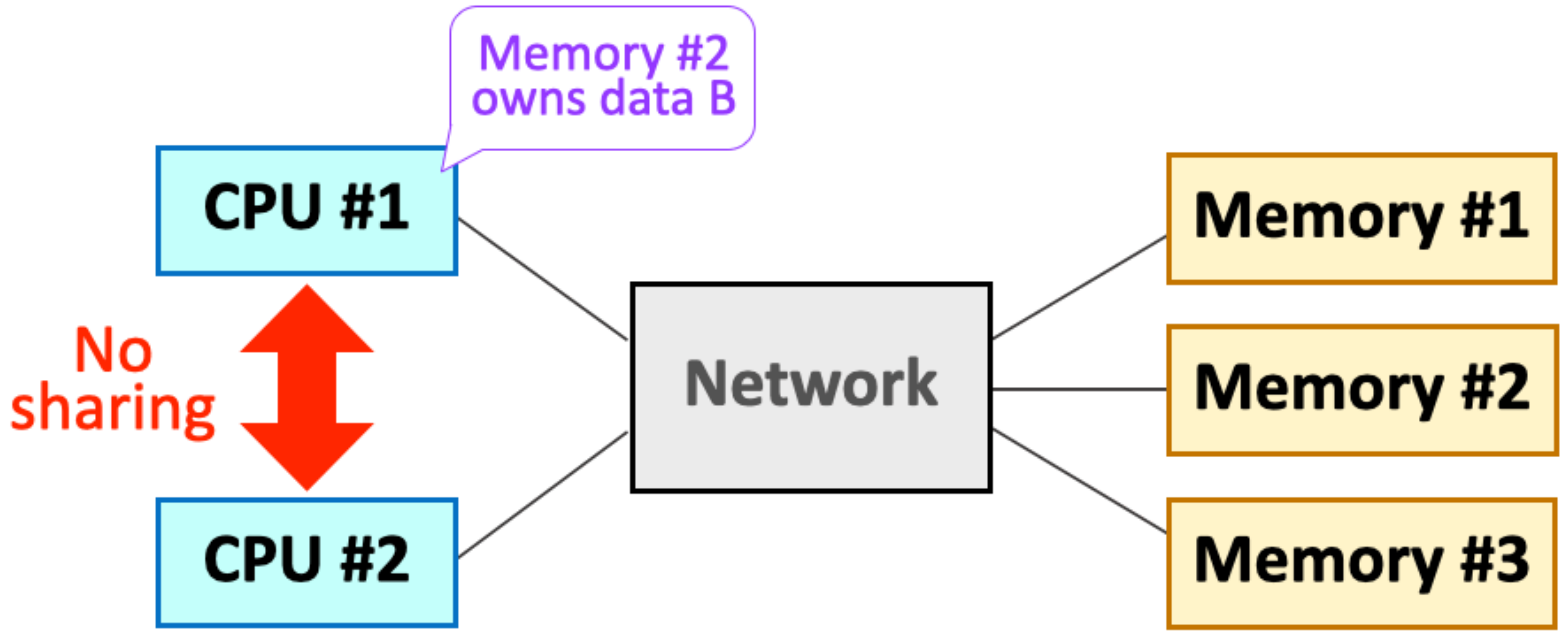
Distributed shared memory (DSM)



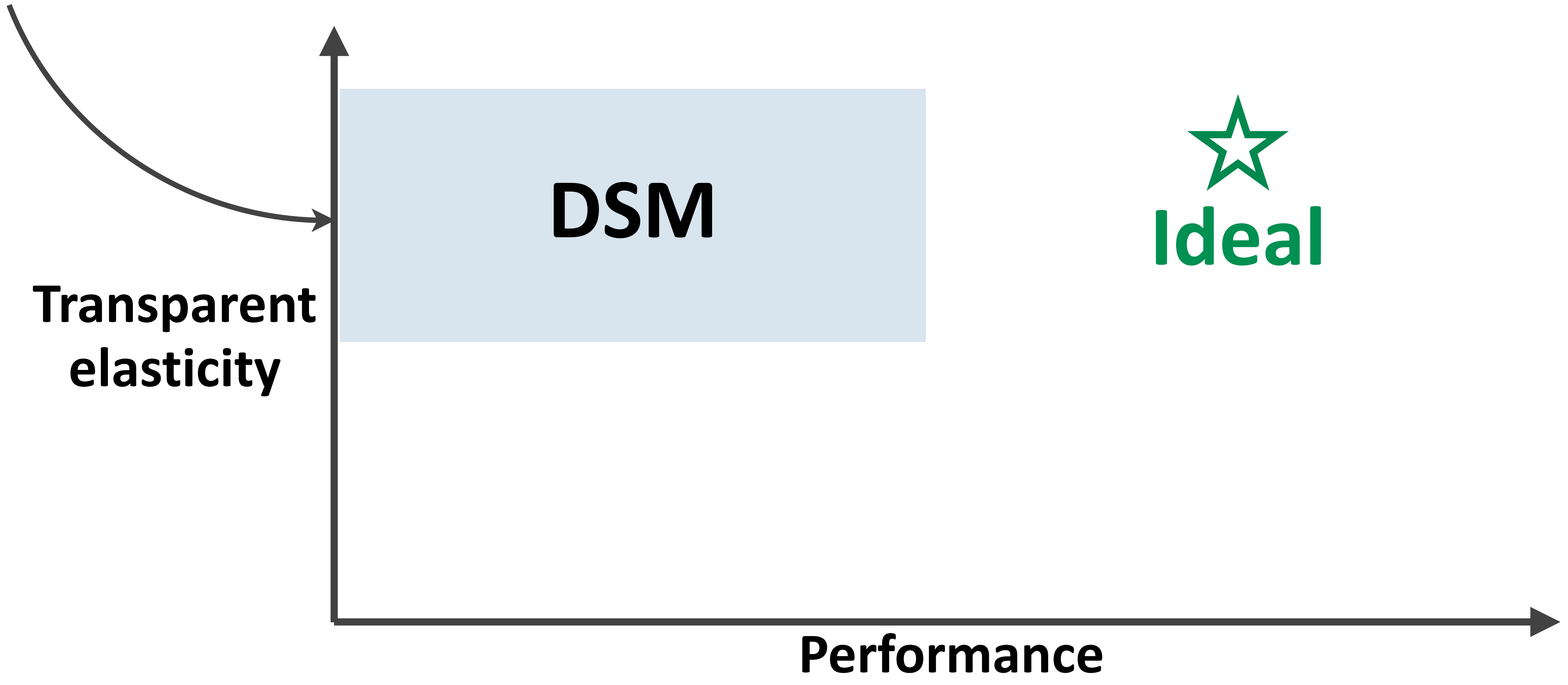
Existing Approaches


CPU #1
Memory
Partitioned metadata (data C)
Network
CPU #1 owns data B
CPU #3
Memory
Partitioned metadata (data B)
CPU #2
Memory
Partitioned metadata (data A)

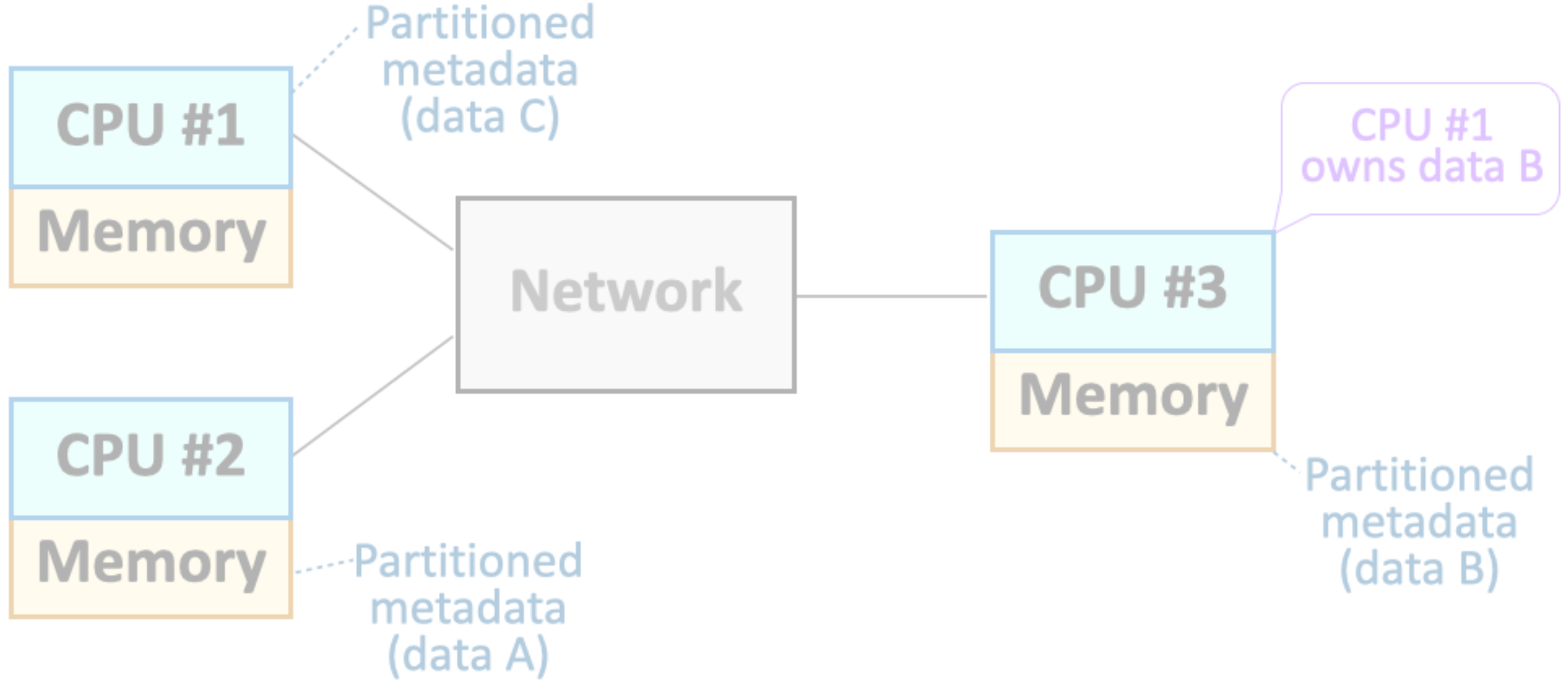
Distributed shared memory (DSM)



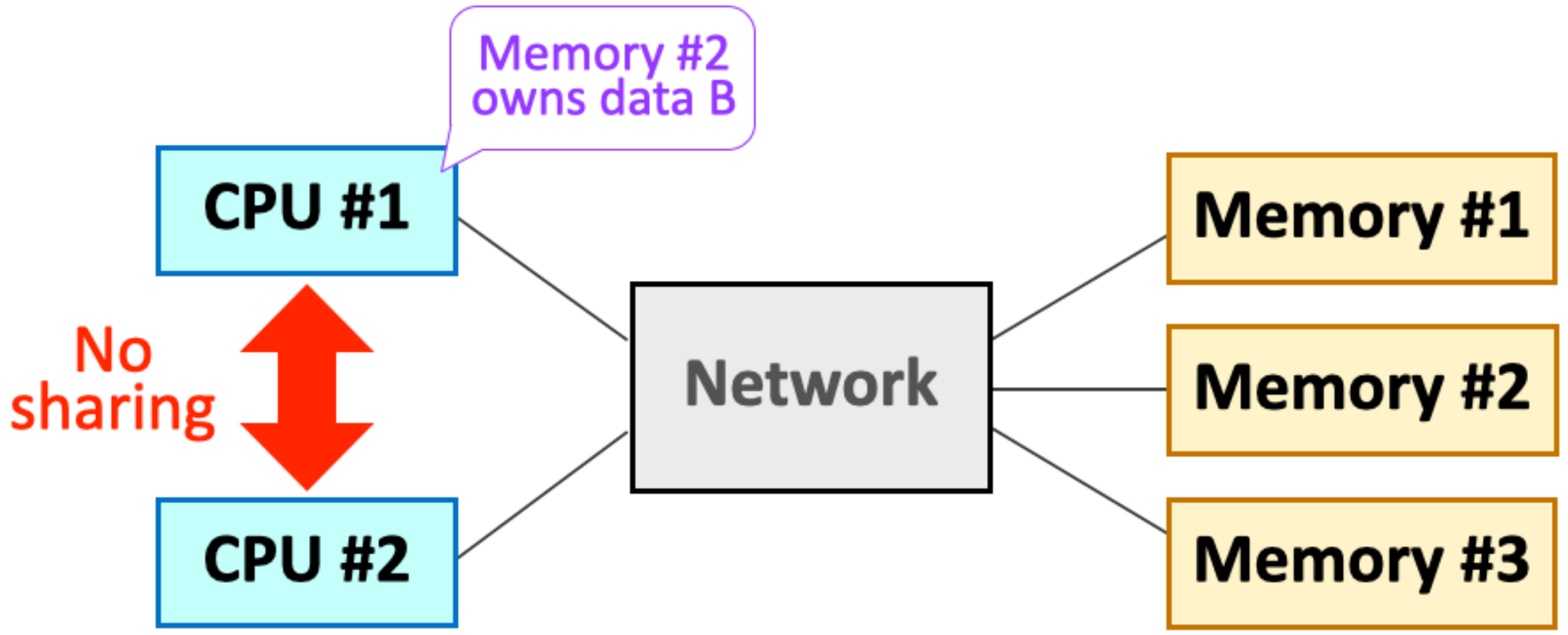
Recent disaggregated memory schemes



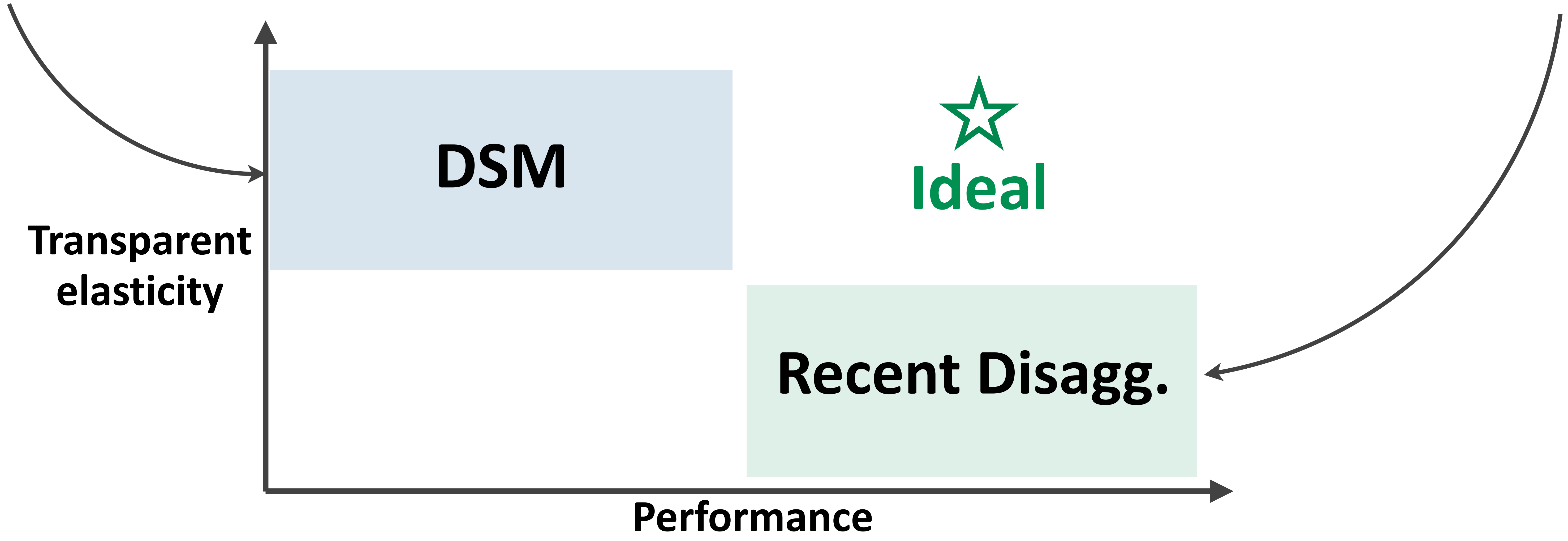
Existing Approaches


Partitioned metadata (data C)
CPU #1
Memory
Network
CPU #1 owns data B
CPU #3
Memory
CPU #2
Memory
Partitioned metadata (data A)
Partitioned metadata (data B)

Distributed shared memory (DSM)

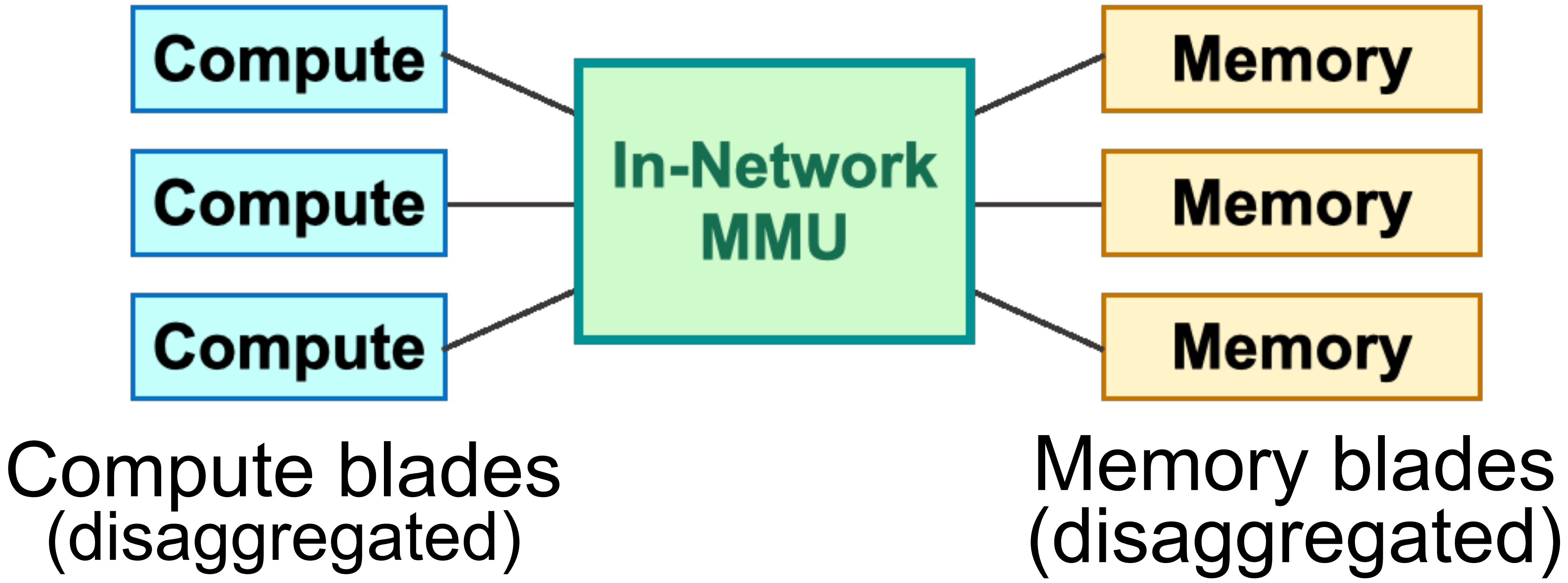


Recent disaggregated memory schemes



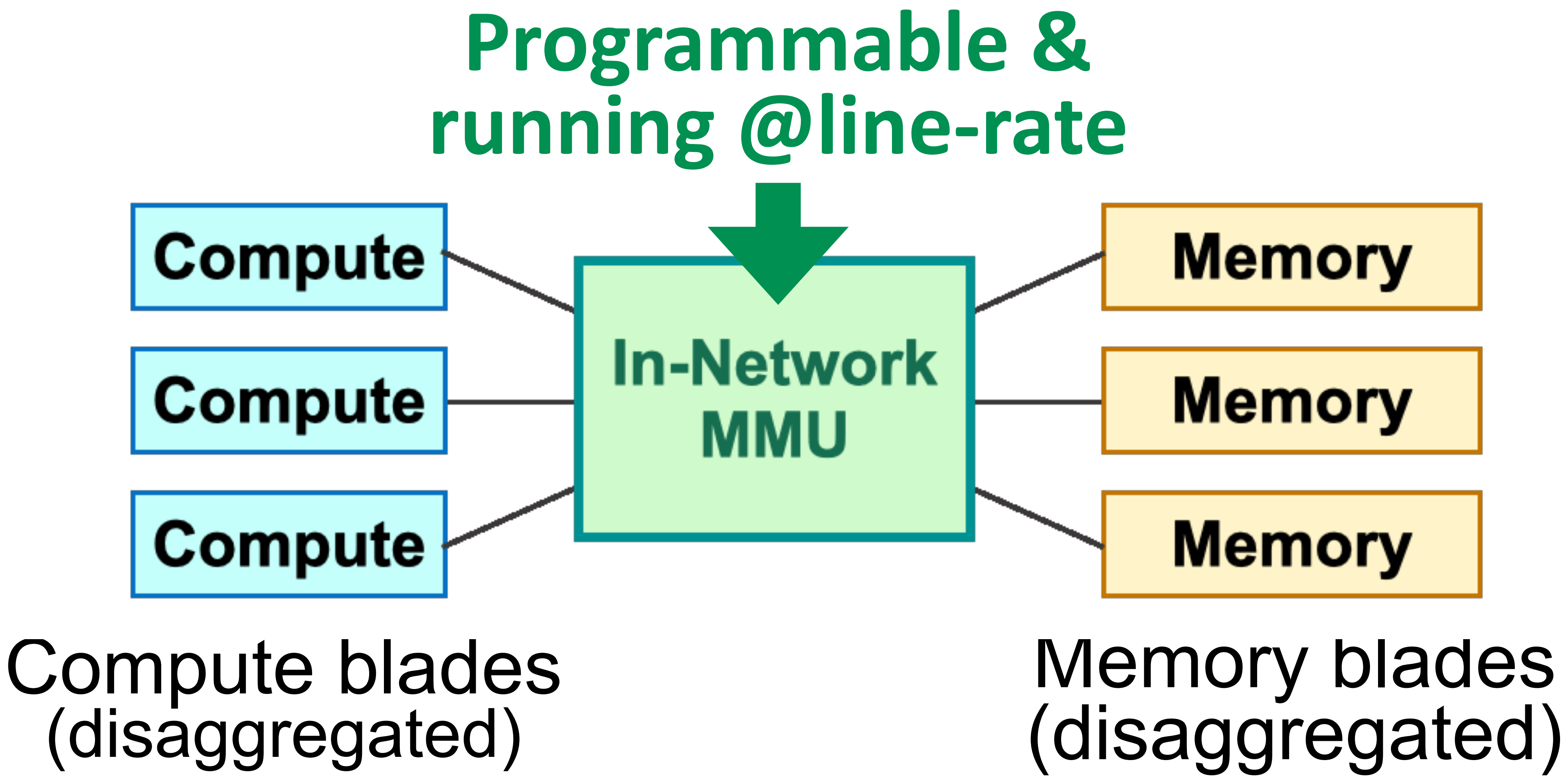
Key Insight — In-network Memory management

- **Central location:** global view & processing directly in the data path



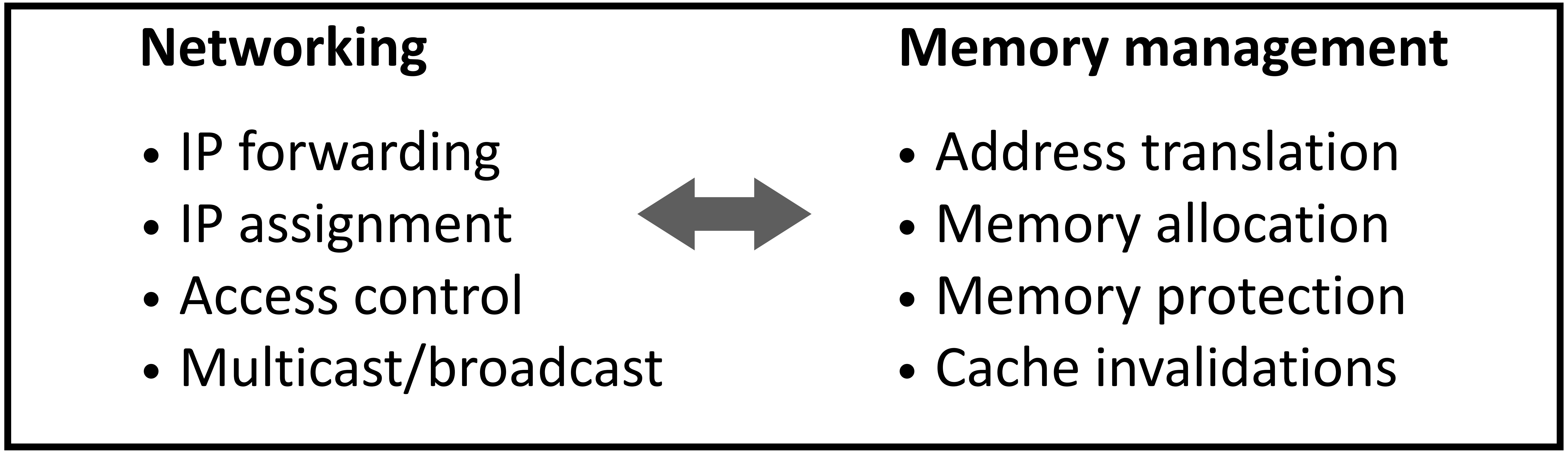
Key Insight — In-network Memory management

- **Central location:** global view & processing directly in the data path
- **Programmable switching ASIC:** flexible processing at line rate



Key Insight — In-network Memory management

- **Central location:** global view & processing directly in the data path
- **Programmable switching ASIC:** flexible processing at line rate
- **Similarity between network functions and memory management**



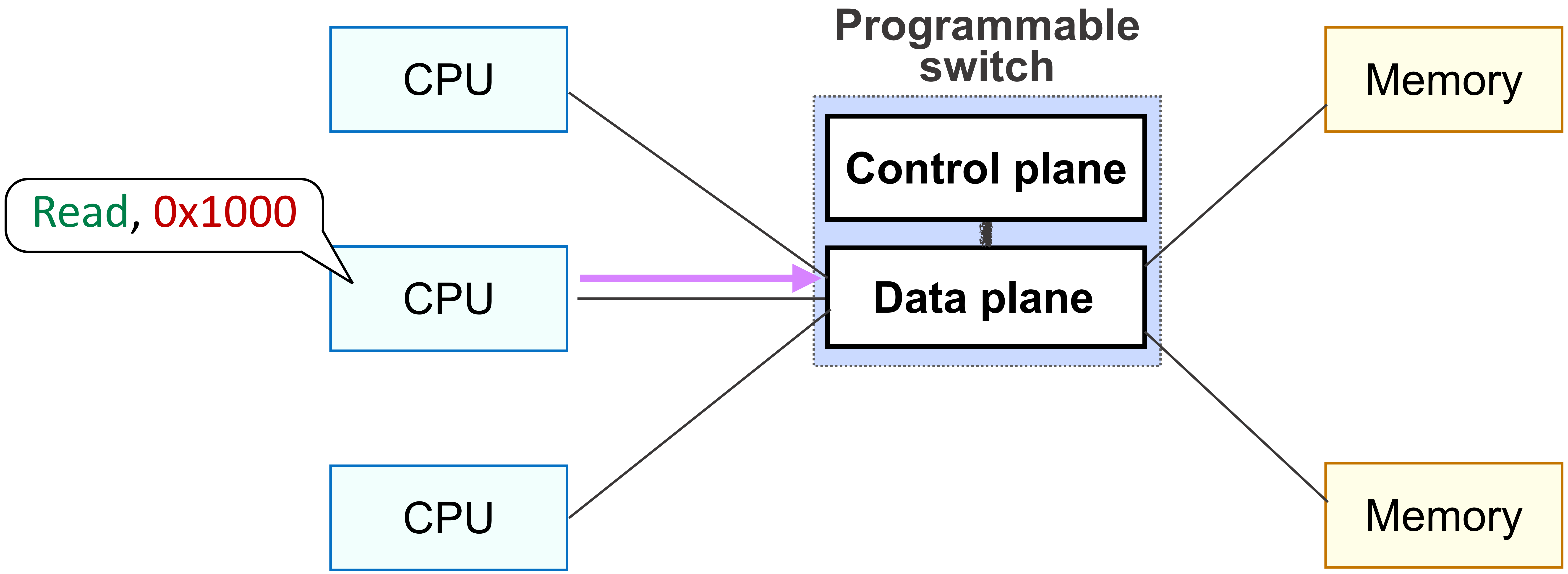
Networking

- IP forwarding
- IP assignment
- Access control
- Multicast/broadcast

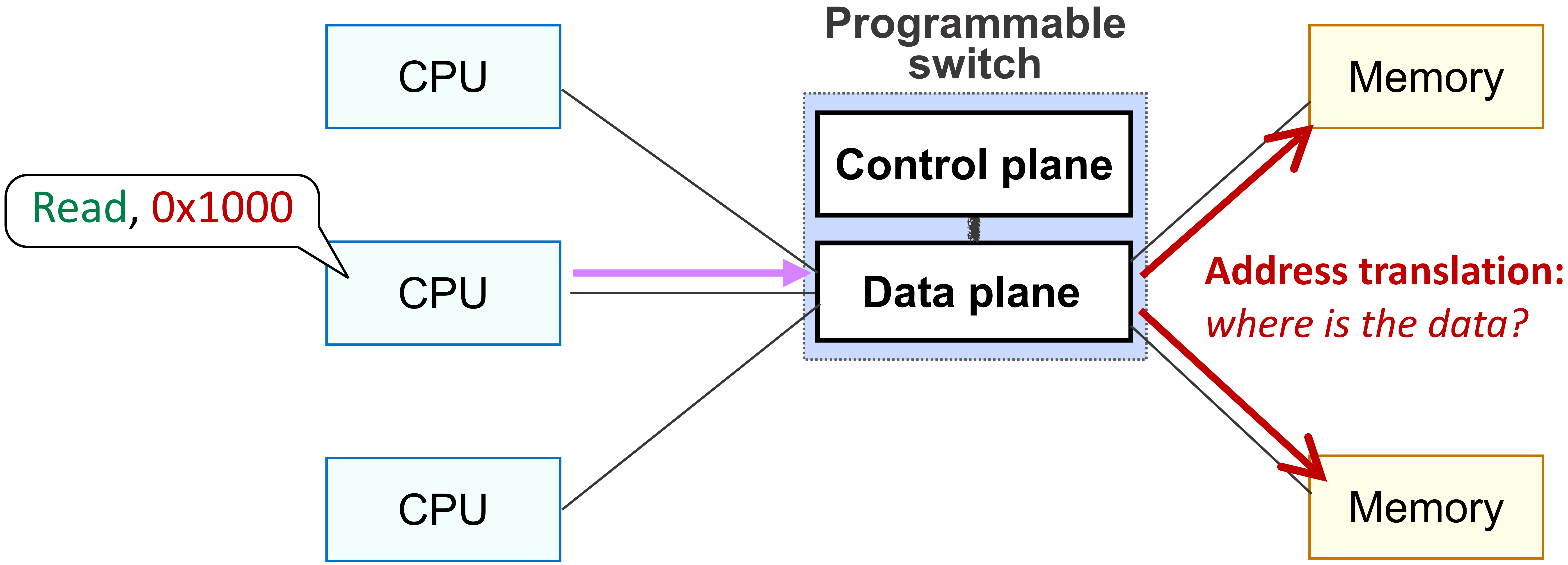
Memory management

- Address translation
- Memory allocation
- Memory protection
- Cache invalidations

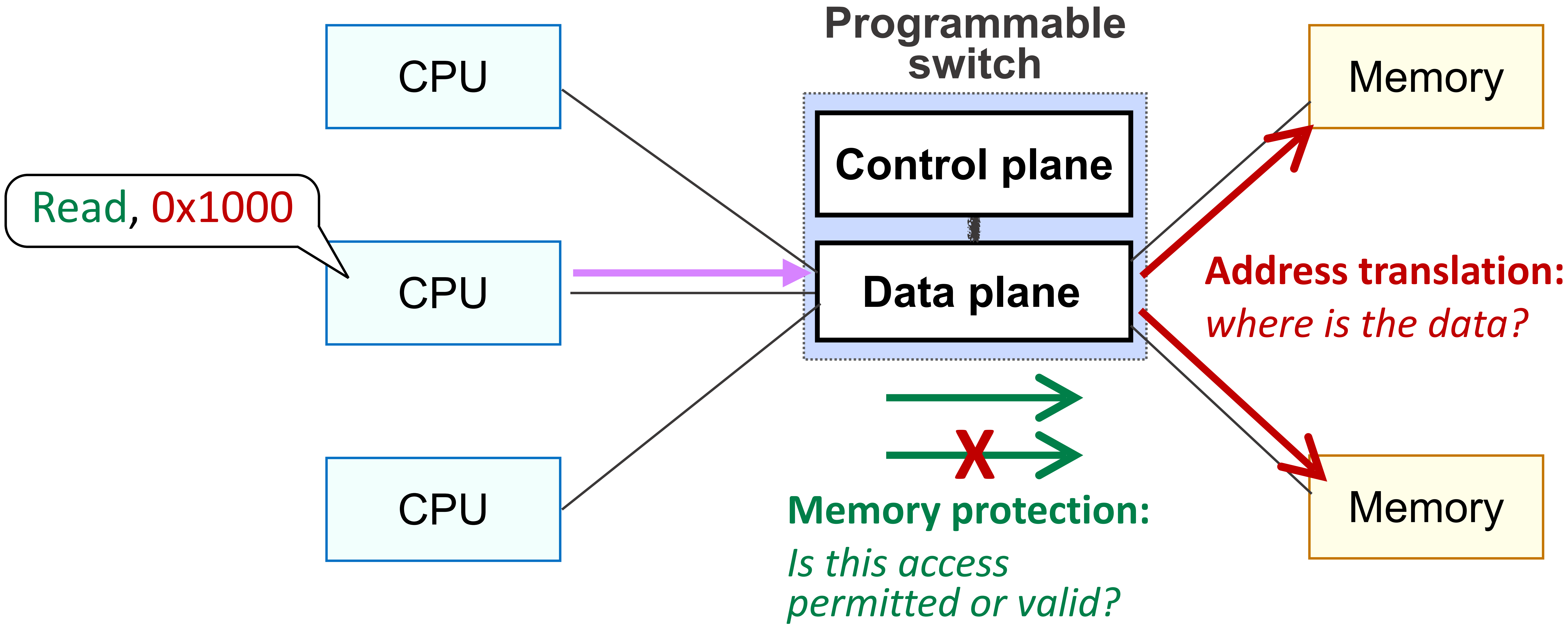
MIND: In-network Memory Management



MIND: In-network Memory Management



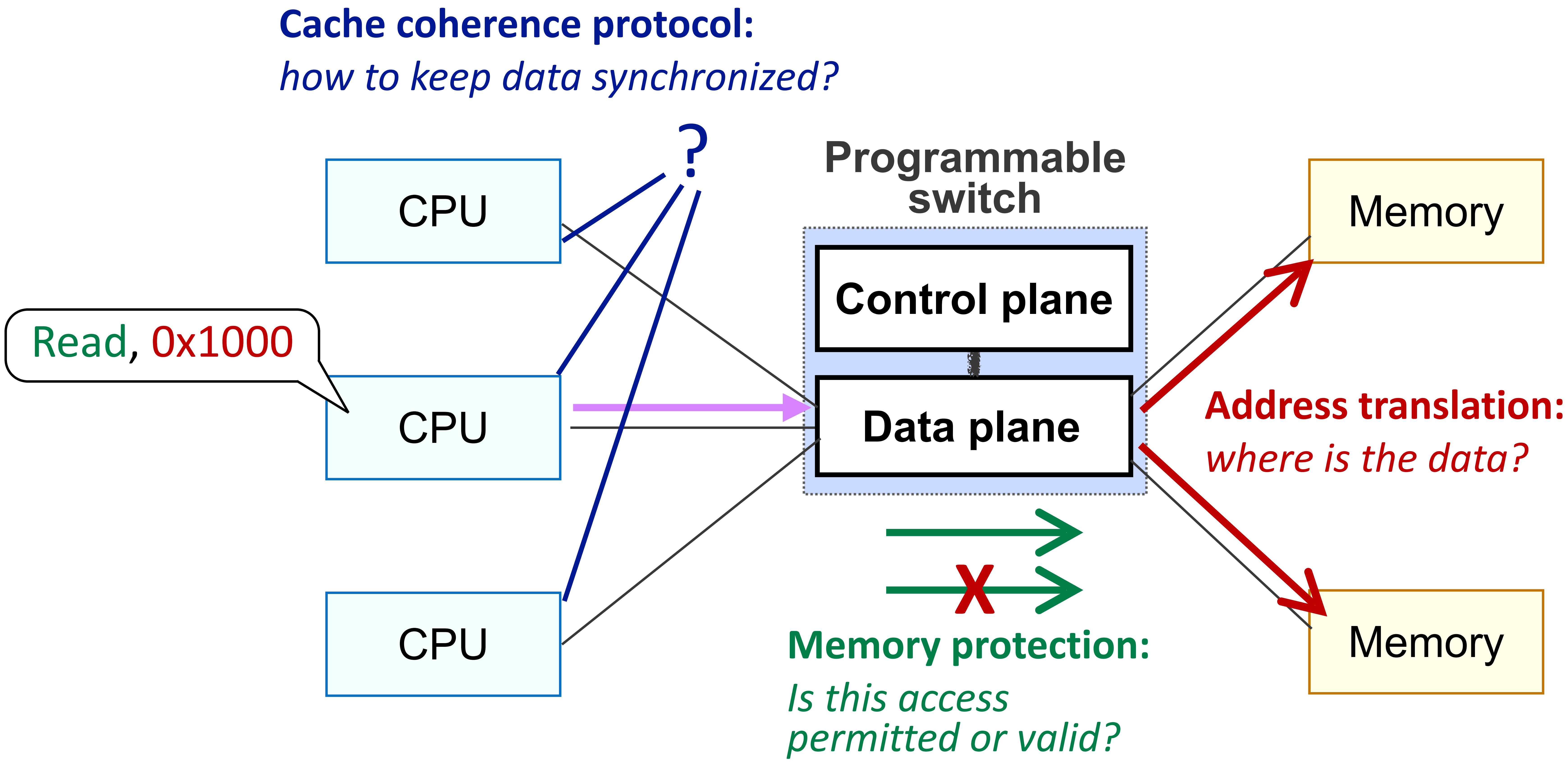
MIND: In-network Memory Management



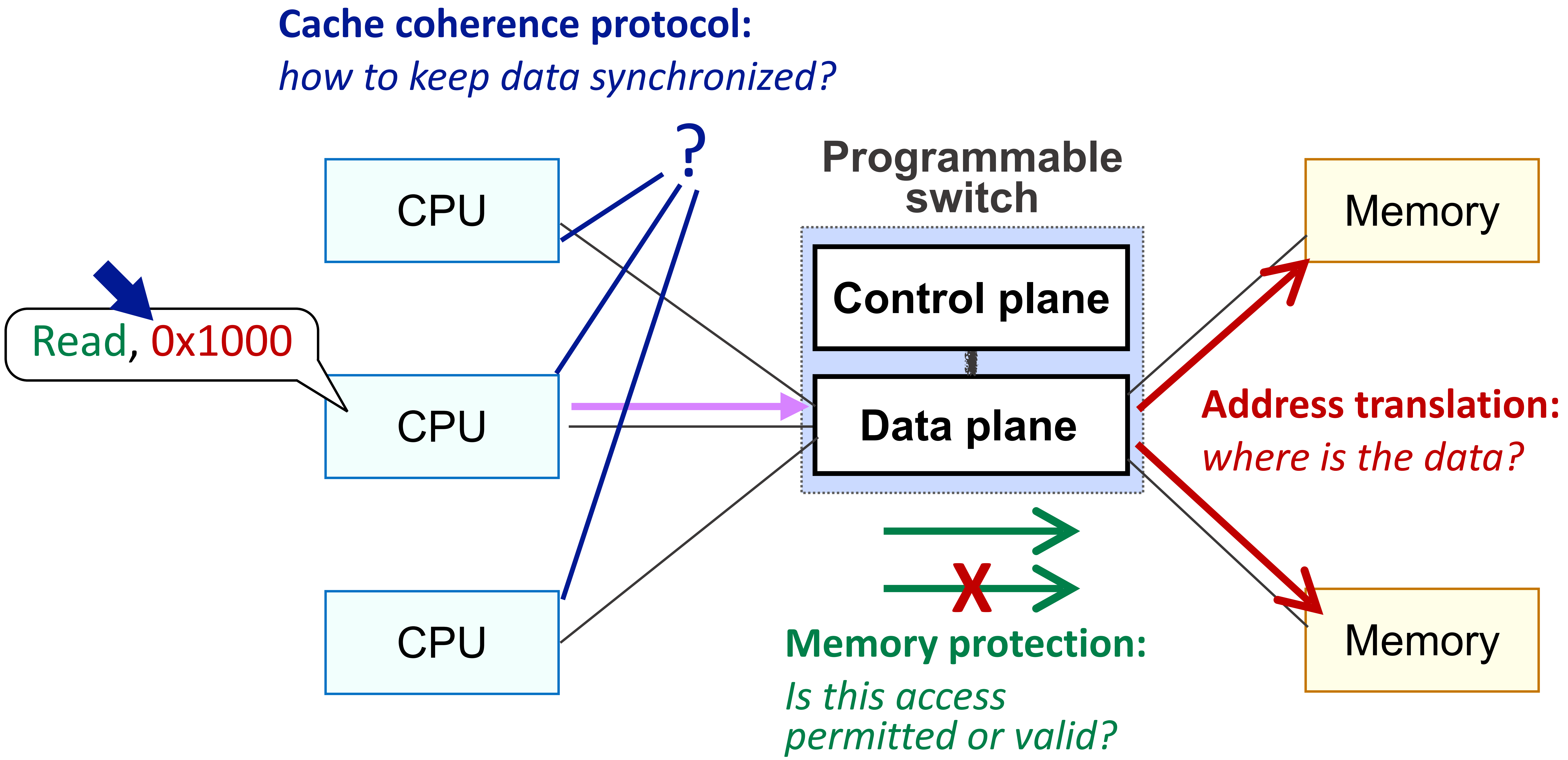
Address translation:
where is the data?

Memory protection:
Is this access permitted or valid?

MIND: In-network Memory Management



MIND: In-network Memory Management



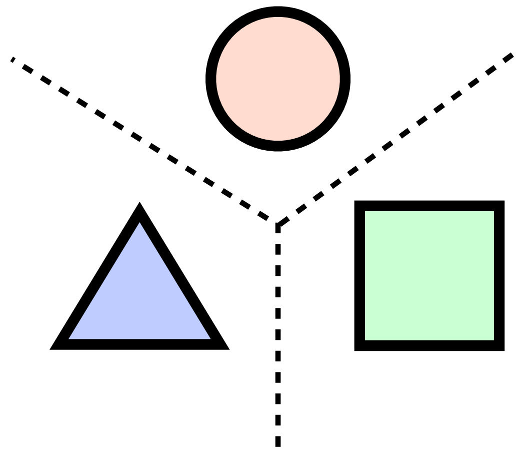
Challenges in Building In-network MMU

- **Limited amount of in-network resources**
  - *Limited size of in-network memory*
    - Tens of MB → not sufficient to store metadata in a traditional way
      - *E.g., page table: 4 KB for 4GB of memory → 1M entries*

Challenges in Building In-network MMU

- **Limited amount of in-network resources**
  - *Limited size of in-network memory*
    - Tens of MB → not sufficient to store metadata in a traditional way
      - *E.g., page table: 4 KB for 4GB of memory → 1M entries*
  - *Limited computation capability*
    - Switching ASIC → not sufficient to directly port traditional MMU functions
      - *E.g., limited number of operations for line rate processing*
        ↔ complicated cache coherence protocol

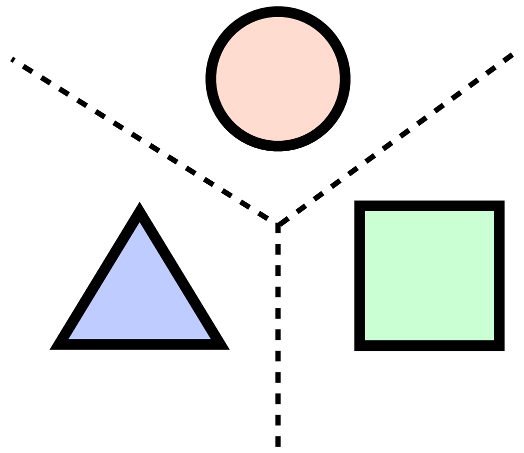
3 Principles for System Design



- **P1 - Decouple memory management functionalities**

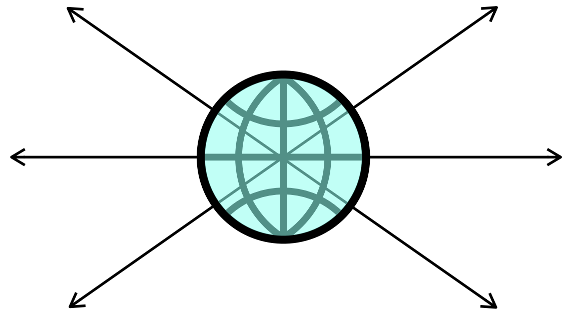
  → Each function has the data structure suitable to its purpose

3 Principles for System Design



- **P1 - Decouple memory management functionalities**

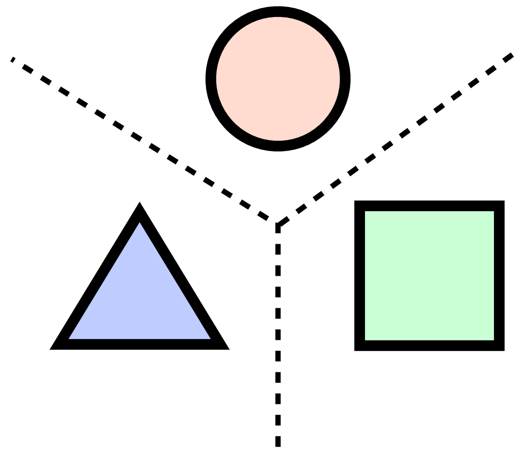
  → Each function has the data structure suitable to its purpose



- **P2 - Leverage global view of network**

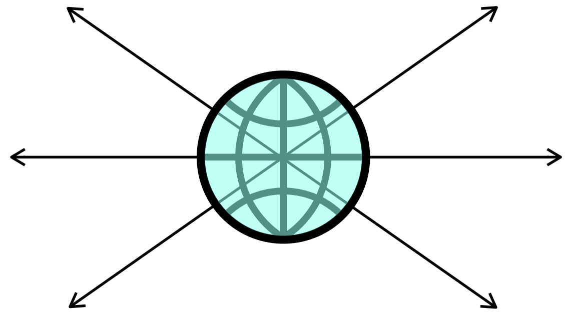
  → Make better decisions for memory management

3 Principles for System Design



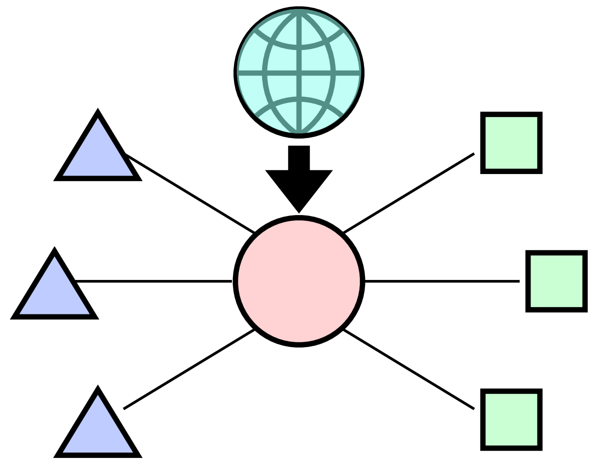
- **P1 - Decouple memory management functionalities**

  → Each function has the data structure suitable to its purpose



- **P2 - Leverage global view of network**

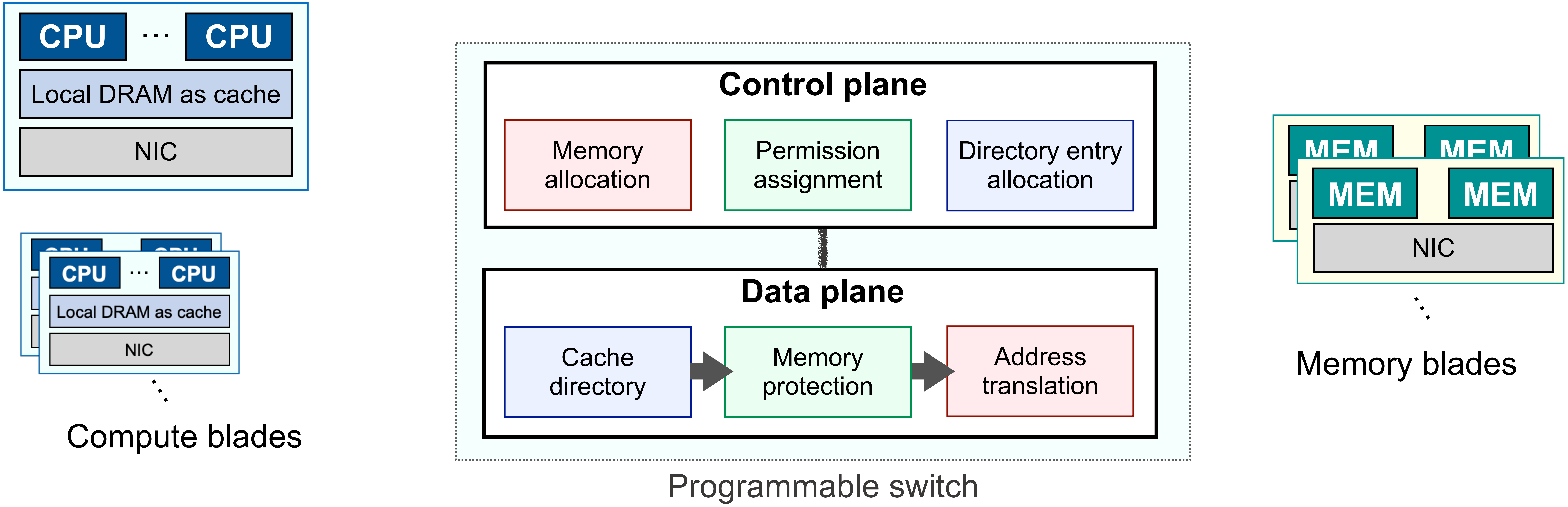
  → Make better decisions for memory management



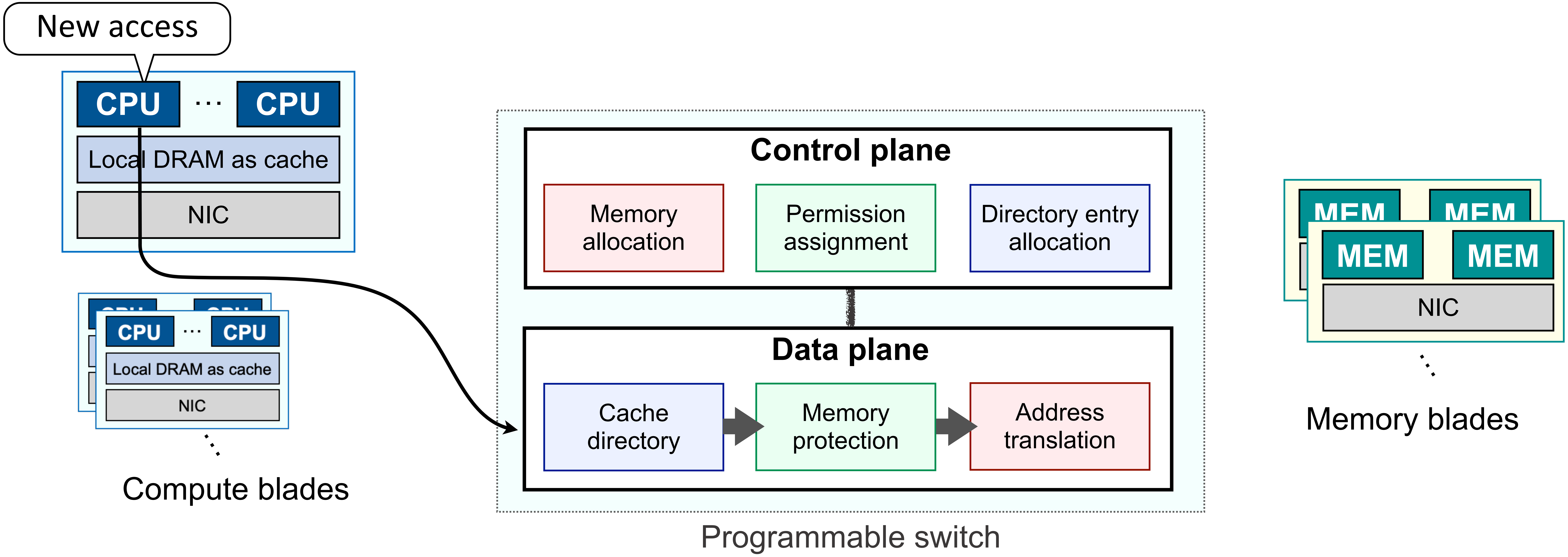
- **P3 - Exploit network-centric hardware primitives**

  → Reuse network hardware highly optimized for network functions

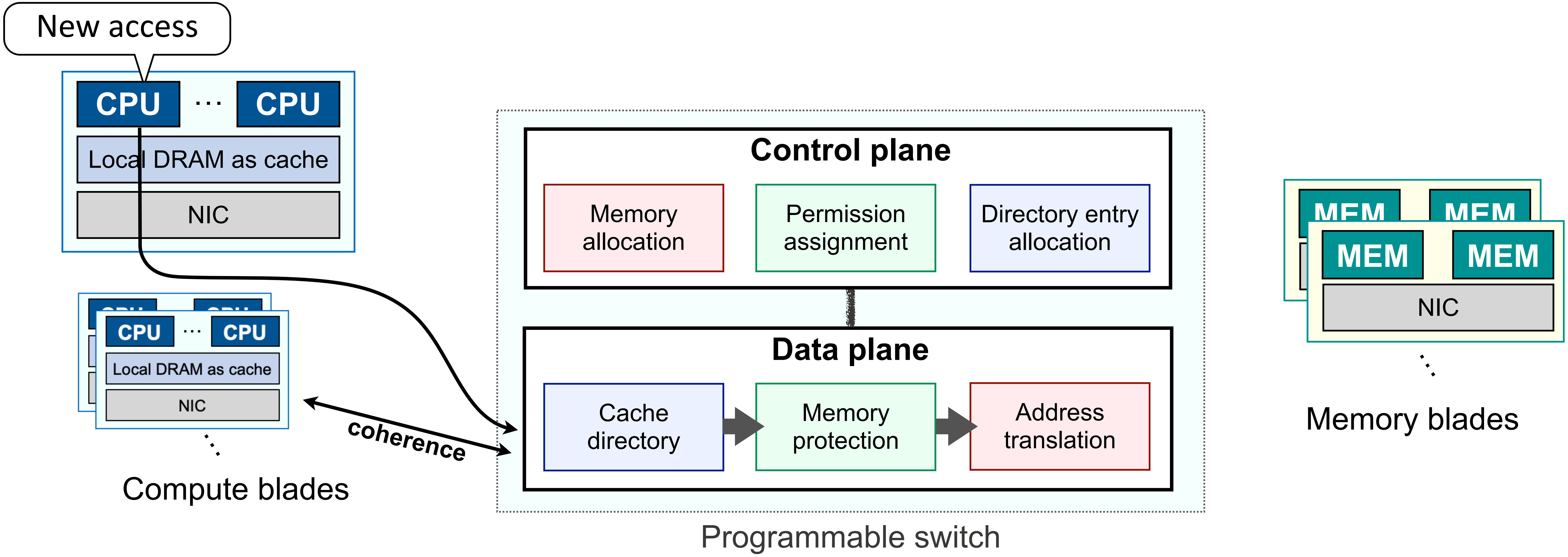
MIND overview: challenges and solutions



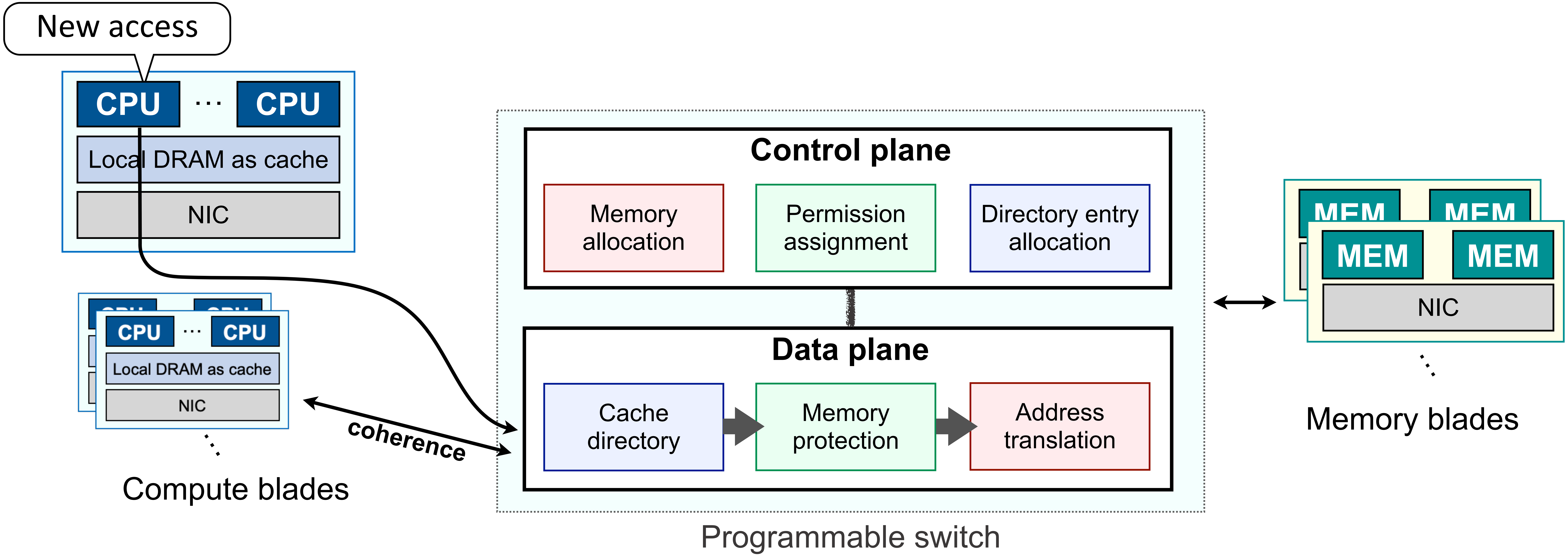
MIND overview: challenges and solutions



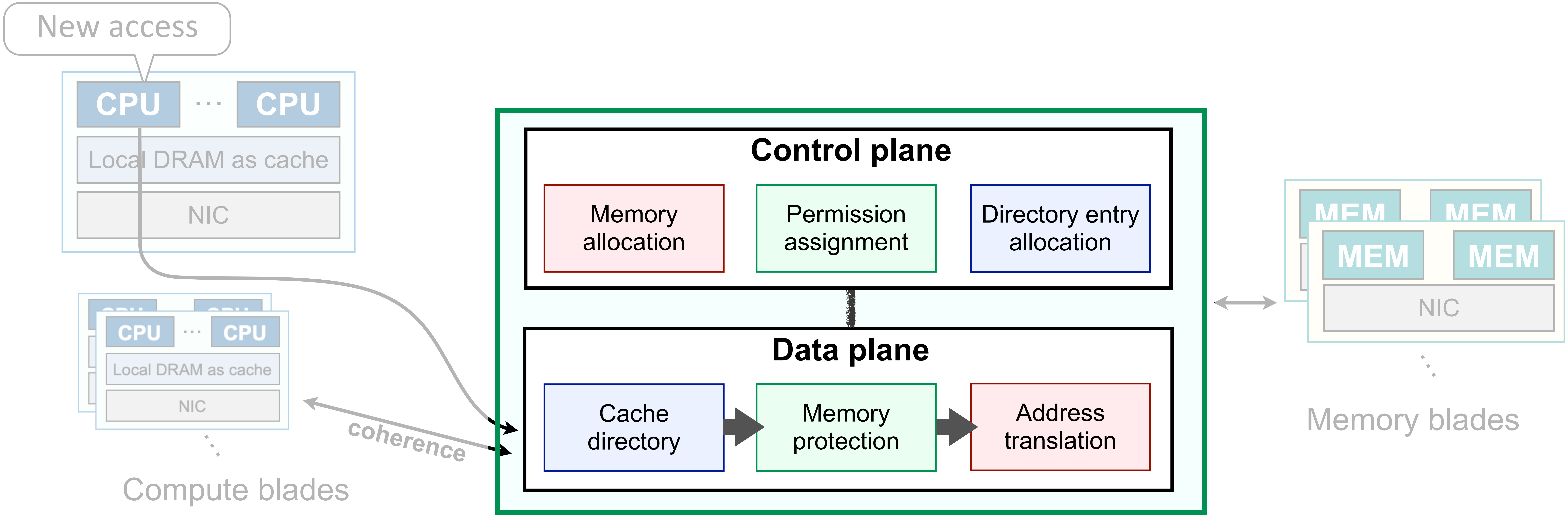
MIND overview: challenges and solutions



MIND overview: challenges and solutions

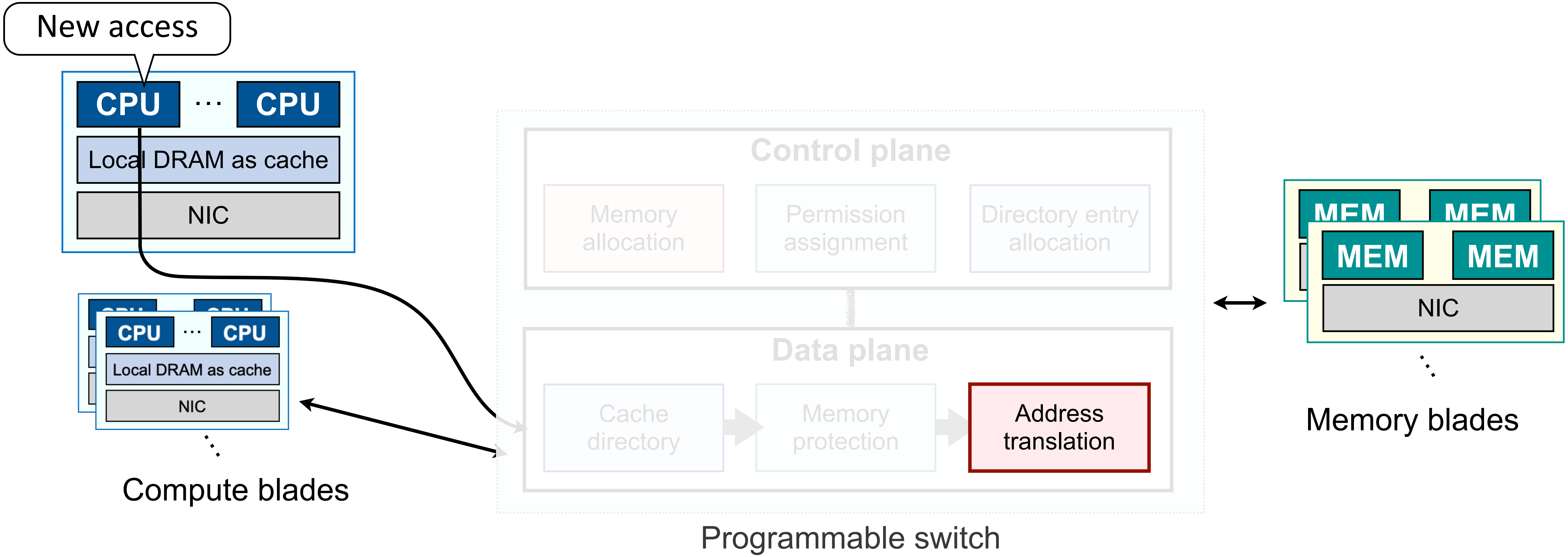


MIND overview: challenges and solutions

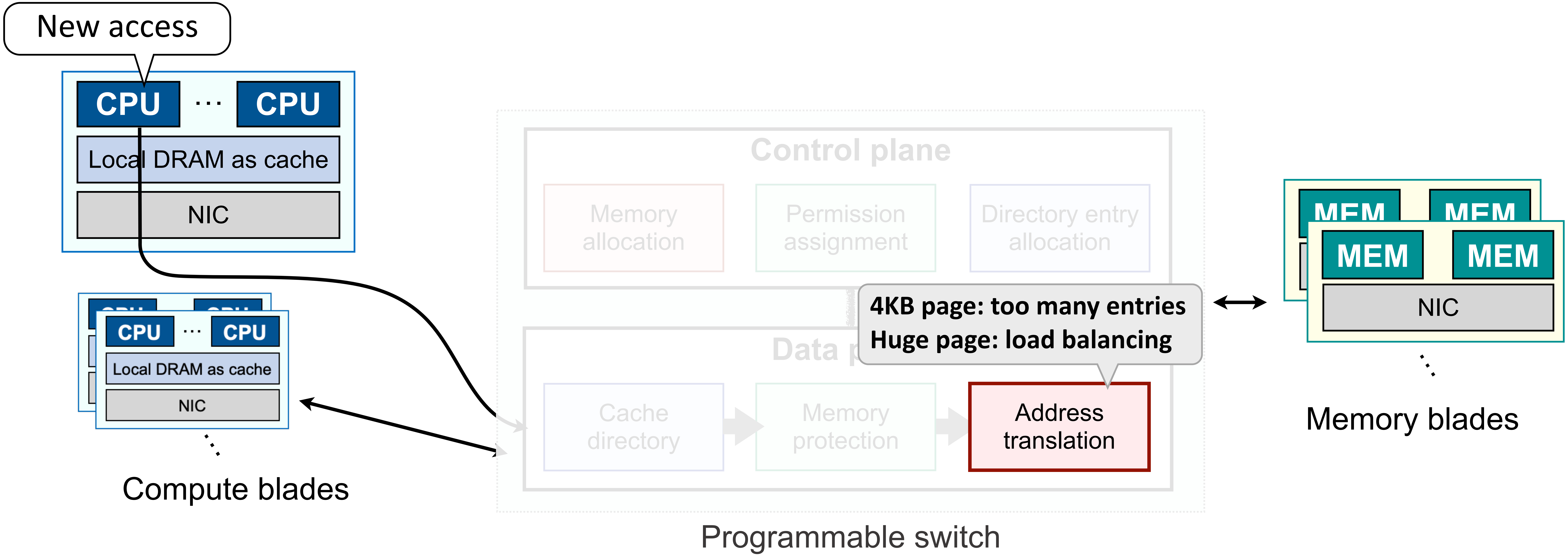


Programmable switch

MIND overview: challenges and solutions

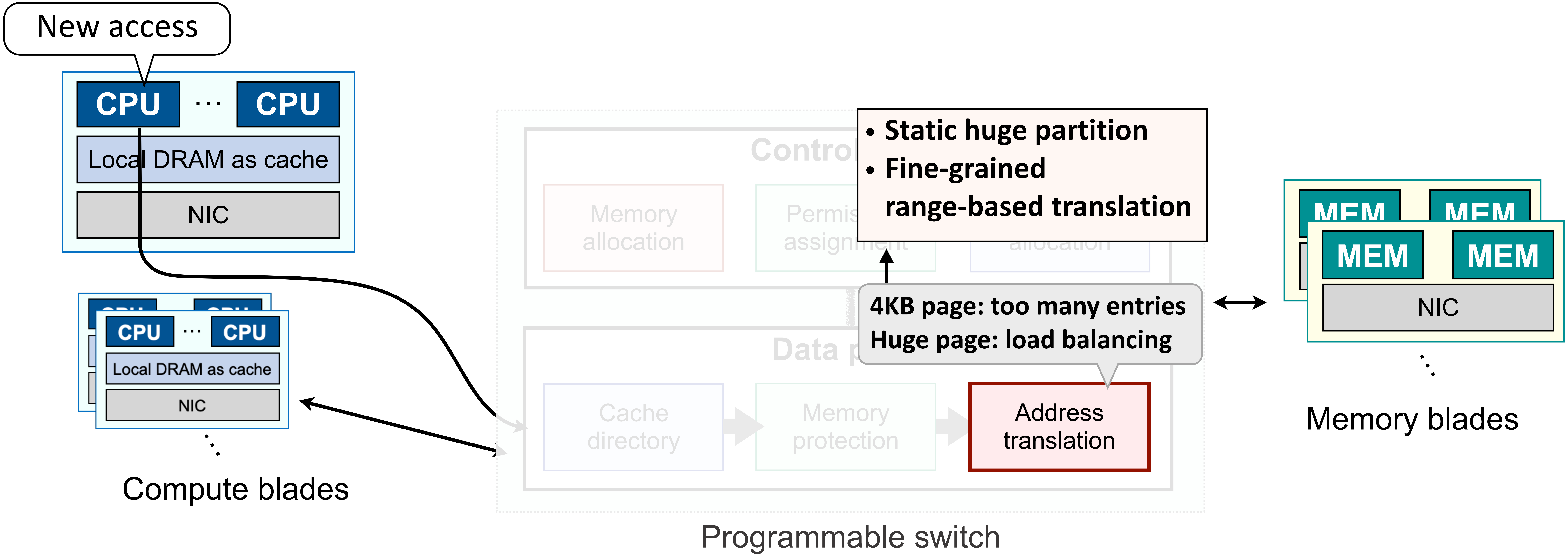


MIND overview: challenges and solutions



4KB page: too many entries
Huge page: load balancing

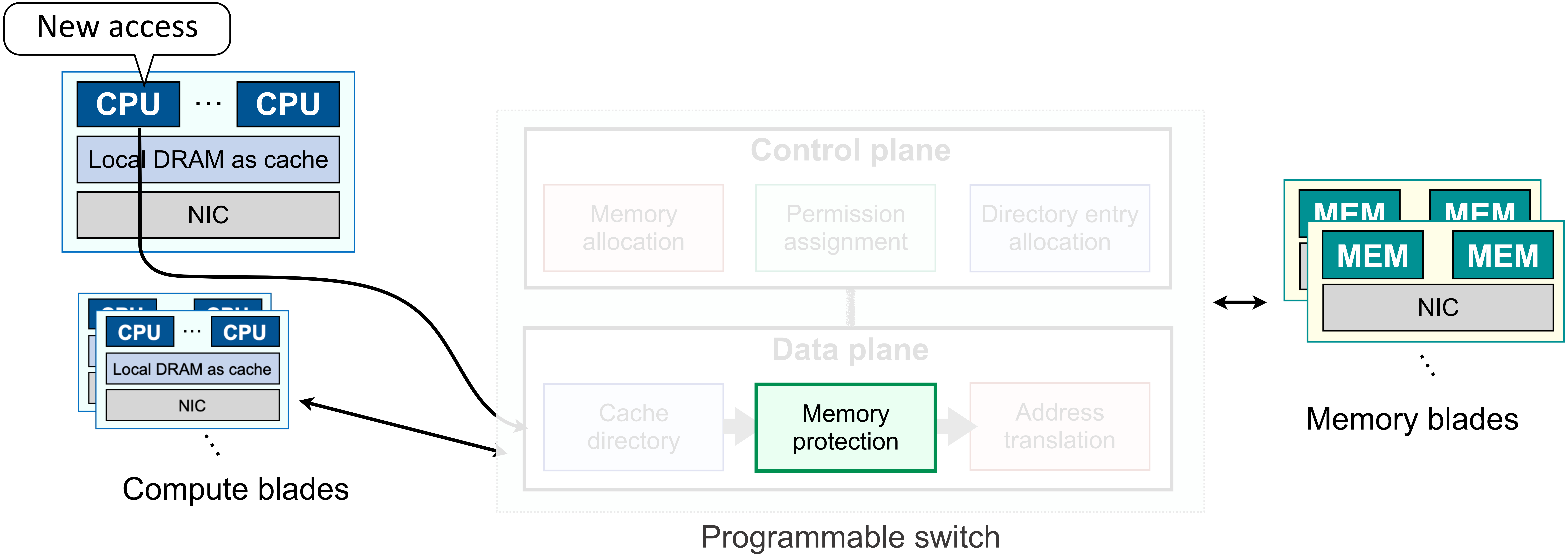
MIND overview: challenges and solutions



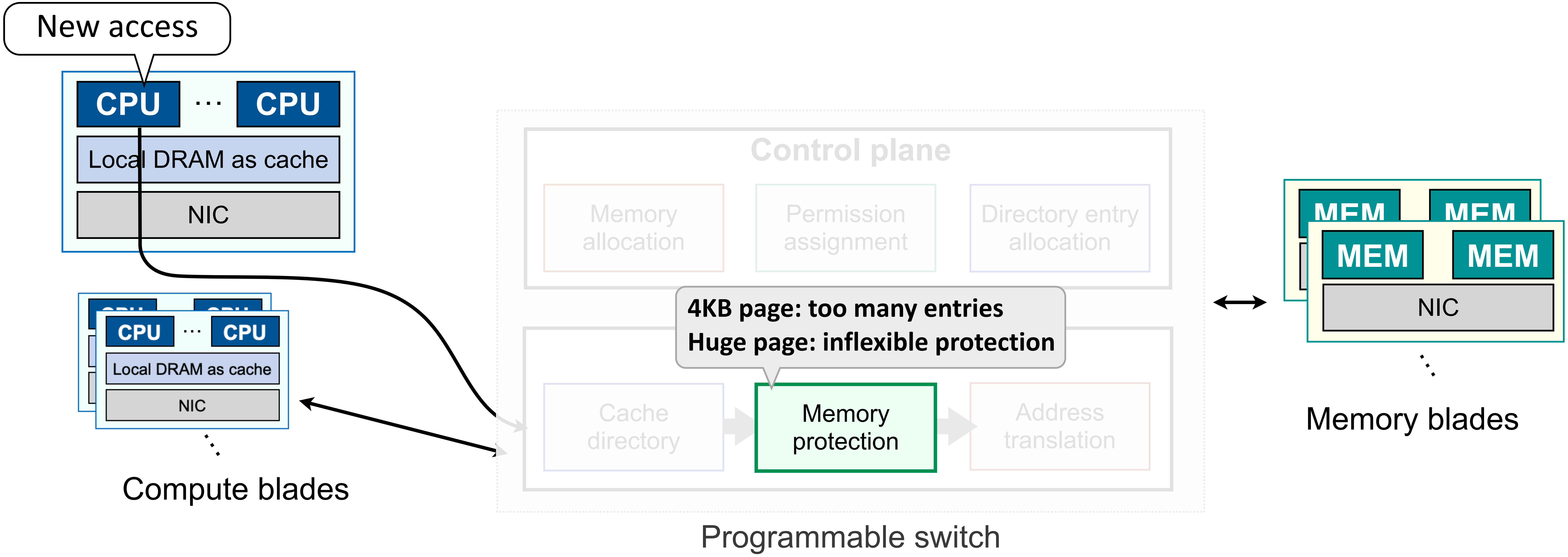
- **Static huge partition**
- **Fine-grained range-based translation**

4KB page: too many entries
Huge page: load balancing

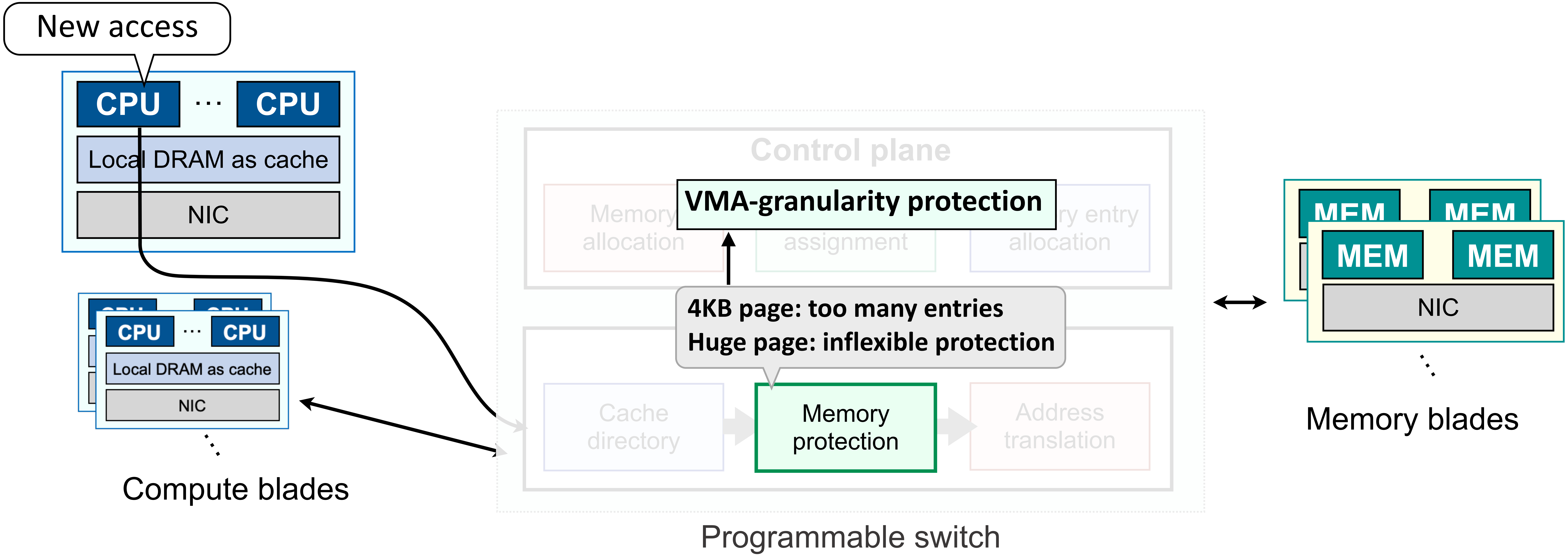
MIND overview: challenges and solutions



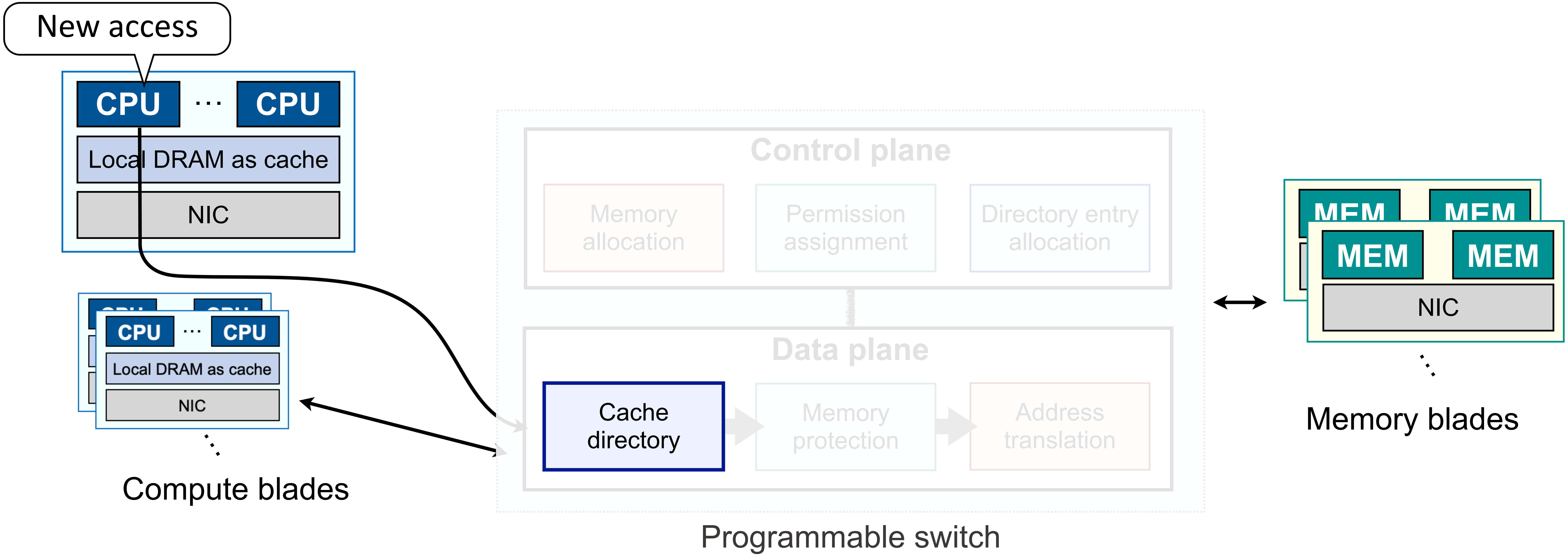
MIND overview: challenges and solutions



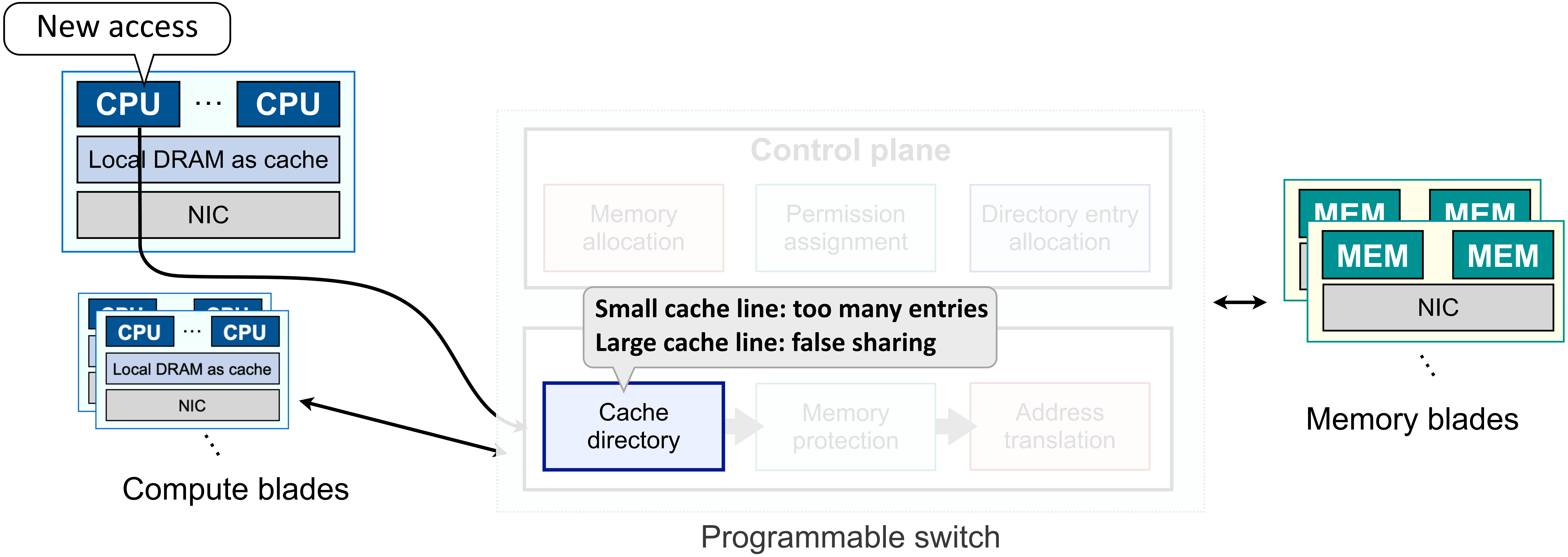
MIND overview: challenges and solutions



MIND overview: challenges and solutions

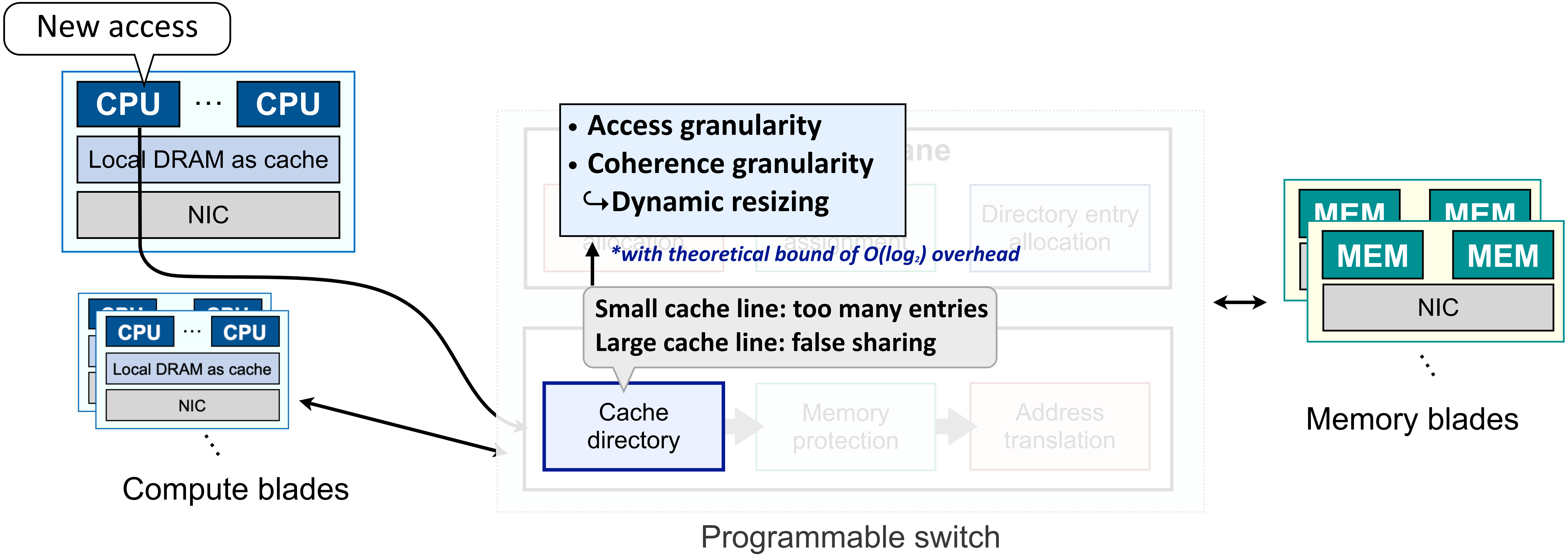


MIND overview: challenges and solutions



Small cache line: too many entries
Large cache line: false sharing

MIND overview: challenges and solutions



- Access granularity
- Coherence granularity

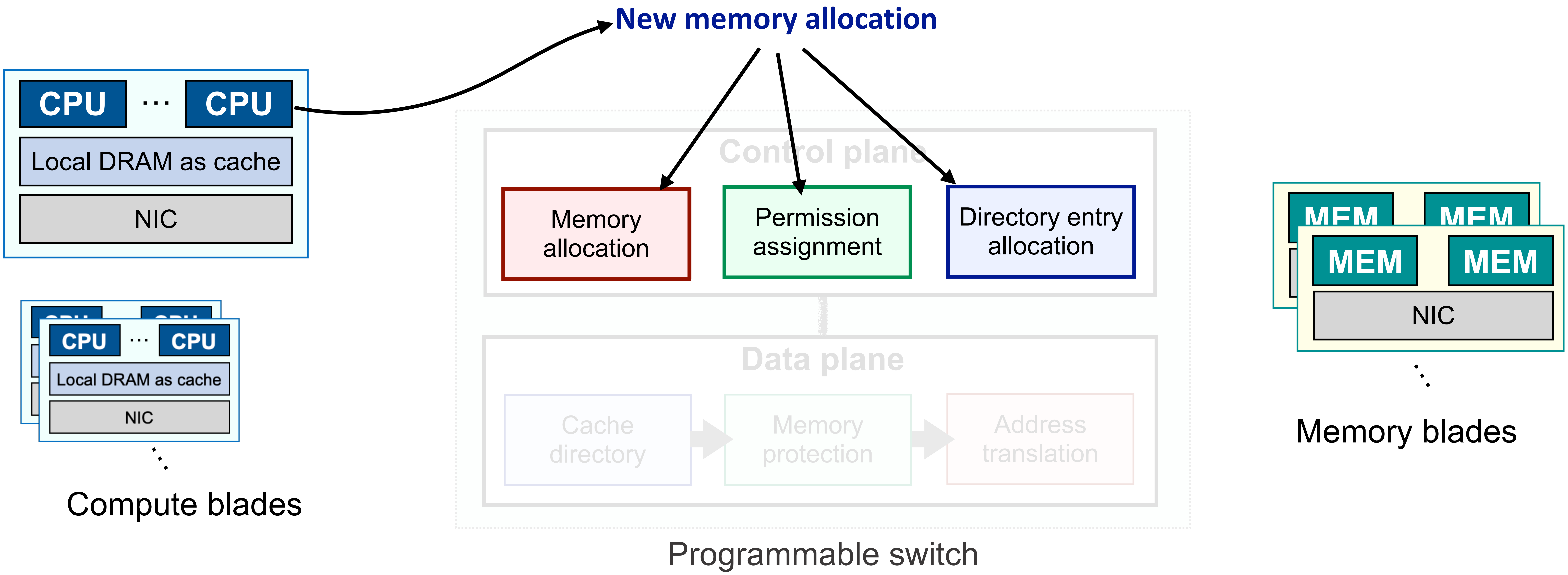
↪Dynamic resizing

***with theoretical bound of $O(\log_2)$ overhead**

Small cache line: too many entries

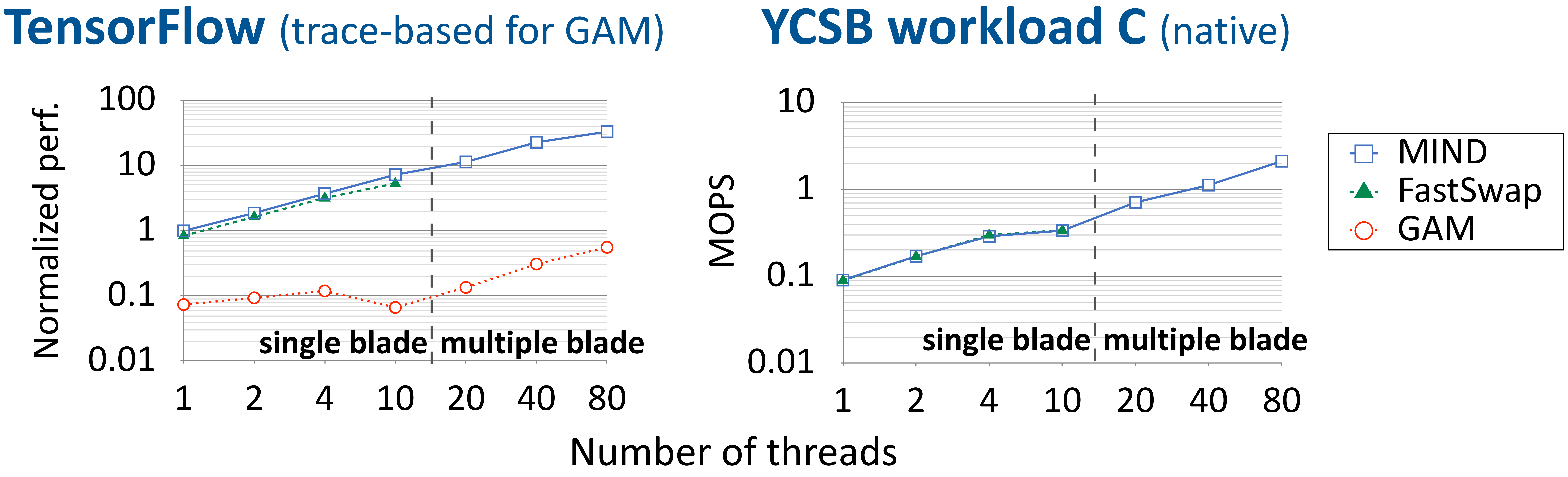
Large cache line: false sharing

MIND overview: challenges and solutions



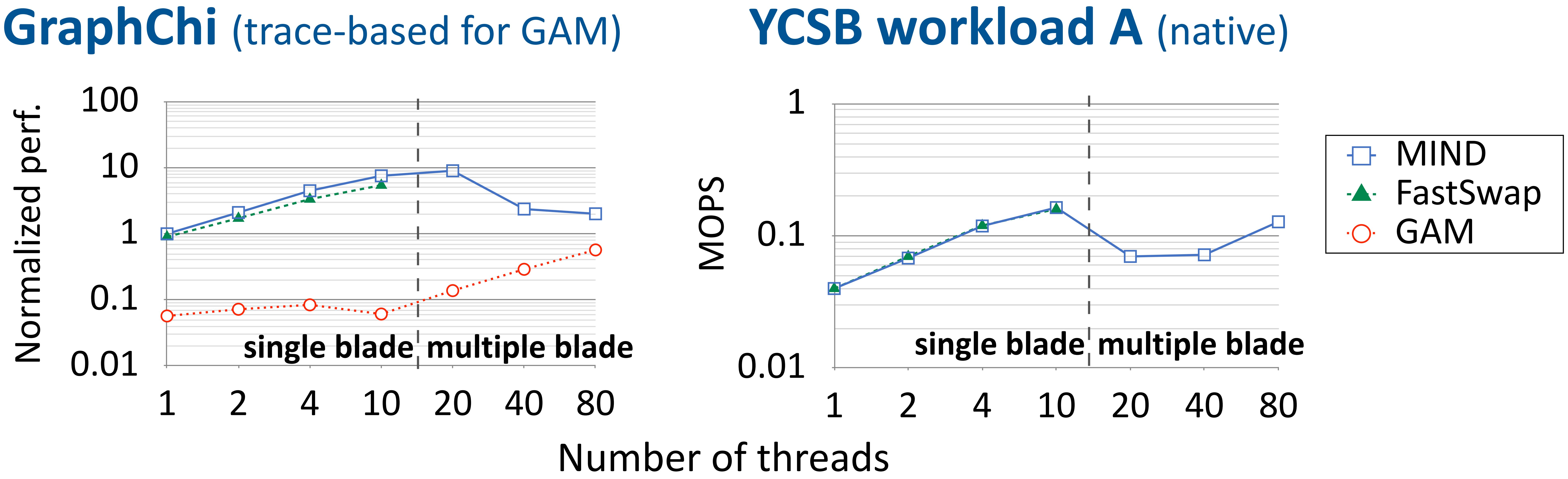
Performance Evaluation

- **Workloads with low contention**
  - TensorFlow (ResNet50), YCSB workload C (read only)



Performance Evaluation

- **Workloads with high contention**
  - GraphChi (PageRank), YCSB workload A (50 % update, 50 % read)



Detailed trade offs for design choices in the paper!

Conclusion & Summary

- **Trade-off between resource elasticity and performance** in memory disaggregation

Conclusion & Summary

- **Trade-off between resource elasticity and performance** in memory disaggregation
- We designed MIND, an **in-network MMU** by leveraging programmable network

Conclusion & Summary

- **Trade-off between resource elasticity and performance** in memory disaggregation

- We designed MIND, an **in-network MMU** by leveraging programmable network

- Our prototype of MIND can **match the performance of prior proposals and provide transparent elasticity**