DINOMO: An Elastic, Scalable, High-Performance Key-Value Store for Disaggregated Persistent Memory

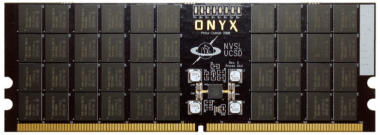
Sekwon Lee*, Soujanya Ponnapalli, Sharad Singhal,
Marcos K. Aguilera, Kimberly Keeton, Vijay Chidambaram

TEXAS The University of Texas at Austin · Google · Hewlett Packard Labs · vmware

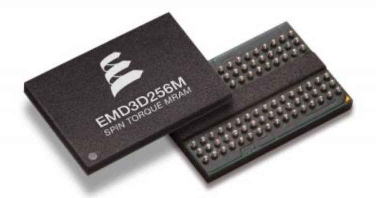
*On the job market

Emerging storage technology & Disaggregation

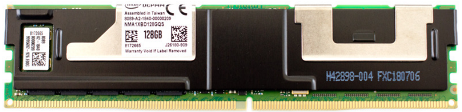
- Persistent Memory (PM)
  - Non-volatile like storage and byte-addressable like DRAM
  - High performance close to DRAM
  - Cost per GB >>>> HDD or SSD



PCM



STT-MRAM

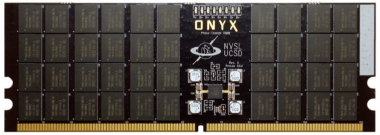


Intel Optane DC PM

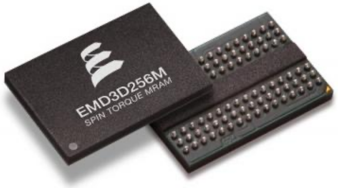
CXL-SSD

Emerging storage technology & Disaggregation

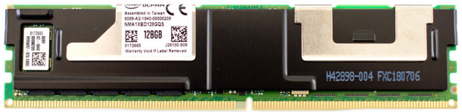
- Persistent Memory (PM)
  - Non-volatile like storage and byte-addressable like DRAM
  - High performance close to DRAM
  - Cost per GB >>>> HDD or SSD
    - Ensuring high utilization is more critical for cost efficiency



PCM



STT-MRAM

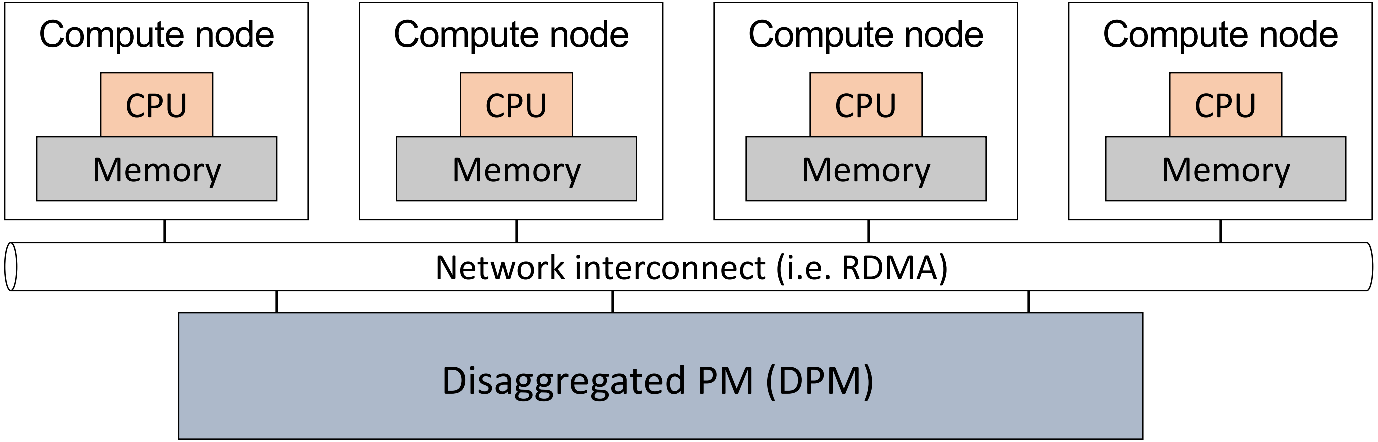


Intel Optane DC PM

CXL-SSD

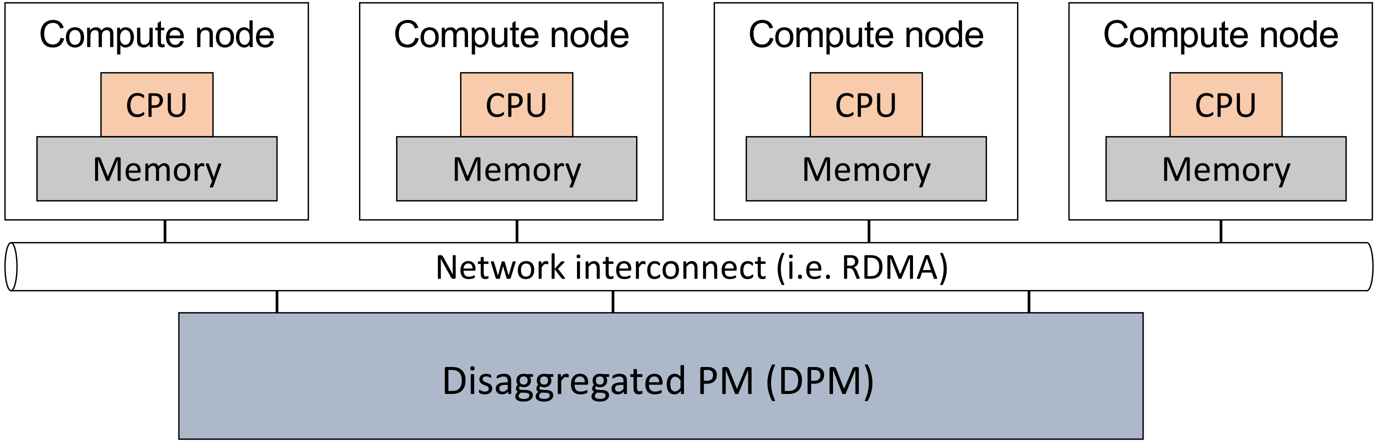
Emerging storage technology & Disaggregation

- Disaggregated Persistent Memory (DPM)
  + Share PM → Increase utilization, Reduce TCO (Total Cost Ownership)



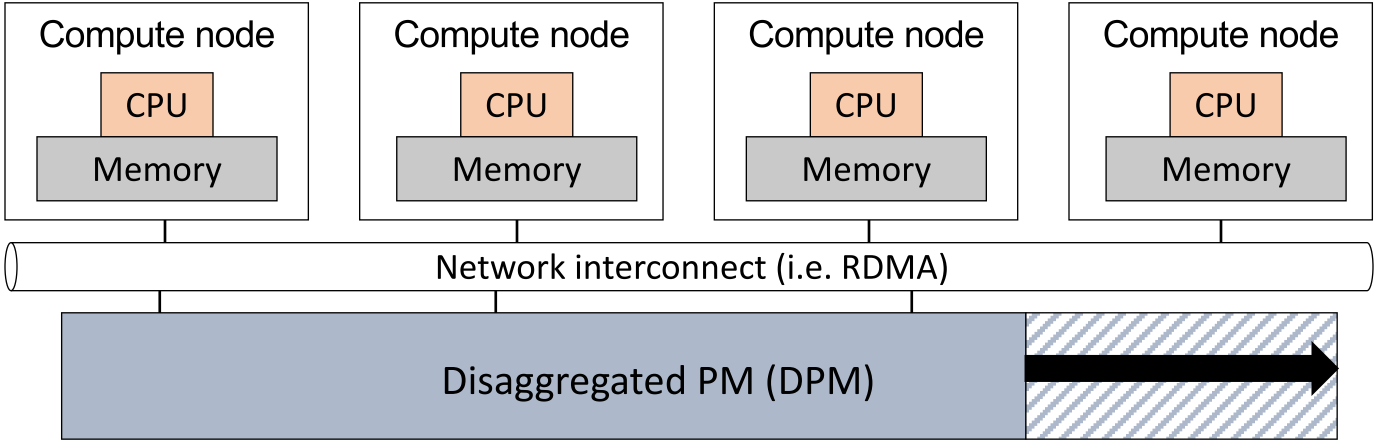
Emerging storage technology & Disaggregation

- Disaggregated Persistent Memory (DPM)
  - + Share PM → Increase utilization, Reduce TCO (Total Cost Ownership)
  - + Disaggregate PM → Scale resources independently, Separate failure domains



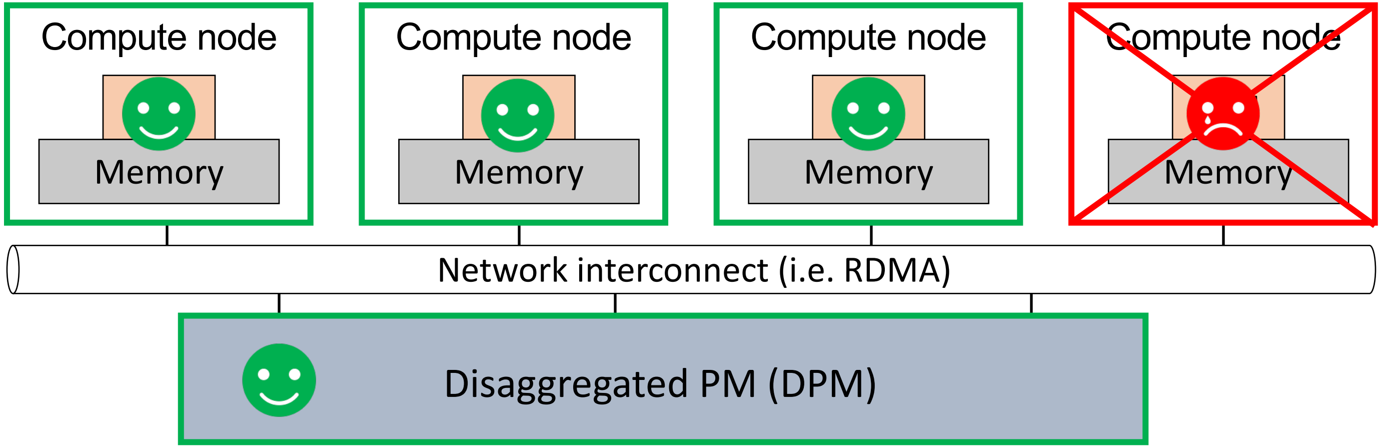
Emerging storage technology & Disaggregation

- Disaggregated Persistent Memory (DPM)
  - + Share PM → Increase utilization, Reduce TCO (Total Cost Ownership)
  - + Disaggregate PM → Scale resources independently, Separate failure domains



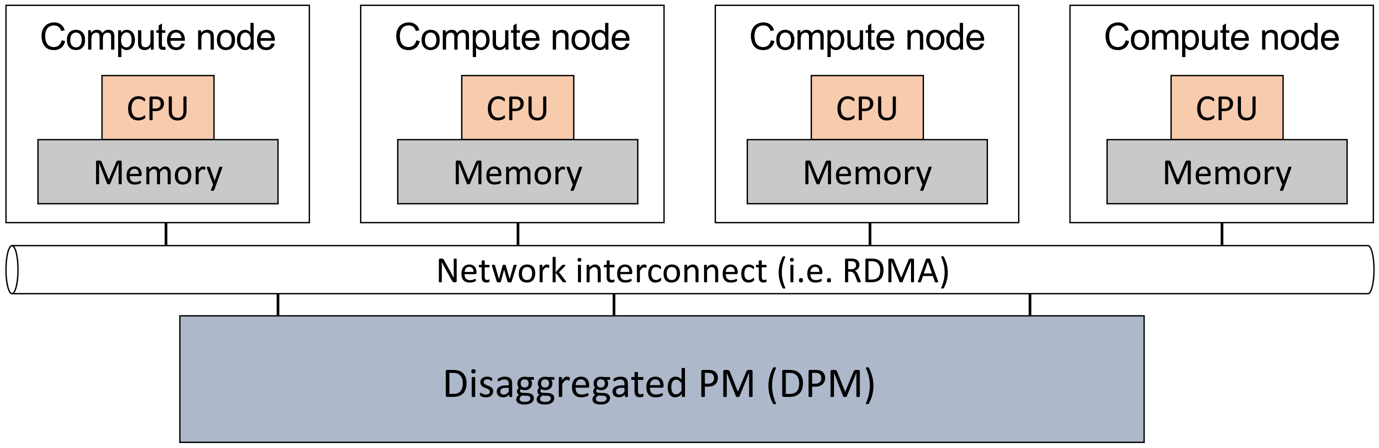
Emerging storage technology & Disaggregation

- Disaggregated Persistent Memory (DPM)
  - + Share PM → Increase utilization, Reduce TCO (Total Cost Ownership)
  - + Disaggregate PM → Scale resources independently, Separate failure domains



Emerging storage technology & Disaggregation

- Disaggregated Persistent Memory (DPM)
  - + Share PM → Increase utilization, Reduce TCO (Total Cost Ownership)
  - + Disaggregate PM → Scale resources independently, Separate failure domains
  - – Access PM over network → Network latency >> PM latency

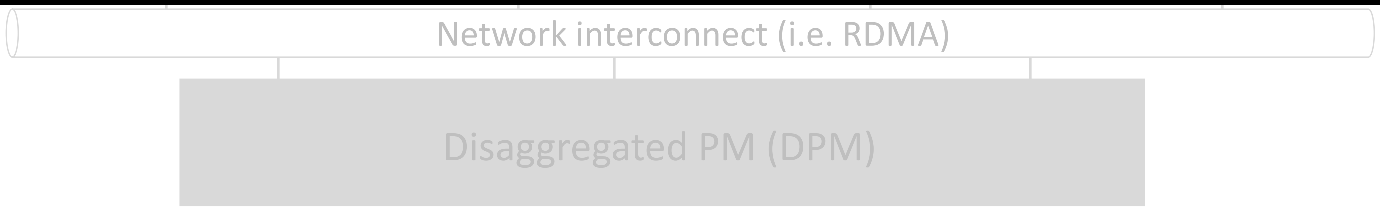


Emerging storage technology & Disaggregation

Data processing system done right for DPM

To benefit from the independence of scaling resources and failure, it must be elastic and scalable

Despite expensive networking overheads, it must provide high performance



Key-Value Store (KVS) for DPM

- Simple key-value APIs: get, put, update, delete …

Key-Value Store (KVS) for DPM

- Simple key-value APIs: get, put, update, delete …
- Support various applications under cloud environment
  - Require dynamic working sets/sizes and non-uniform workload patterns with varying skew

Key-Value Store (KVS) for DPM

- Simple key-value APIs: get, put, update, delete …
- Support various applications under cloud environment
  - Require dynamic working sets/sizes and non-uniform workload patterns with varying skew
- Goals of ideal DPM KVS
  - High common-case performance
  - Scalability with the amount of provisioned resource
  - Fast reconfiguration to change the amount of resource elastically

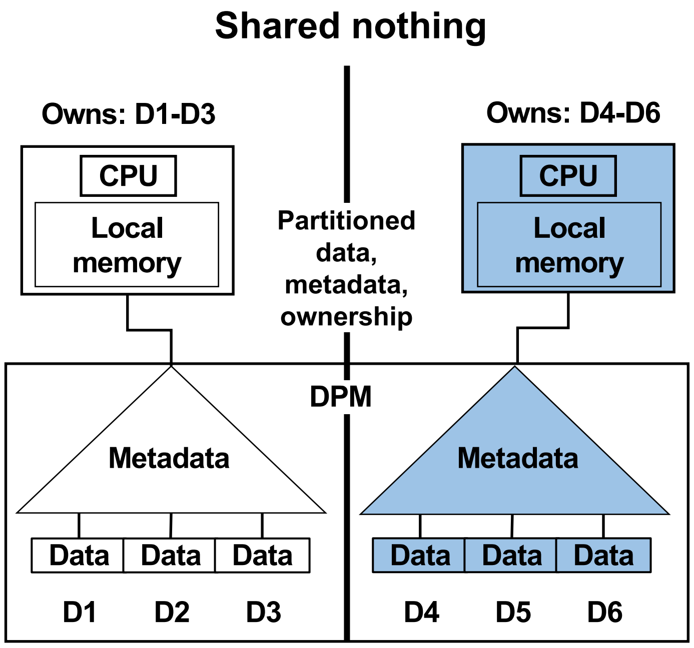
Prior DPM KVS

- Architectural limitations in achieving all three goals

Goals \ KVSs	AsymNVM	Clover
High performance	✓	X
Scalability	✓	X
Lightweight reconfiguration	X	✓

Prior DPM KVS

- AsymNVM[1]
  - Exclusive ownership to partitioned data/metadata

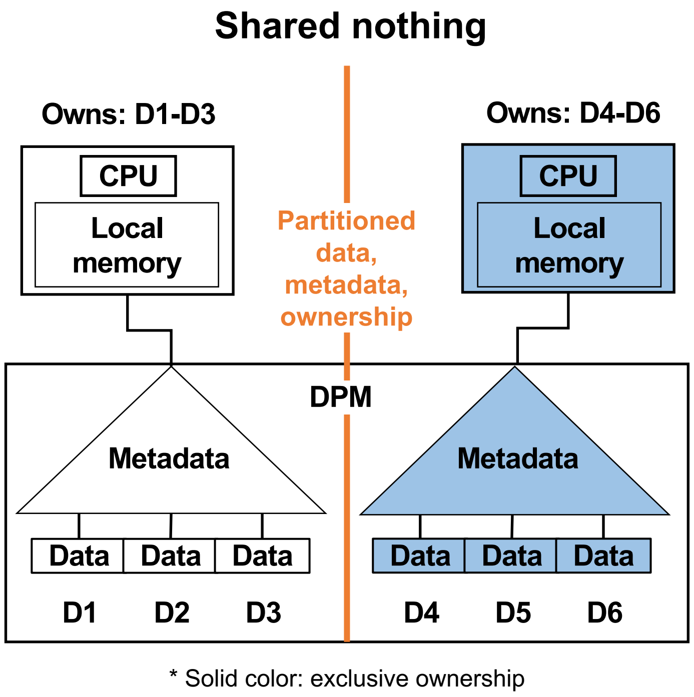


* Solid color: exclusive ownership

[1]Teng Ma, et al., AsymNVM: An Efficient Framework for Implementing Persistent Data Structures on Asymmetric NVM Architecture, ASPLOS'20

Prior DPM KVS

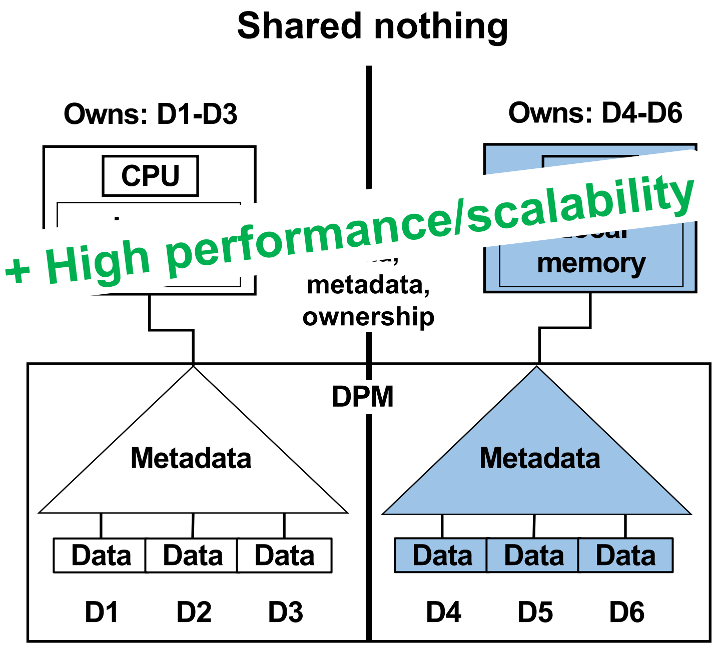
- AsymNVM[1]
  - Exclusive ownership to partitioned data/metadata



[1]Teng Ma, et al., AsymNVM: An Efficient Framework for Implementing Persistent Data Structures on Asymmetric NVM Architecture, ASPLOS'20

Prior DPM KVS

- AsymNVM[1]
  - Exclusive ownership to partitioned data/metadata
    - + Cache data without consistency overheads
    - + Preserve data locality of caches

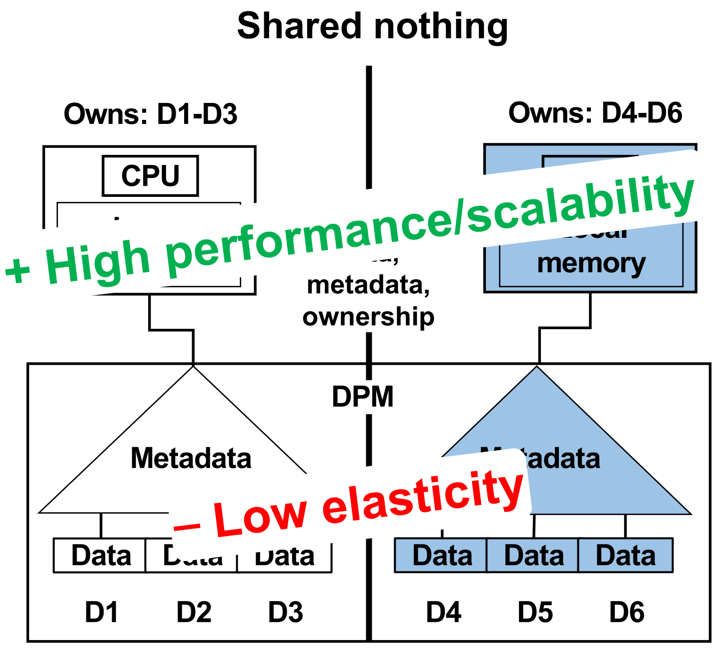


* Solid color: exclusive ownership

[1]Teng Ma, et al., AsymNVM: An Efficient Framework for Implementing Persistent Data Structures on Asymmetric NVM Architecture, ASPLOS'20

Prior DPM KVS

- AsymNVM[1]
  - Exclusive ownership to partitioned data/metadata
    - \+ Cache data without consistency overheads
    - \+ Preserve data locality of caches
    - – Expensive data reorganization upon reconfigurations

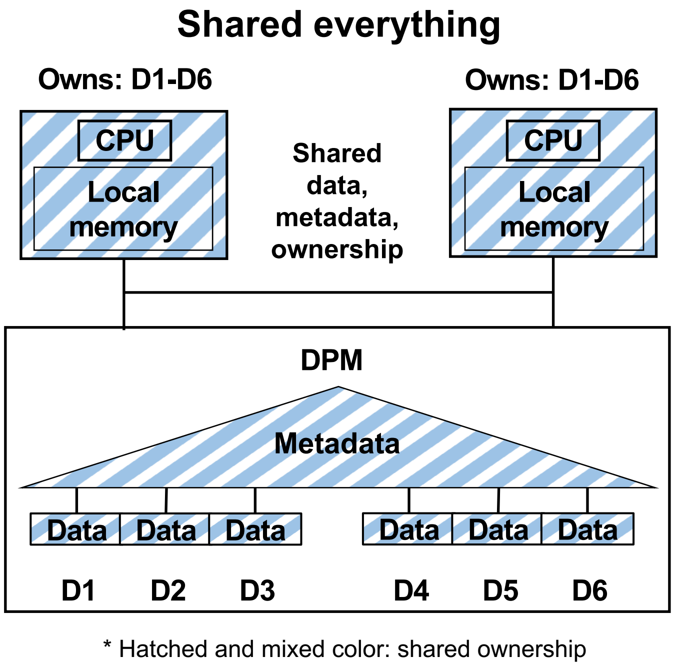


* Solid color: exclusive ownership

[1]Teng Ma, et al., AsymNVM: An Efficient Framework for Implementing Persistent Data Structures on Asymmetric NVM Architecture, ASPLOS'20

Prior DPM KVS

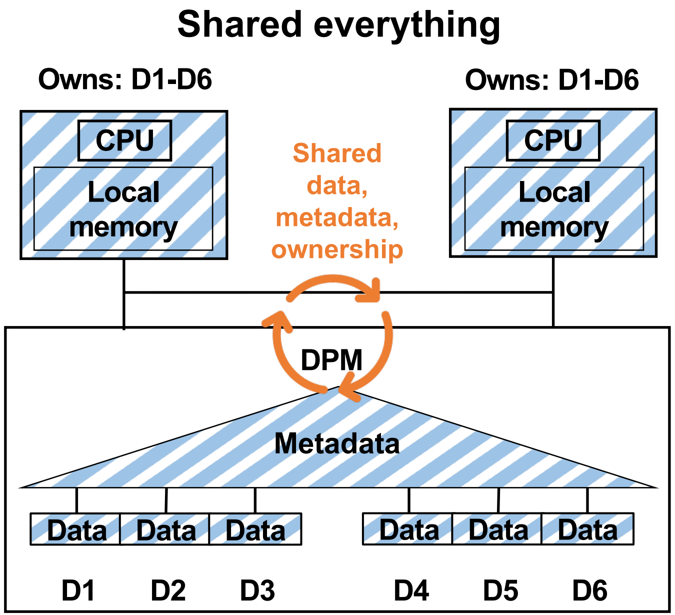
- Clover[1]
  - Share data, metadata, and ownership



[1]Shin-Yeh Tsai, et al., Disaggregating Persistent Memory and Controlling Them Remotely: An Exploration of Passive Disaggregated Key-Value Stores, ATC'20

Prior DPM KVS

- Clover[1]
  - Share data, metadata, and ownership

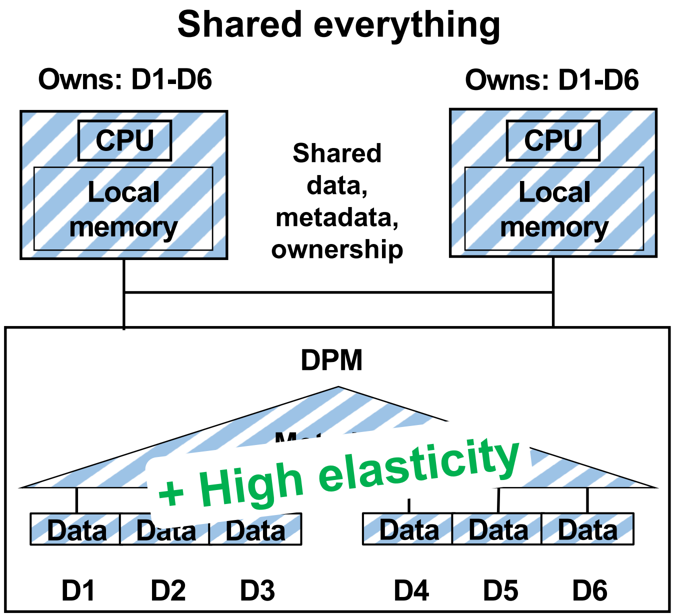


* Hatched and mixed color: shared ownership

[1]Shin-Yeh Tsai, et al., Disaggregating Persistent Memory and Controlling Them Remotely: An Exploration of Passive Disaggregated Key-Value Stores, ATC'20

Prior DPM KVS

- Clover[1]
  - Share data, metadata, and ownership
    - + No data reorganization upon reconfigurations

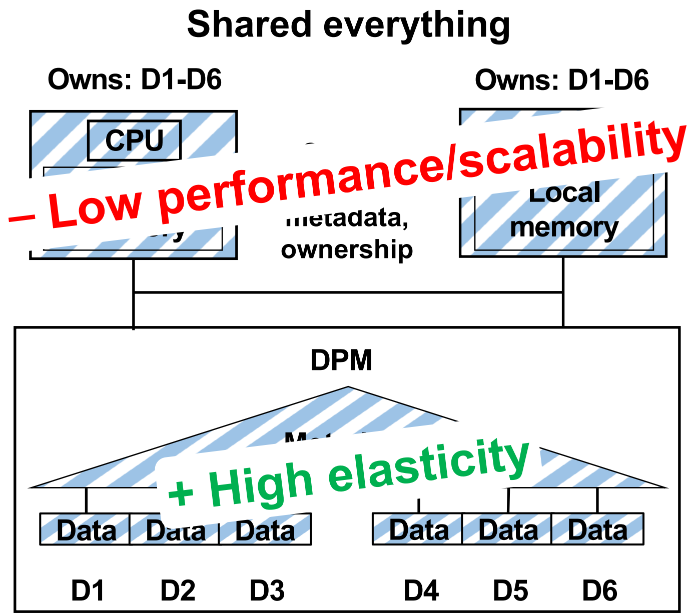


* Hatched and mixed color: shared ownership

[1]Shin-Yeh Tsai, et al., Disaggregating Persistent Memory and Controlling Them Remotely: An Exploration of Passive Disaggregated Key-Value Stores, ATC'20

Prior DPM KVS

- Clover[1]
  - Share data, metadata, and ownership
    - + No data reorganization upon reconfigurations
    - – Expensive data consistency overheads between caches
    - – Lack of data locality of caches



* Hatched and mixed color: shared ownership

[1]Shin-Yeh Tsai, et al., Disaggregating Persistent Memory and Controlling Them Remotely: An Exploration of Passive Disaggregated Key-Value Stores, ATC'20

Prior DPM KVS

- Architectural limitations in achieving all three goals

Goals \ KVSs	Clover	AsymNVM
High performance	X	✓
Scalability	X	✓
Lightweight reconfiguration	✓	X

DINOMO

First DPM KVS achieving high performance, scalability, and fast reconfiguration simultaneously

Goals \ KVSs	**DINOMO**	AsymNVM	Clover
High performance	✓	✓	X
Scalability	✓	✓	X
Lightweight reconfiguration	✓	X	✓

https://github.com/utsaslab/dinomo

DINOMO

First DPM KVS achieving high performance, scalability, and fast reconfiguration simultaneously

Adapt techniques **(ownership partitioning, adaptive caching, etc.)** from storage research community for DPM

Full end-to-end implementations including KVS control plane, data plane, and client

Upto **10x** better performance at scale and elasticity

https://github.com/utsaslab/dinomo

Outline

- Ownership partitioning
- Disaggregated adaptive caching
- Evaluation
- Discussion

Outline

- Ownership partitioning
- Disaggregated adaptive caching
- Evaluation
- Discussion

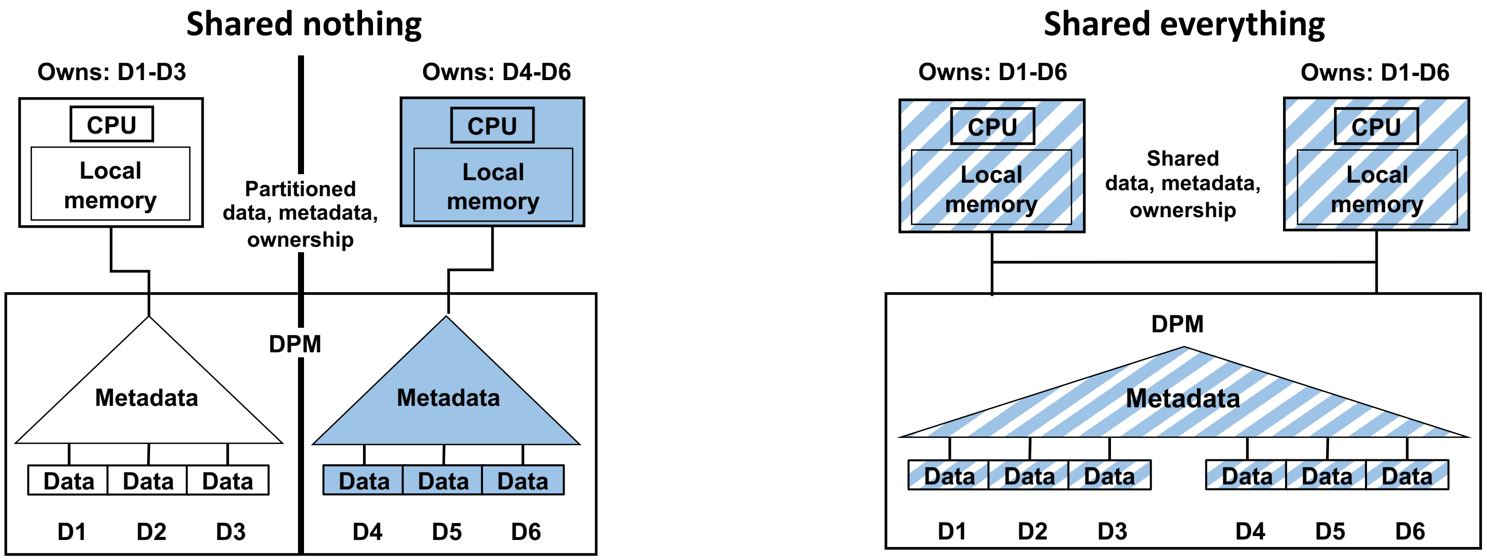
Goals and design techniques

Goal	Dinomo technique
High performance	
Scalability	
Lightweight reconfiguration (Elasticity)	

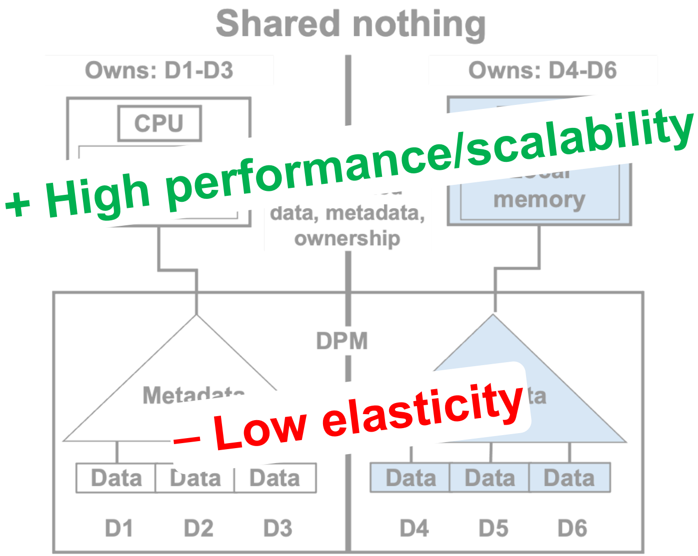
Goals and design techniques

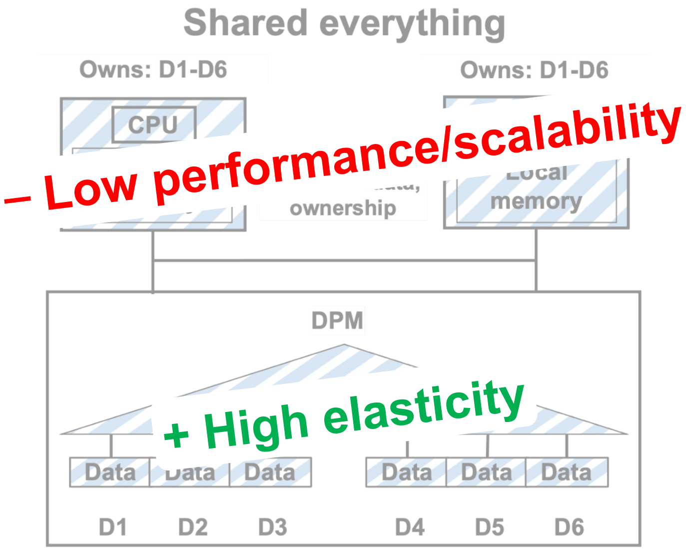
Goal	Dinomo technique
High performance	
	Ownership partitioning
Scalability	**Ownership partitioning**
Lightweight reconfiguration (Elasticity)	**Ownership partitioning**

Hybrid Architecture

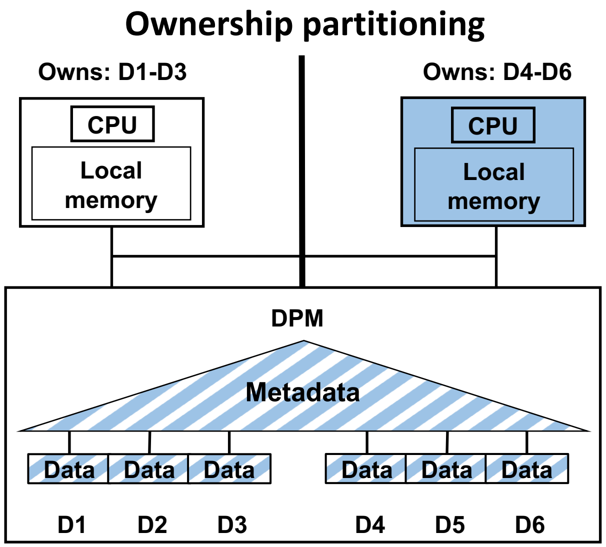


Hybrid Architecture





Hybrid Architecture

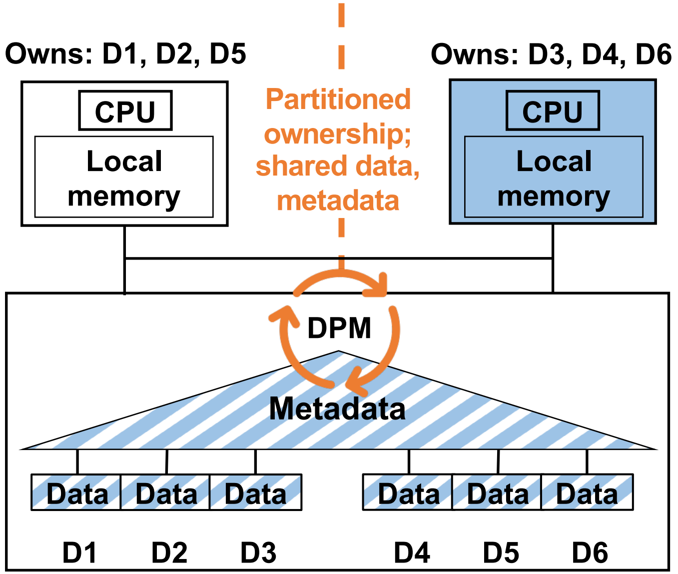


Insight: Data access and ownership can be an independent consideration owing to disaggregation

Approach: Partition ownership across compute nodes while sharing access to data through DPM

Ownership Partitioning

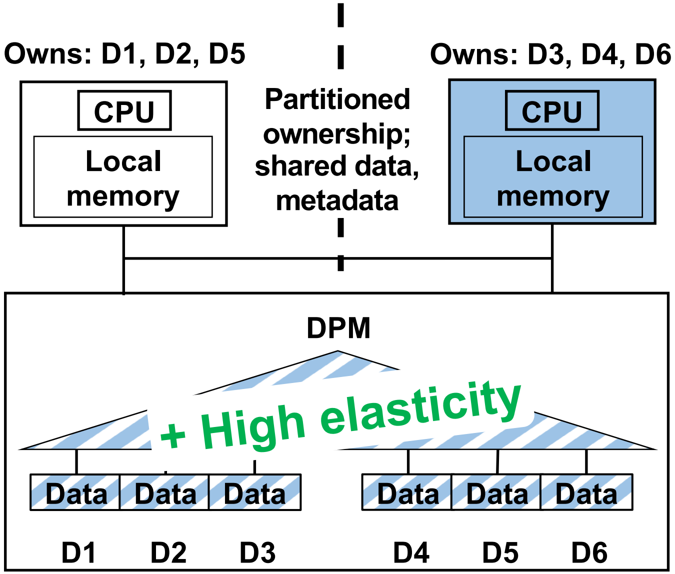
- Shared data/metadata
- Partitioned ownership



* Solid color: exclusive ownership
Hatched and mixed color: shared ownership

Ownership Partitioning

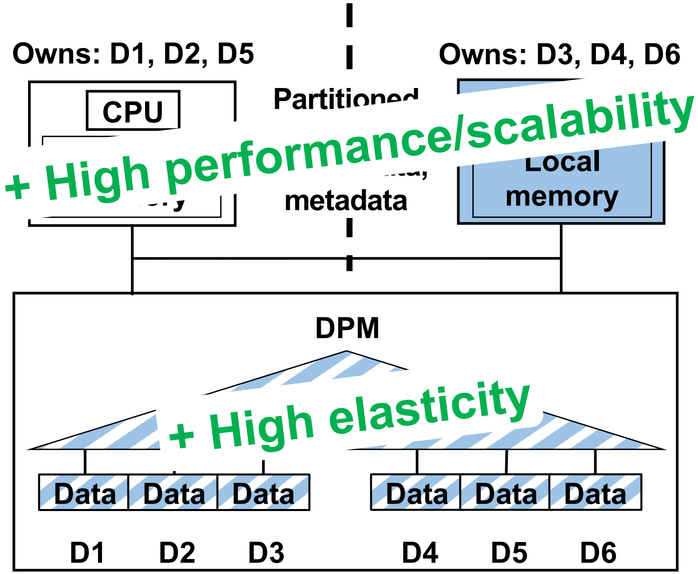
- Shared data/metadata
  - + Allow fast reconfiguration without expensive data reorganization
- Partitioned ownership



* Solid color: exclusive ownership
Hatched and mixed color: shared ownership

Ownership Partitioning

- Shared data/metadata
  - \+ Allow fast reconfiguration without expensive data reorganization
- Partitioned ownership
  - \+ Avoid consistency overheads between caches
  - \+ Preserve data locality of caches



* Solid color: exclusive ownership
Hatched and mixed color: shared ownership

Goals and design techniques

Goal	Dinomo technique
High performance	**Disaggregated adaptive cache**
	Ownership partitioning
Scalability	Ownership partitioning
Lightweight reconfiguration (Elasticity)	Ownership partitioning

Caching

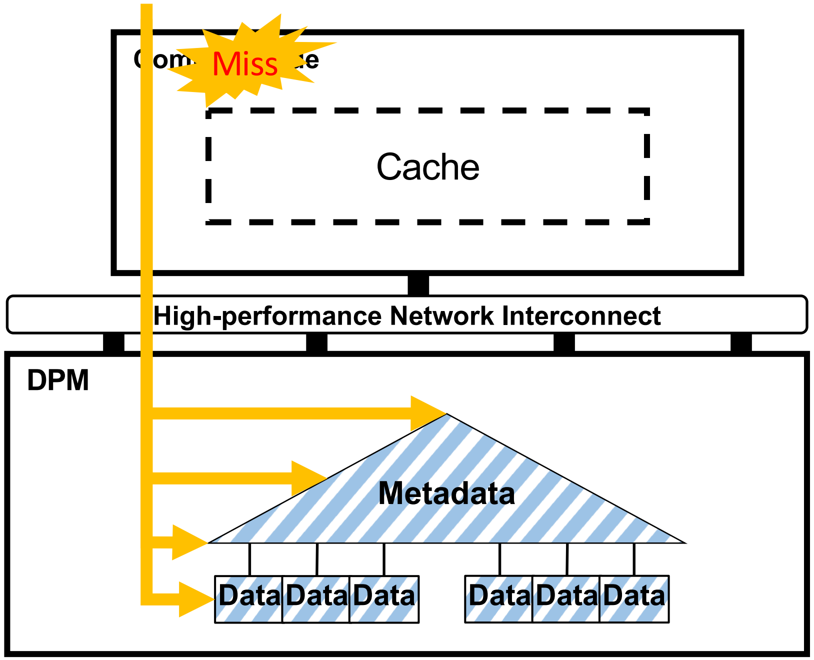
- Number of network round trips significantly impacts on overall system performance

Caching

- Number of network round trips significantly impacts on overall system performance
- Cache data or metadata into the memory of compute nodes to reduce round trips to DPM
  - Important to minimize cache misses

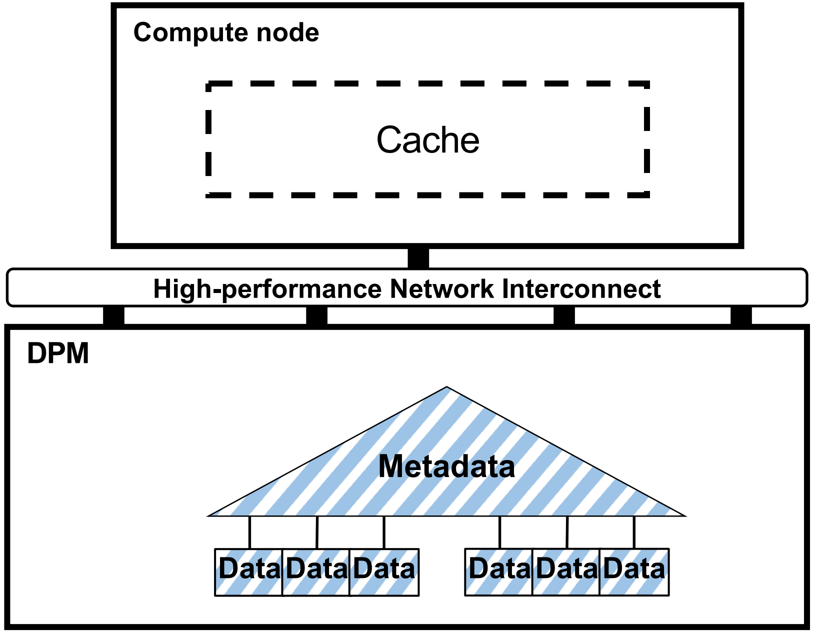
Caching

- Cache miss → multiple RTs
  - Traverse metadata index structures in DPM
  - Fetch data from DPM



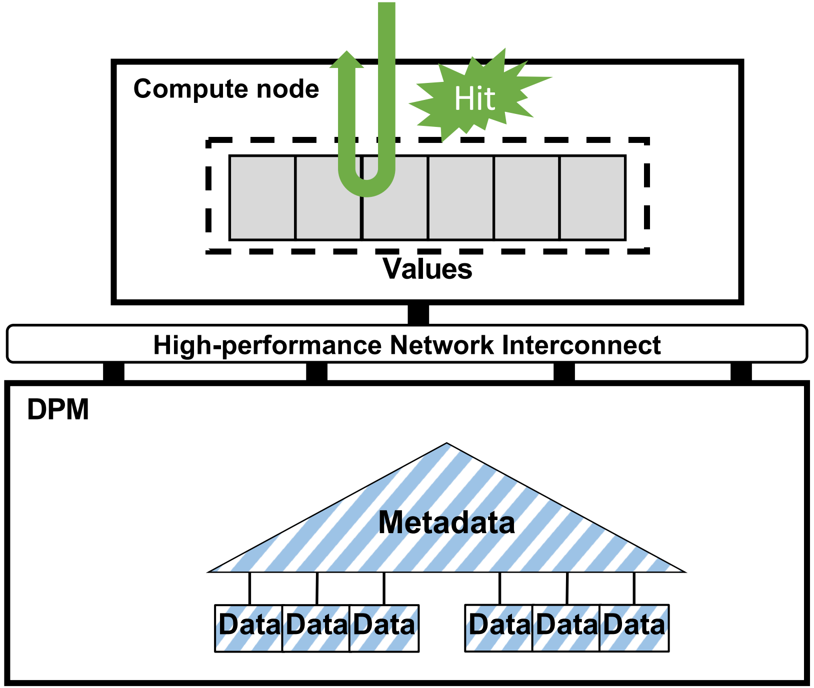
Caching

- Static policy
  - Value
  - Shortcut



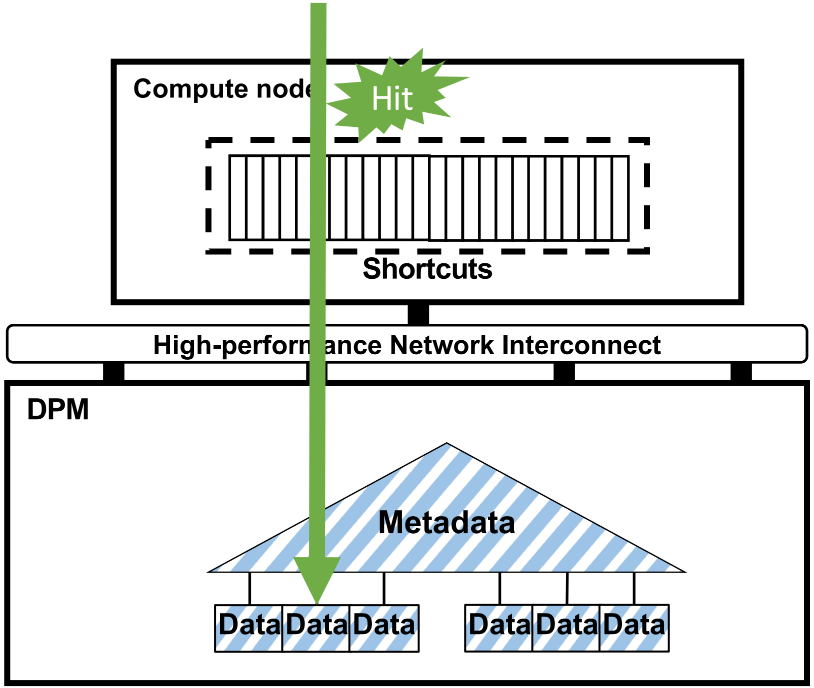
Caching

- Static policy
  - Value
    - Zero round trip, but more space
  - Shortcut



Caching

- Static policy
  - Value
    - Zero round trip, but more space
  - Shortcut
    - One round trip, but less space



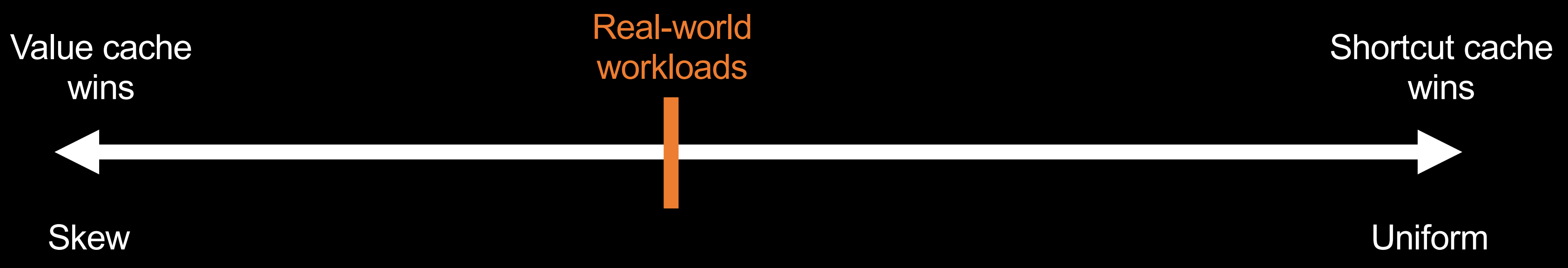
Is it better to cache a few values without overheads on hits, or a larger number of shortcuts with fixed hit overheads?

Is it better to cache a few values without overheads on hits, or a larger number of shortcuts with fixed hit overheads?

Value cache wins ⟷ Shortcut cache wins

Skew ⟷ Uniform

Is it better to cache a few values without overheads on hits, or a larger number of shortcuts with fixed hit overheads?



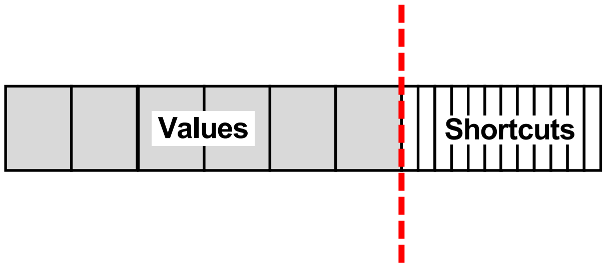
Is it better to cache a few values without overheads on hits, or a larger number of shortcuts with fixed hit overheads?

Answer: Efficient ratio depends on workload patterns and available memory space

We need an **adaptive policy** changing ratio between values and shortcuts!

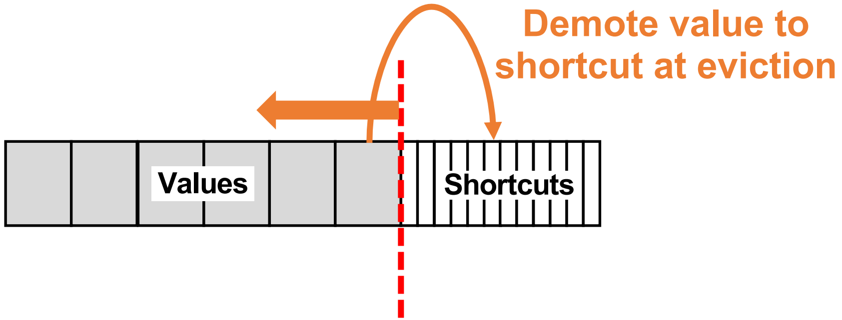
Disaggregated Adaptive Caching

- Adaptive policy
  - Change the boundary via demotion and promotion



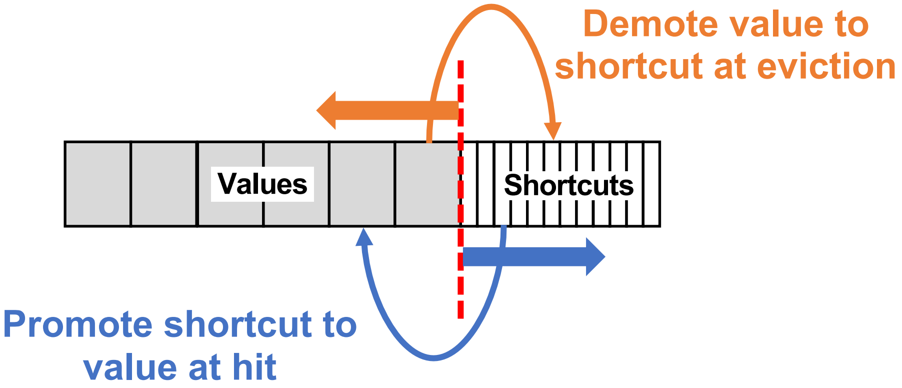
Disaggregated Adaptive Caching

- Adaptive policy
  - Change the boundary via demotion and promotion



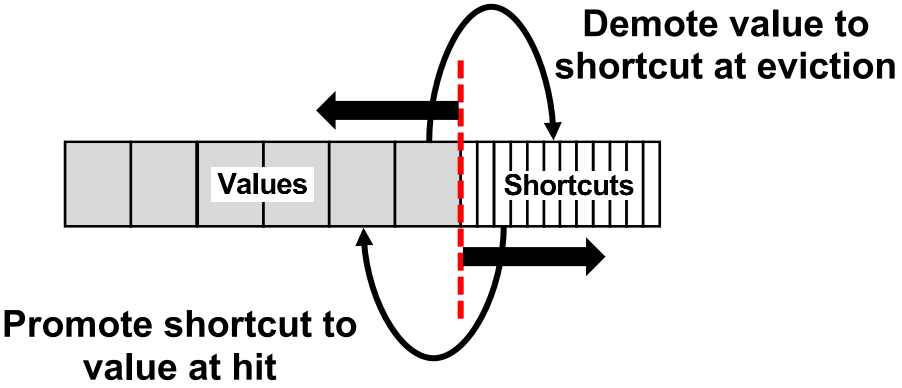
Disaggregated Adaptive Caching

- Adaptive policy
  - Change the boundary via demotion and promotion



Disaggregated Adaptive Caching

- Adaptive policy
  - Change the boundary via demotion and promotion
  - Promotion policy considering sizes, hit costs, and miss costs



Outline

- Ownership partitioning
- Disaggregated adaptive cache
- Evaluation
- Discussion

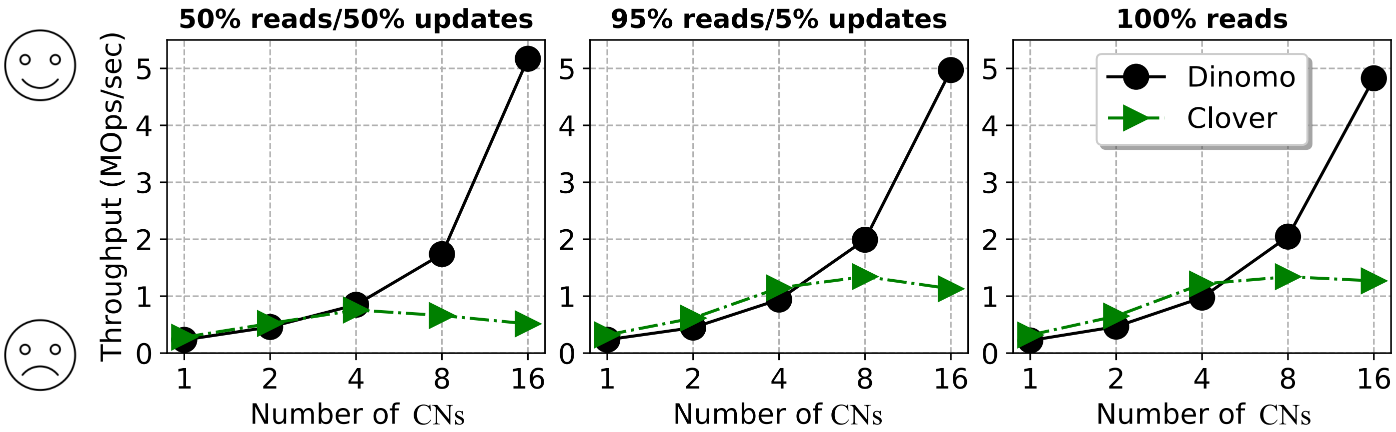
Evaluation

- How does DINOMO fare against the state-of-the-art in terms of performance and scalability?
- How elastic and responsive is DINOMO while handling changes in workloads?

Evaluation setup

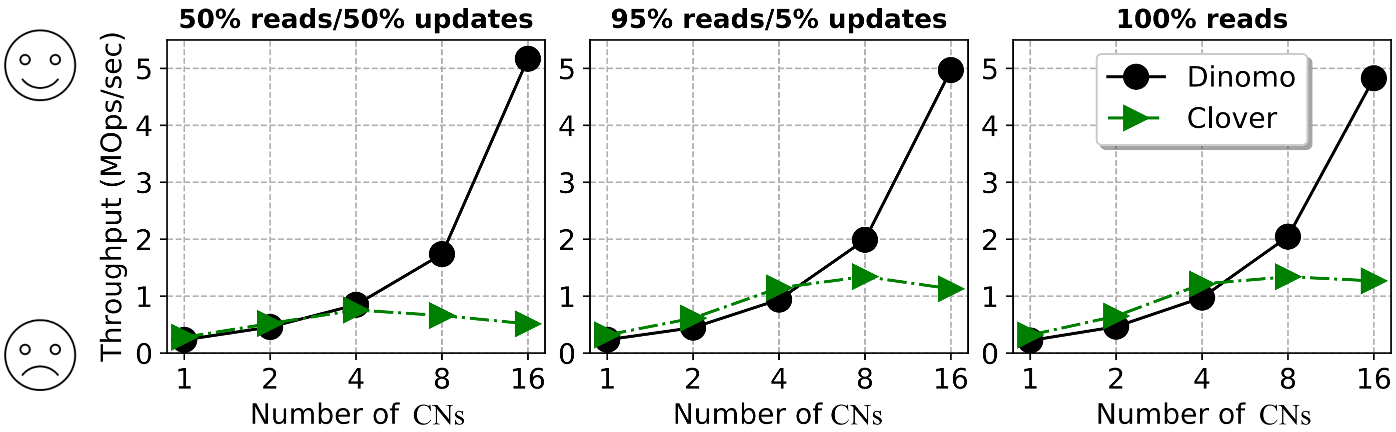
- System configuration
  - DPM: 4 threads, 110GB of DRAM to emulate PM
  - 16 CNs: 8 threads, 1GB of DRAM for caching (≈1% of the DPM)
  - Connected via 56Gbps ConnectX-3 RNICs
- Baseline
  - Performance/scalability: Clover (shared everything, shortcut-only cache)
  - Elasticity: DINOMO-N (DAC, but partition data/metadata)
- Workload
  - YCSB workloads with 8B keys and 1KB values

Performance and Scalability



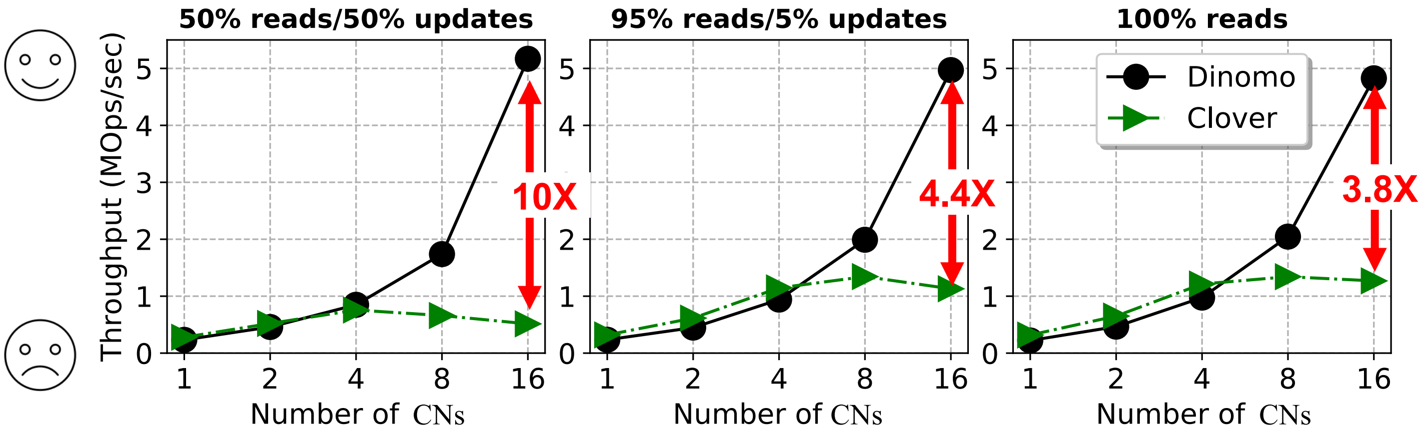
Performance and Scalability

- DINOMO scales to 16 CNs, but Clover does not beyond 4 CNs

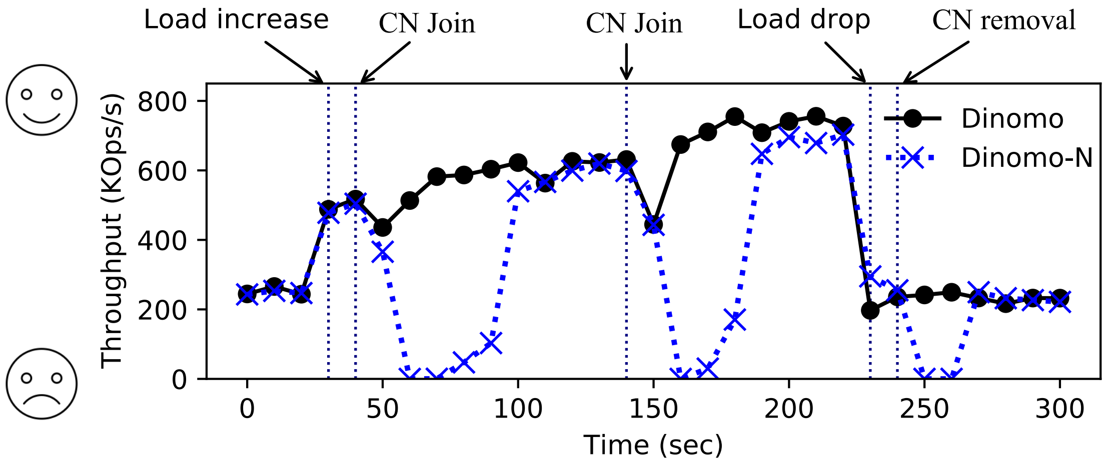


Performance and Scalability

- DINOMO scales to 16 CNs, but Clover does not beyond 4 CNs
- With 16 CNs, DINOMO outperforms Clover upto 10x



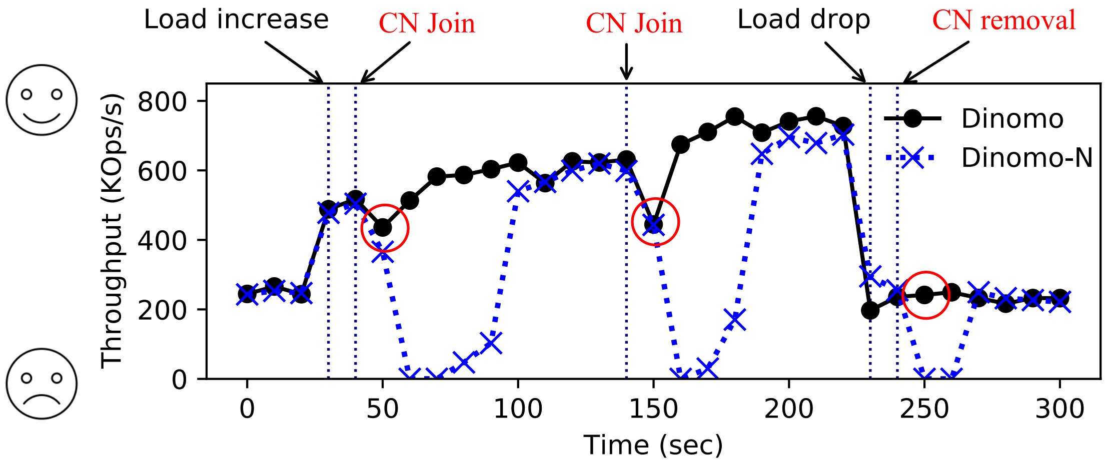
Elasticity



Load increase
CN Join
CN Join
Load drop
CN removal
Throughput (KOps/s)
800
600
400
200
0
0
50
100
150
200
250
300
Time (sec)
Dinomo
Dinomo-N

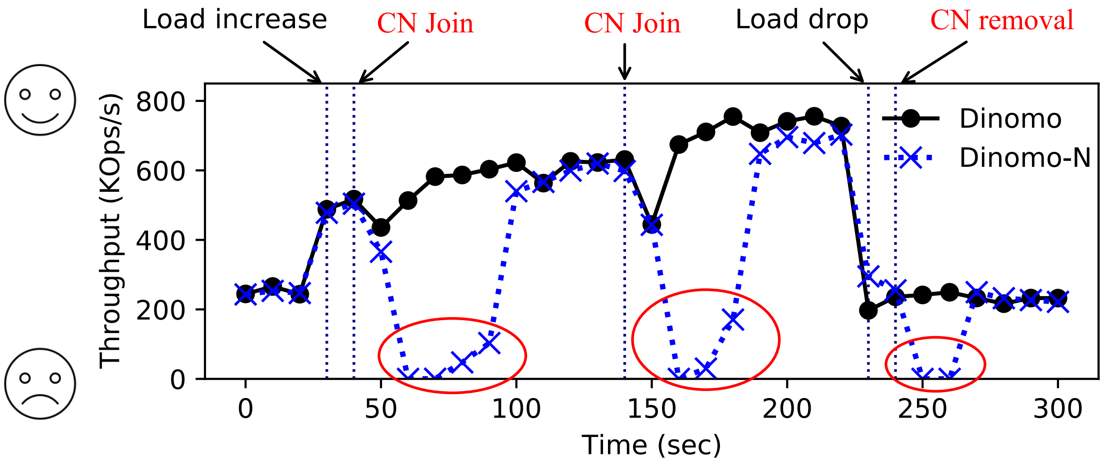
Elasticity

- DINOMO: Brief throughput dips when adding/removing CNs



Elasticity

- DINOMO: Brief throughput dips when adding/removing CNs
- DINOMO-N: Throughput dips for 20-40 seconds due to expensive data reorganization

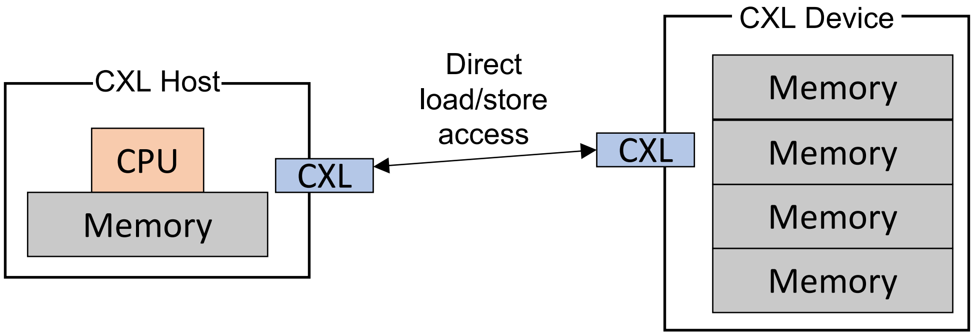


Outline

- Ownership partitioning
- Disaggregated adaptive cache
- Evaluation
- Discussion

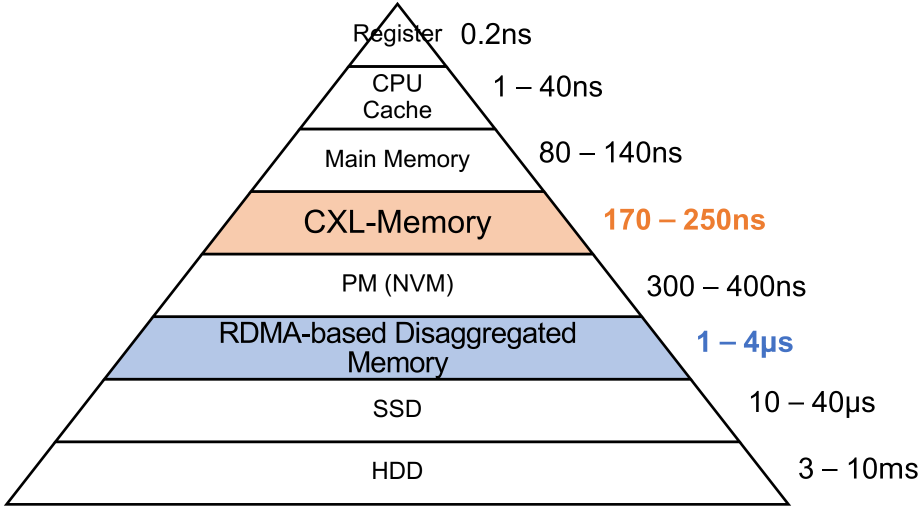
Discussion

- CXL (Compute Express Link)
  - A cache-coherent interconnect over PCIe
  - CXL.memory for memory expansion (Type 3)



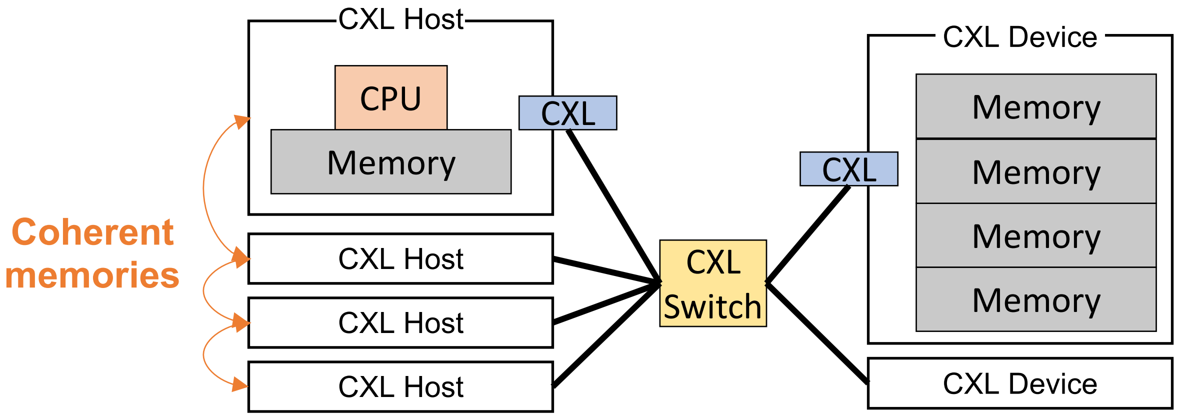
Discussion

- CXL VS. RDMA-based disaggregation
  - Lower access latency
    - CXL (170 - 250ns), RDMA (1 - 4μs)



Discussion

- CXL VS. RDMA-based disaggregation
  - Lower access latency
    - CXL (170 - 250ns), RDMA (1 - 4us)
  - Hardware-guaranteed coherence



Discussion

- **Ownership partitioning** in the context of CXL
  - CXL enables coherent memory sharing

Discussion

- **Ownership partitioning** in the context of CXL
  - CXL enables coherent memory sharing
  - Just because we can share doesn't mean we should

Discussion

- **Ownership partitioning** in the context of CXL
  - CXL enables coherent memory sharing
  - Just because we can share doesn't mean we should
  - Design principle of scalable system
    - Avoid cache coherence overheads

Discussion

- **Ownership partitioning** in the context of CXL
  - CXL enables coherent memory sharing
  - Just because we can share doesn't mean we should
  - Design principle of scalable system
    - Avoid cache coherence overheads
  - Ownership partitioning → Avoid coherence traffic between CXL-enabled hosts and devices

Discussion

- **Disaggregated adaptive cache** in the context of CXL
  - Caching data in compute nodes
    - A key to improve performance in RDMA-based disaggregation

Discussion

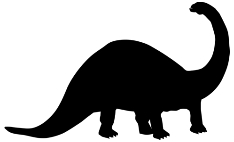
- **Disaggregated adaptive cache** in the context of CXL
  - Caching data in compute nodes
    - A key to improve performance in RDMA-based disaggregation
  - Much lower access latency to disaggregated memory over CXL

Discussion

- **Disaggregated adaptive cache** in the context of CXL
  - Caching data in compute nodes
    - A key to improve performance in RDMA-based disaggregation
  - Much lower access latency to disaggregated memory over CXL
  - Unclear if caching would be still useful for low-latency medias
    - e.g., page cache bypass in PM file systems to avoid cache-management and data-copy overheads

Discussion

- **Disaggregated adaptive cache** in the context of CXL
  - Caching data to the local memory of compute nodes has been a key to improve performance in disaggregation settings
  - With CXL, access latency to disaggregated memory becomes much lower than RDMA
  - Unclear if caching would be still useful for low-latency medias
    - e.g., bypassing page cache in PM file systems to avoid cache-management and data-copy overheads
    - Future work: How to utilize host local memory well

DINOMO 

- First KVS for DPM achieving high performance, scalability, and elasticity simultaneously
- Use a novel combination of techniques, ownership partitioning and disaggregated adaptive cache
- Experimentally show DINOMO can scale performance and efficiently react to reconfigurations
- Try our KVS: https://github.com/utsaslab/dinomo