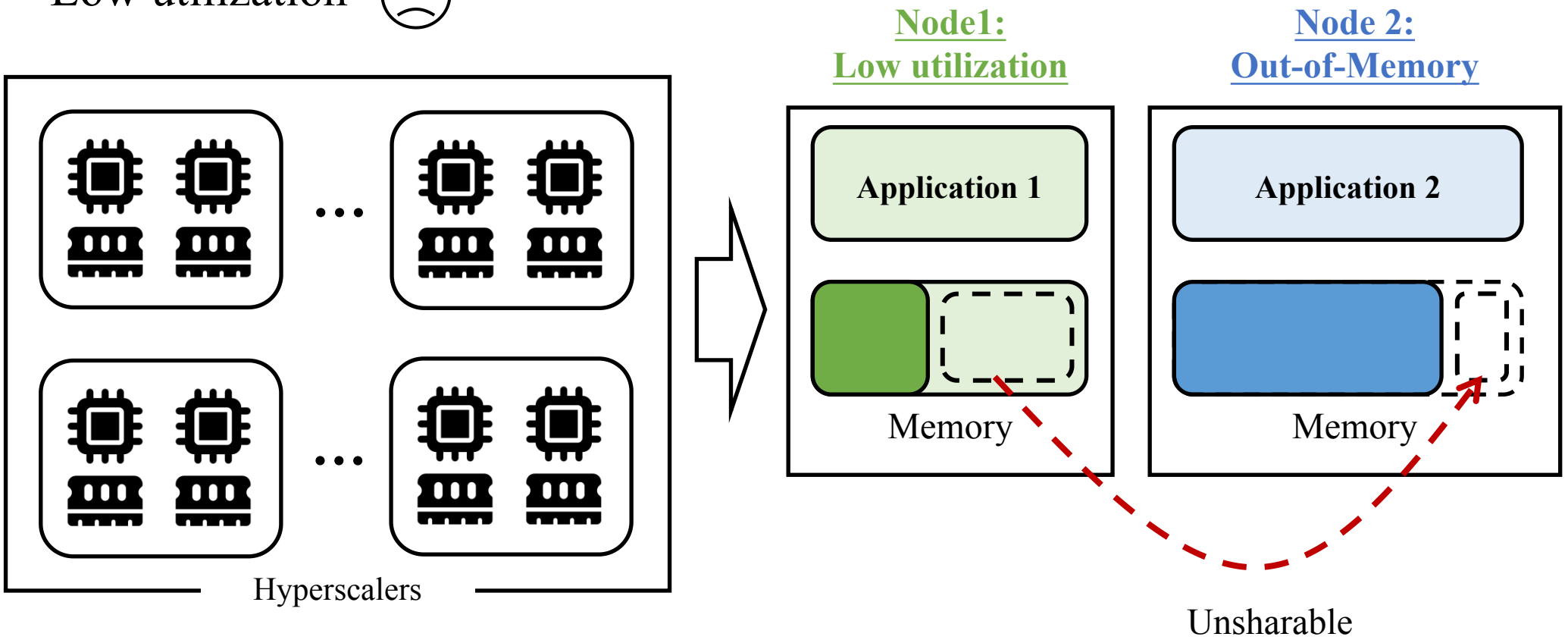
FineMem: Breaking the Allocation Overhead vs. Memory Waste Dilemma in Fine-Grained Disaggregated Memory Management

Xiaoyang Wang, Yongkun Li, Kan Wu, Wenzhe Zhu, Yuqi Li, Yinlong Xu

Memory Costs in Hyperscale Systems

Monolithic Servers: Coupled CPU and Memory

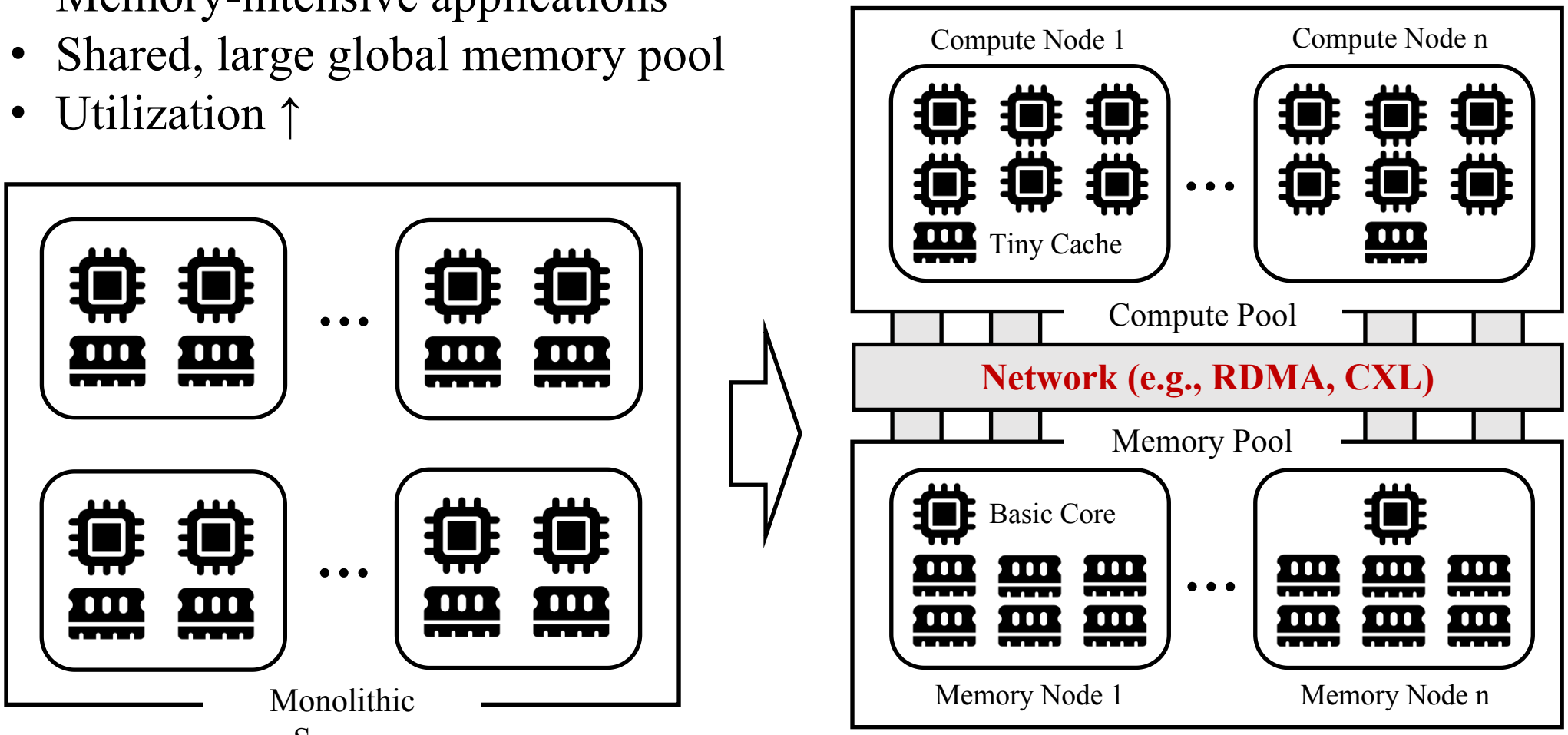
- Memory-intensive applications
- Large per-node memory capacity
- Low utilization ☹



Memory Disaggregation

Memory Disaggregation: Decoupled Compute Nodes and Memory Nodes

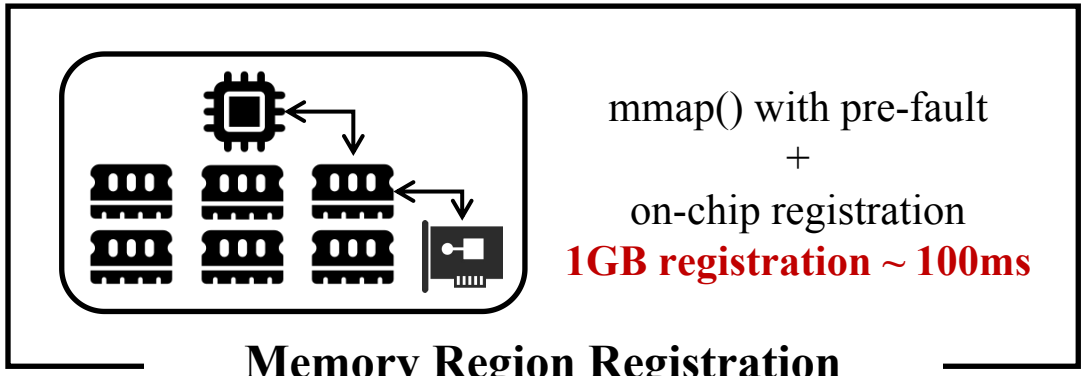
- Memory-intensive applications
- Shared, large global memory pool
- Utilization ↑

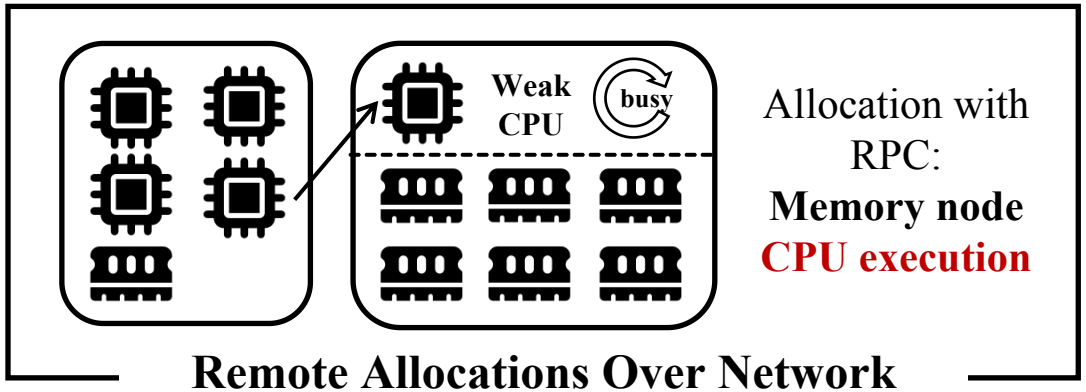


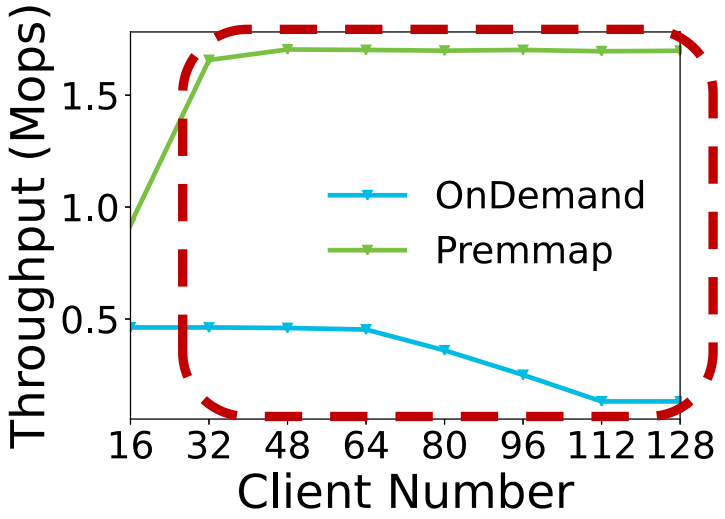
Memory Disaggregation - Allocation Overheads

RDMA-based disaggregated memory allocation: performance challenges

- **MR registration latency**
- **RPC execution overheads**





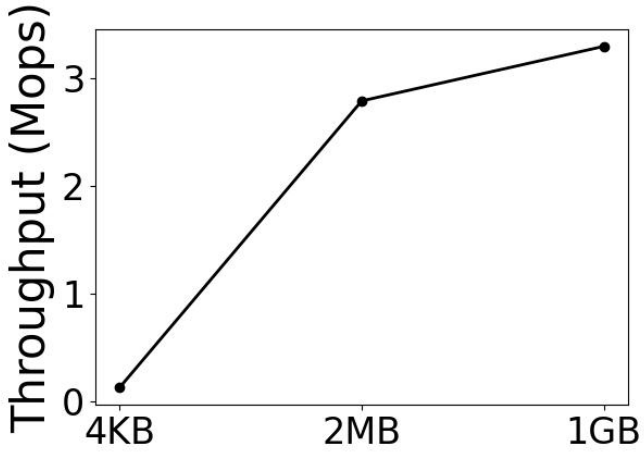


DM KV-Store in a YCSB-A workload

Trade-off: Coarse-grained Memory Management

Existing DM systems: **Coarse-grained (MB-GB) Memory Management**

- Reduce allocation frequency → Lower allocation overheads



DM KV-Store with different block allocation size (YCSB-A)

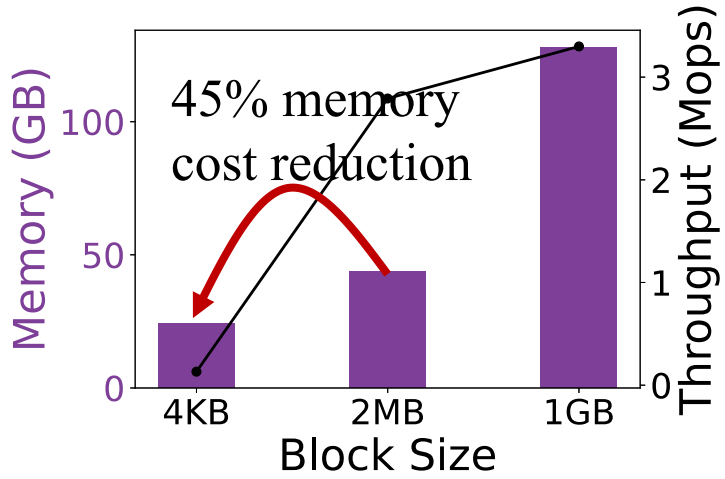
State-of-the-Art Works	Management Granularity
FaRM@NSDI'14	2GB
Fastswap@Furosys'20	32GB
FUSEE@FAST'23	16MB
CXL-SHM@SOSP'24	64MB

- **Performance**
- **Memory Efficiency ?**

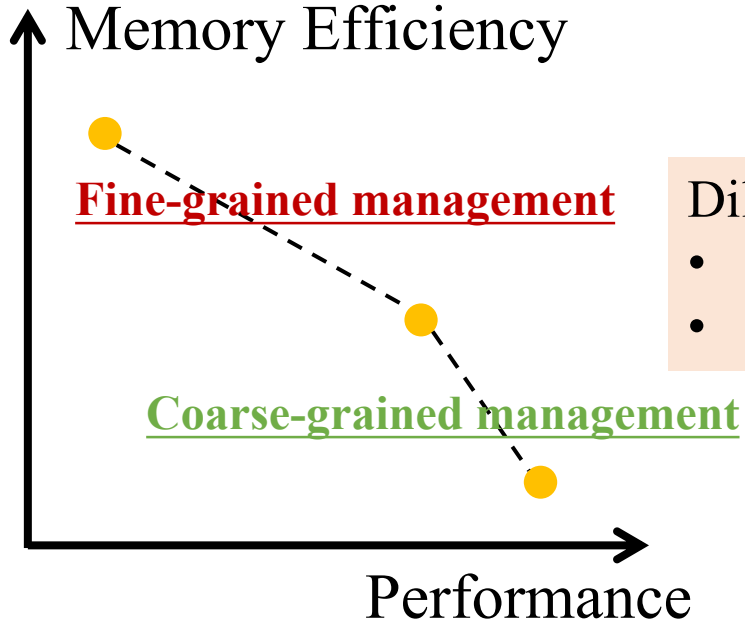
Exploring the DM Dilemma: Performance vs. Efficiency

Existing DM systems: **Coarse-grained (MB-GB) Memory Management**

- Reduce allocation frequency → **Causing significant memory waste**



DM KV-Store with different block allocation size (YCSB-A)



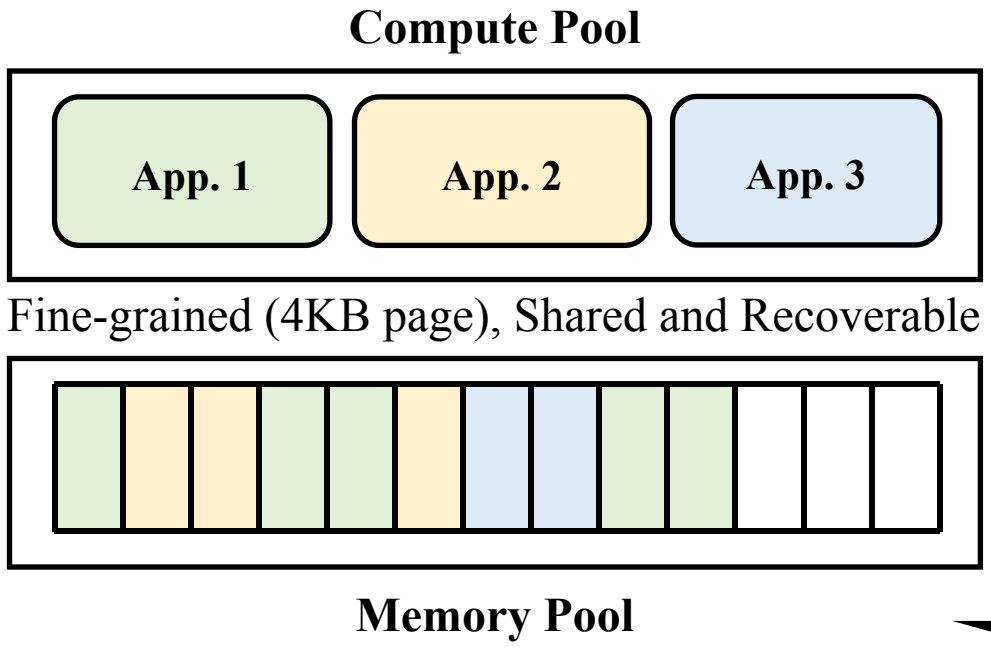
Dilemma:

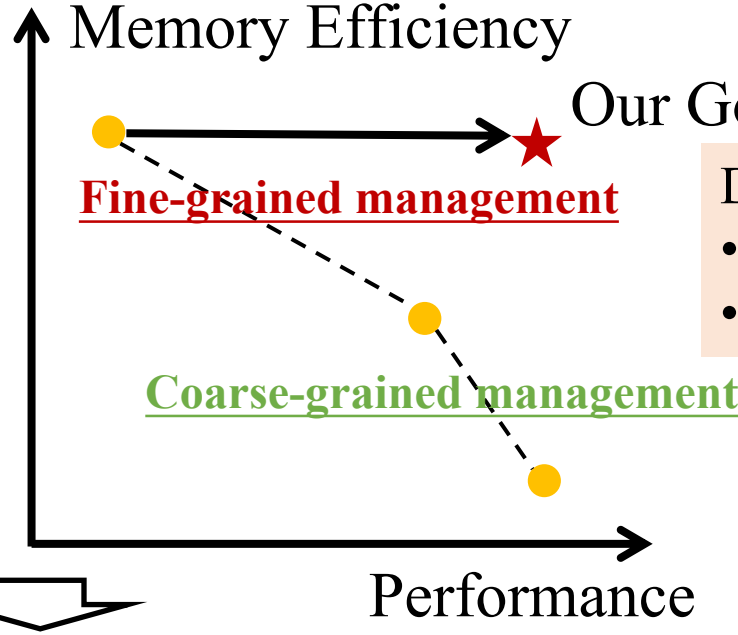
- Performance
- Memory Efficiency

DM Management Dilemma: Allocation Overheads vs. Memory Waste

Solving the DM Dilemma with Fine-grained Management

FineMem: A fine-grained remote memory management system in RDMA-based DM





Dilemma:
- Performance
- Memory Efficiency

Key Challenges

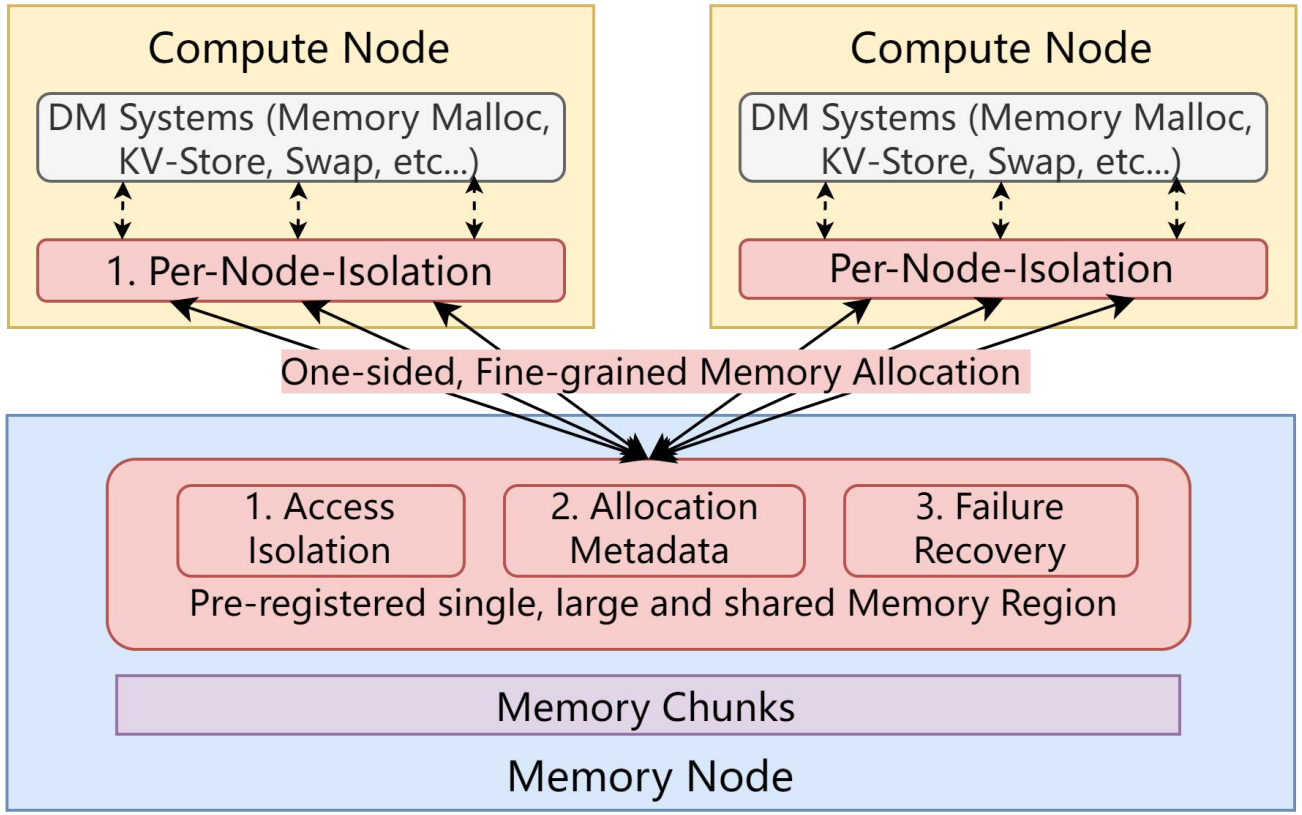
MR Registration Latency

Fine-grained Allocations Over Network

Memory Recovery during Compute Node Failures

Key Design Elements of FineMem

FineMem: A fine-grained remote memory management system in RDMA-based DM



1. Global MR while Overcoming Isolation Challenges

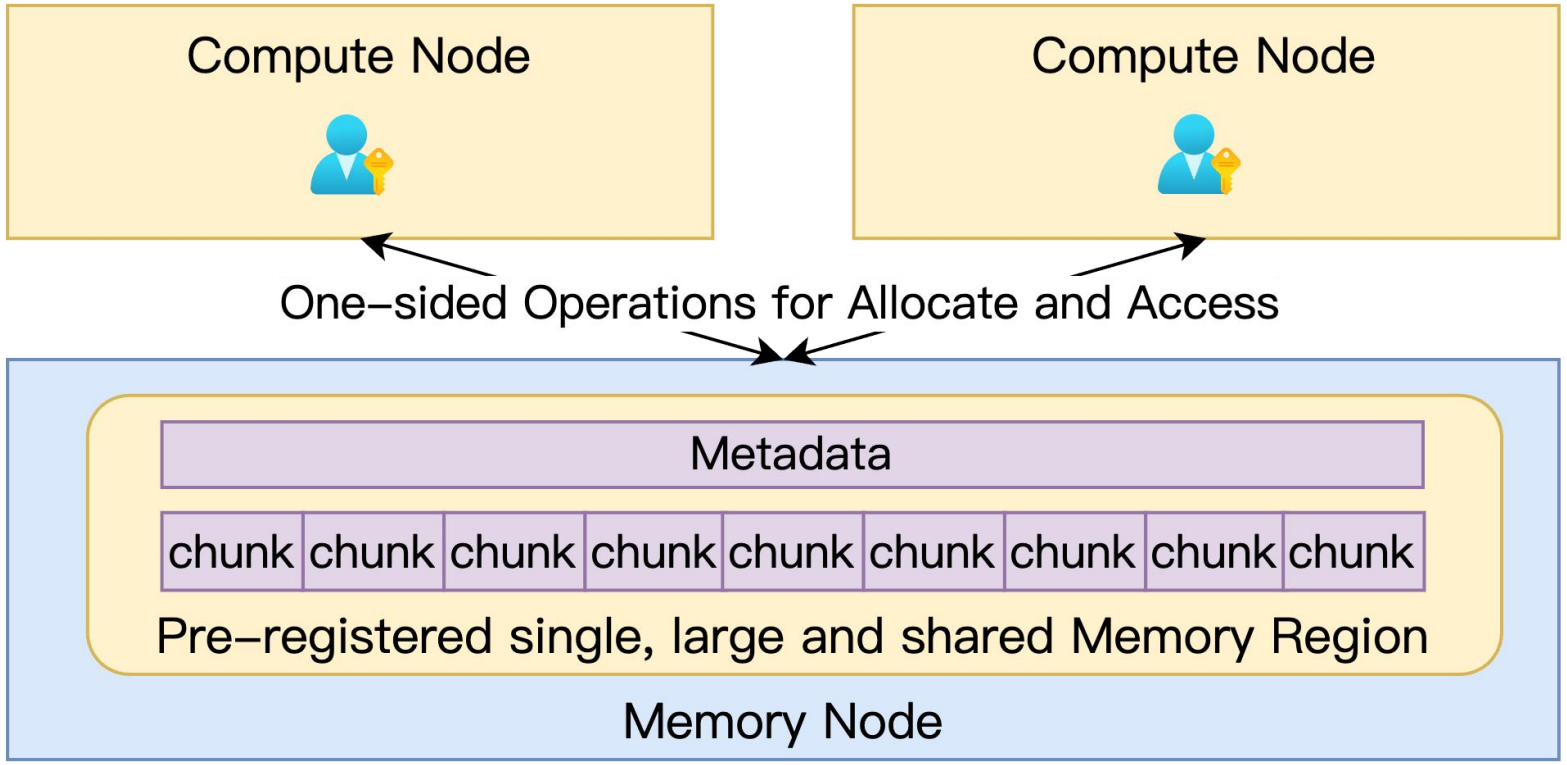
2. One-sided, Efficient Remote Memory Chunk Allocation

3. Memory Metadata Consistency

Memory Pool sharing a Single Memory Region

Strawman Solution: Pre-registered single, large and shared memory region

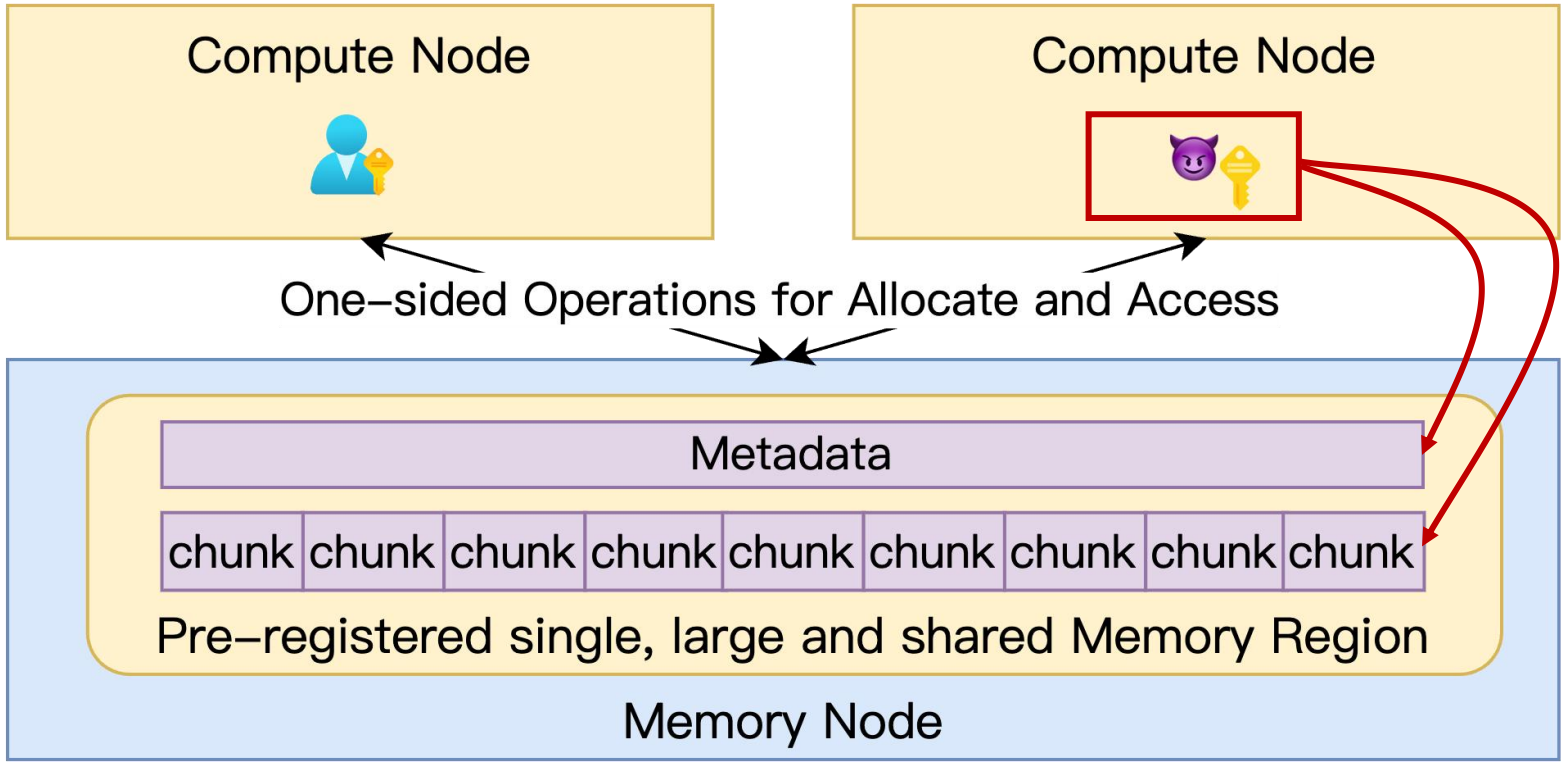
- Fine-grained memory chunks among multiple nodes
- Allocation via directly modifying shred allocation metadata



Memory Pool sharing a Single Memory Region

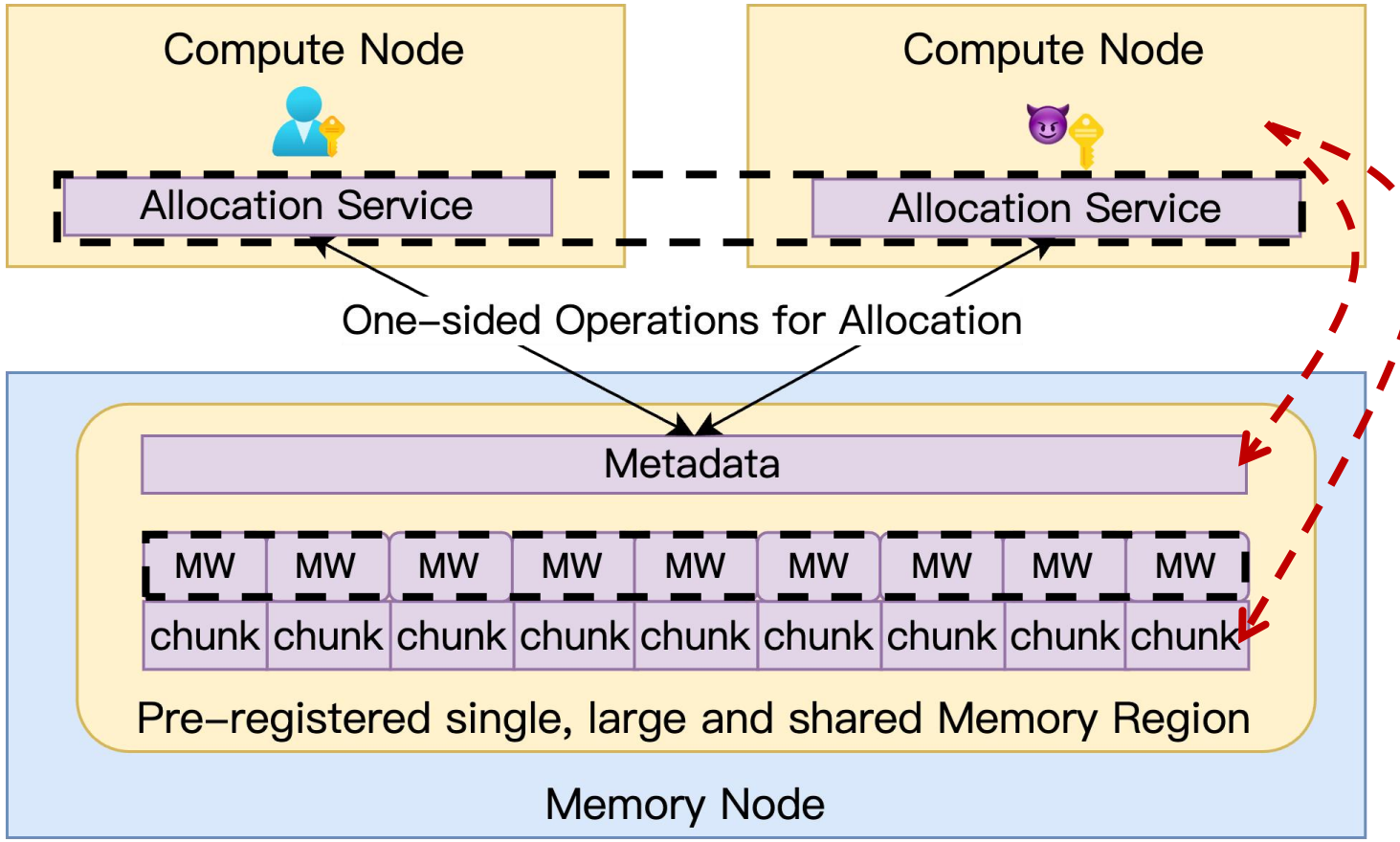
Isolation Challenges:

- Privacy and security risks among nodes
- Memory chunks and memory allocation metadata **unprotected**



FineMem: Global Shared Memory Pool

FineMem Solution: MW-based Chunk Isolation + Compute-Node Runtime Service



Chunk Isolation

Fine-grained Memory Window

- Fast MW (re)generation
- Each chunk a MW rkey

Metadata Isolation

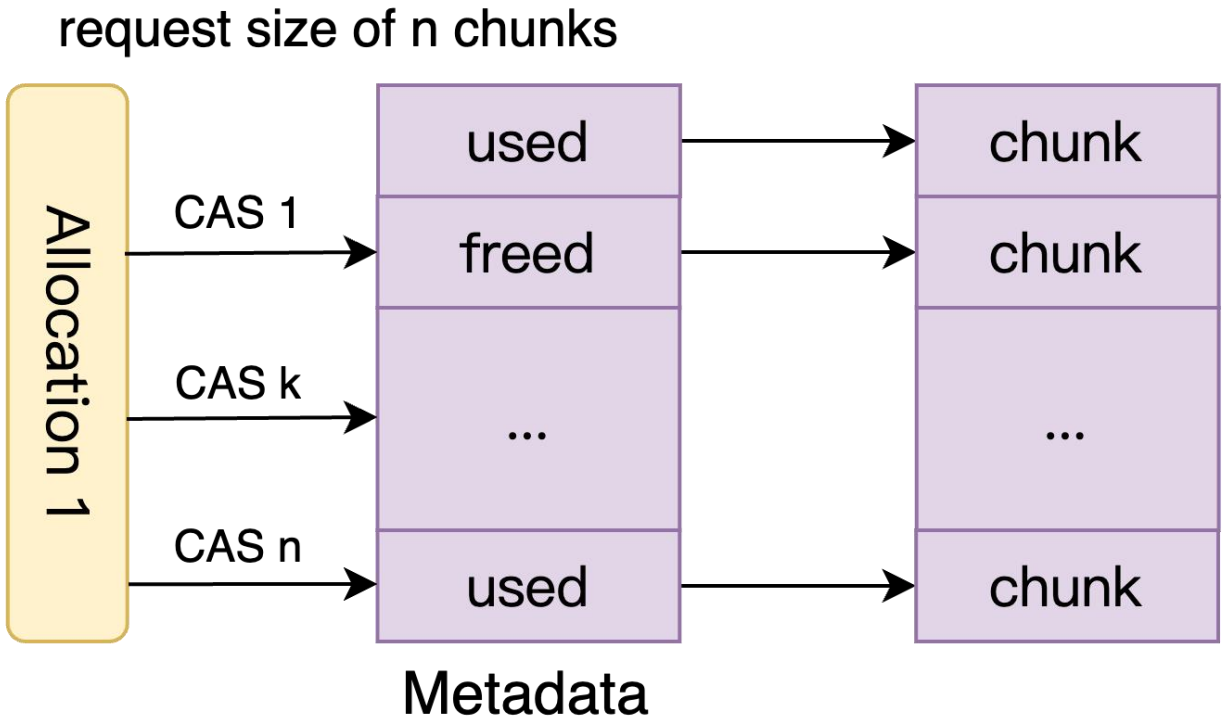
Per-Node Allocation Service

- Trusted service
- Users call service via IPC

One-sided DM Allocation: Strawman Solutions

Strawman Solution: One Chunk per Entry (8B)

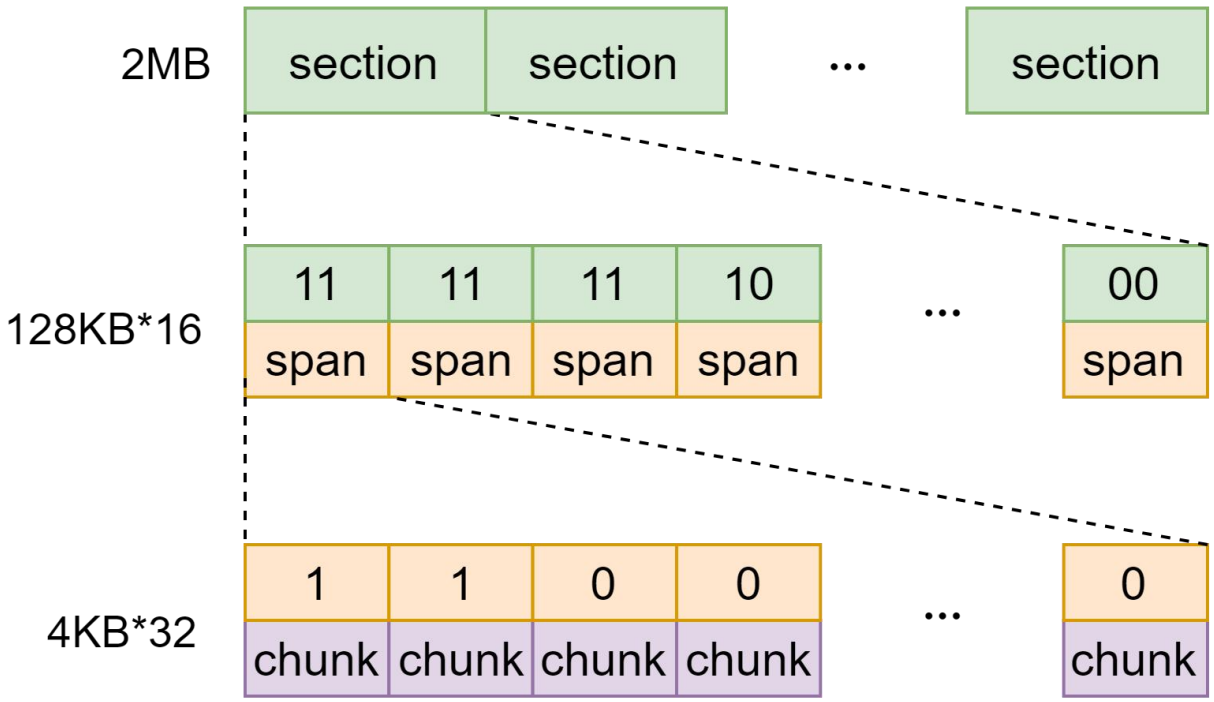
- One RDMA Compare-And-Swap (CAS) update to occupy one chunk
- **Variant-size Allocation Challenge**
  - E.g., 4KB base chunk, 2MB allocation requires 512 CAS



FineMem's One-sided DM Allocation: Two-Layer Bitmap

FineMem Solution: Two-Layer Bitmap

- [First-Layer] Section: 32-bit bitmap (16 spans, 2-bit per span) + 32-bit redo-log
- [Second-Layer] Span: 32-bit bitmap (32 chunks, 1-bit per chunk) + 32-bit redo-log
- [Base-Block] Chunk



One-layer bitmap: limited to 64 chunks

Two-layer bitmap:

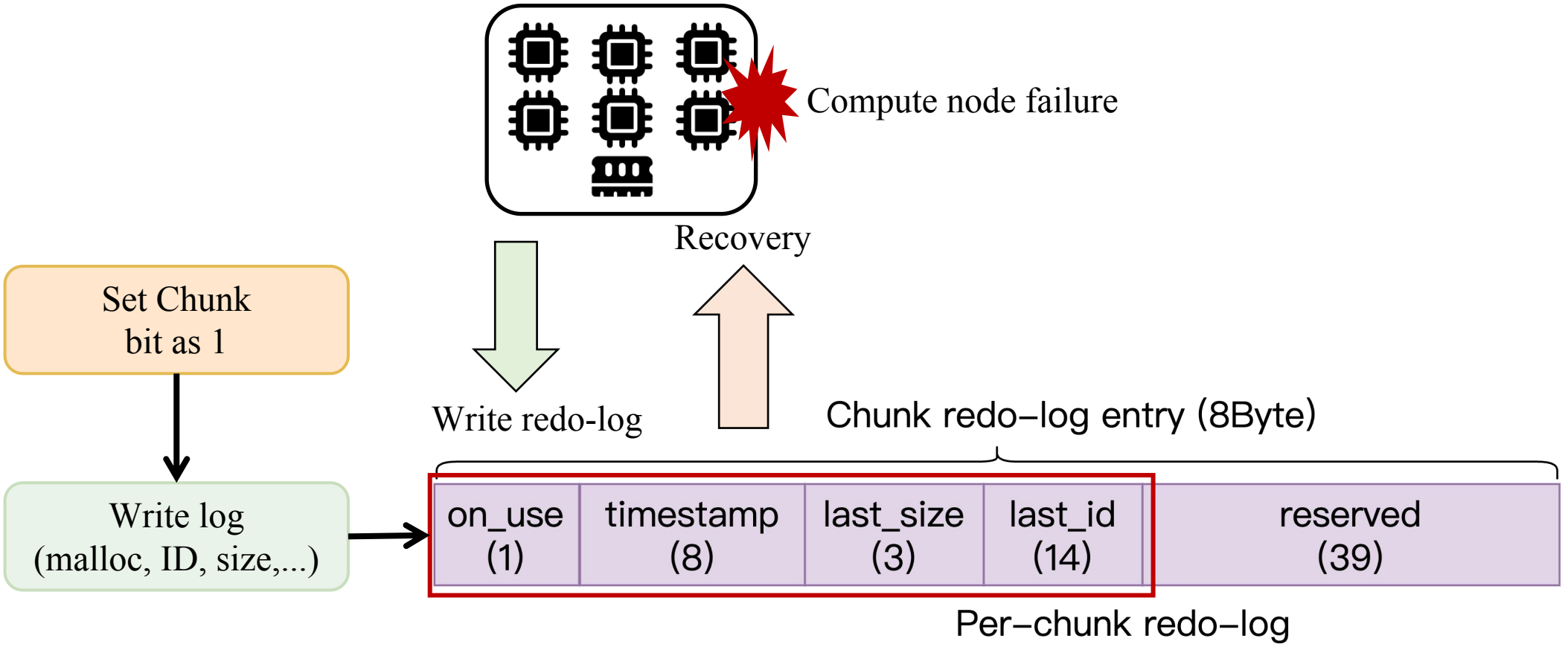
- Search/allocate 512 (16*32) chunks with up to 2 RDMA request
- Reserve 32-bit as redo-log, discuss later

Span bits	Span status
00	Empty
01	Partially Allocated
10	Contended
11	Full

Compute Node Failure Recovery via Redo-Log

Starwman Solution: Recovery Memory Chunks using Redo-Log

- Redo logging on memory nodes, written by compute nodes via one-sided RDMA,

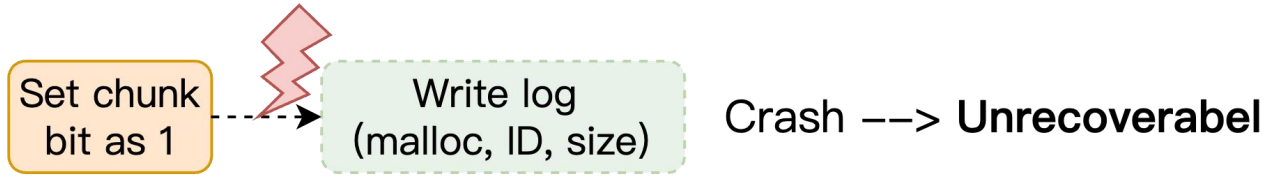


Metadata Consistency during Compute Node Failure

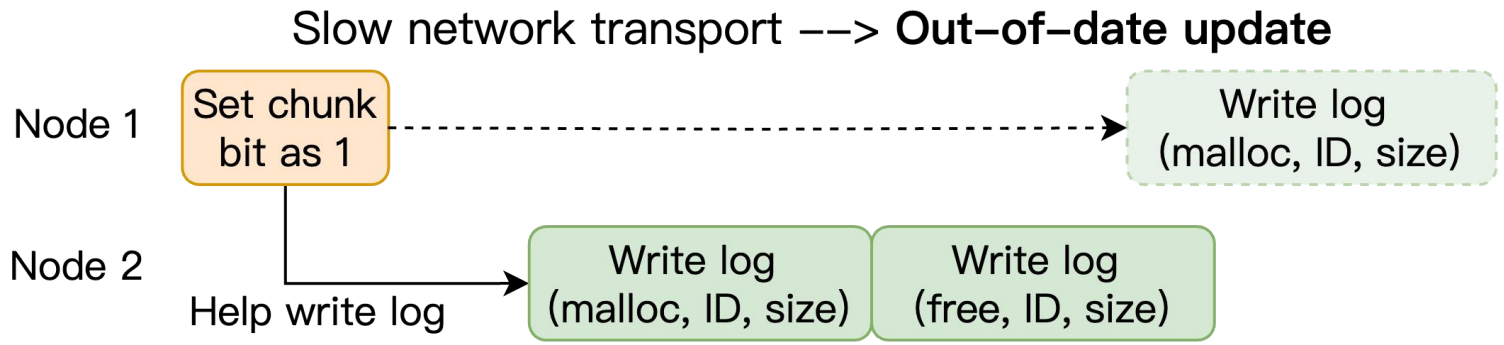
Meatadata Consistency Challenges:

1. **Crash failure**: chunk unrecoverable
2. **Fail-slow failure**: outdated log writing

Case 1:



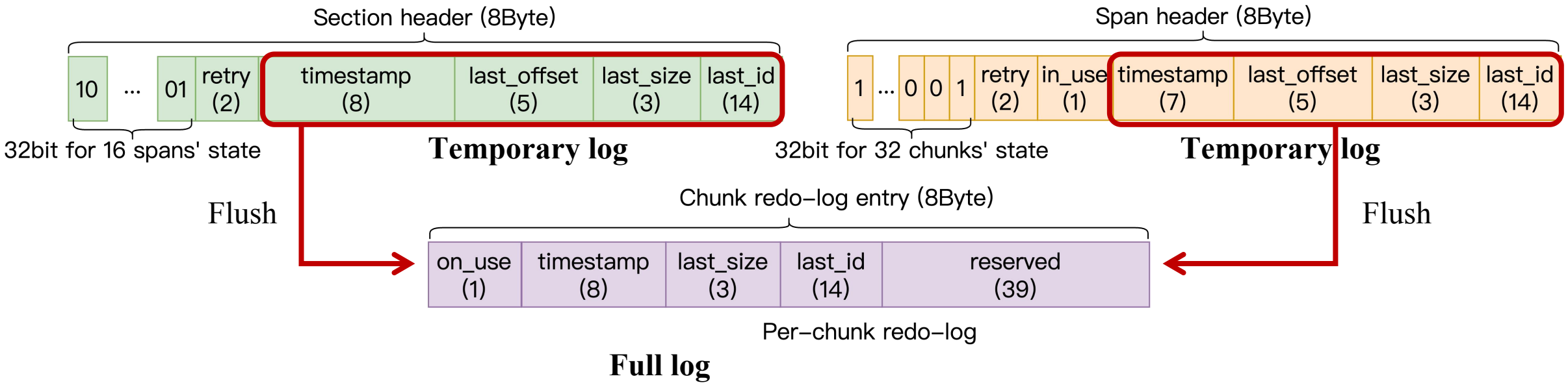
Case 2:



Metadata Consistency in FineMem

FineMem Solution: Commit point with temporary log

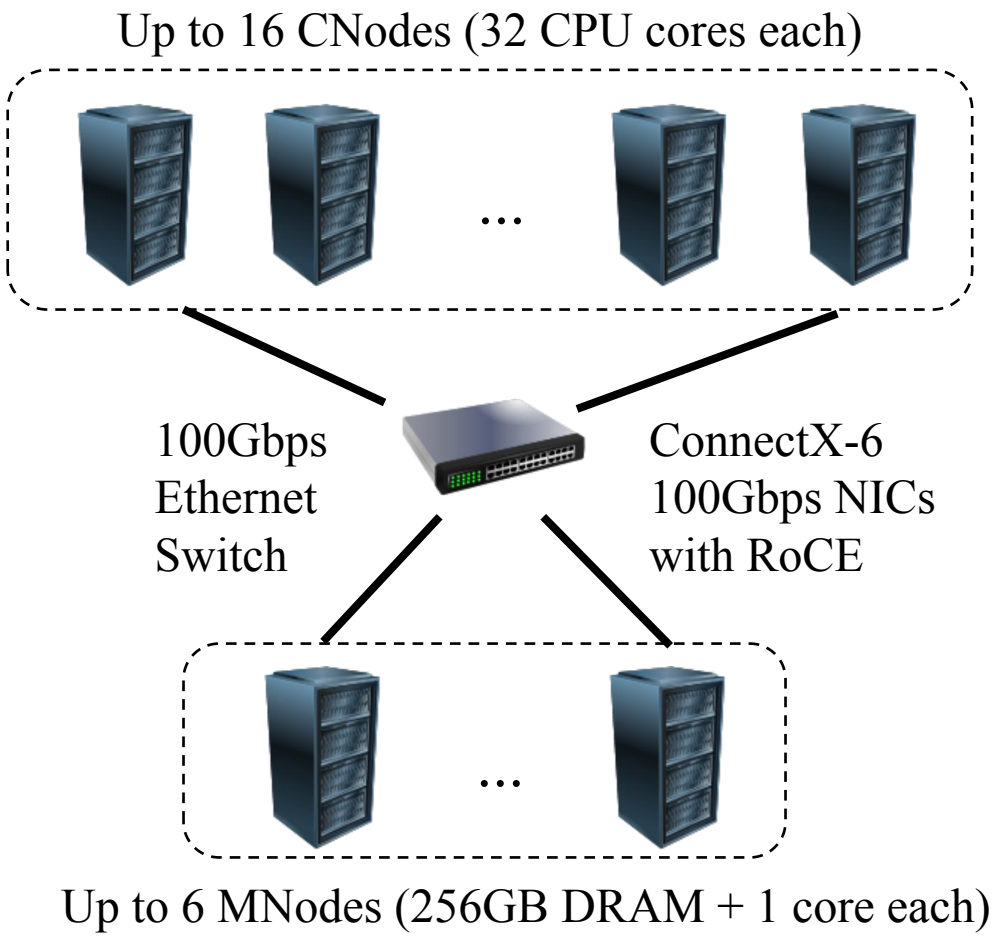
- **Temporary log** in the section/span headers
  - Flush to **full log** asynchronously (via any node)
  - **Timestamp** for out-of-date detection



Evaluation Setups

Baselines

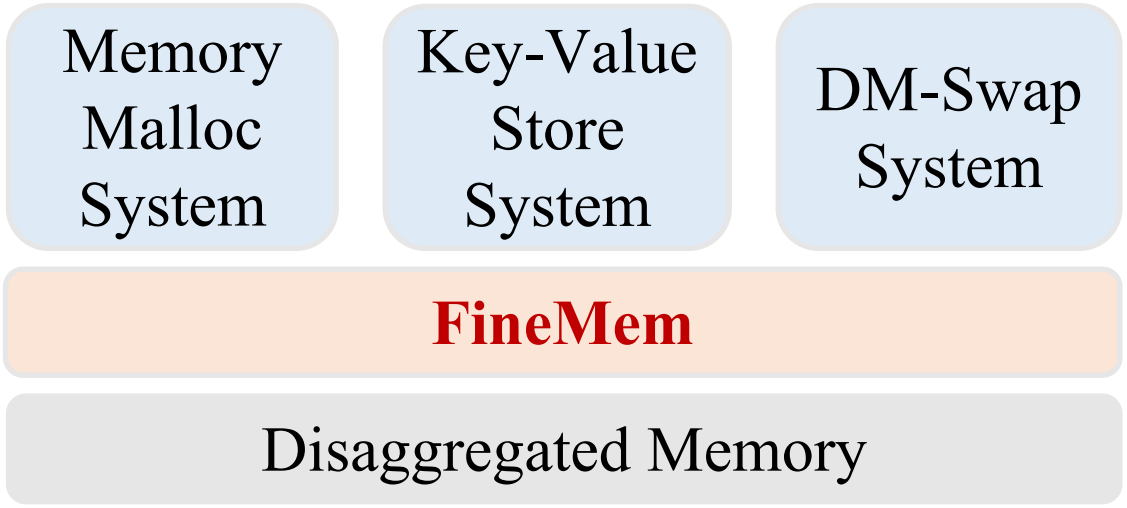
1. **OnDemand-RPC**
   - On-demand MR registration
   - RPC allocation
   - [CoRM@SIGMOD'21]
2. **Premmap-RPC**
   - Pre-registered MR
   - RPC allocation
   - [FUSEE@FAST'23]
3. **Premmap-One-sided**
   - Pre-registered MR
   - Array-based one-sided allocation
   - [CXL-SHM@SOSP'24]



Evaluation Setups: DM Systems Working with FineMem

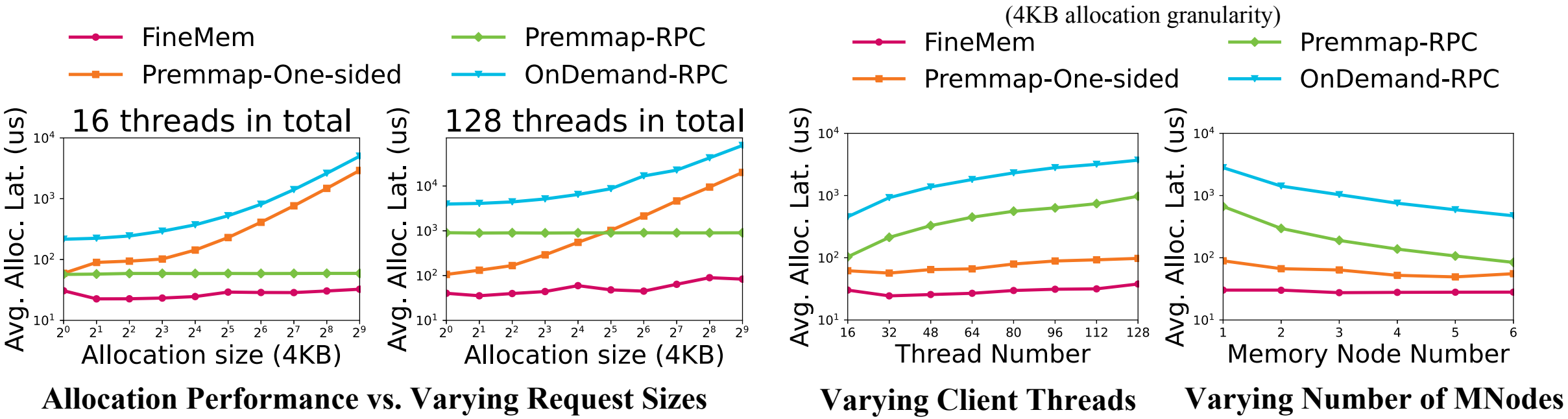
DM Systems Case Study:

DM Usage Cases	Based on	Workloads
Memory Malloc System	Mimalloc	Malloc benchmarks
Key-Value Store System	FUSEE	YCSB workloads
DM-Swap System	Fastswap	Unmodified applications



Evaluation: Microbench

Microbenchmarks: different allocation granularity and scalability

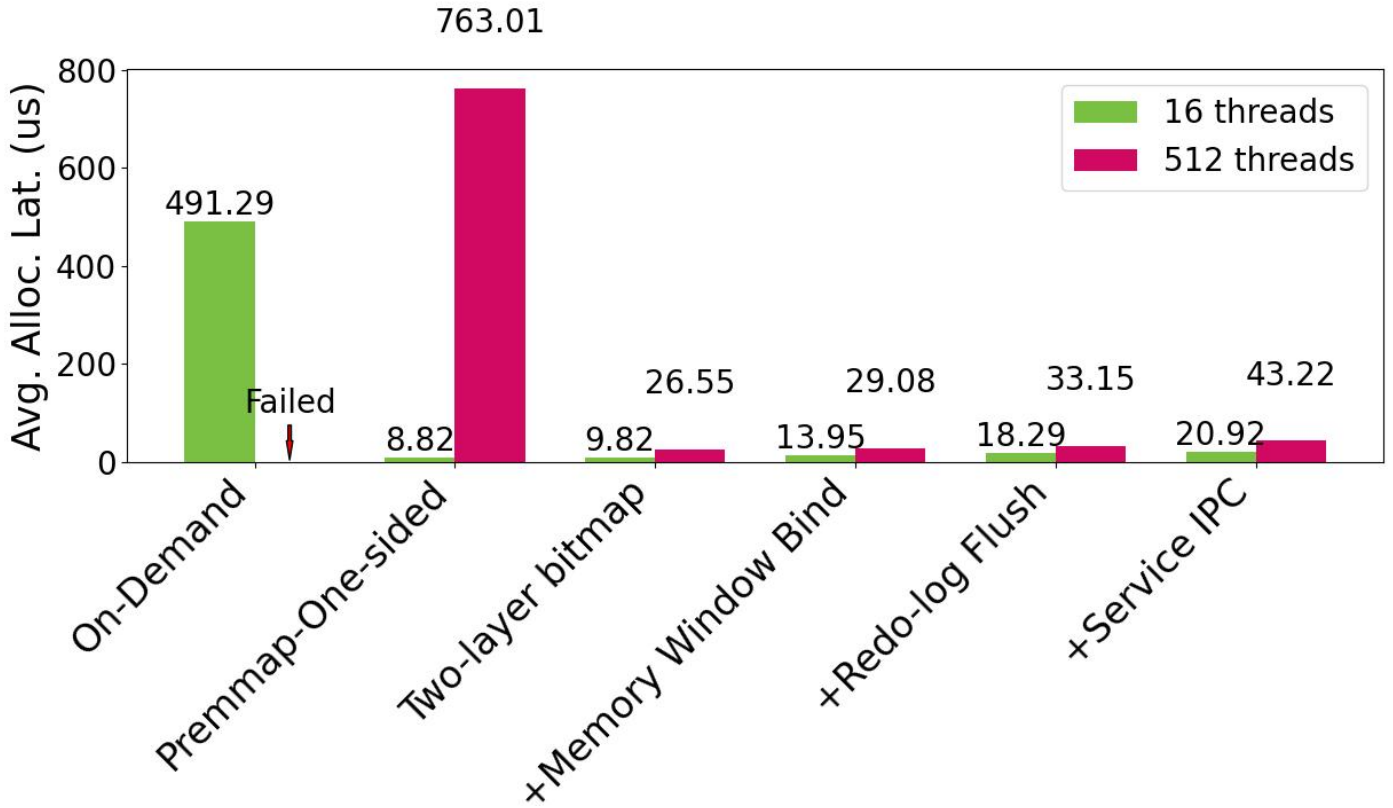


Allocation Performance vs. Varying Request Sizes **Varying Client Threads** **Varying Number of MNodes**

- FineMem outperforms across varying allocation granularities (4KB-2MB) and levels of parallelism (~128 Client Threads and ~6 memory Nodes)
- Reduces allocation latency by approximately 95%

Evaluation: Microbench

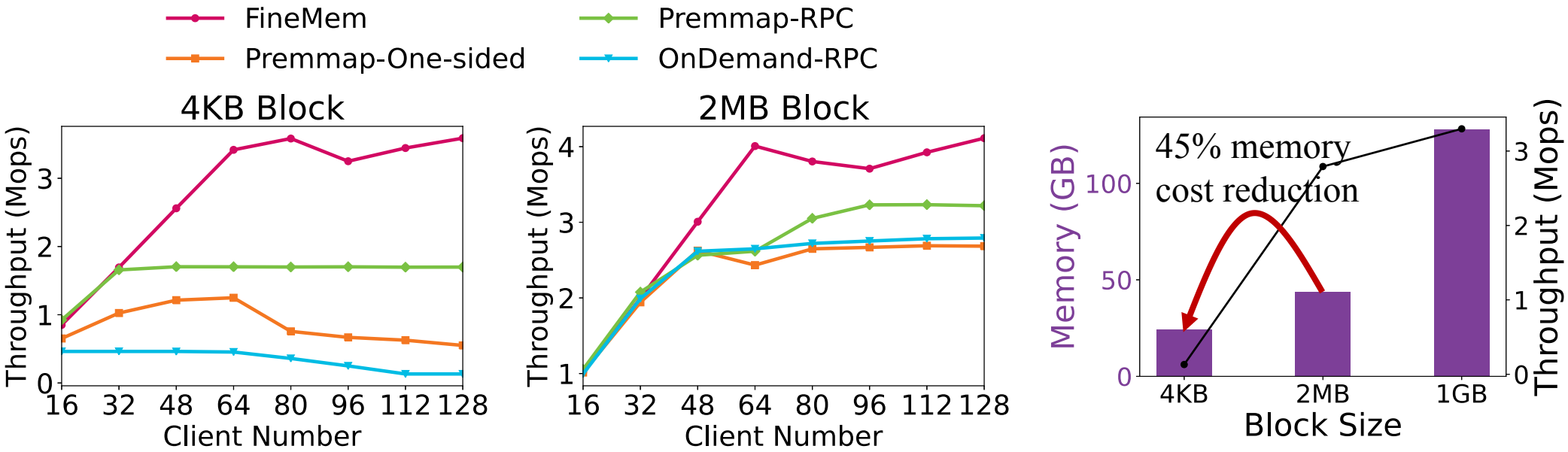
Factor analysis of FineMem's design mechanisms



FineMem overheads: MW bind + allocation service + redo-log write ~ 10-20us

Case Study: Key-value Store System

FUSEE with FineMem as the memory block allocation backend

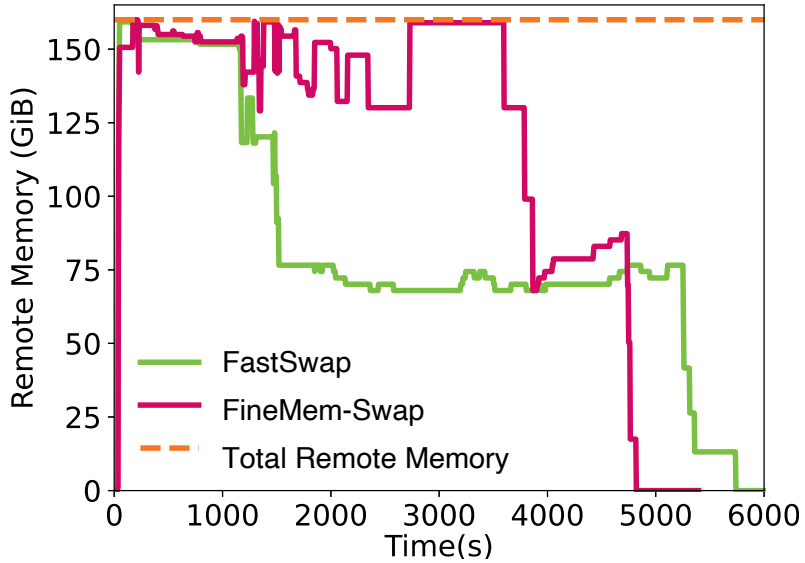


YCSB-A (Search:Update=50:50)

- Reduces the allocation overheads in KV-update operations
  - A bandwidth improvement of around 27% to 110%
- Achieves 45% reduction in memory cost with 4KB fine-grained block

Case Study: DM-Swap System

Fastswap with FineMem as the dynamic swap-out page allocation backend



Dynamic swap system vs. Static premmap swap system

- Higher memory utilization: 41.39% → 74.06%
- More jobs running concurrently in DM: job throughput improvement 8.38% ~10.69%

Summary

Background: Existing RDMA-based DM systems face a tough choice: either endure significant allocation overheads or accept substantial memory waste

FineMem: A distributed remote memory management system in RDMA-connected DM
To support fine-grained, high-performance memory allocation

- Removing MR registration while overcoming critical isolation challenges
- Enabling one-sided, efficient remote memory chunk allocation
- Maintaining metadata consistency during compute node failures

Evaluation: FineMem reduces remote memory allocation latency by as much as 95%, enables DM systems to achieve low memory waste with minimal overhead.

Available at: https://github.com/ADSLMemoryDisaggregation/FineMem

Thanks for your attention!

For questions, please contact:
Xiaoyang Wang — wxy1999@mail.ustc.edu.cn