MAC: Metadata Acceleration for Sustainable Performance in Big-Data Systems with CXL DRAM

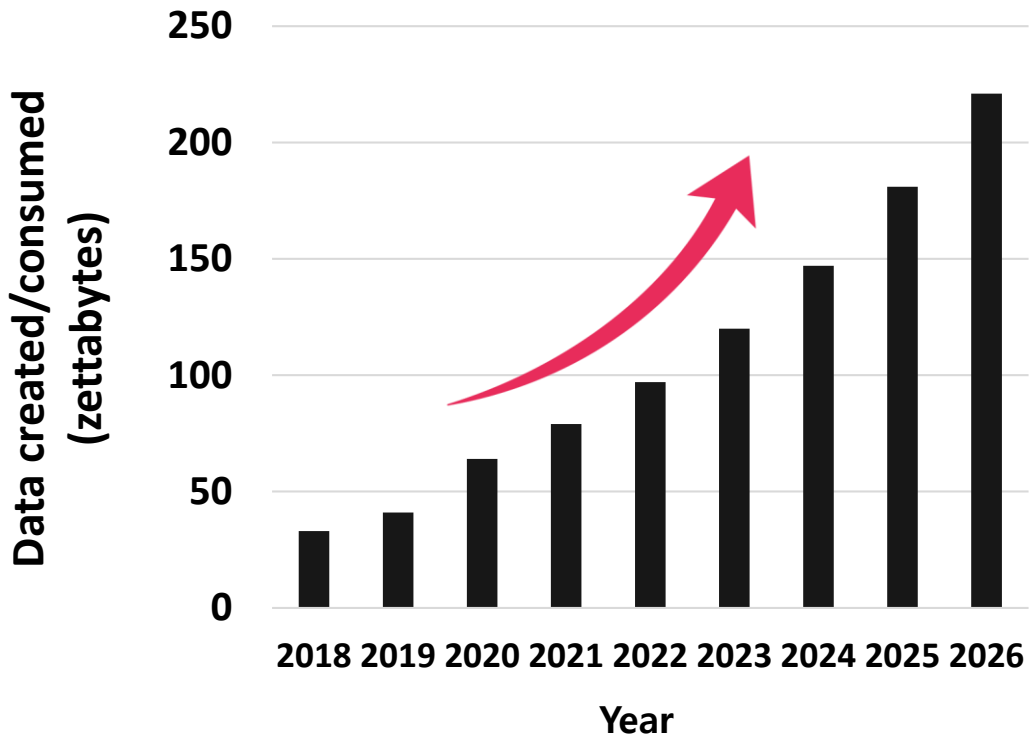
Dusol Lee, †Yan Sun, †Houxiang Ji, †Vinit Gupta, †Austin Antony Cruz, Inhyuk Choi,
†Nam Sung Kim, and Jihong Kim
Seoul National University, †*University of Illinois Urbana-Champaign*





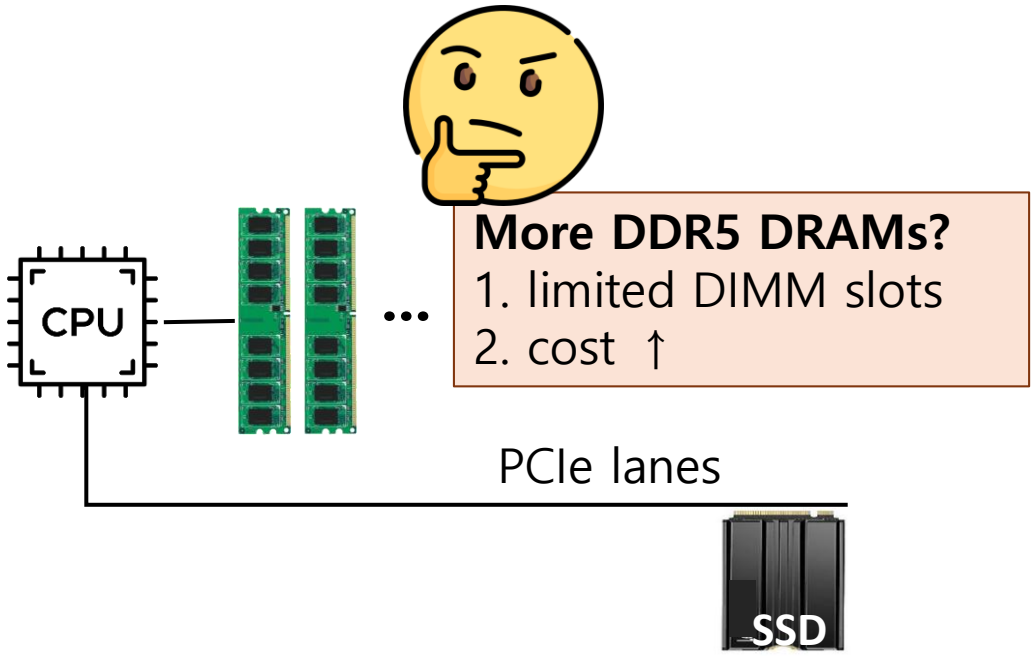
Rising Memory Demands and Memory-Expansion Challenges

- Databases (e.g., SQL, No-SQL, ..)
- Graph processing engines
- …



Big Data Statistics [1]

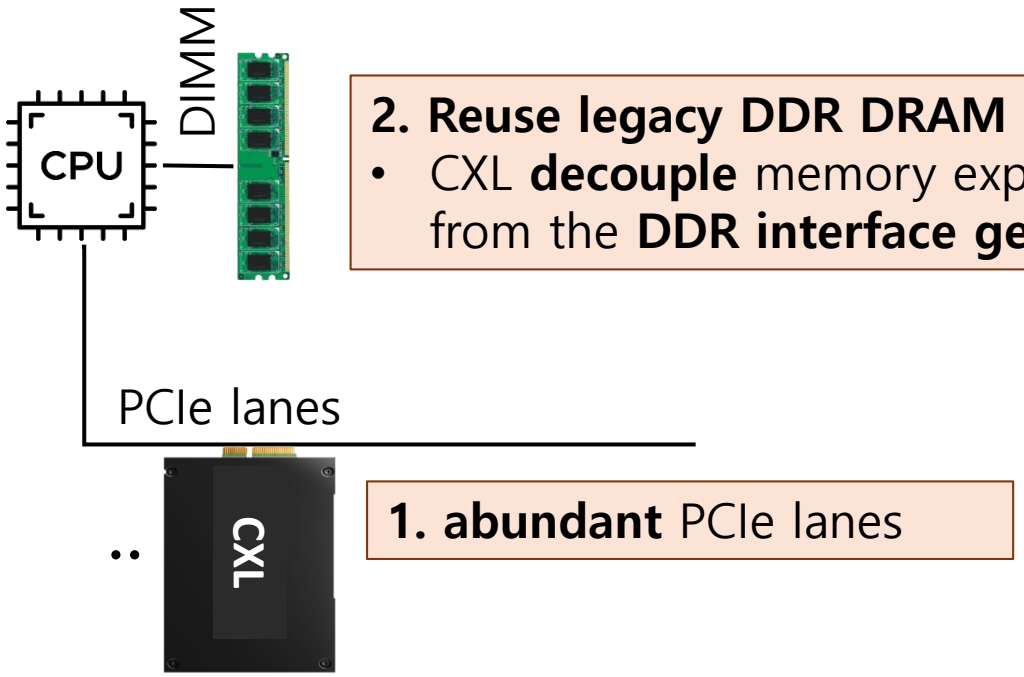
- What if we add more DDR5 DRAM?



[1] Big Data Statistics 2026, Demandsage

Alternative: Memory Expansion with CXL DRAM

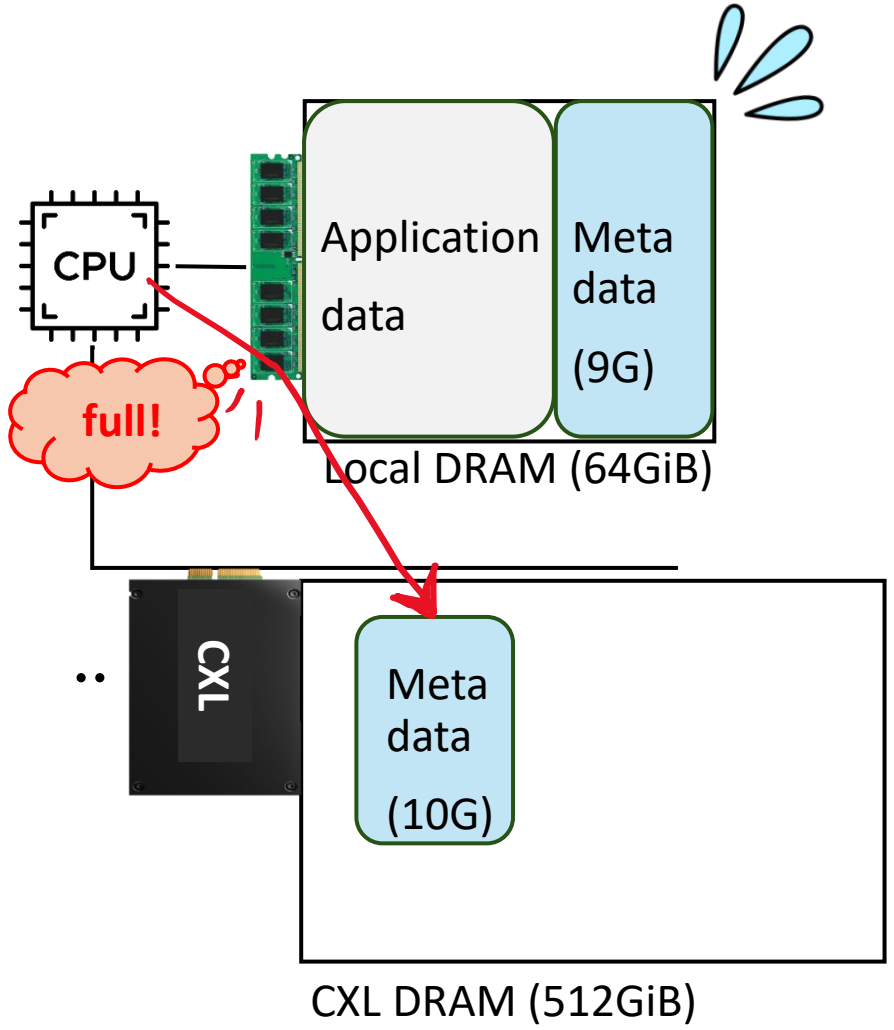
- 1. Enables **2x - 4x** memory **capacity** expansion
- 2. **50% ↓ cost** at DDR DRAM : CXL DRAM = 1:4



2. Reuse legacy DDR DRAM
- CXL **decouple** memory expansion from the **DDR interface generation**

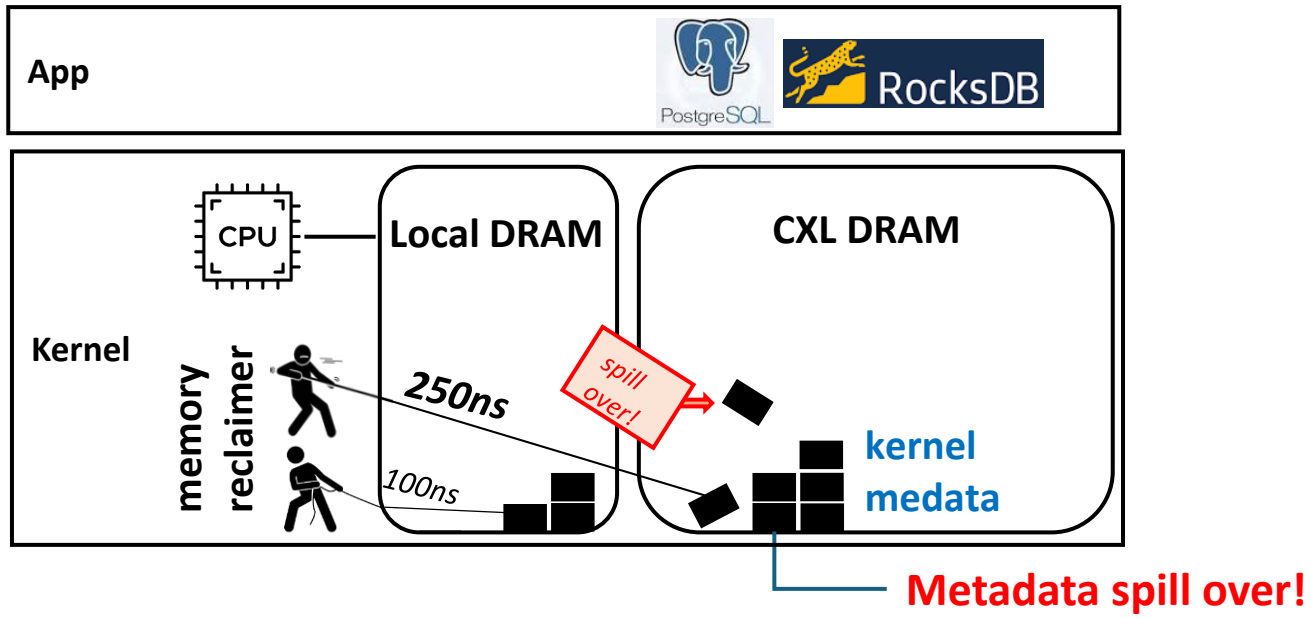
1. abundant PCIe lanes

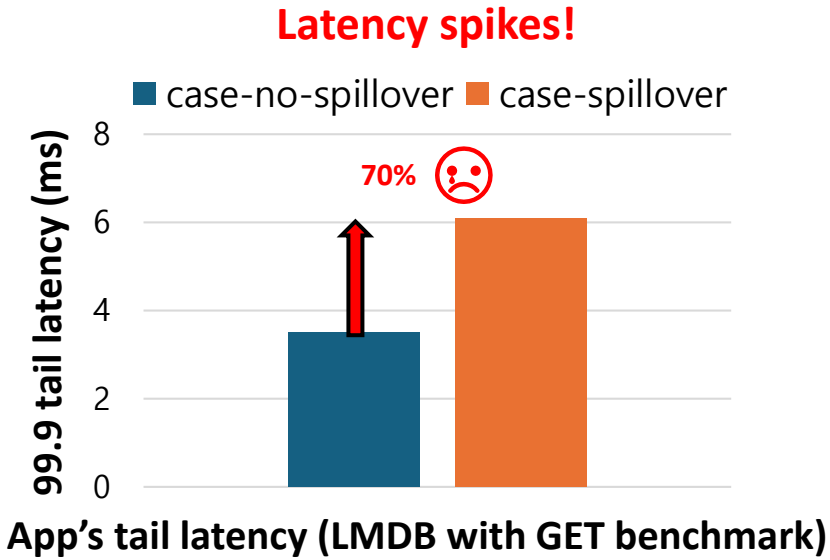
- However, system data (e.g., metadata for memory mgmt.) can **fallback** to CXL DRAM



Observations: Excessive Metadata and Latency Spikes

- Metadata is **too large** to fit entirely in DDR-DRAM
  - As **CXL DRAM** and **file** size ↑, **Metadata** size ↑
    - Up to **20-30%** of DDR DRAM (DDR:CXL = 1:4 or 1:8)
  - **Spill over** to CXL DRAM
- Remote access during memory reclamation → Latency spikes!



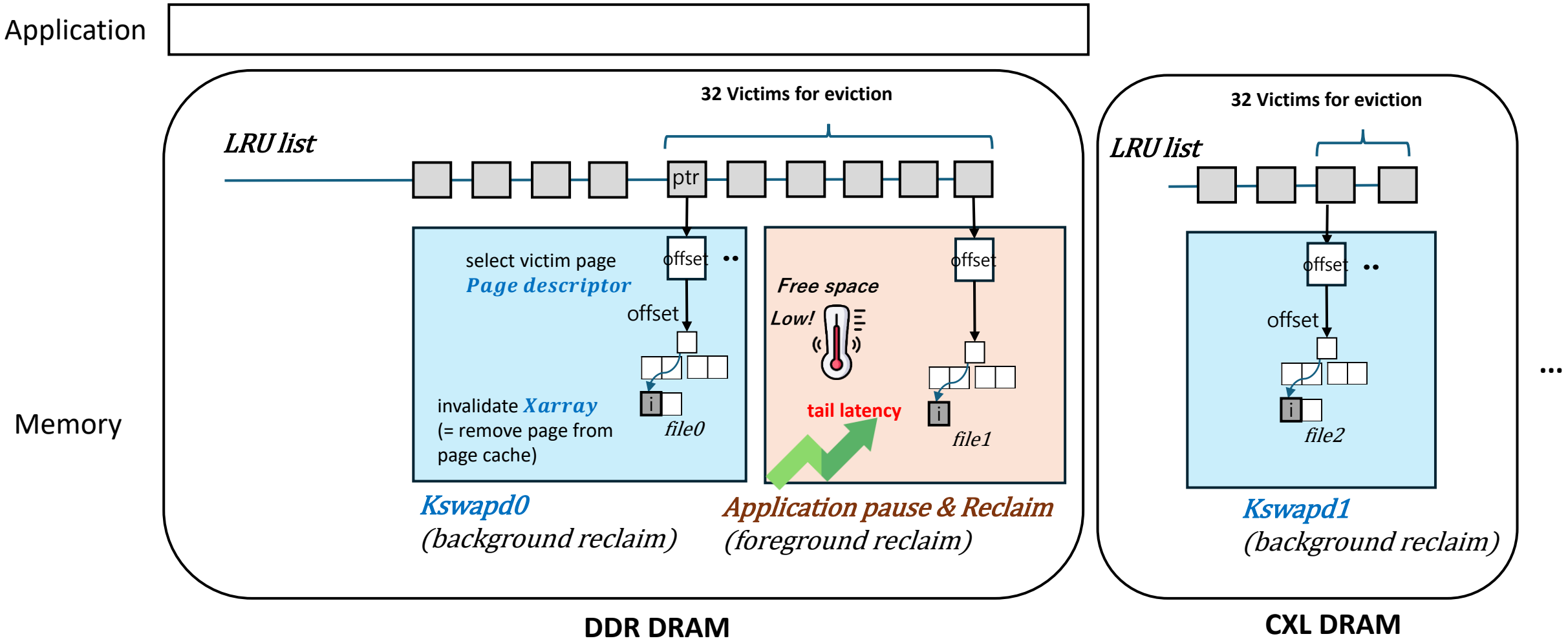


Outline

- Introduction
  - The Rise of CXL DRAM Systems and Their Challenges
- **Background & Motivation:**
  - **Memory reclamation slowdown and latency spikes in CXL DRAM systems**
- MAC Design & Challenges
- Experimental Evaluations

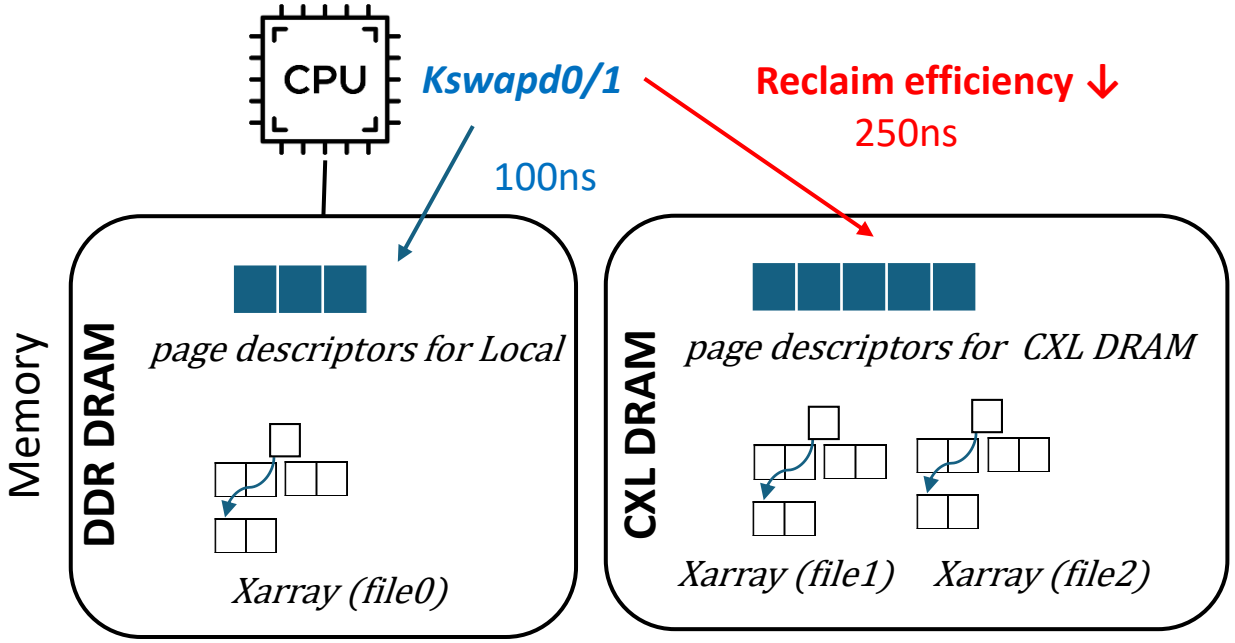
How Does Latency Spikes Occur Under Memory Pressure?

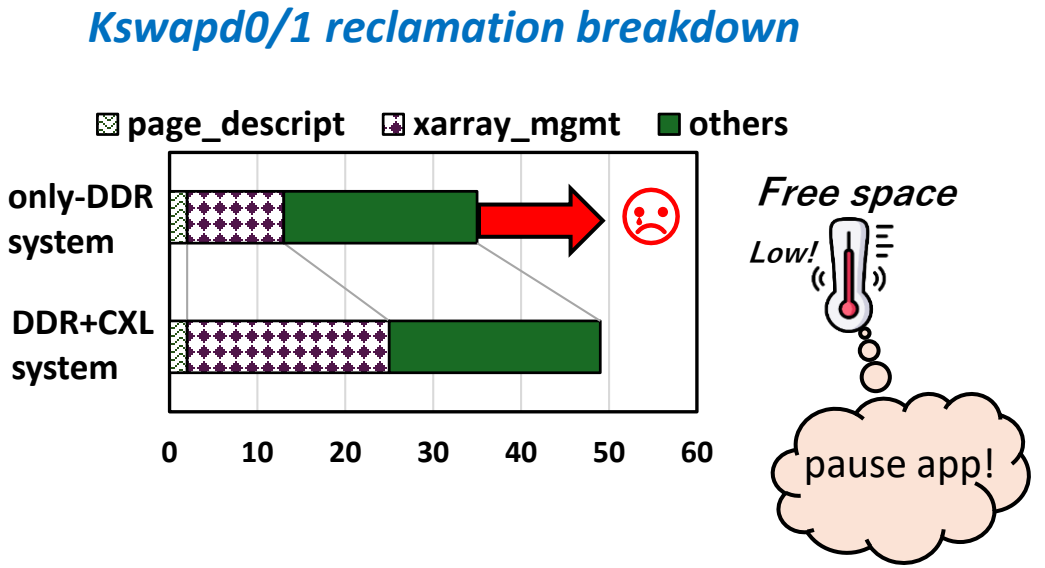
- Background reclaimer (Kswapd) accesses metadata iteratively
  - Free **page descriptor** and invalidate **Xarray** indexing



How Does Latency Spikes Occur Under Memory Pressure? (Cont.)

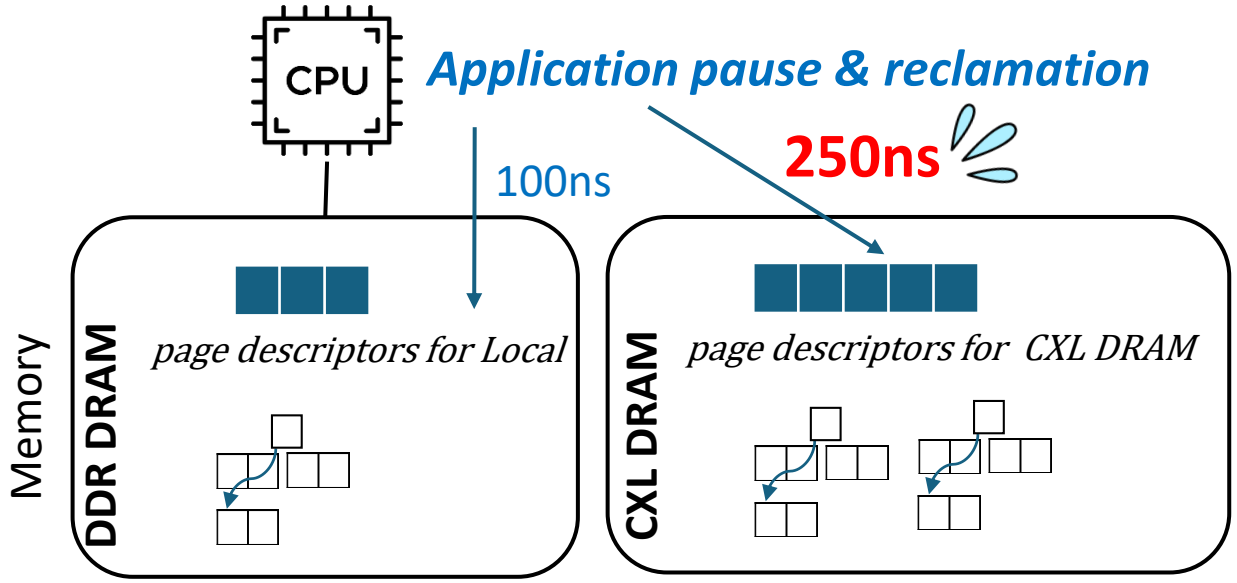
- If Metadata Spills Over..

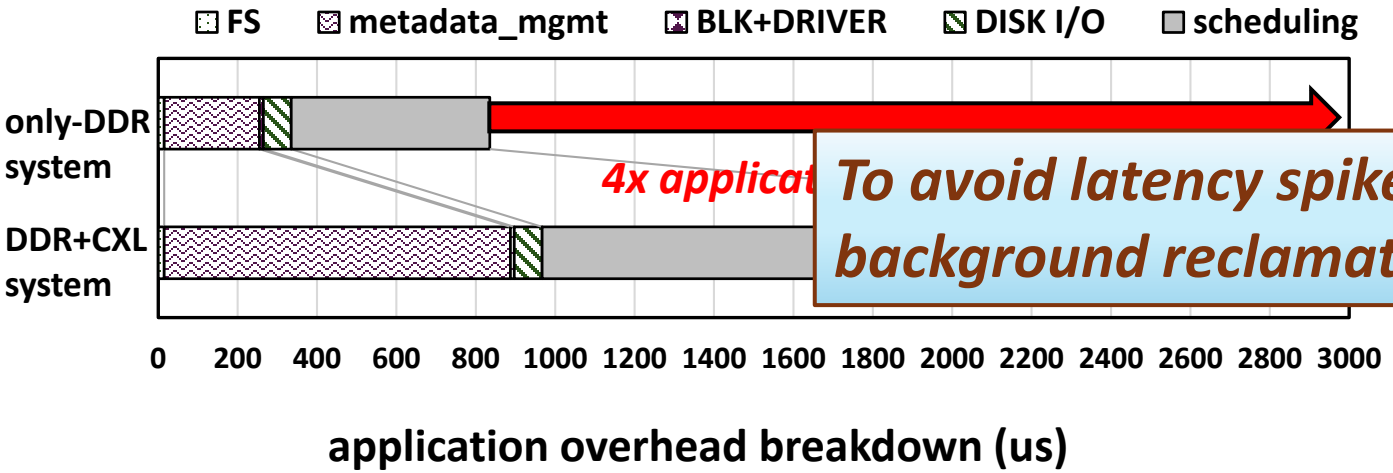


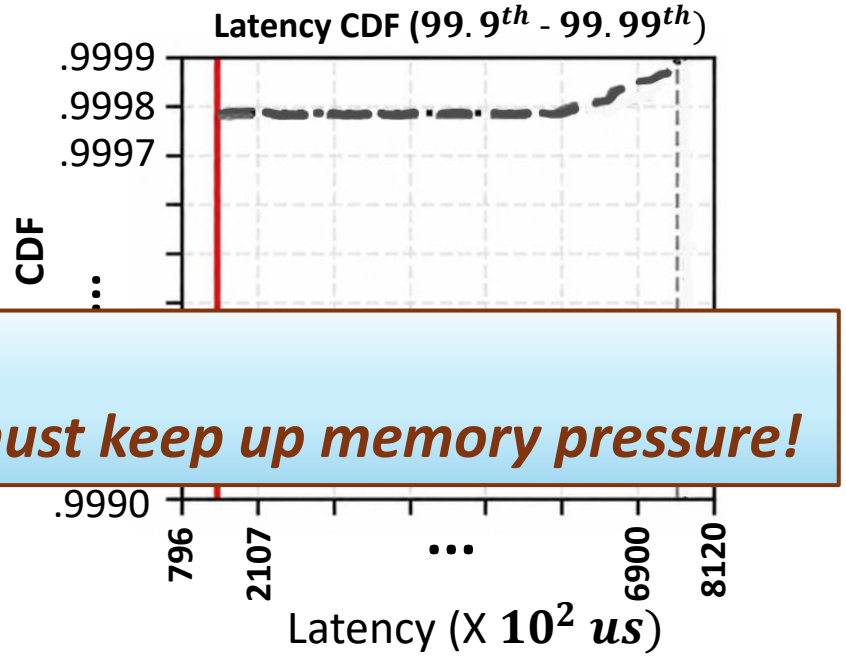


Reduced background reclamation efficiency due to remote memory access
→ Trigger foreground page reclamations

Application Latency Spikes Due to Foreground Reclamation





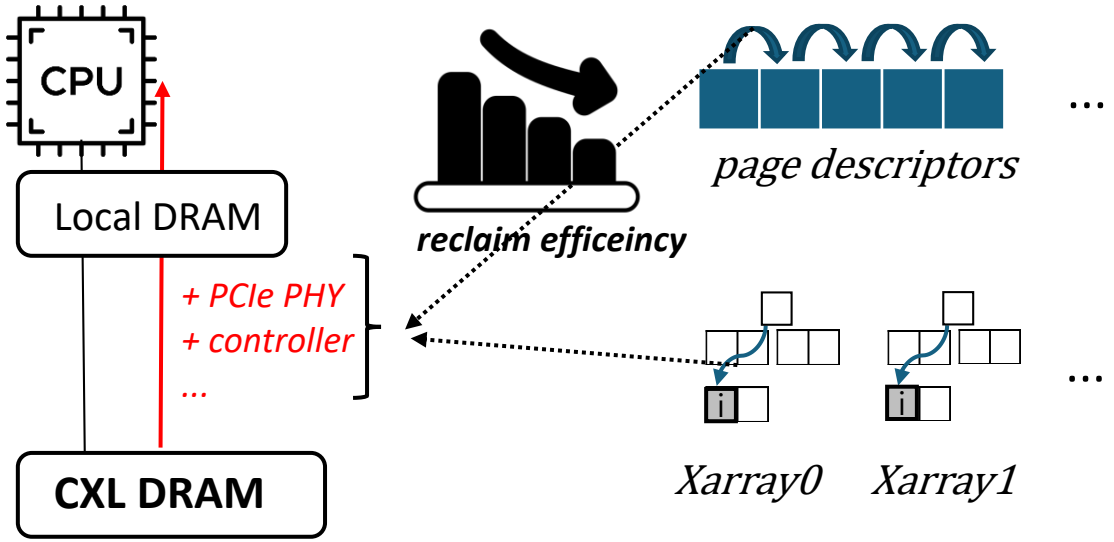


To avoid latency spikes, background reclamation must keep up memory pressure!

Requirement for Fast Memory Reclamation and Opportunities

Key requirement:

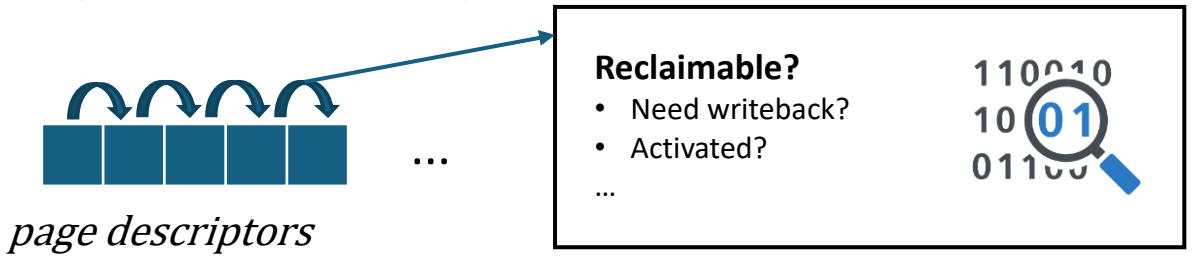
Repeated metadata traversals must be fast.



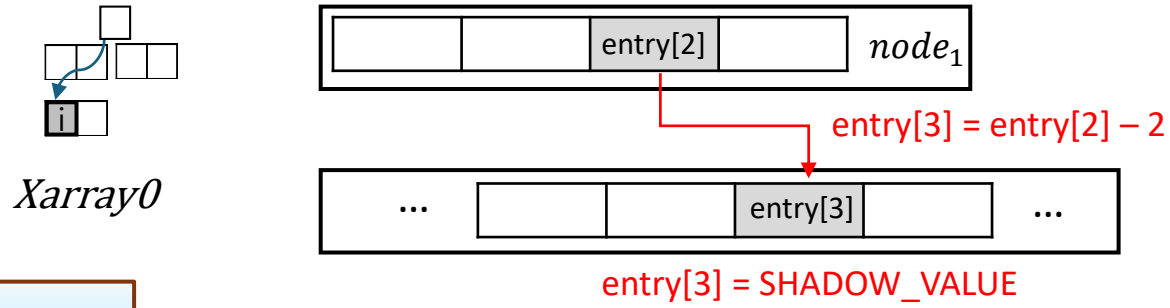
An NMP-based solution is needed on the CXL DRAM side where metadata resides

Observations:

1. Operations are simple



Only shift/bitwise operations on flags



Only arithmetic/store operations

2. Operations on each metadata objects are independent

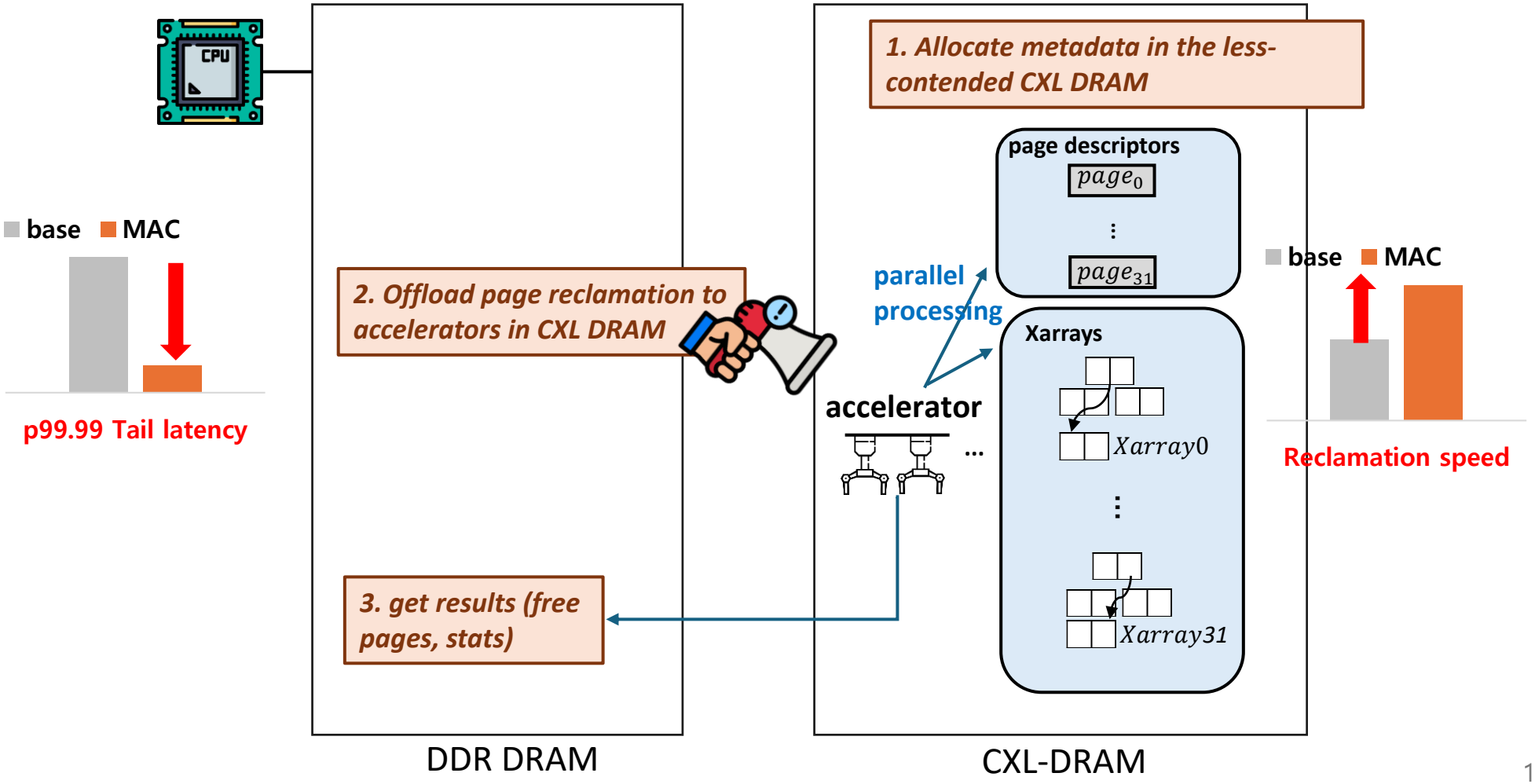
- e.g., Each XArray traversal can be performed in parallel

Outline

- Introduction
  - The Rise of CXL DRAM Systems and Their Challenges
- Background & Motivation:
  - Memory reclamation slowdown and latency spikes in CXL DRAM systems
- **MAC Design & Challenges**
- Experimental Evaluations

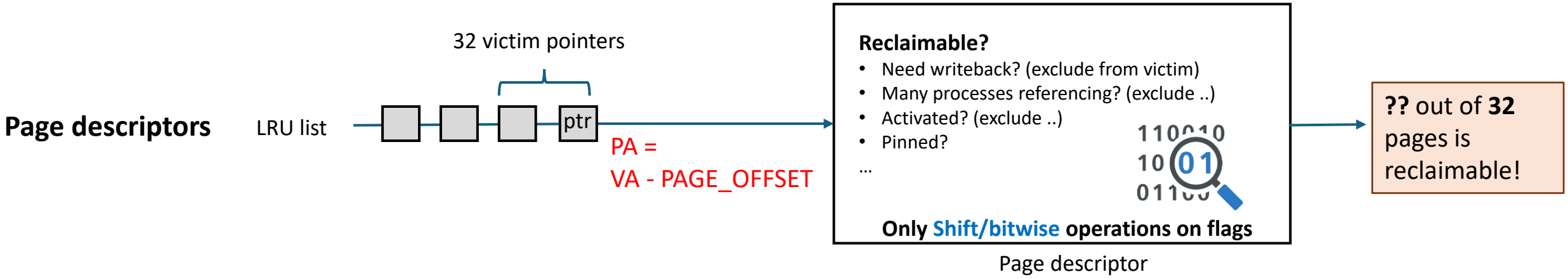
MAC: Metadata Acceleration inside the CXL DRAM

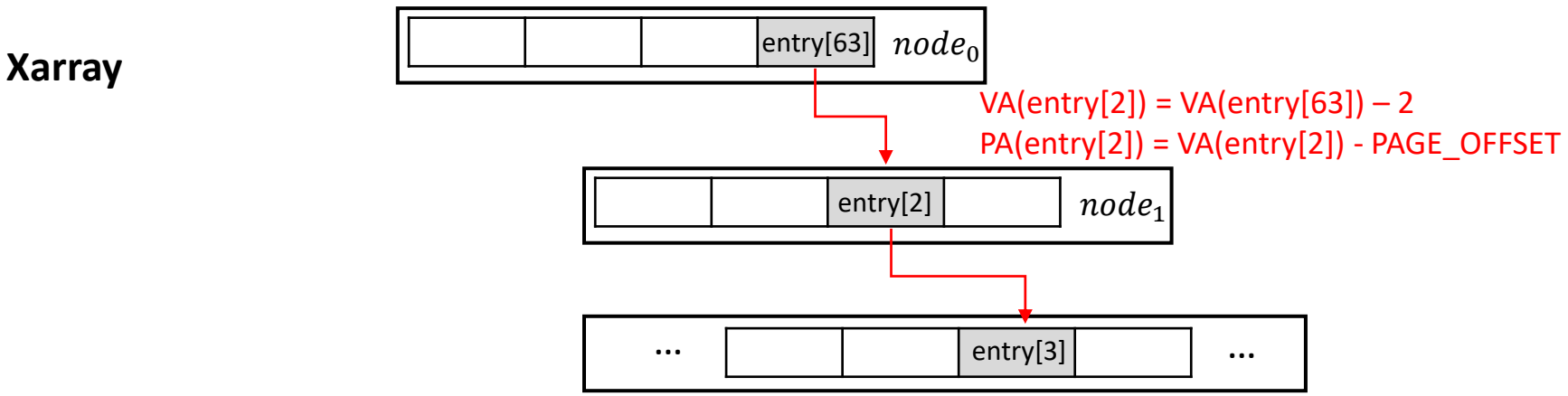
- Place metadata (page descriptor and Xarray) on CXL DRAM *and* offload page reclamation to NMP accelerators.



Challenge1: **How** the accelerator **process** metadata?

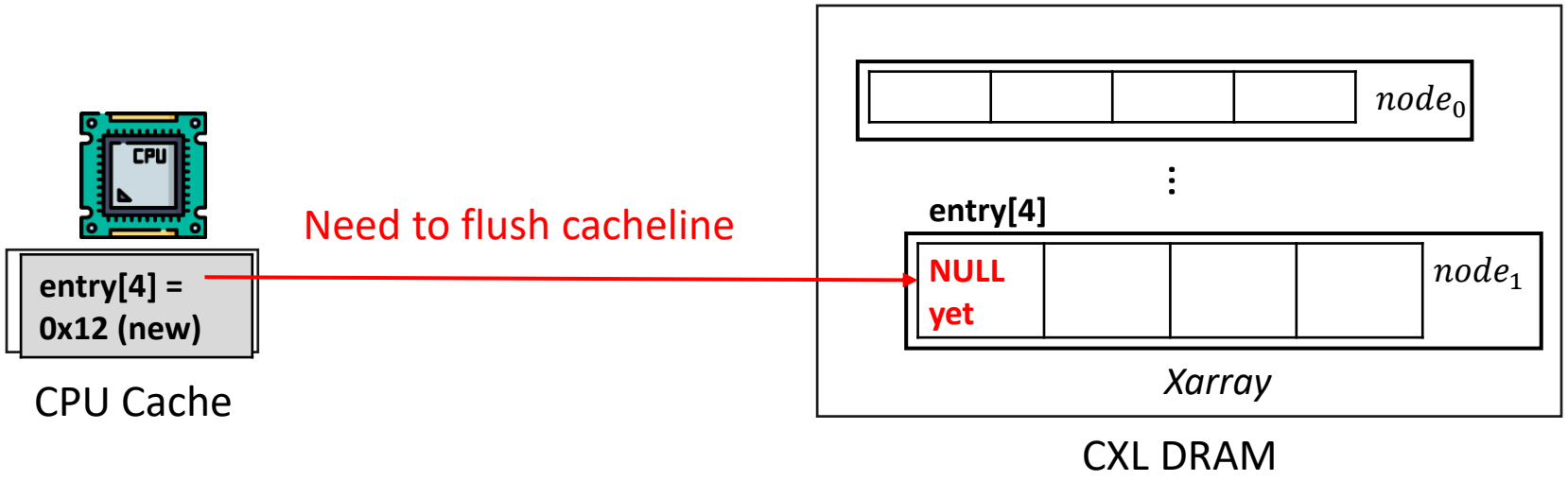
- Only shift/bitwise needed in the kernel address area for **address translation** and **processing**



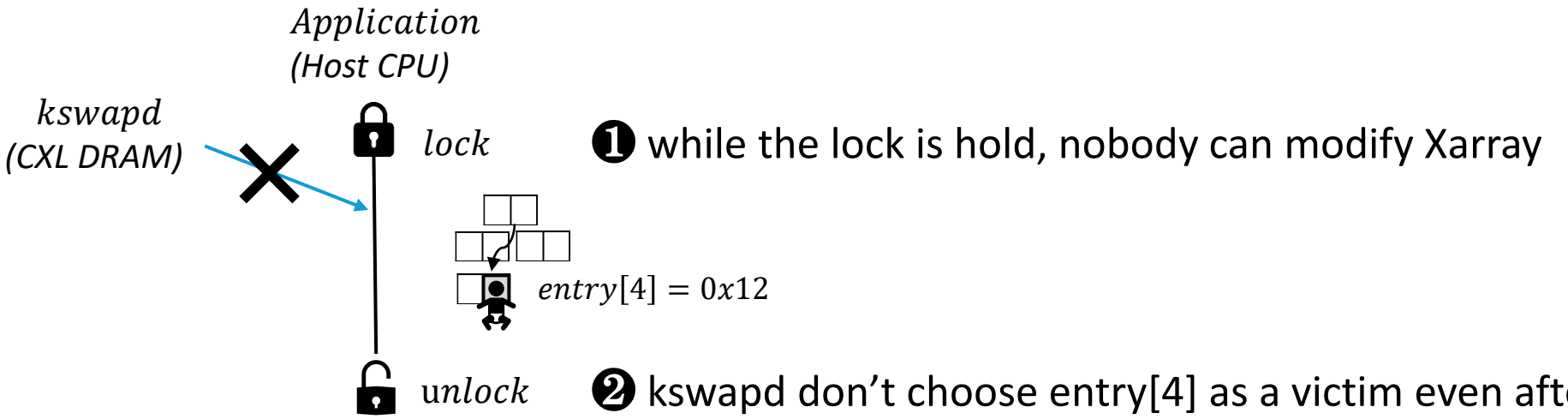


Challenge2: Cache Coherency Overhead

- Case1: Host CPU (application) updated the Xarray
  - When new file page is read by the application, the updated Xarray's entry is only in the CPU cache

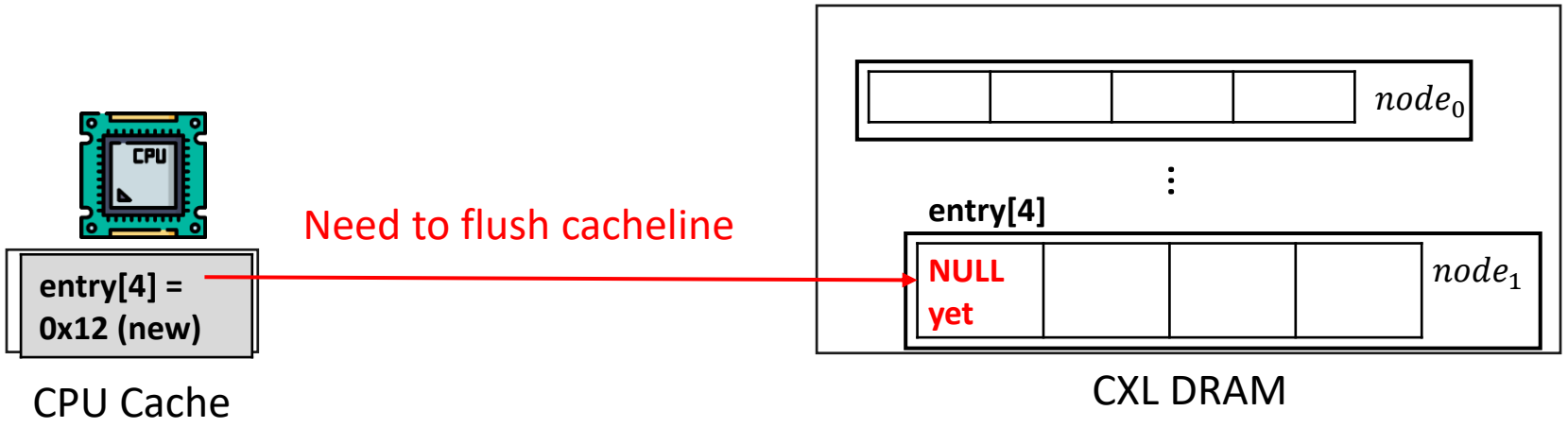


- Observation: No need for immediate synchronization.

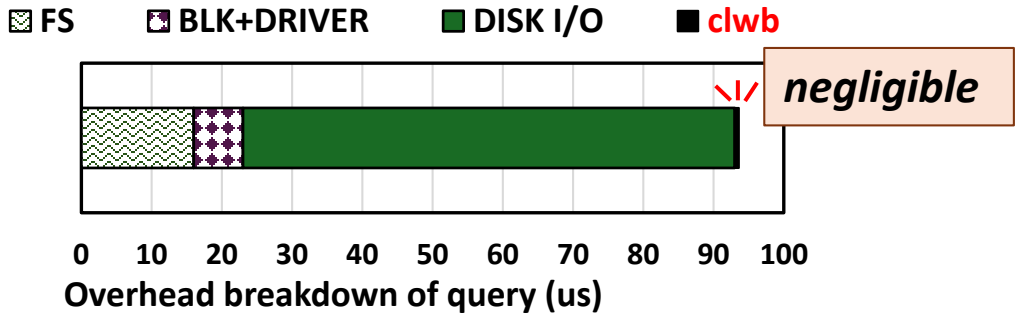


Challenge2: Cache Coherency Overhead

- Case1: Host CPU (application) updated the Xarray
  - When new file page is read by the application, the updated Xarray's entry is only in the CPU cache

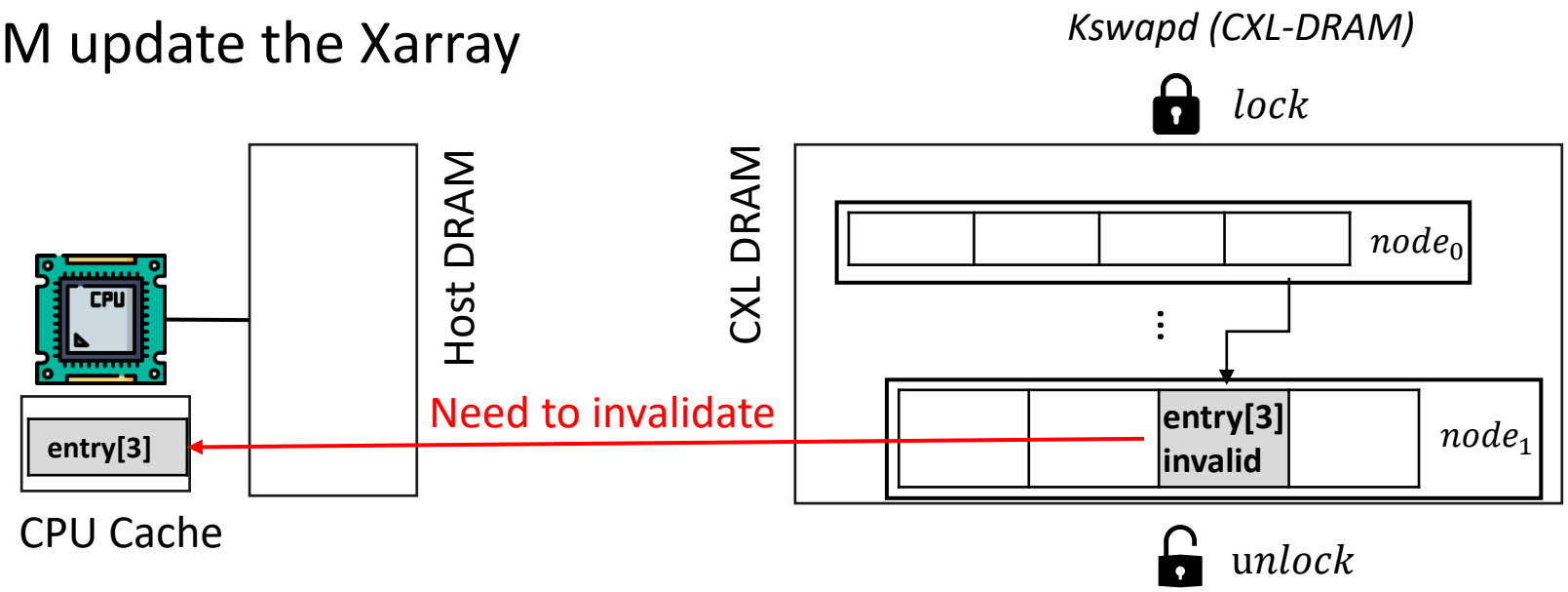


- Solution: Asynchronous **clwb** (cache line writeback)

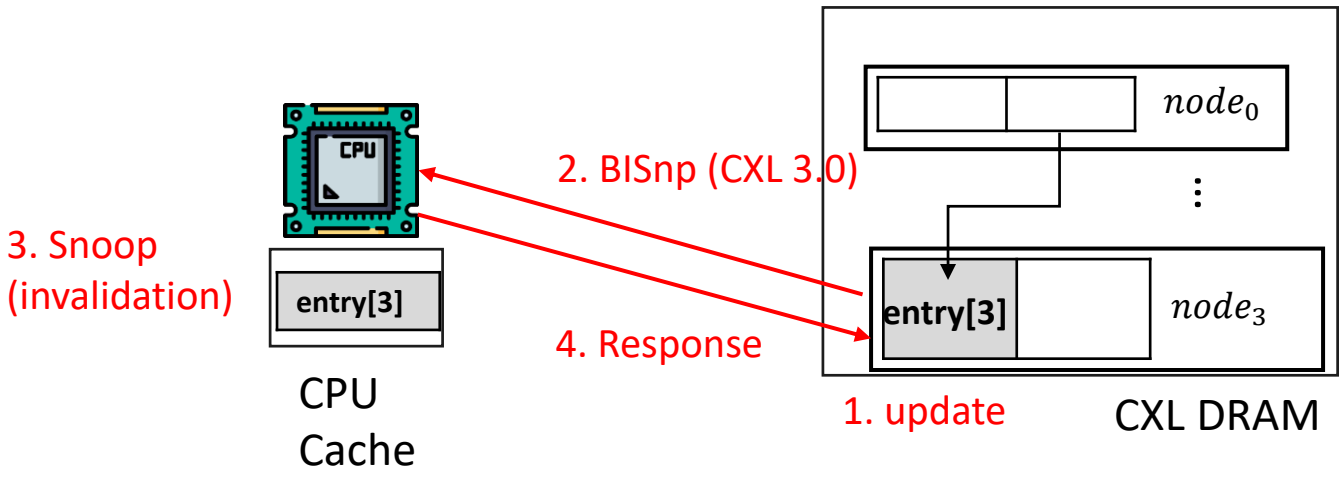


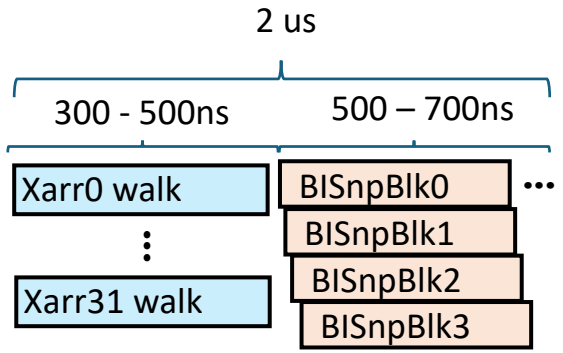
Challenge2: Cache Coherency Overhead (Cont.)

- Case2: CXL-DRAM update the Xarray



- Solution: Utilize BISnp (Back Invalidation Snoop protocol)





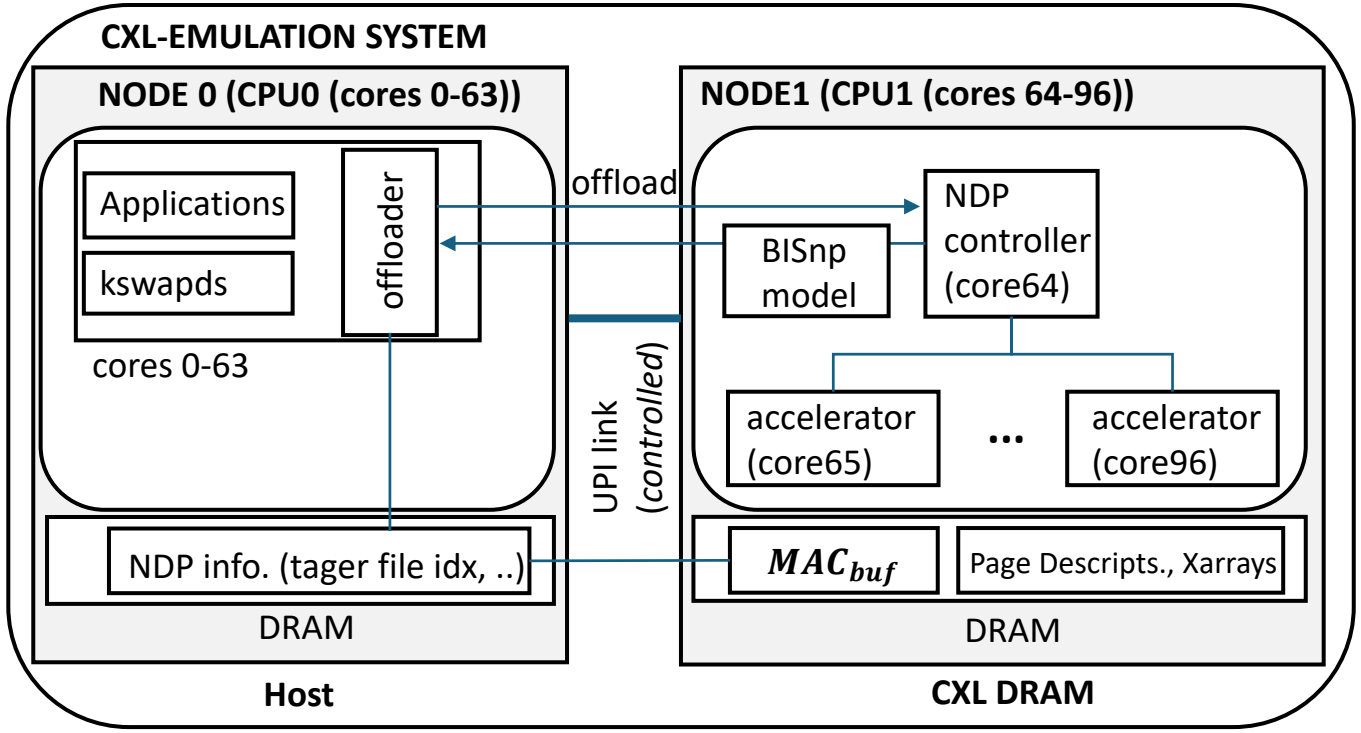
Pipelined BISnp

Outline

- Introduction
  - The Rise of CXL DRAM Systems and Their Challenges
- Background & Motivation:
  - Memory reclamation slowdown and latency spikes in CXL DRAM systems
- MAC Design & Challenges
- **Experimental Evaluations**

Experimental Setup

- **Evaluation Platform**
  - Linux Server (Kernel 6.14)
  - Intel Xeon 6th 128 cores
  - 256 GiB DRAM, 8-TiB NVMe SSD
  - **UPI frequency is controlled to emulate CXL-DRAM**

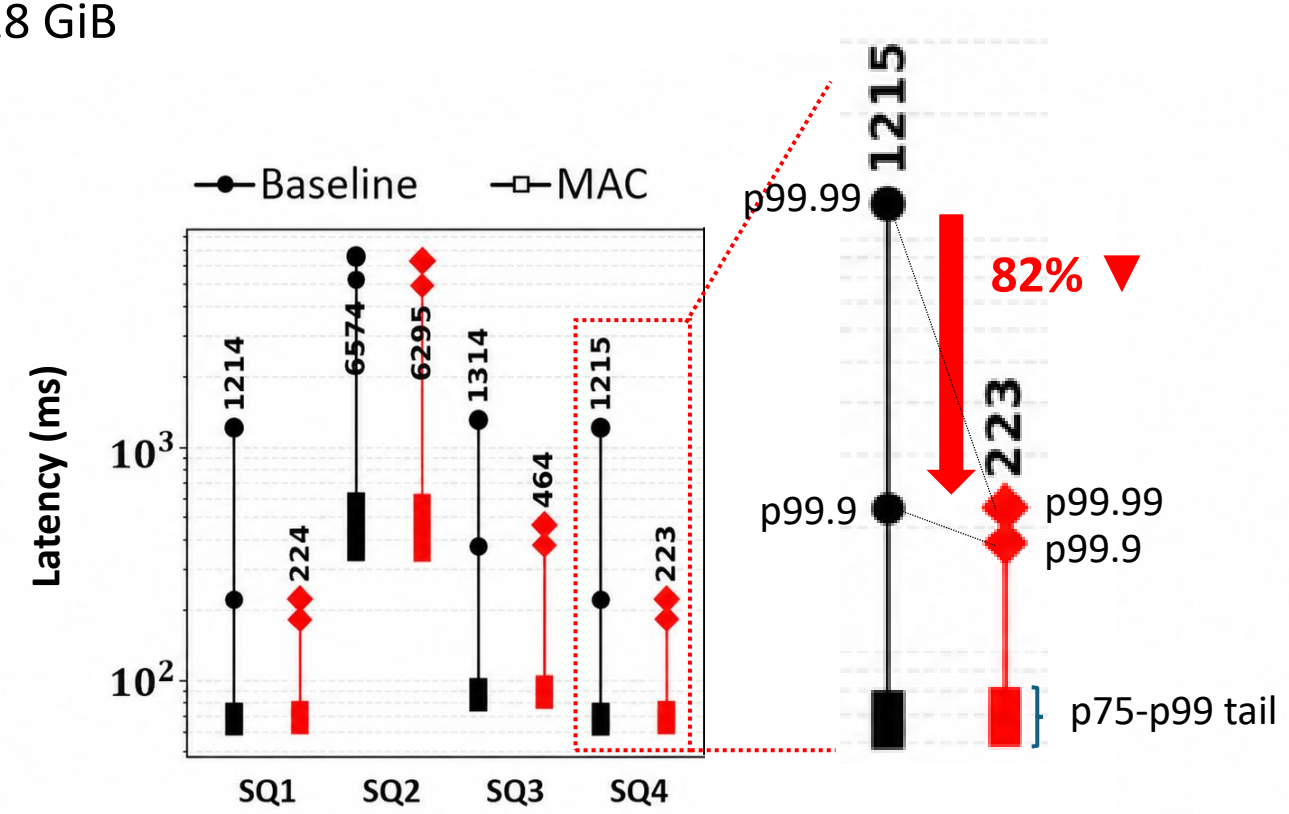


- **Workloads**
  - Databases: PostgreSQL, RocksDB, LMDB (key-value store), and Neo4j
  - Workloads: YCSB (read/write intensive), pgbench, *mmap*-based GET, and Interactive Short/Complex Queries

- **Comparison Schemes**
  - **Baseline**: Default kernel (preferentially allocate metadata on DDR, but spill over under DDR pressure)
  - **MAC**: +NMP-based page reclamation

Experiment 1: Evaluating Tail-Latency Reduction for Database Application

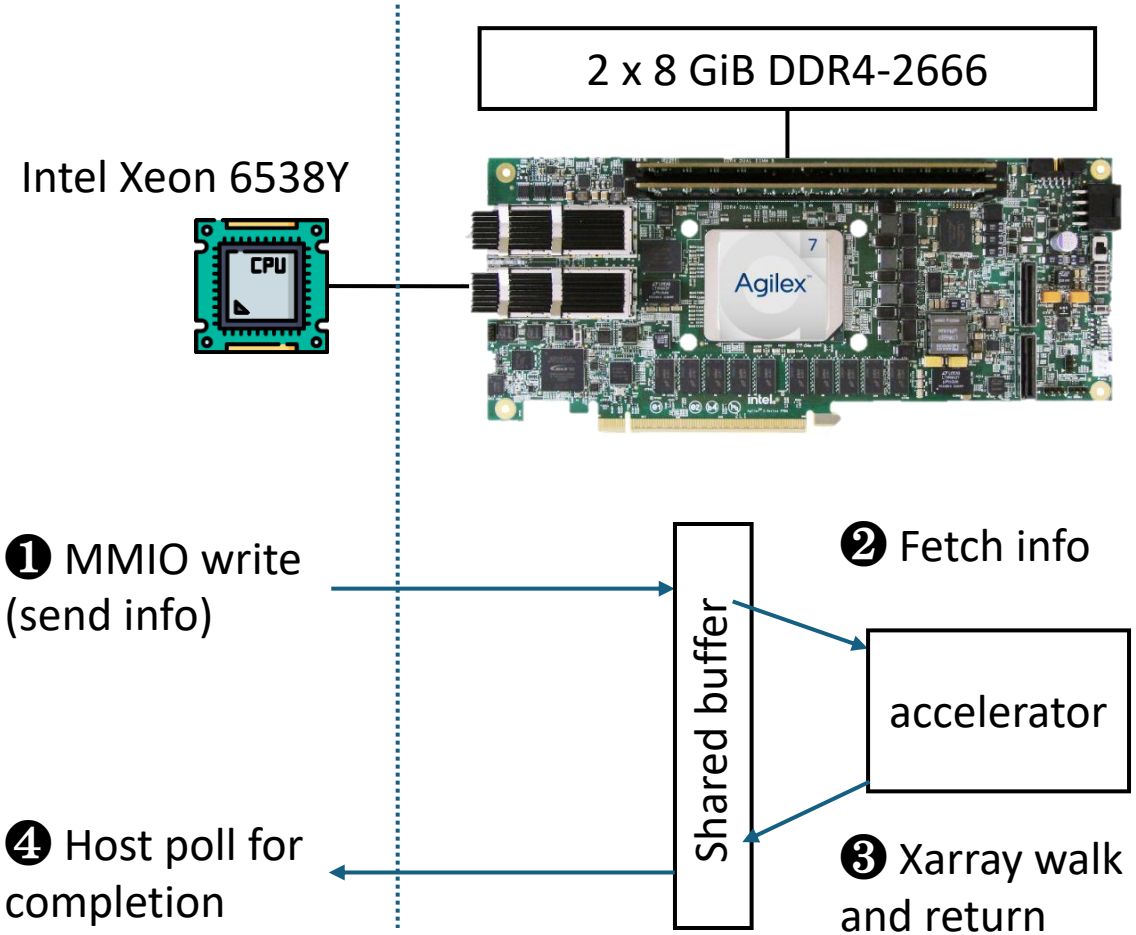
- **Interactive Short (IS)** workload on **Neo4j**
  - **Workload:** 400K ops with 200 threads
  - DDR : CXL = 64 GiB : 128 GiB

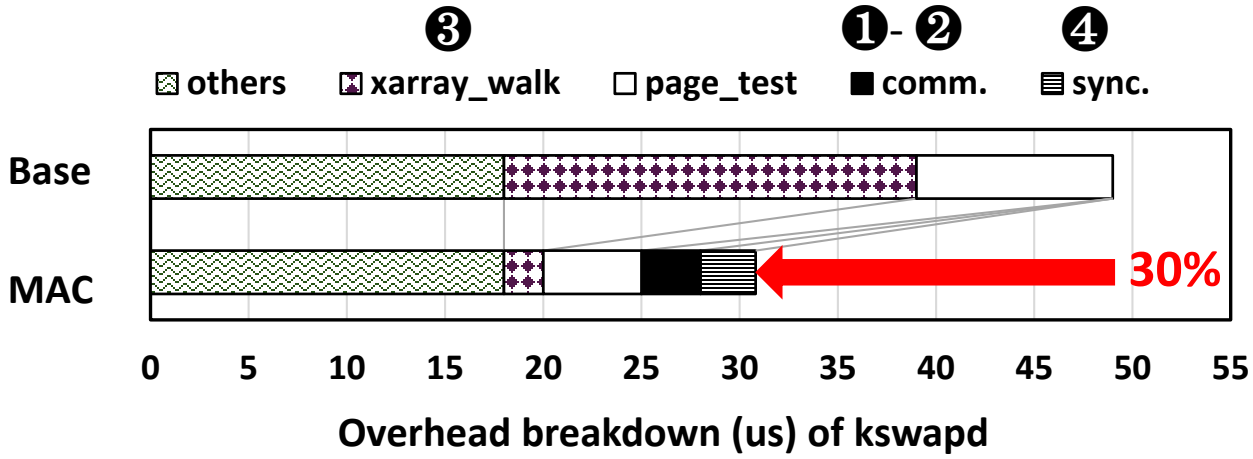


Parallel NMP inside CXL-DRAM reduces tail latency by 82%

Experiment 2: Real Overhead Measurement Using FPGA

- Intel Agilex 7 I-series FPGA





Conclusions

- **Introduced the metadata overhead problem in the CXL DRAM systems.**
- **Proposed MAC, a HW/SW co-designed near-memory metadata acceleration architecture.**
  - Substantially reduced latency spikes.
    - NMP-based parallel metadata processing (▼ number of foreground reclamations)
    - Light-weight consistency optimizations (▼ consistency overhead)
- **Evaluated on a two-NUMA-node emulation platform and an FPGA prototype.**
  - Reduced the p99.99 latency by up to 98% across four data-intensive applications.
  - Validated a 30% reduction in page reclamation overhead using an FPGA prototype.

Thank You

Q/A