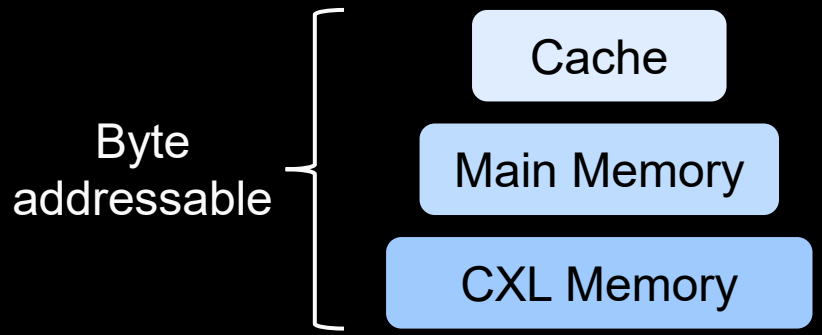
Hot Data Detection for CXL Memory

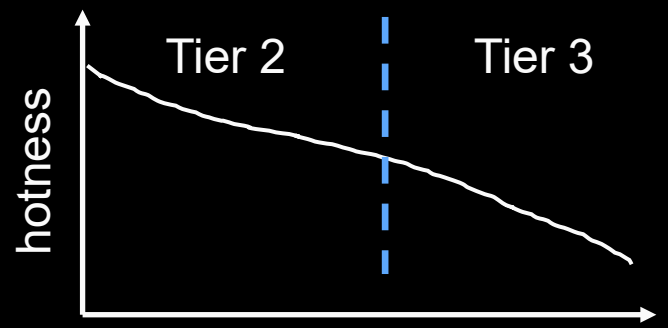
Jørgen Sværke Hansen, Sr. Technologist, Western Digital Research

August 2024

Increasing Capacity with CXL Tiered Memory

- Single memory range with separate tiers where lower tiers are less performant
- Data temperature should reflect tier placement
  - Identify application working set - track application memory access to classify data temperature
  - Kernel-level heuristics for adapting data layout to application behavior
- APIs to allow applications to control data layout
  - libnuma, Scalable Memory SDK by Samsung, Heterogenous Memory SDK by SK Hynix





OS Level Data Placement

Current state in the Linux kernel

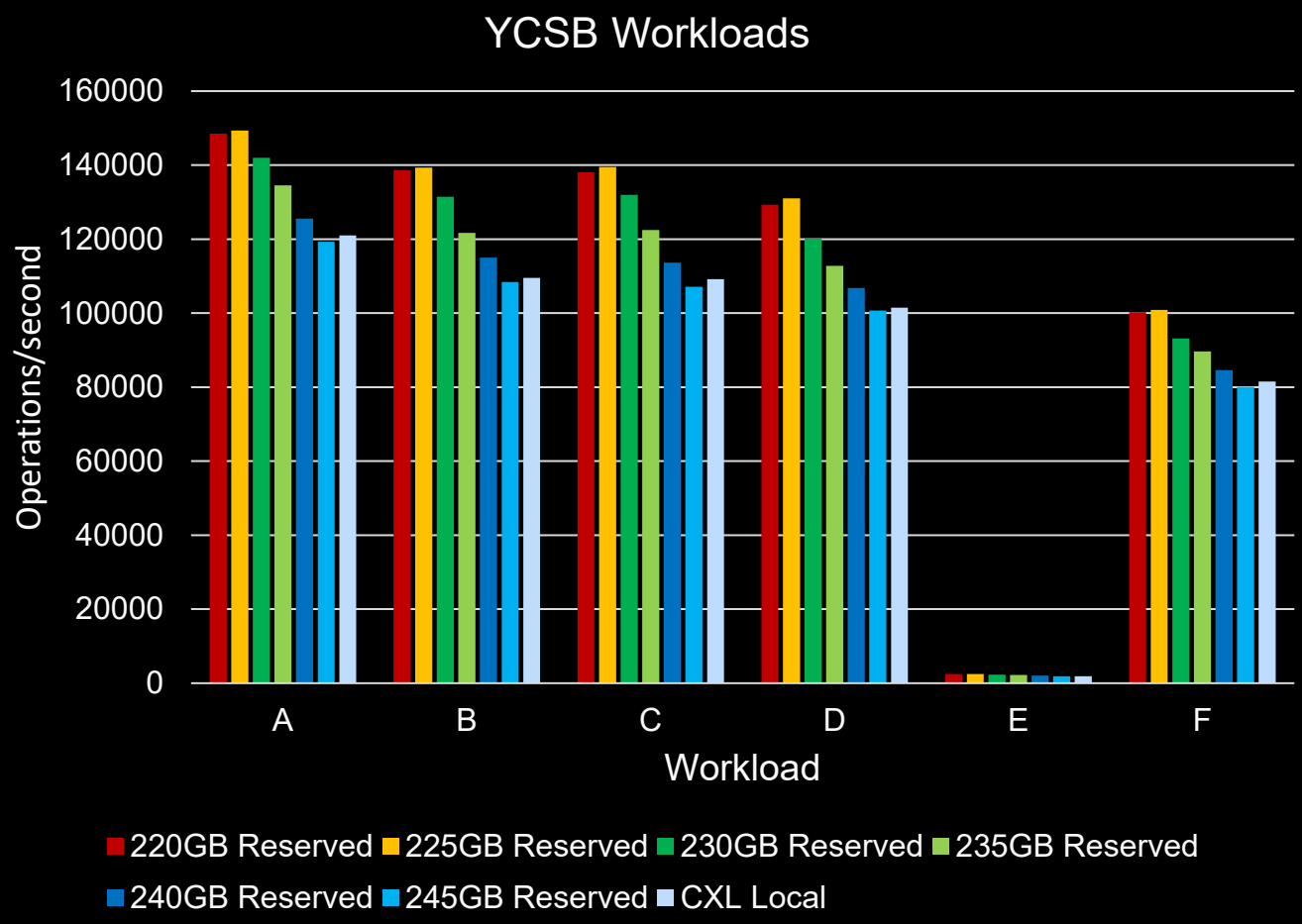
- NUMA Balancing
  - Keep data close to node where thread is scheduled

    - or -
  - Schedule thread where data is located.
  - Fault-based detection of access
  - LRU-based demotion of pages (optional)

- Tiered Memory Balancing
  - Migrate hot pages to faster tier
  - Estimate hotness of page based on page fault frequency
  - Move cold memory to slower tiers using LRU-based demotion of pages
  - Based on the work on TPP - *["TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory" in Proceedings of ASPLOS 2023 by Hasan Al Maruf et al.]*

Getting a Sense of Tiered Memory Balancing

- YCSB workloads using Redis
  - Server and client on the same node
  - 128GB Local DRAM (latency 115 ns) and 128GB Local CXL memory (latency 245 ns)
- Control CXL memory usage by adjusting kernel memory reservation
  - Workload approximately 23GB memory footprint
  - Slowdown between 4%-6% for every 5GB local DRAM removed – no indication of flattening curve.
  - Large potential for improvements with more precise hotness monitoring
- Hotness-based page promotion increases transaction rates by approx. 2% compared to regular NUMA (235GB reserved)

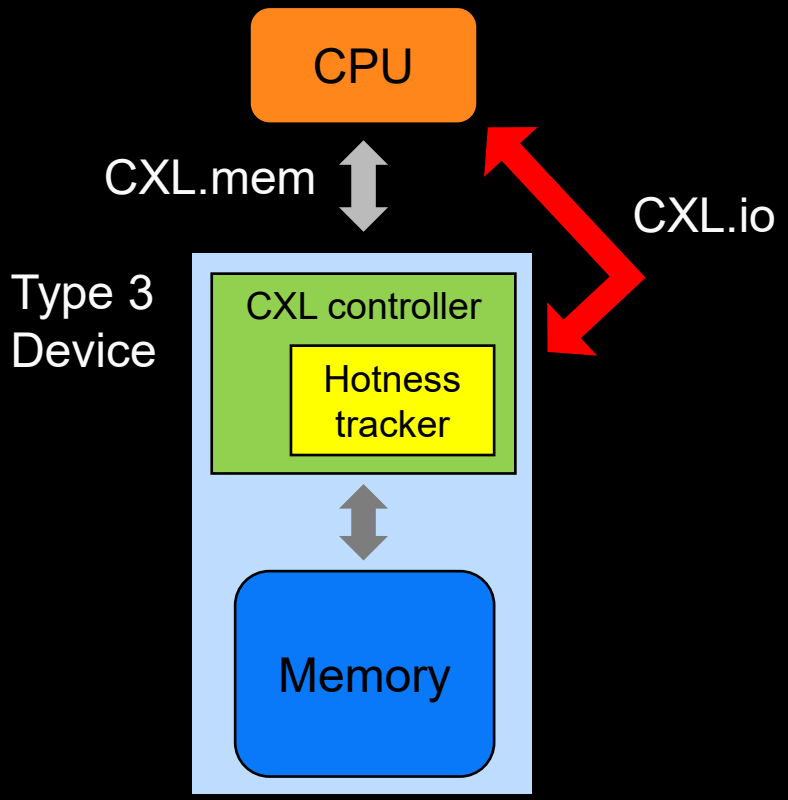


What if Memory Could Speak?

- Sampling on the processor side isn't perfect:
  - Page-fault based tracking has high overhead
  - Performance counter based (PEBS/IBS) tracking isn't tracking all memory accesses:
    - Caching traffic (read-ahead/writeback) not accounted for
- Distribute the responsibility of tracking memory temperature to the memory devices
  - The memory device is in the path of every relevant memory access
- Composable Memory System (CMS) subgroup of the Open Compute Project (OCP) proposes a hotness tracking extension to CXL:
  - Whitepaper available at: https://www.opencompute.org/wiki/Server/CMS#Whitepapers

CMS/OCP Proposal

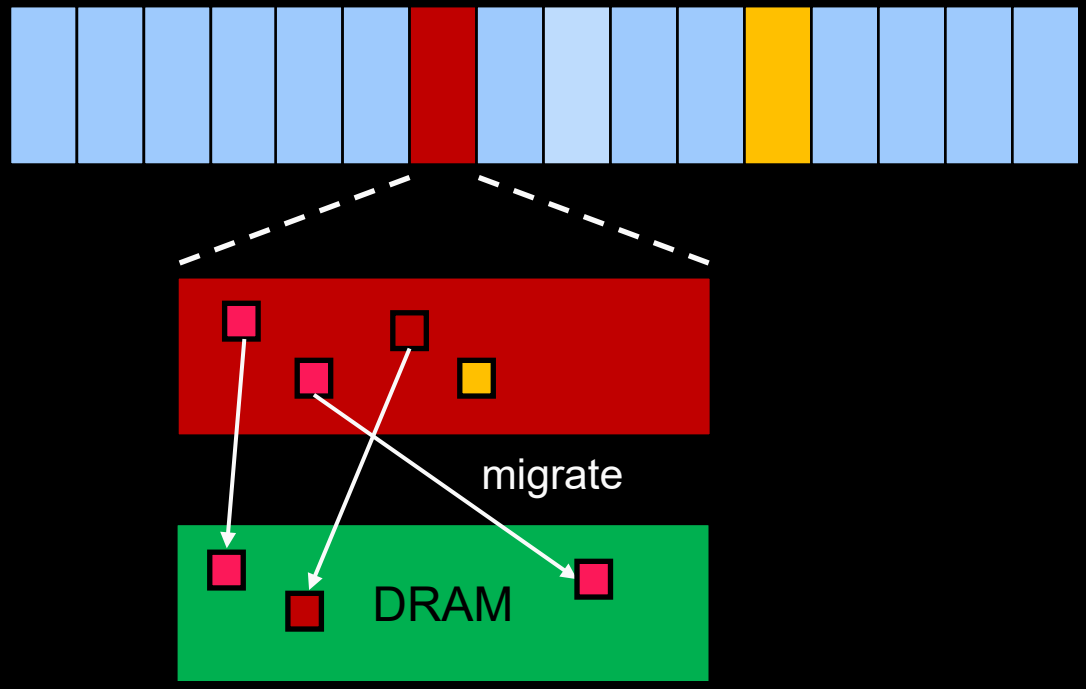
- Extend CXL Memory with a hotness tracking widget
  - Monitoring of the memory configured by the host side
  - Hot pages communicated to the host
- Proposal focuses on the host-device interaction:
  - Doesn’t provide complete specification of interaction
  - Device implementation left to hardware vendors



Flow of Operations

Monitoring done through four steps:

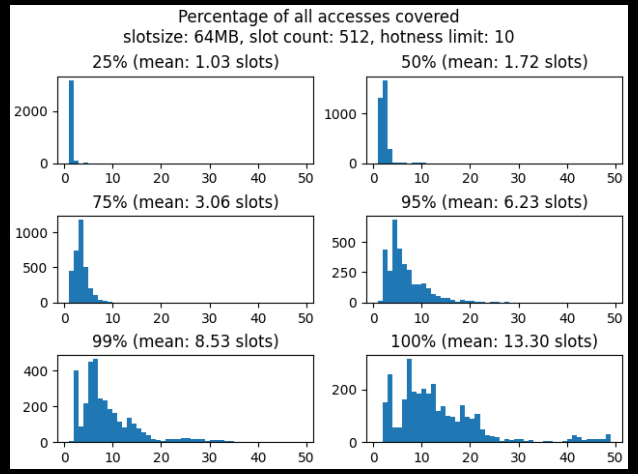
1. Coarse-grained monitoring to identify areas of high activity
2. Fine-grained monitoring within these areas identifies a pool of hot pages
3. The host side uses this information to migrate one or more pages
4. Migrated pages removed from the pool of hot pages

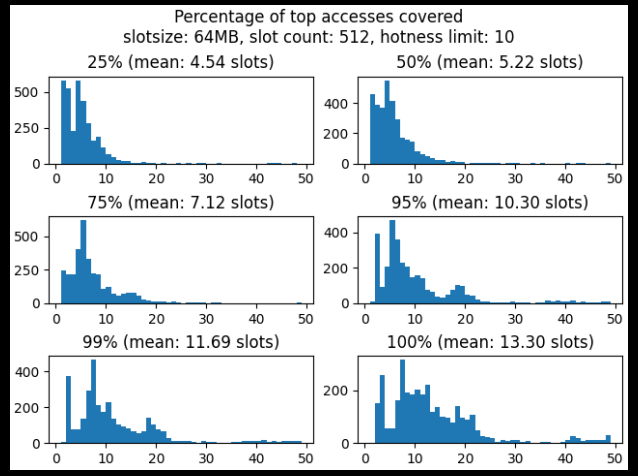


How Many Active Regions are Enough?

Trace-based Analysis of cachebench Workload using QEMU

- If detecting the pages with most accesses, the minimal number of page promotions can be used to achieve a given access coverage
- As accesses may be spread over semi-contiguous page ranges, the same access coverage can be achieved using fewer slots at the expense of having more migrations
- Indicating that several independent monitoring ranges necessary to identify majority of hot pages
- Consider having a tree of ranges, where continuous coarse-grained monitoring of complete memory, allows for selecting hot regions and drill down on those





Summary

- Identifying hot pages is critical for good performance when CXL memory is used as a slower tier.
- Region-based tracking of hot pages on the CXL memory device is likely to provide good coverage with limited resources
- However, there are other promising approaches as well
  - Intel® Flat Memory Mode provides hardware-managed tiering support at cacheline granularity - *[”Managing Memory Tiers with CXL in Virtualized Environments” in Proceedings of OSDI 24 by Yuhong Zhong et al.]*
  - Instead of using regions, others have shown that hash-based techniques can be used to identify hot pages on the device side - *[”Toward CXL-Native Memory Tiering via Device-Side Profiling” by Zhe Zhou et al. (https://doi.org/10.48550/arXiv.2403.18702)]*


TM