Understanding memory usage in datacenters and Enabling software for CXL-Memory

EMPOWERING OPEN.

OCP GLOBAL SUMMIT | OCTOBER 18-20, 2022 SAN JOSE, CA

ENDLESS MEMORY CXL™ FORUM at OCP Summit

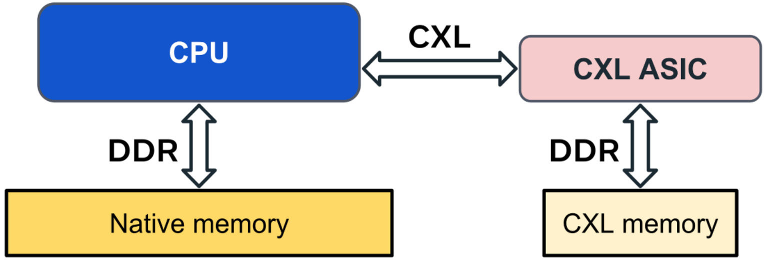
Understanding memory usage in data centers and Enabling software for CXL-Memory

Abhishek Dhanotia, Performance Engineer, Meta

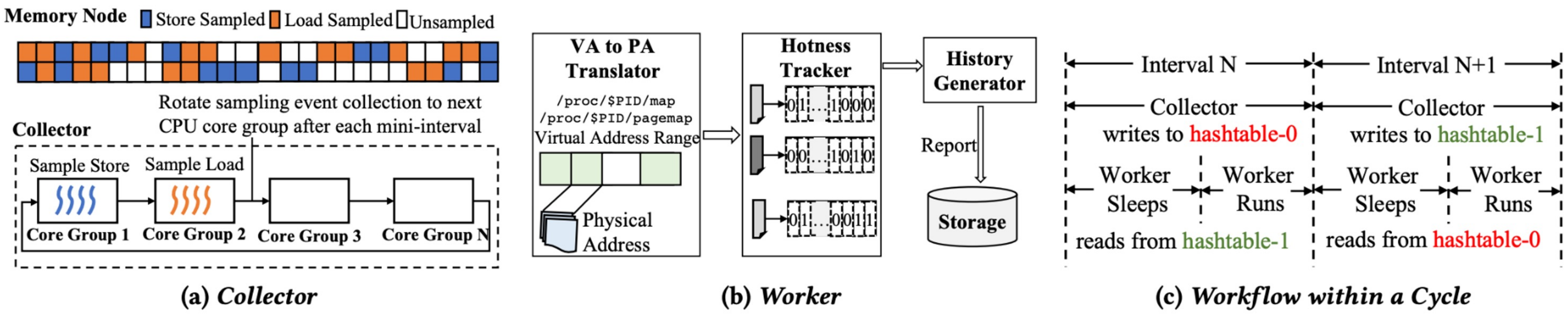
Hao Wang, Performance Engineer, Meta

CXL-Memory: How to get software production-ready?

- CXL enables much better latency vs. previous technologies
  - but still, extra latency makes CPU run slower.
- To avoid performance regression, ideally we want
  - “hot” memory pages in native memory
  - “cold” memory pages in CXL memory
- To guide the design of such a smart page placement mechanism, we built a profiling tool to understand the memory usage across our server fleet.



Understand memory usage - Chameleon tool for datacenter workloads

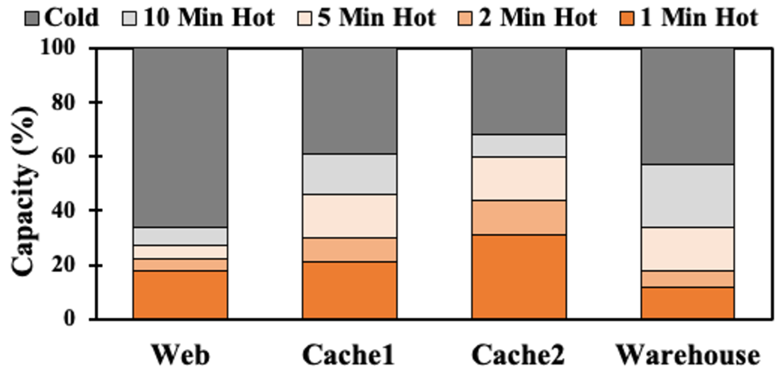


Too complicated. We wish there is better hardware support for this.

- physical address precise event sampling
- more hardware buffers for better sampling rate
- support for store side precise events

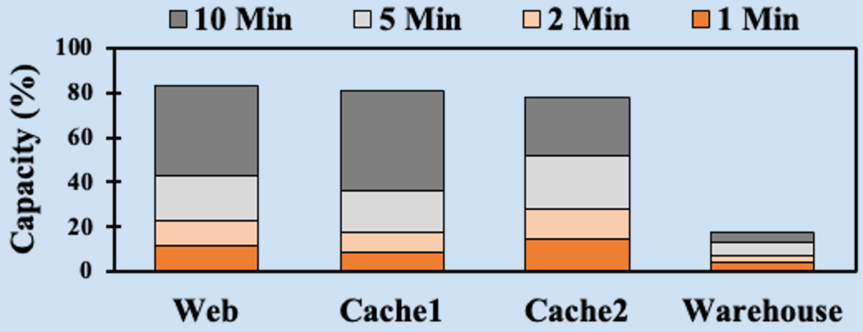
Insight 1: Opportunity for memory tiering

- Meaningful portion of application memory is not accessed frequently, i.e. “cold”.
- Tiered memory could help!
  - tier-1: native DDR memory
  - tier-2: CXL memory w/ flexibility & $$$ savings

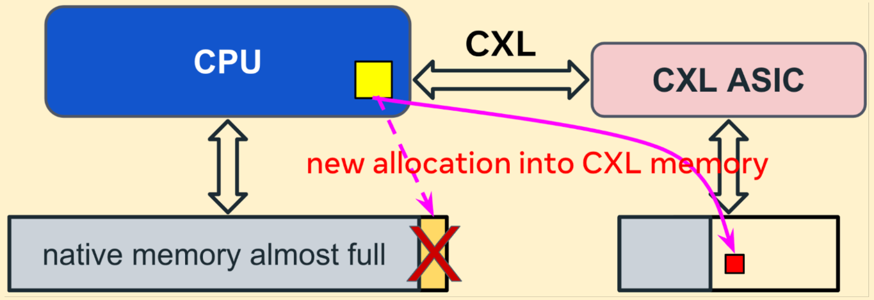


Insights 2 & 3: How to get hot pages in native memory

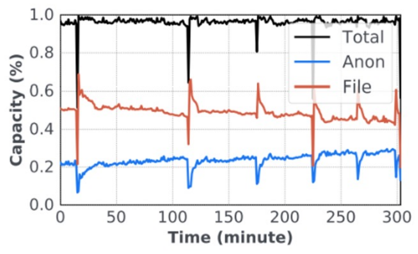
- Code page re-access time varies for workloads.
- Page movement support is needed to avoid perf degradation.



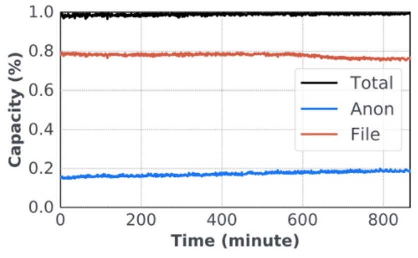
- Long-lived memory
  - shards, ML models, cache objects
- Short-lived memory
  - task memory for request processing
  - RPC related buffers



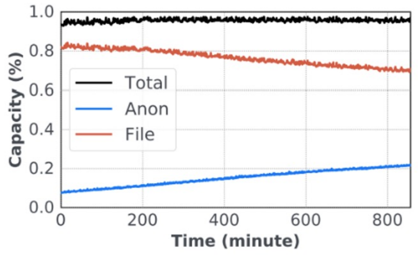
Insight 4: Need for fast convergence



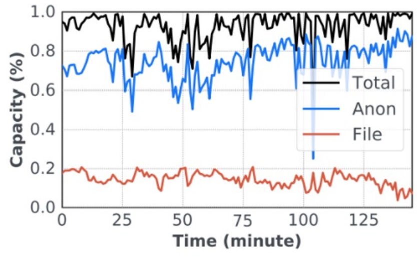
(a) *Web*



(b) *Cache1*



(c) *Cache2*



(d) *Data Warehouse*

Application needs are different and vary over time.

Need page placement mechanisms that are

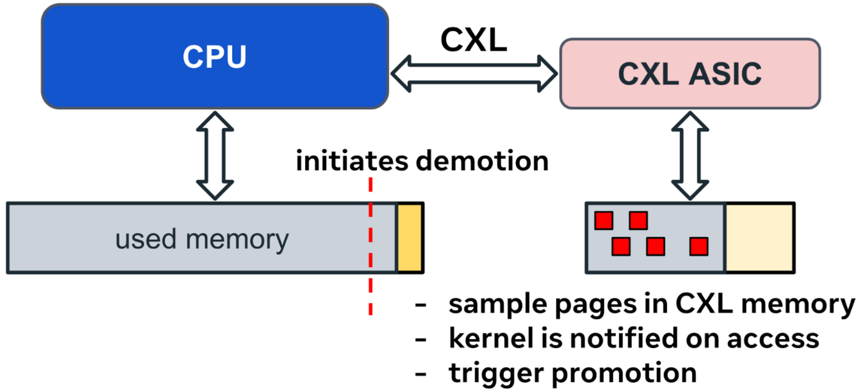
- transparent to users
- work across a wide range of use cases
- reasonably fast to converge.

Design Principles

- Transparent to applications - Page detection & movement handled by kernel.
- Converge quickly - short-lived memory won't give you much time.
- Reliable - Testing with real applications in production environment.
- Works globally - something 'upstream-able' to Linux, not Meta specific.

Transparent Page Placement (TPP) Design

- Cold page detection
  - rely on existing kernel mechanism
- Demotion
  - moving cold pages to CXL memory
  - proactively doing this to maintain free space headroom in native memory
- Promotion
  - moving hot pages to native memory
  - further optimization to avoid ping-pong movement.



Production Application Testing Results

Workload	**Efficiency Win from Capacity Increase**	
	All Native Low Capacity	CXL w/ TPP
Web	reference baseline	~0% perf (not memory bound, yet)
Cache1		+4% QPS win +2% hit ratio win
Cache2		+50% QPS win (at iso-retention time)
Data Warehouse		~30% perf win

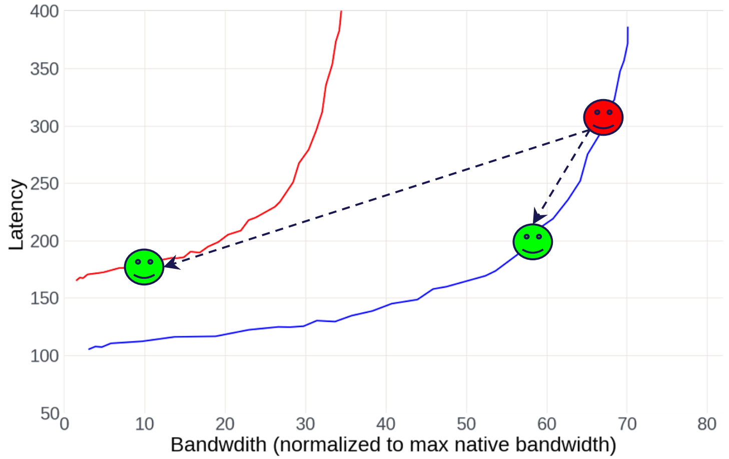
Production Application Testing Results

Workload	Efficiency Win from Capacity Increase		Performance Impact of Tiering at Iso-Total-Capacity		
	All Native Low Capacity	CXL w/ TPP	All Native High Capacity	CXL w/ default Linux	CXL w/ TPP
Web	reference baseline	~0% perf (not memory bound, yet)	reference baseline	-15%	-0.6%
Cache1		+4% QPS win +2% hit ratio win		-5%	0%
Cache2		+50% QPS win (at iso-retention time)		-2%	-0.4%
Data Warehouse		~30% perf win		-15%	-0.9%

- Less than 1% performance degradation vs. having all memory in native node.

What about the additional Bandwidth by CXL memory?

- Some applications are memory bandwidth bound
  - CXL memory provides meaningful extra bandwidth.
- To use the additional CXL bandwidth
  - need to place some hot pages in CXL memory.
- To actually get a performance win
  - need to place the "right" amount of hot pages in CXL memory.
  - optimal point varies based on the latency-bandwidth curves of native vs. CXL memory.



TPP-Bandwidth Solution

- Kernel support for N:M interleave
  - to control the exact fraction of memory traffic goes to native vs. CXL memory.
- Optimal N is mostly hardware dependent
  - i.e. the latency-bandwidth curves of two tiers.
  - transparent to application.

	All memory in Native DDR	CXL w/ 1:1 Interleave (Linux default)	CXL w/ TPP's 5:1 Interleave
% Traffic to CXL memory	0%	50%	17%
Memory Latency	reference baseline	+74%	-19%
Performance		-44%	+7%

Call to Action

- Read more about TPP
  - TPP: Transparent Page Placement for CXL-Enabled Tiered Memory on arXiv
- Improve upon our open-sourced solutions
  - patch set of Transparent Page Placement for Tiered-Memory for capacity expansion
  - patch set of N:M interleave policy for tiered memory nodes for bandwidth expansion

Thank you!

ENDLESS MEMORY
CXL™ FORUM at OCP Summit

EMPOWERING OPEN.

OCP GLOBAL SUMMIT | OCTOBER 18-20, 2022
SAN JOSE, CA