THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

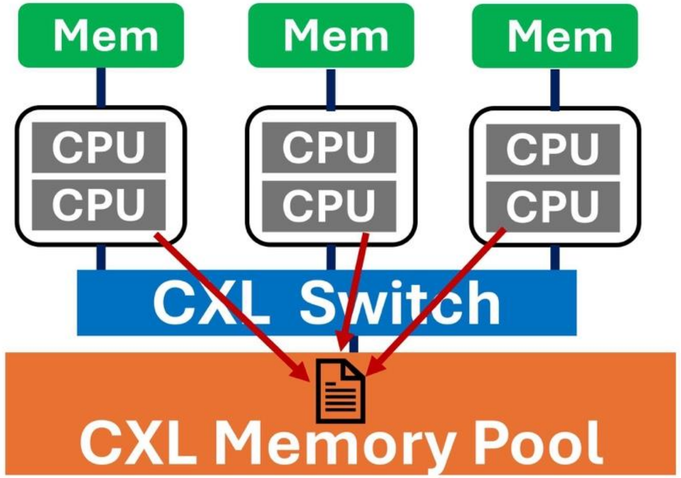
Systems-Nuts

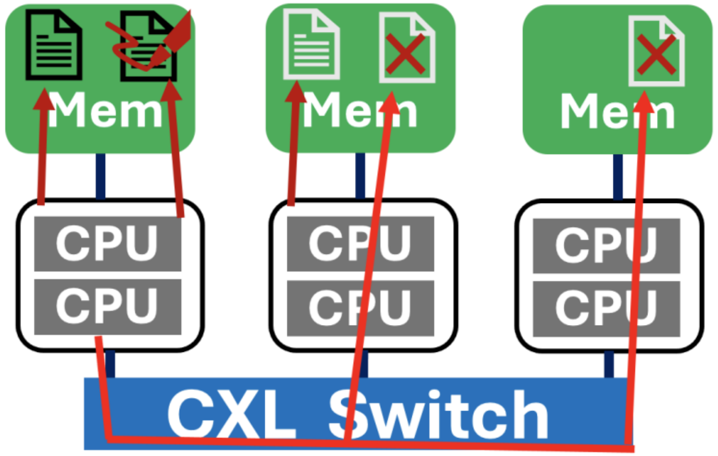
Rethinking Applications' Address Space with CXL Shared Memory Pools

Tong Xing and Antonio Barbalace

The University of Edinburgh

Introduction: Hardware-managed Coherence vs Software-managed Coherence





Direct Access

- **Hardware managed CC**
- **Data is directly shared within multiple Nodes, like in DRAM**

Pros:

- Shared data access at near-remote memory latency for reads/writes.

Cons:

- Maintaining per-cache-line directories or snoop filters at large scale is impractical

Page Replication

- **Software maintain a single coherent memory space.**
- **Page-granularity: Pages are replicated across nodes; each shared data has its replicas at each nodes**

Pros:

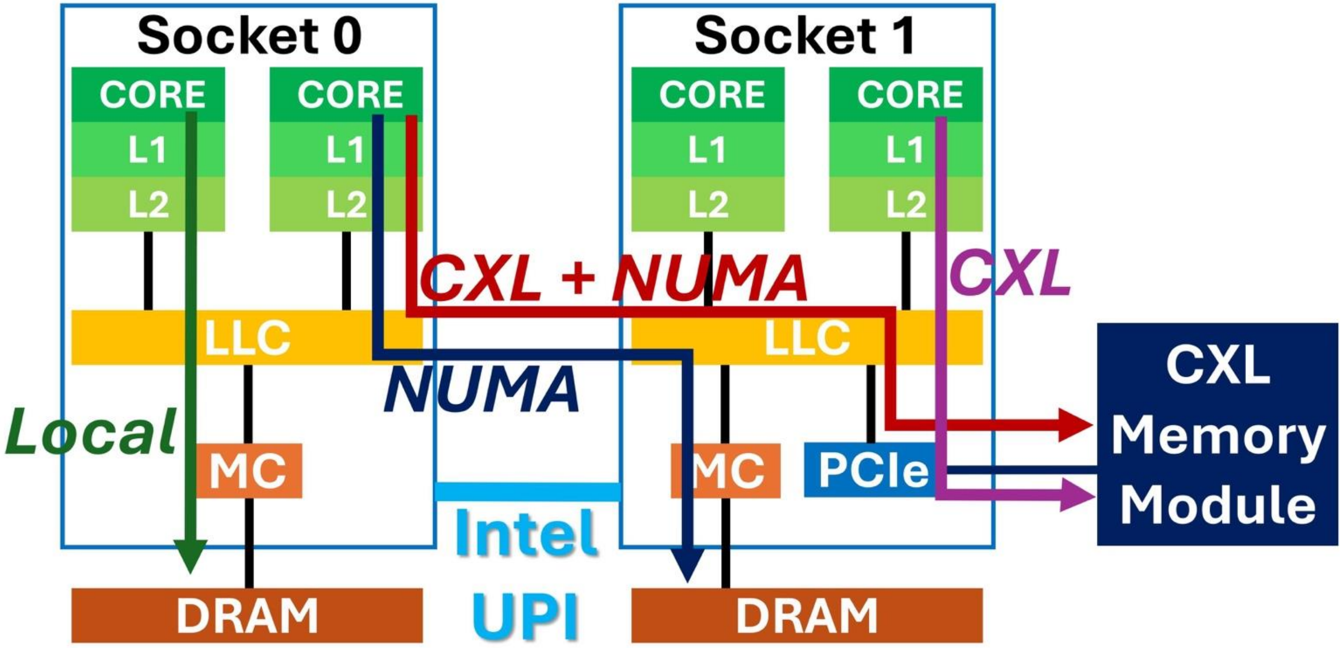
- Low-latency accesses to local data copies.

Cons:

- Frequent writes trigger invalidations across all replicas, leading to high overhead.

Software-managed Coherence can be implemented at CXL based inter-connections, What is the trade-off of choose between those two solutions?

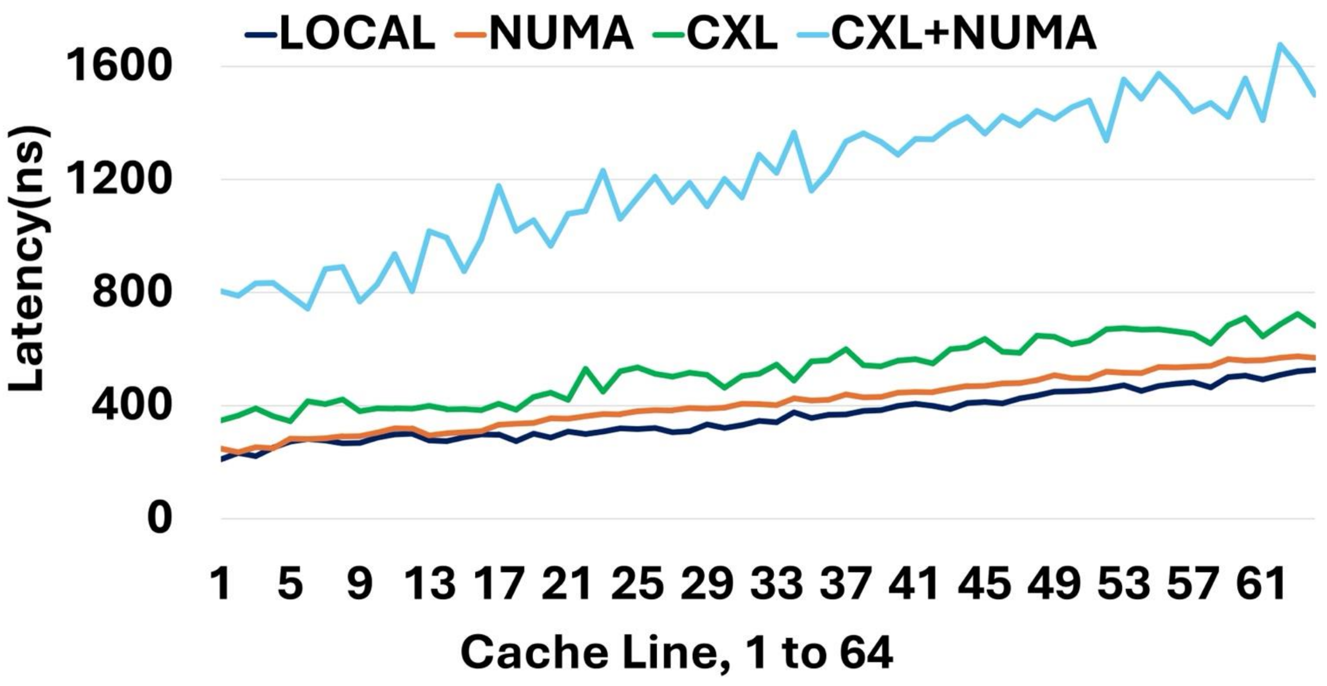
Evaluation: Setup



Local accesses directly-attached memory on the same NUMA node as the running thread
NUMA accesses directly-attached memory on a remote NUMA node (1 hop)
CXL accesses CXL-attached memory on the same NUMA node as the running thread
CXL+NUMA accesses CXL-attached memory on a remote NUMA node (2 hops)

Evaluation: Direct Access

- Latency varies across different memory tiers
- CXL+NUMA is approximately 4x more expensive than accessing local DRAM, aligning with Liu et al.[1]



Takeaway 1:
The multi-tier latency variations may become main factors influencing both system design and flexibility in next-generation cloud data centers

[1] Jinshu Liu et al. Dissecting cxl memory performance at scale: Analysis, modeling, and optimization. arXiv preprint arXiv:2409.14317

Evaluation: Page Replication

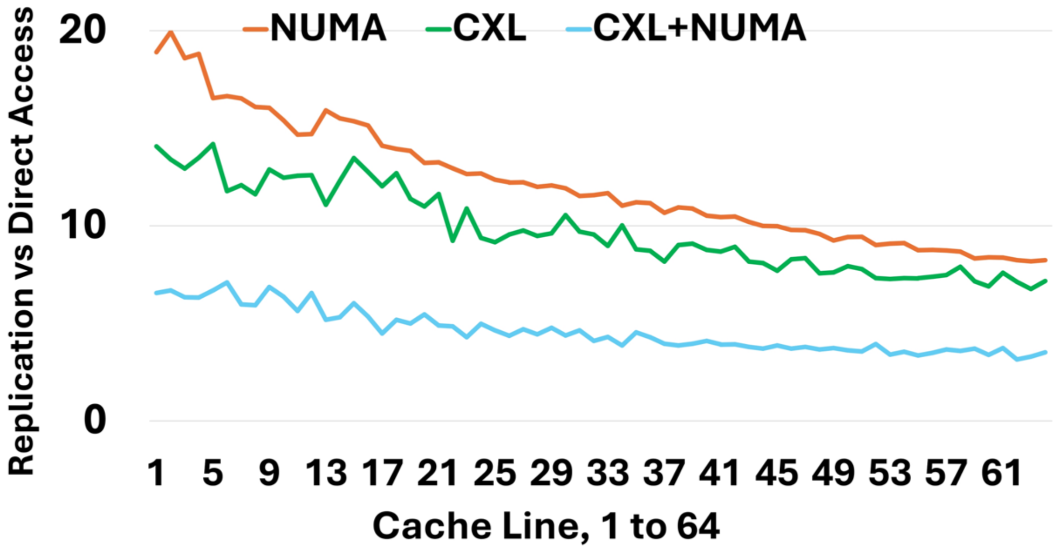
Breakdown OS overhead (page unmap/remap) + data copy

From	NUMA	CXL	CXL+NUMA
`migrate_pages()`	4826ns	4966ns	5272ns
`memcpy()`	887ns	942ns	1158ns

- Modified kernel's handle_page_fault() to migrate pages on demand
- Use migrate_pages() to copy, unmap, and remap pages
- Pollute L3 caches so copies fetch from remote memory and force write back after copy

Takeaway 2:
The overall page replication time is almost <u>independent</u> from the source or destination because it is <u>dominated</u> by OS management routines

Evaluation: Direct Access vs Page Replication



Number of cacheline(1-64) fetches equivalent to the cost of page replication over NUMA, CXL and CXL+NUMA.

Key Takeaways 3:

- Higher remote latency makes page replication more attractive
- Highly polluted caches will enforce the CPU to fetch data from remote again and again for direct access
- For read mostly data, page replication is a more favorite solution
- **Dynamic selection** of coherence (hardware vs. software) may be ideal

Adaptive Coherence Management Design

Single Memory Consistency Model

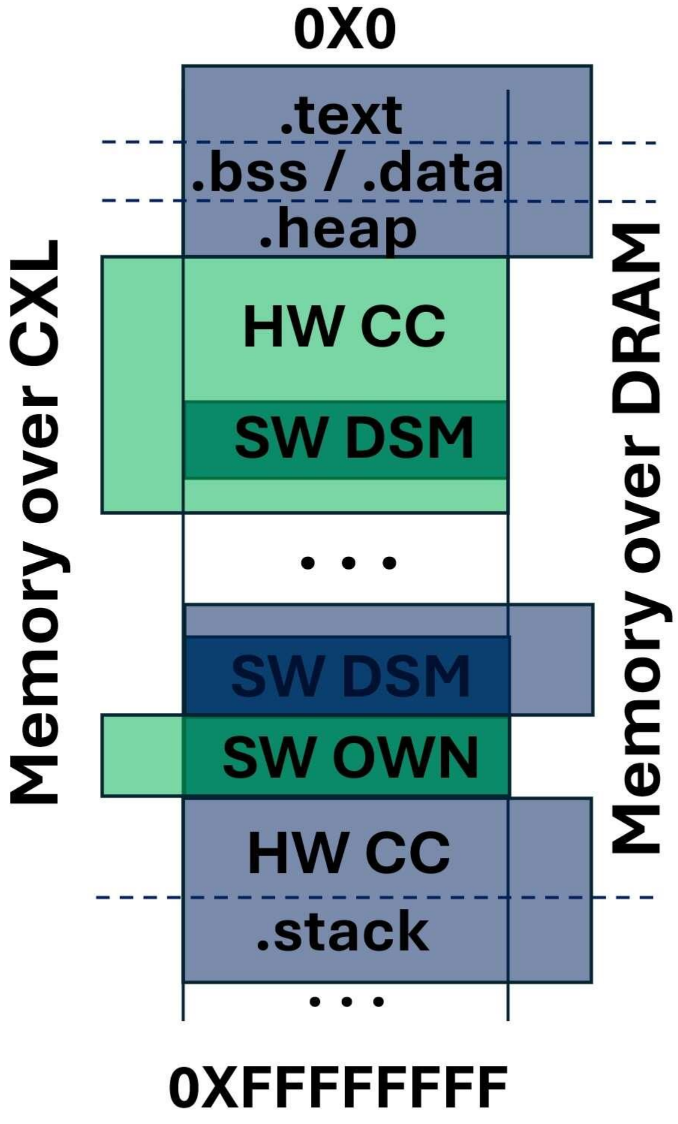
- Provide a unified, application-transparent consistency view

Per Address Space-Area Handling

- Divide App's virtual address space
- Different coherence mechanism

Lightweight Runtime Profiling and Adaptation

- Access patterns profiling
- System metrics monitoring (latency, bandwidth, usage)



Conclusion

Systems-Nuts

Software-Managed Coherence Still Matters

Even with CXL 3.0's hardware cache coherence, software-based approaches can be advantageous

No "One-Size-Fits-All" Solution

Hardware and Software-managed coherence have trade-offs; neither is universally optimal

Adaptive Coherence Management

Dynamically selects between hardware and software coherence based on runtime profiling (e.g., hot/cold pages, CXL memory latency)

Contact

tong.xing@ed.ac.uk