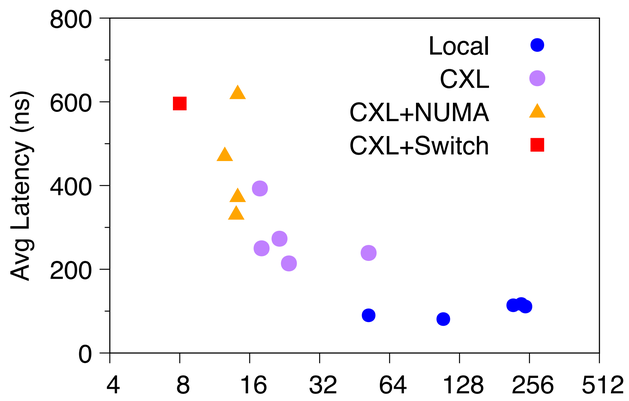
Systematic CXL Memory Characterization and Performance Analysis at Scale

Jinshu Liu, Hamid Hadian, Yuyue Wang, Daniel S. Berger*, Marie Nguyen†, Xun Jian, Sam H. Noh, Huaicheng Li
jinshu@vt.edu

Heterogenous CXL Latency and Bandwidth



CXL introduces diverse and higher latency, what is the performance implication of CXL memory across CXL devices, processors, and workloads at scale?

Research Questions:

Is CXL latency as stable/predictable as regular DRAM?

How does CXL latency affect workload performance?

How does CXL latency affect CPU pipeline (e.g., prefetching)?

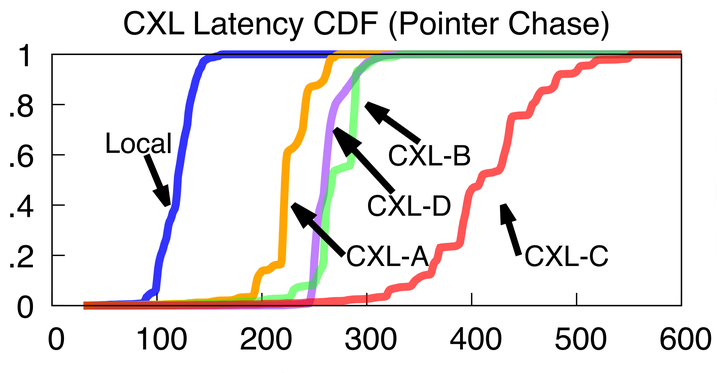
Melody Overview

A comprehensive framework for CXL characterization and analysis

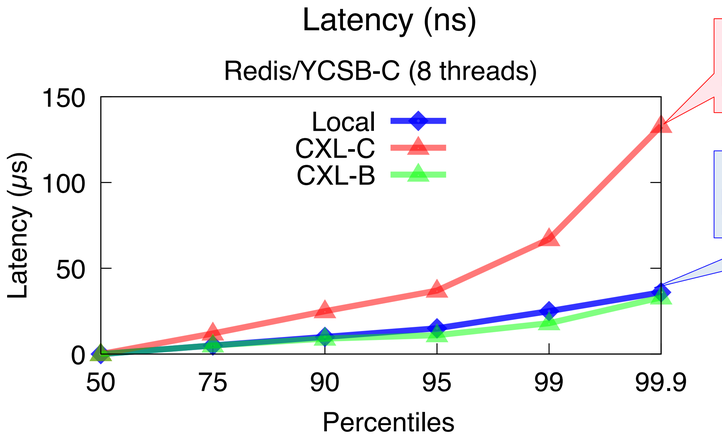
265 workloads across 4 CXL devices under 7 memory latency levels on 5 processors

1. Unstable and unpredictable latency introduced by CXL
   - µs-scale memory tail latency even when bandwidth is not saturated
2. Extensive CXL characterization across diverse workloads
   - Quantitative slowdowns due to latency or bandwidth boundness
3. SPA: A simple and accurate performance analysis approach
   - 9 CPU counters for accurate slowdown estimation (<5% inaccuracy for over 95% workloads)
   - Dissect the root causes of CXL slowdown
   - Disclose CPU prefetching inefficiency

CXL Tail Latency



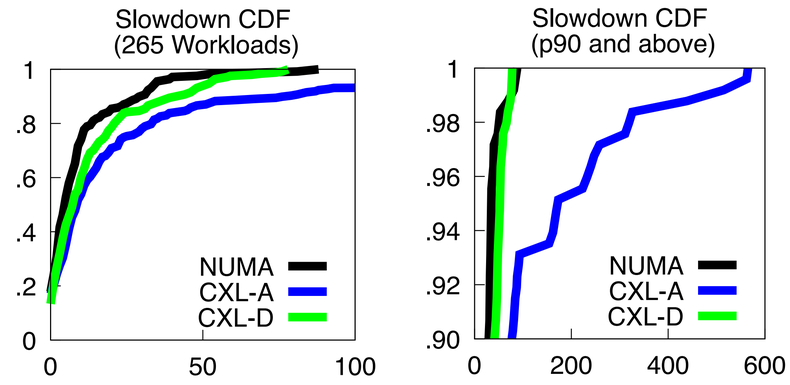
CXL devices exhibit unstable/unpredictable latency compared to regular DRAM.



CXL tail latency can lead to unpredictable application performance.

Workload Characterization on CXL

CXL	#ch DDR	Size GB	Lat ns	BW GB/s
CXL-A	2×DDR4	128	**214**	24
CXL-B	1×DDR5	128	**271**	22
CXL-C	1×DDR4	16	**394**	18
CXL-D	2×DDR5	756	**239**	52



The bandwidth limitation on CXL causes some workloads with high slowdown

The performance gap between CXL(-D) and NUMA diminishes due to its higher bandwidth even though its latency is worse

CXL memory can be used as a viable alternative to NUMA memory

SPA: Stall-based CXL Performance Analysis

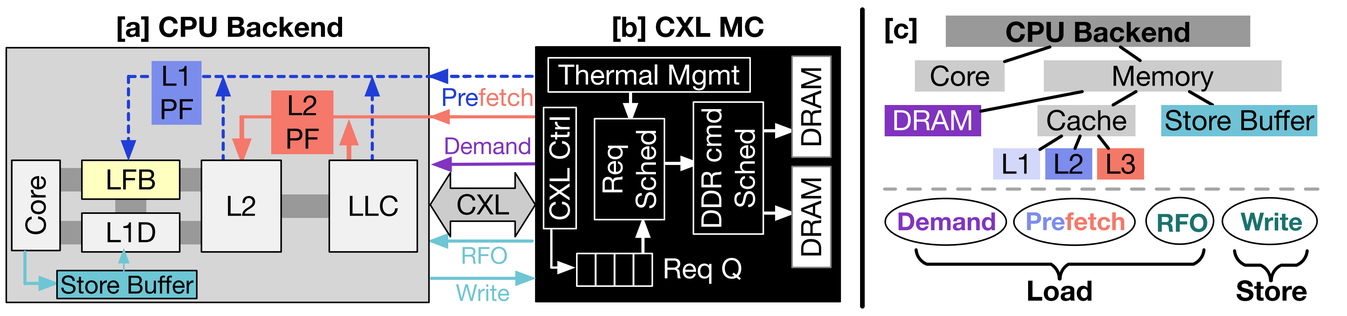
1. Workload slowdown breakdown

Δ CPU Cycles ≈ Δ DRAM-Stalls + Δ Cache-Stalls + Δ Store-Stalls

Slowdown (S) = Δ CPU Cycles / Cycles on Local

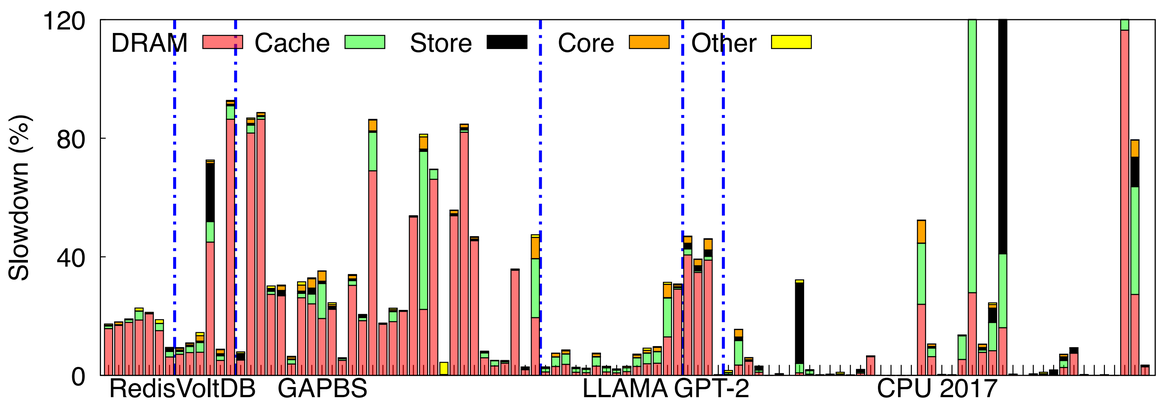
Slowdown (S) = S_{DRAM} + S_{Cache} + S_{Store}

S_{DRAM}	Demand read miss on L3
S_{Cache}	Less efficient prefetching under longer memory latency
S_{Store}	Intensive RFO on Store Buffer with limited size



2. CXL slowdown for real-world workloads

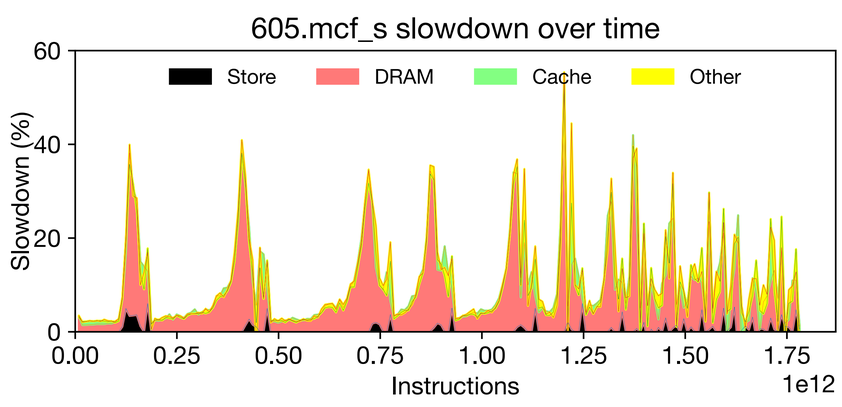
The sources of slowdown vary across workloads



3. Cache slowdown reasoning

0 CXL's longer access latency → 1 Reduced L2PF timeliness and coverage → 2 More aggressive L1PF prefetch → 3 Increasing # of delayed L1 hits

4. Dynamic slowdown



More in the paper

Appears in the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25)