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

ASPLOS 2025

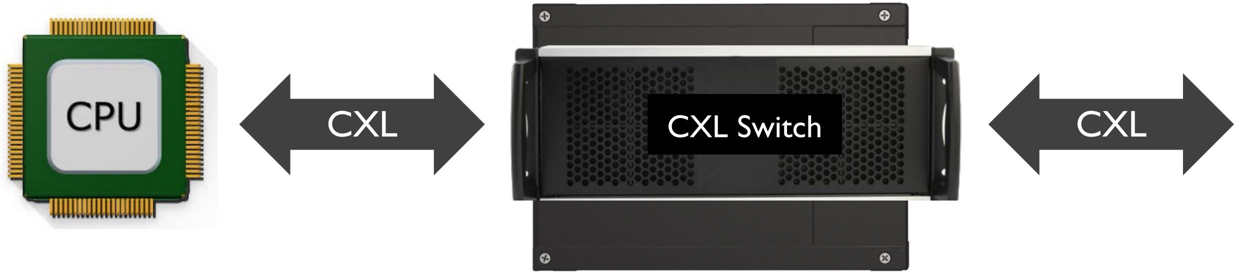
VIRGINIA TECH

[*] Microsoft

[†] SAMSUNG

Compute Express Link (CXL) Use Cases

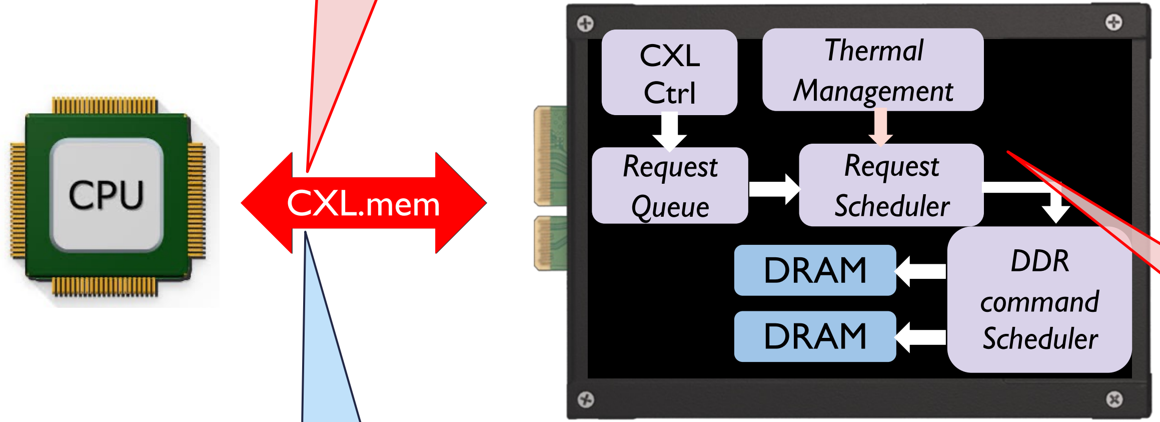
Growing demand from memory-intensive applications

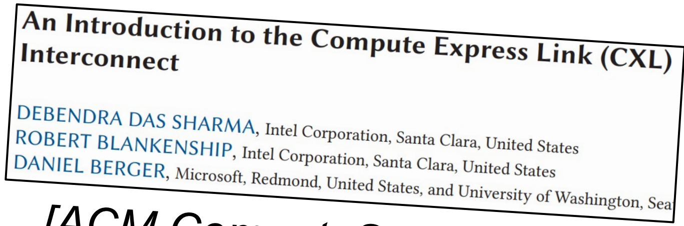


How is CXL Implemented?

PCIe electricals + low-latency protocol layers

Faster than PCIe, slower than DDR




An Introduction to the Compute Express Link (CXL) Interconnect

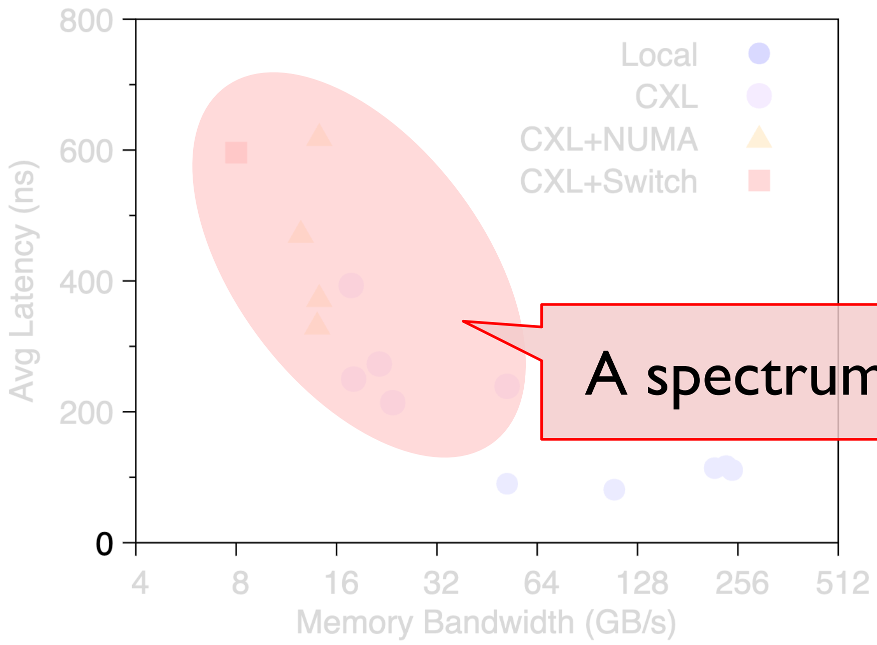
DEBENDRA DAS SHARMA, Intel Corporation, Santa Clara, United States
ROBERT BLANKENSHIP, Intel Corporation, Santa Clara, United States
DANIEL BERGER, Microsoft, Redmond, United States, and University of Washington, Sea

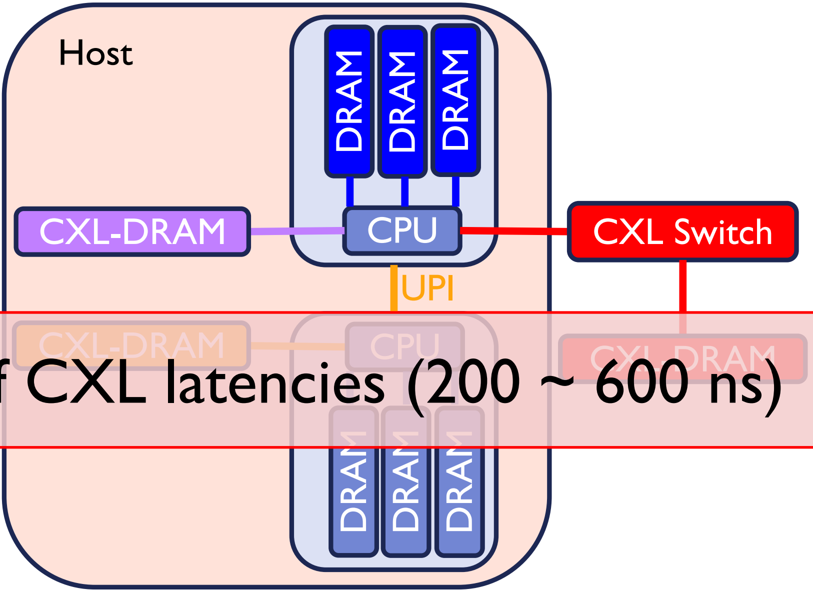
[ACM Comput. Surv. 56, 11]

Latency variability due to request processing

Transaction layer: queueing, processing, and ordering
Link layer: transaction reliability, data integrity

Heterogenous CXL Latency and Bandwidth





A spectrum of CXL latencies (200 ~ 600 ns)

What is the performance implication of CXL memory across *CXL devices, processors, and workloads* at scale?

State-of-the-Art CXL Study

I. Measure average latency and bandwidth for single CXL device

Overlook performance variation

II. Quantify the performance of a ~10 workloads

Limited scope of workloads

III. Observational approaches for performance analysis

Lack of root-cause analysis

[1] Demystifying CXL Memory with Genuine CXL-Ready Systems and Devices ***[MICRO '23]***

[2] Exploring Performance and Cost Optimization with ASIC-Based CXL Memory ***[EuroSys '24]***

[3] A Mess of Memory System Benchmarking, Simulation and Application Profiling ***[MICRO '24]***

Melody Overview

A comprehensive framework for CXL characterization and analysis

265 workloads across 4 CXL devices under 7 memory latency configurations on 5 CPUs!

Unstable and unpredictable latency introduced by CXL

μs-scale tail latency even when bandwidth is not saturated

Extensive CXL characterization across diverse workloads

Quantitative slowdowns due to latency or bandwidth boundness

SPA: A simple and accurate performance analysis approach

9 CPU counters for accurate slowdown estimation *(>95% accuracy for over 95% workloads)*

Dissect the root causes of CXL slowdown

Disclose CPU prefetching inefficiency

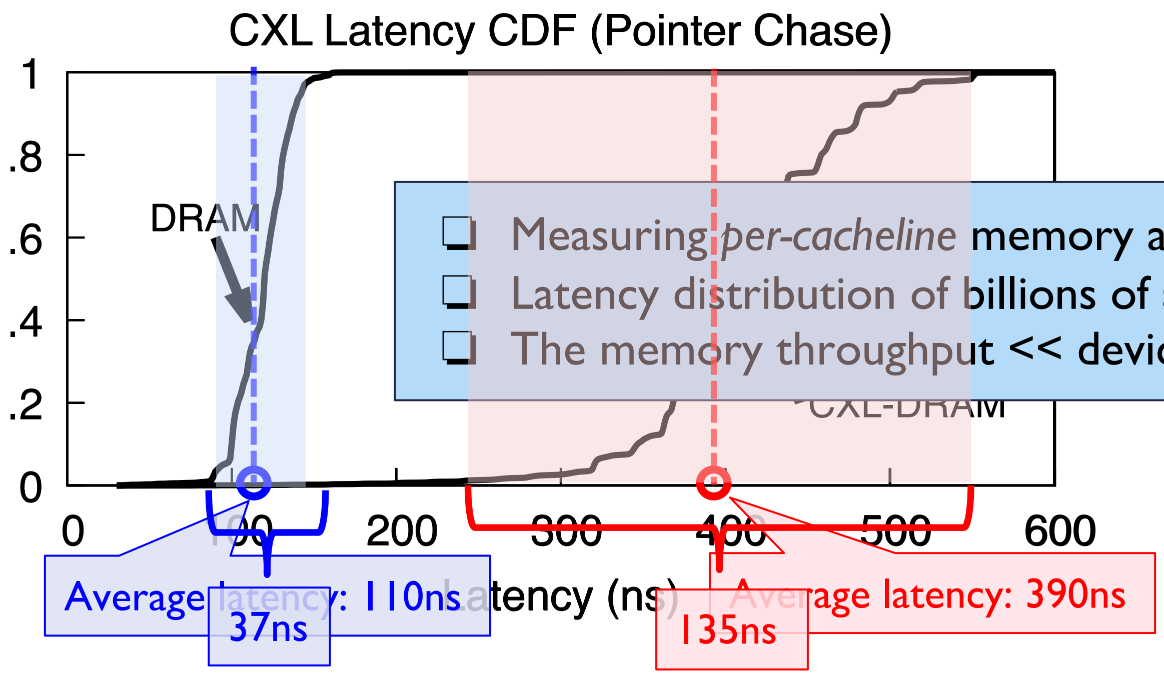
Melody overview

CXL tail latency

Workload characterization

SPA: Stall-based CXL performance analysis

CXL Latency Variation

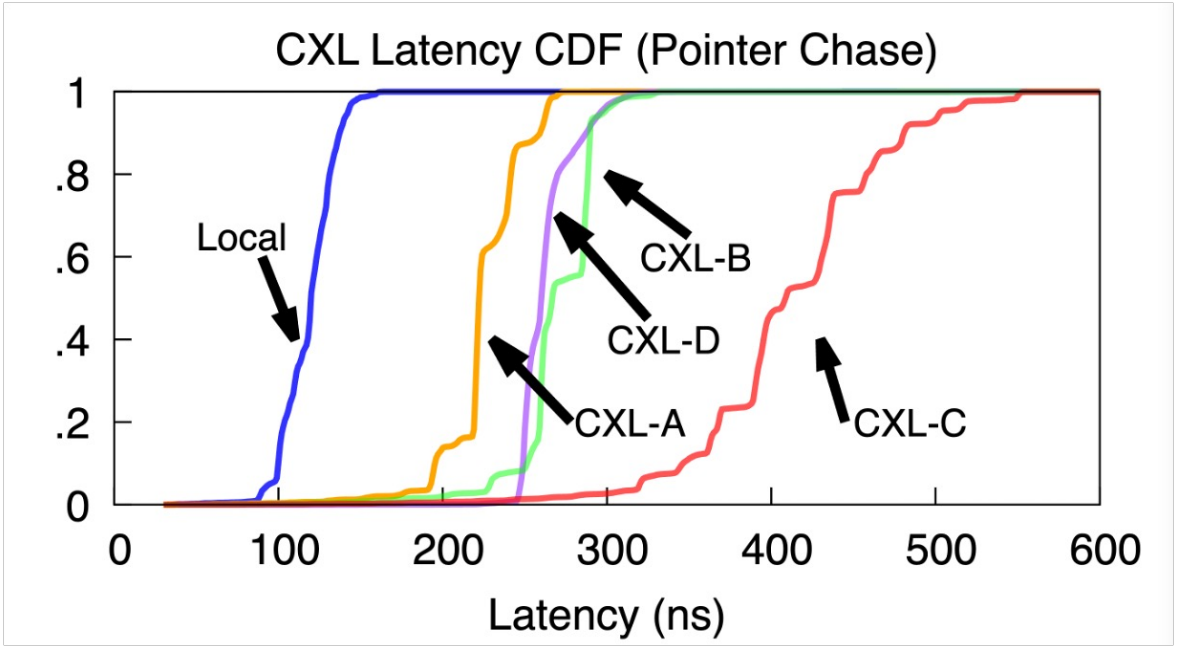


- ❑ Measuring *per-cacheline* memory access latency
- ❑ Latency distribution of billions of accesses
- ❑ The memory throughput << device bandwidth

Average latency is not enough to capture CXL performance variations

Some CXL devices exhibit unstable latency compared to regular DRAM

Tail Latency across CXL Devices

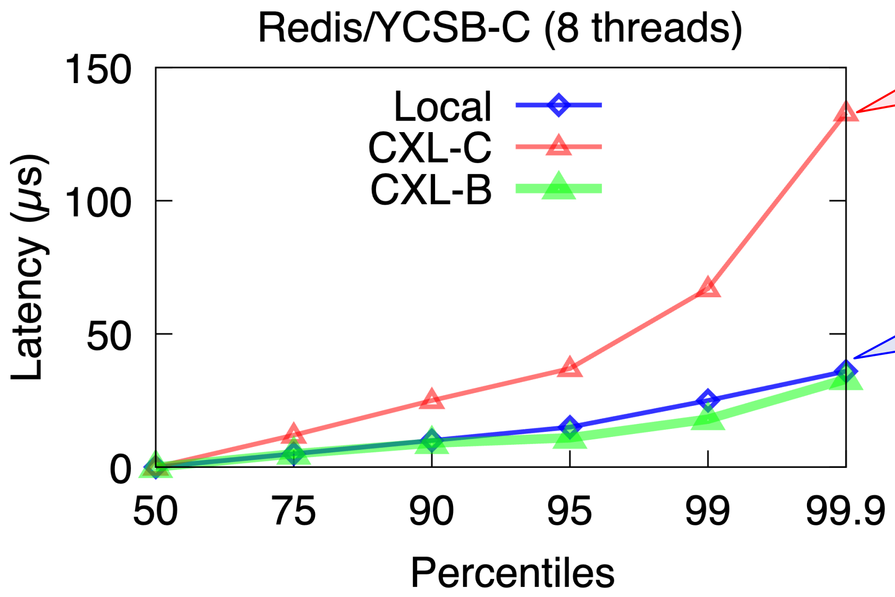


In paper:

- higher load
- interference CDFs

Some CXL devices have lower tail latency (CXL-A, CXL-D)

CXL Tail Latency in Workloads



p99.9 – p50 = 133μs

p99.9 – p50 = 40μs

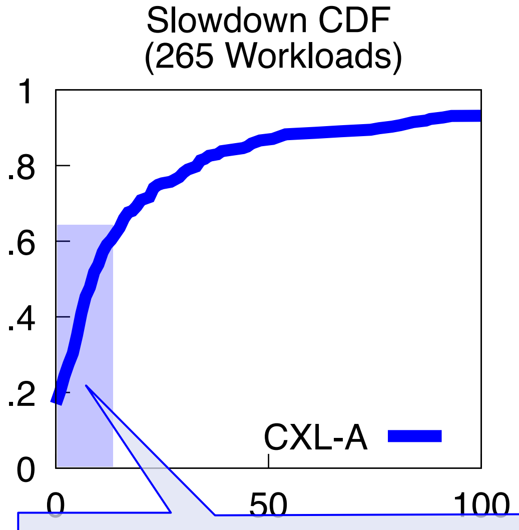
CXL tail latency can lead to unpredictable application performance

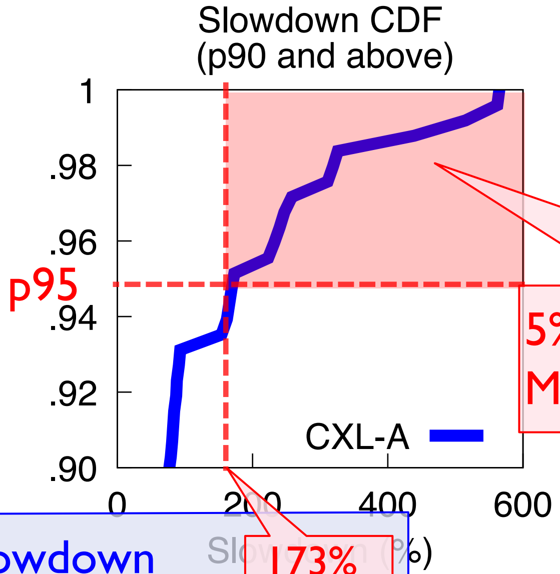
Outline

Workload Characterization on CXL

- $Slowdown = (Time_{CXL} / Time_{DRAM} - 1) * 100\%$
- Workload categories:
  - SPEC CPU 2017
  - PARSEC
  - Graph (*GAPBS, PBBS*)
  - Database (*Redis, Voltdb*)
  - ML/AI (*GPT-2, Llama, MLPerf*)
  - Data analytics (*Spark*)
  - Phoronix

Workload Characterization on CXL





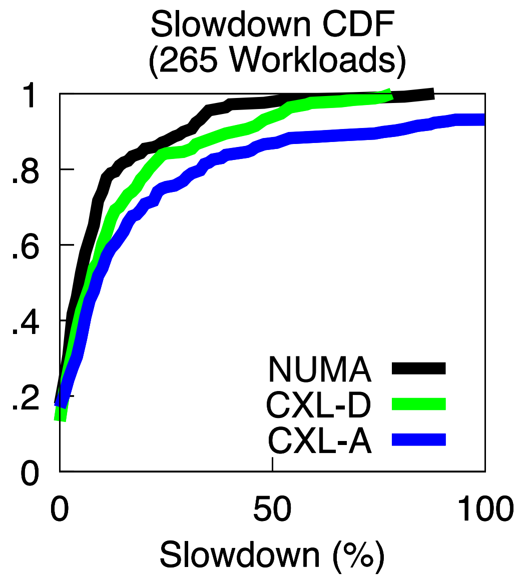
CXL-A (*214ns, 24GB/s*)

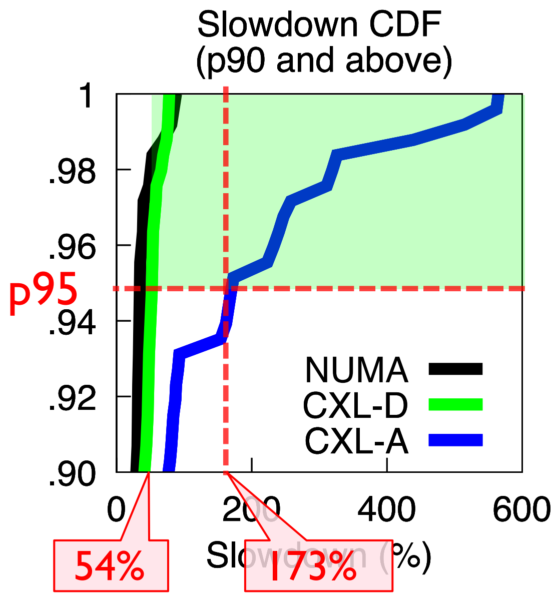
5% workloads >173% slowdown
Mainly bounded by bandwidth

173%

60% workloads <13% slowdown
Bounded by neither latency nor bandwidth

Workload Characterization on CXL





NUMA (*212ns, 119GB/s*)

CXL-A (*214ns, 24GB/s*)

CXL-D (*239ns, 52GB/s*)

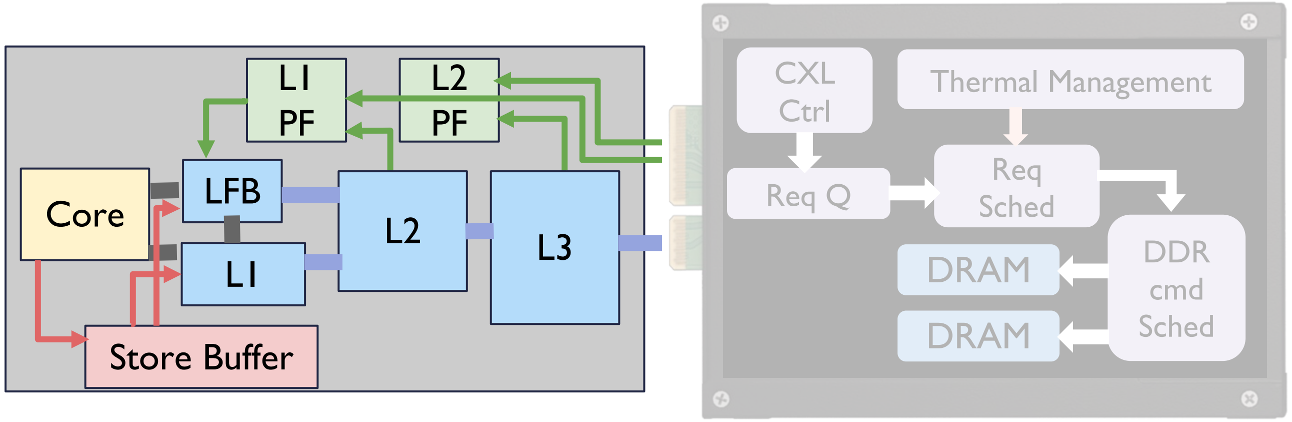
Higher CXL bandwidth (24GB/s → 52GB/s) partially mitigates slowdowns tails

CXL≈NUMA: The performance gap between (high-bandwidth) CXL and NUMA is closing!

Outline

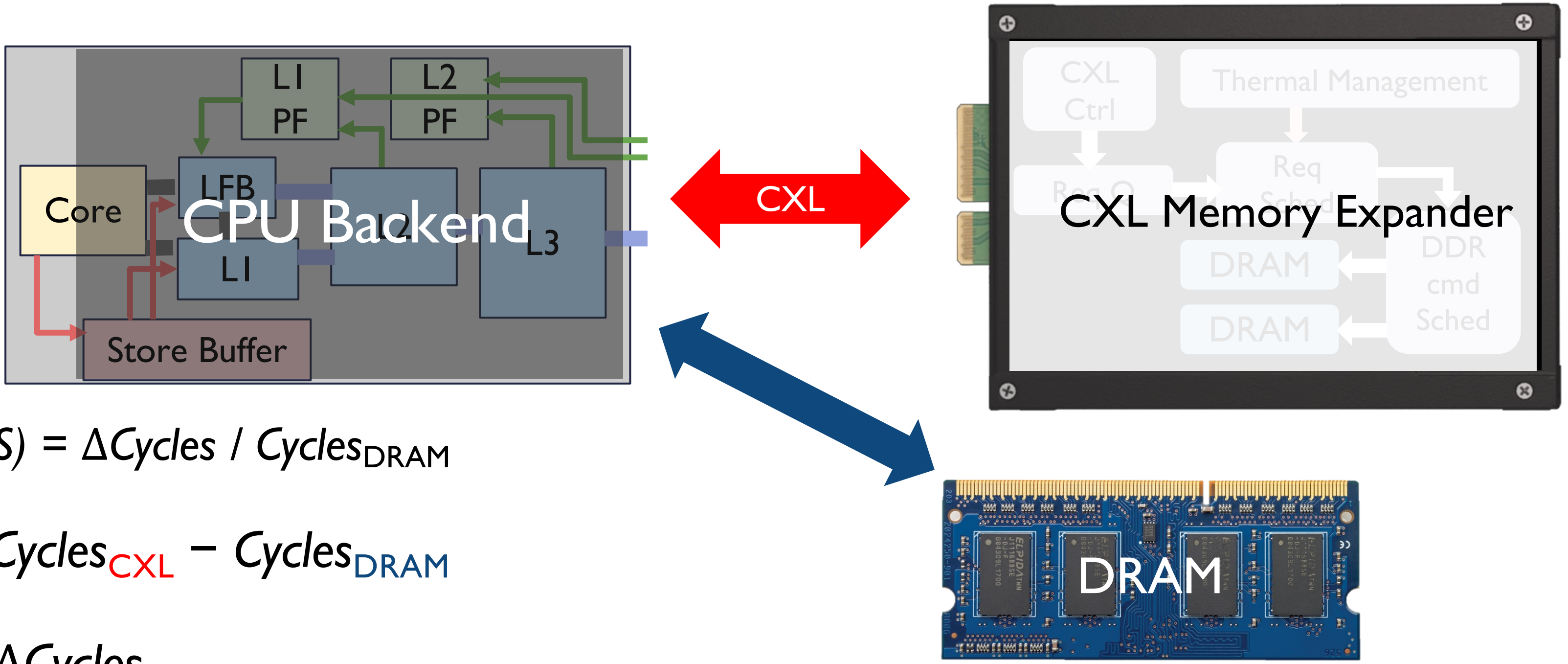
CXL Slowdown Analysis

How does CXL latency affect CPU pipeline efficiency?



CXL Slowdown Analysis

How does CXL latency affect CPU pipeline efficiency?



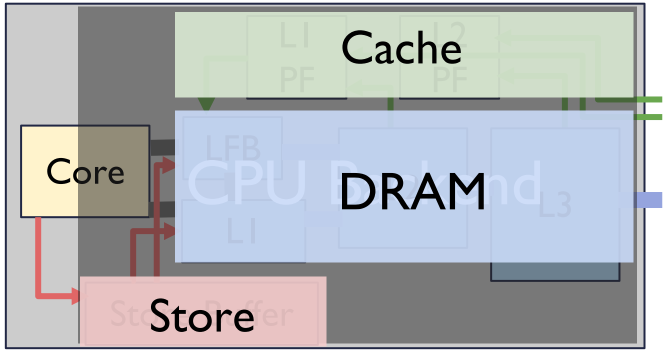
$$\textit{Slowdown (S)} = \Delta \textit{Cycles} \,/\, \textit{Cycles}_{\text{DRAM}}$$

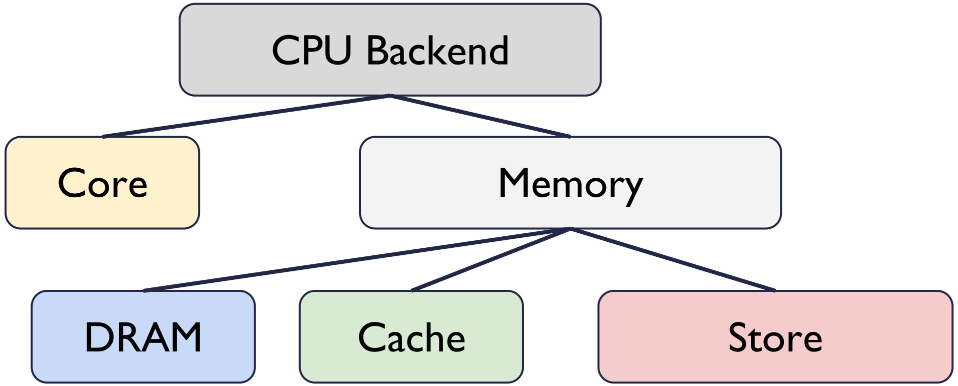
$$\Delta \textit{Cycles} = \textit{Cycles}_{\text{CXL}} - \textit{Cycles}_{\text{DRAM}}$$

$$\approx \Delta \textit{Cycles}_{\text{Backend}}$$

CXL Slowdown Analysis

How does CXL latency affect CPU pipeline efficiency?





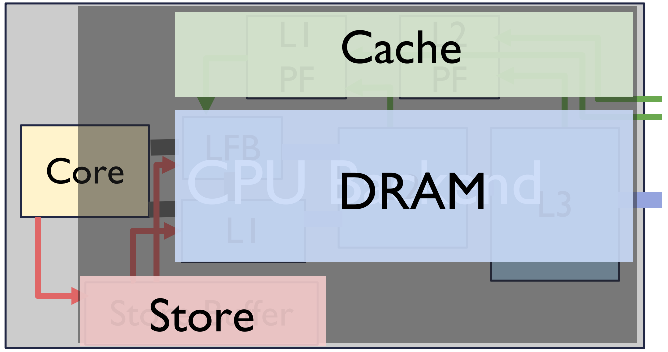
$$\Delta Cycles = Cycles_{\text{CXL}} - Cycles_{\text{DRAM}}$$

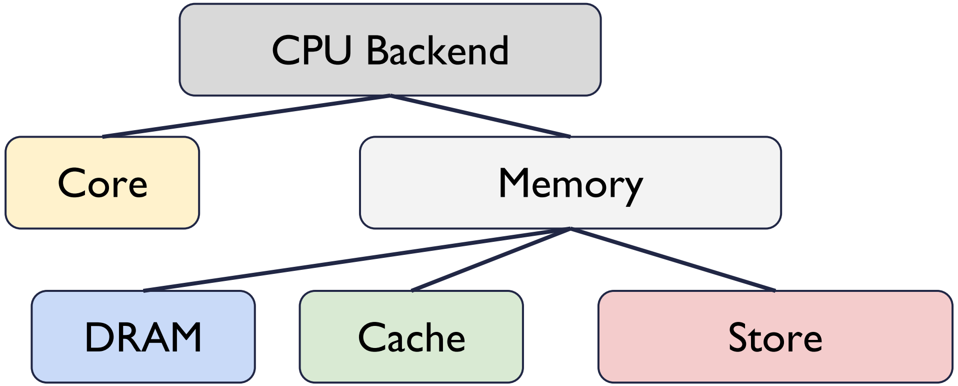
$$\approx \Delta Cycles_{\text{Backend}}$$

$$\approx \Delta Stalls_{\text{DRAM}} + \Delta Stalls_{\text{Cache}} + \Delta Stalls_{\text{Store}}$$

CXL Slowdown Analysis

How does CXL latency affect CPU pipeline efficiency?





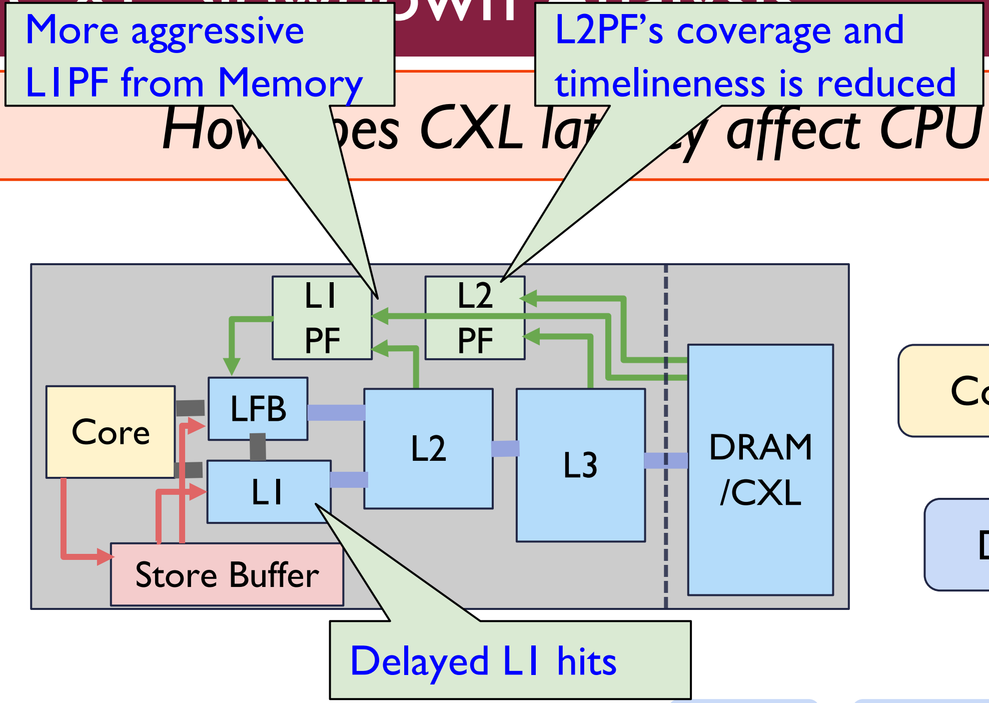
$$\Delta Cycles = Cycles_{CXL} - Cycles_{DRAM}$$

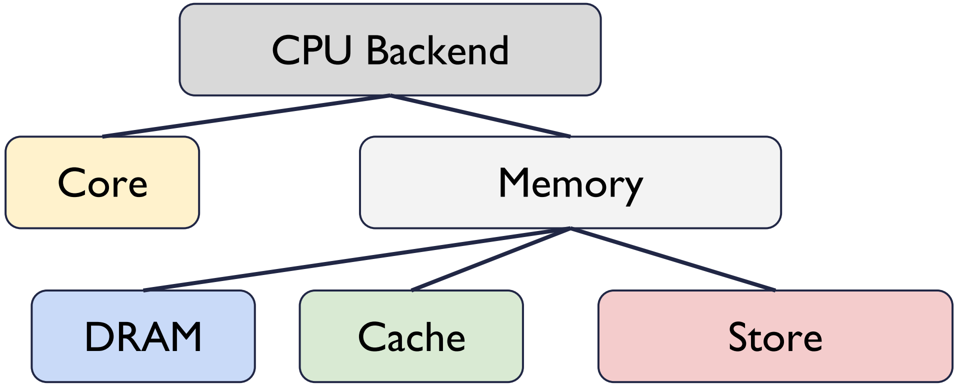
$$\approx \Delta Cycles_{Backend}$$

$$\approx S_{DRAM} + S_{Cache} + S_{Store}$$

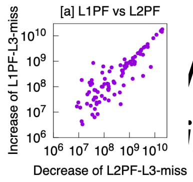
CXL Slowdown Analysis

How does CXL latency affect CPU pipeline efficiency?


More aggressive L1PF from Memory
L2PF's coverage and timeliness is reduced
L1 PF
L2 PF
LFB
Core
L1
L2
L3
DRAM /CXL
Store Buffer
Delayed L1 hits

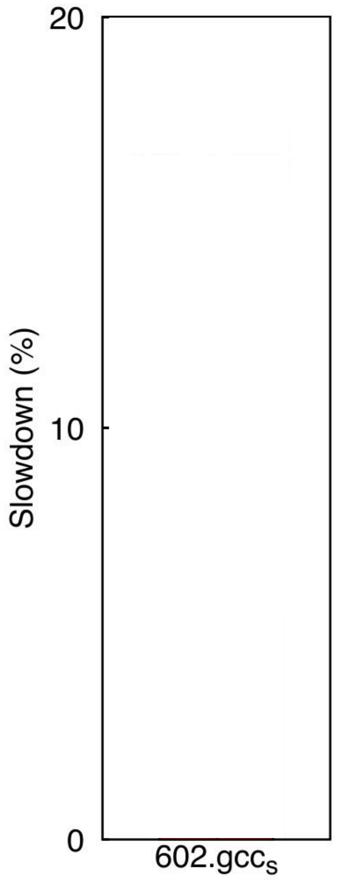


S_{DRAM}	Demand read miss on L3
S_{Cache}	*Less efficient* prefetching under longer memory latency
S_{Store}	Store buffer stays full for longer due to slower RFOs



More details in the paper

CXL Slowdown Breakdown of Real Applications



S_{Store}

+

S_{Cache}

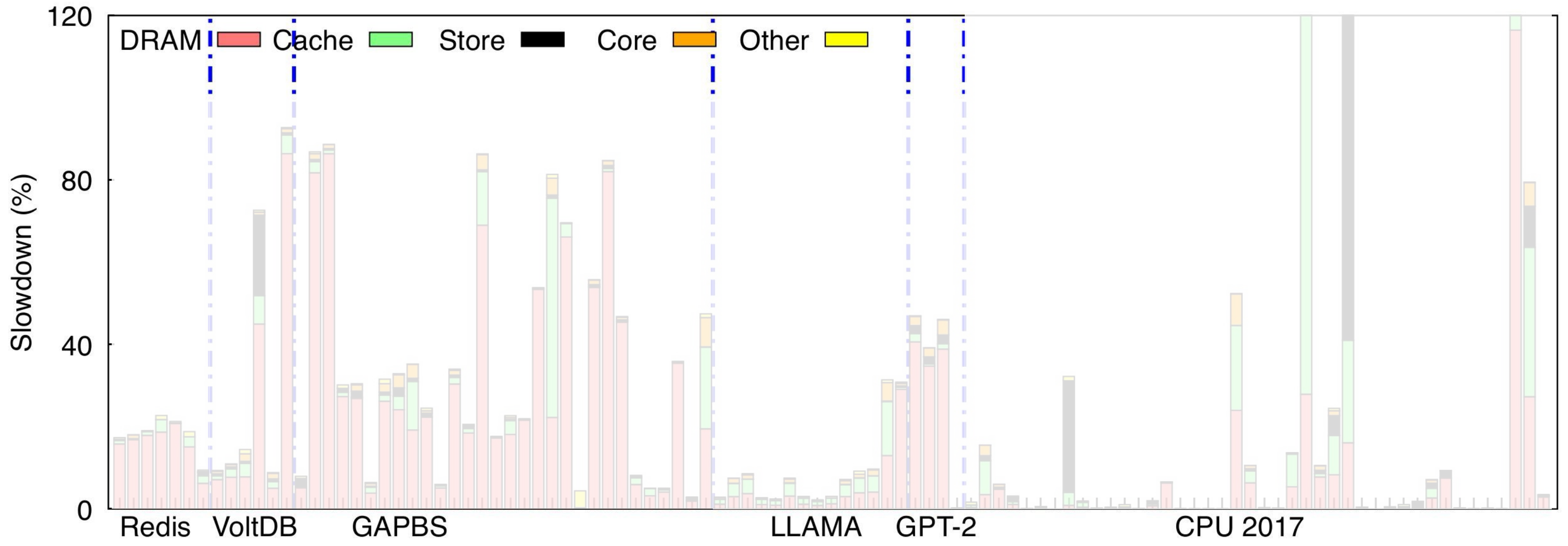
+

S_{DRAM}

→

Overall Slowdown (S)

The Sources of Slowdown Vary across Workloads

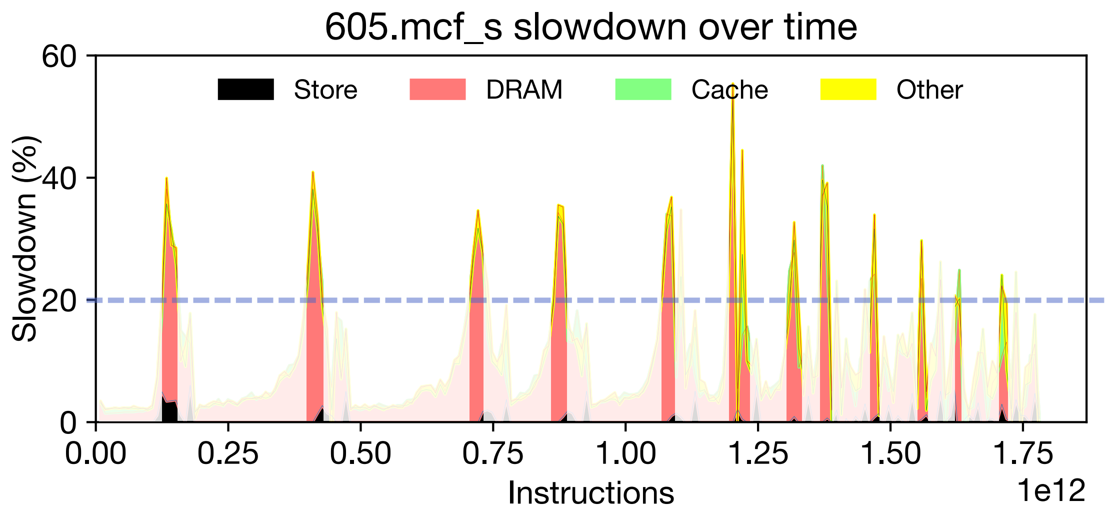


Redis, VoltDB, GPT-2: Slowdown is mainly from *demand read*

GAPBS, LLAMA: Part of slowdown is caused by *prefetching* inefficiency

CPU 2017: Diverse slowdown from *demand read*, *prefetching* and *store*

SPA for Performance Debugging & Optimization



I. Identify the range of instructions with high slowdown (e.g., > *20%*)

→

II. Locate the corresponding lines of code

→

III. Identify corresponding memory objects

→

IV. Allocate those objects to local DRAM

Slowdown will be reduced from 13% to 2%

More in the Paper!

CXL tail latency: analysis and reasoning

Factors for tail latency

Slowdown analysis

Large-scale experimental verification for SPA

Period-based slowdown analysis

SPA use cases and implications

Performance *debugging, tuning, and prediction*

Paper Code

https://github.com/MoatLab/Melody

Thank you! Questions?

Systematic CXL Memory Characterization and Performance Analysis at Scale

Jinshu Liu, Hamid Hadian, Yuyue Wang
Virginia Tech
USA

Daniel S. Berger
Microsoft and University of Washington
USA

Marie Nguyen
Samsung
USA

Xun Jian, Sam H. Noh, Huaicheng Li
Virginia Tech
USA

Abstract

Compute Express Link (CXL) has emerged as a pivotal interconnect technology for enabling scalable memory expansion. Despite its potential, the performance implications of CXL across diverse devices, latency regimes, processor architectures, and workloads remain underexplored. In this paper, we present MELODY, a comprehensive framework for systematic characterization and analysis of CXL memory performance. MELODY leverages an extensive evaluation spanning 265 workloads, 4 real CXL devices, 7 latency levels, and 5 CPU platforms. MELODY yields many key insights: workload sensitivity to sub-microsecond CXL latencies (140-410ns), the first disclosure and quantification of CXL-induced tail latency and its impact, CPU tolerance to CXL latencies, a novel stall-based root cause analysis approach (SPA) for pinpointing CXL bottlenecks, and the identification of CPU prefetcher inefficiencies under CXL.

CCS Concepts: • **Hardware → Emerging technologies**; • **Computer systems organization → Architectures**.

Keywords: Compute Express Link, CXL, Memory, Profiling

ACM Reference Format:
Jinshu Liu, Hamid Hadian, Yuyue Wang, Daniel S. Berger, Marie Nguyen, and Xun Jian, Sam H. Noh, Huaicheng Li. 2025. Systematic CXL Memory Characterization and Performance Analysis at Scale. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '25), March 30–April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3676641.3715987

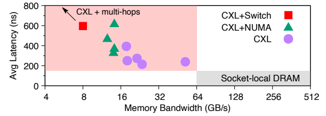


Figure 1. The spectrum of sub-µs CXL latency and bandwidth.

1 Introduction

Driven by the growing requirements of memory-intensive applications, the demand for increased memory capacity is rapidly rising [37]. The surge is further compounded by DRAM scaling challenges [41]. Emerging interconnects like Compute Express Link (CXL) hold the promise of both scale-up and scale-out memory expansion at the server/rack levels [34, 36, 45]. Various memory vendors have introduced CXL memory expanders [3, 4, 8, 15], some of which are being deployed in production systems, facilitating access to significantly larger amounts of DRAM than previously feasible.

Low memory access latency is key to system performance, but CXL memory expansion introduces higher latencies compared to traditional socket-local DRAM [27, 34, 42]. Figure 1 illustrates the substantial heterogeneity in CXL latency and bandwidth, as measured across 4 CXL devices within our platform (Table 1) and 2 more data points from public sources[1][15, 17]. Furthermore, CXL devices can exhibit varying performance characteristics. The variability in latency and bandwidth arises from varying interconnection topologies and vendor optimizations [27, 42]. For instance, the latencies of locally-attached CXL range from ~200-400ns, slightly exceeding NUMA latency. Accessing CXL memory from a remote socket results in increased latency and diminished bandwidth (CXL+NUMA). The use of CXL switch(es) to extend connectivity will introduce additional latencies (CXL+Switch), even elevating latency to approximately 600ns.

The current CPU architecture and memory hierarchy are tailored for typical multi-socket systems, offering ~100ns latency and 100s of GB/s bandwidth. *However, the performance implications of CXL memory with sub-µs latencies remain*

[1] CXL+Switch data is from [15], and bandwidth is averaged for 1 CXL device.