ASPLOS 2026

Performance Predictability in Heterogeneous Memory

Jinshu Liu, Hanchen Xu, Daniel S. Berger*, Marcos K. Aguilera†, Huaicheng Li

VIRGINIA TECH

* Microsoft

† NVIDIA

Heterogeneous Memory

DRAM	NUMA/CXL
Fast-tier	*Slow-tier*
Limited size Fast, low latency	Memory expansion Slow, High latency

Misplacing the data on the wrong tier

- Pipeline stalls
- Bandwidth waste
- SLO violations

Quantify Placement Impact Before It Happens

DRAM	NUMA/CXL
Fast-tier	*Slow-tier*

Without accurate predictors

Cloud operations overprovision DRAM to minimize risk

OS/runtimes correct placement errors after performance has degraded

Predictors are required

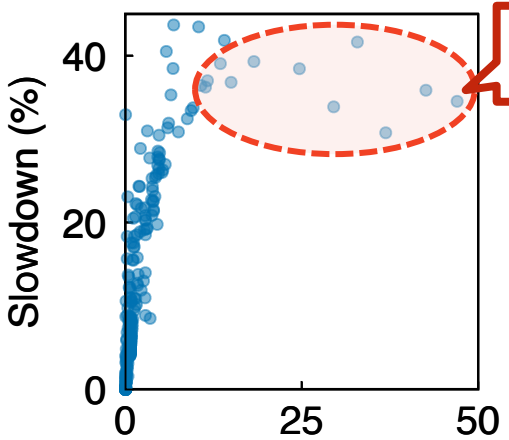
Accurate slowdown predictors can tell how a placement decision will affect application performance

Existing Metrics Fall Short

Memory systems rely on metrics to indicate performance slowdown

Correlation analysis (*metrics vs. slowdown*)

265 workloads



High MPKI with low slowdown

Miss per kilo instruction

Correlation Coefficient 0.40

The Predictability Gap

Correlation analysis (*metrics vs. slowdown*)

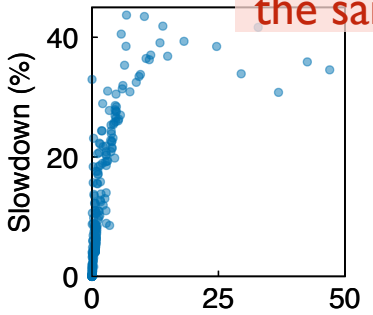
265 workloads

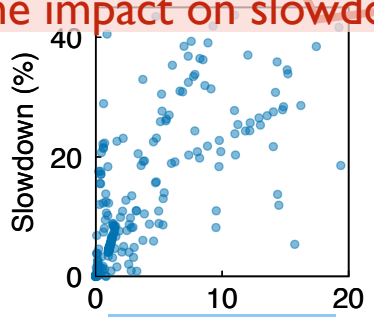
Different slowdown under the same bandwidth/latency

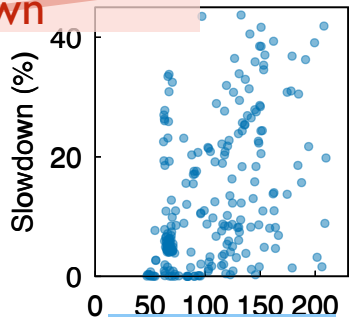
Each memory access does not have the same impact on slowdown

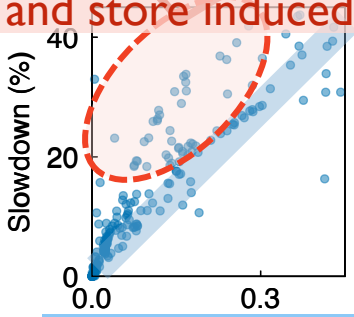
Under-prediction

Cannot capture prefetch and store induced effects









	MPKI	Bandwidth	Latency	Stalls, MLP, latency
Correlation Coefficient	0.40	0.66	0.37	0.88

Simple proxy metrics fail to accurately capture microarchitectural behavior

Memstrata [OSDI'24] *BATMAN [MEMSYS'17]* *Colloid [SOSP'24]* *SoarAlto [OSDI'25]*

Research Question

Is it possible to predict the workload performance on CXL using intrinsic workload signatures?

Insights

Is it possible to predict the workload performance on CXL using intrinsic workload signatures?

Empirical observation → Analytical modeling

Single source signals → Hints from multi-sources

Hints from multi-sources → Predictive interleaving model

CAMP: Causal Analytical Memory Prediction

Metrics for each sub-slowdown

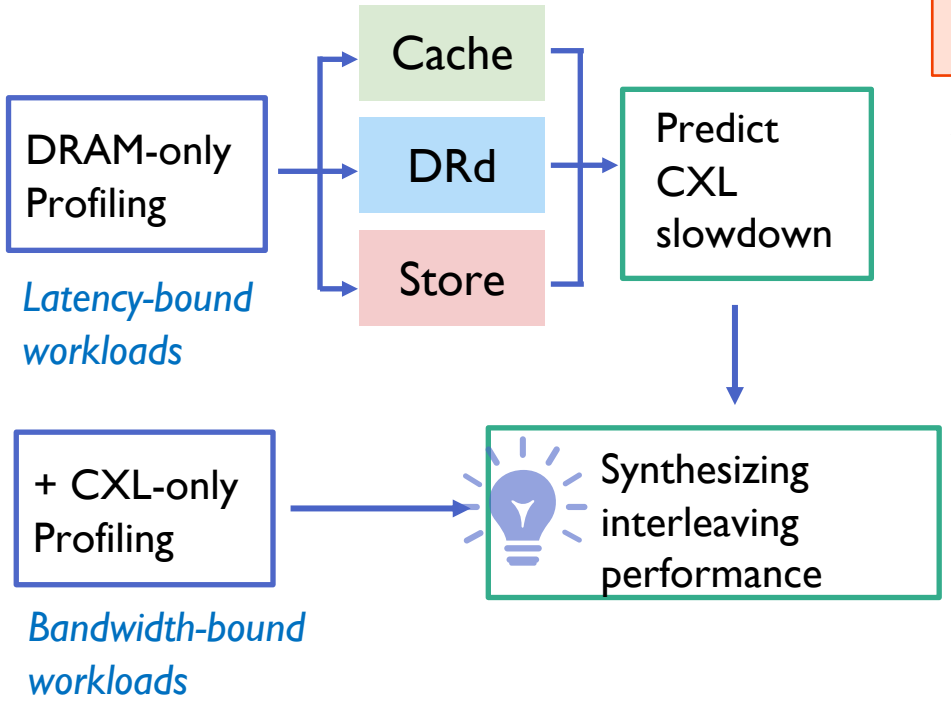
Formulated by microarchitectural analysis

3 sources of slowdown

Melody [ASPLOS'25]

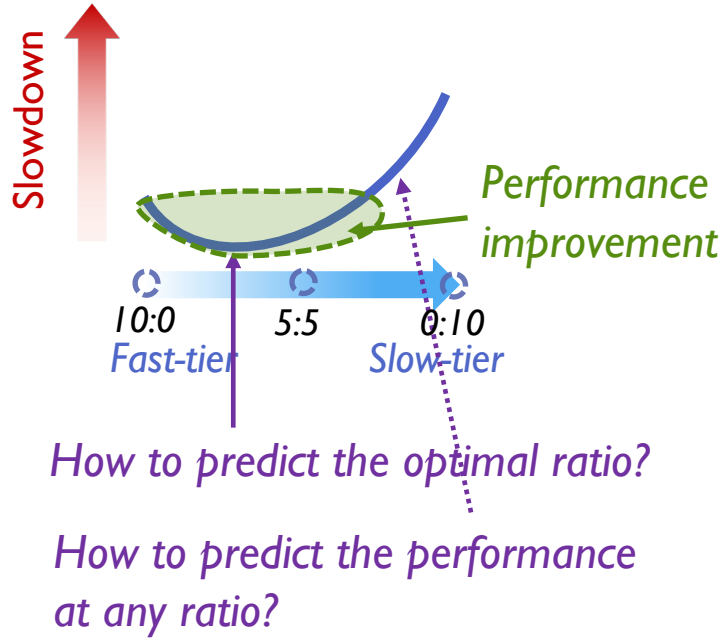
CAMP achieves 0.97 correlation with slowdown with 12 PMU counters

CAMP: Causal Analytical Memory Prediction

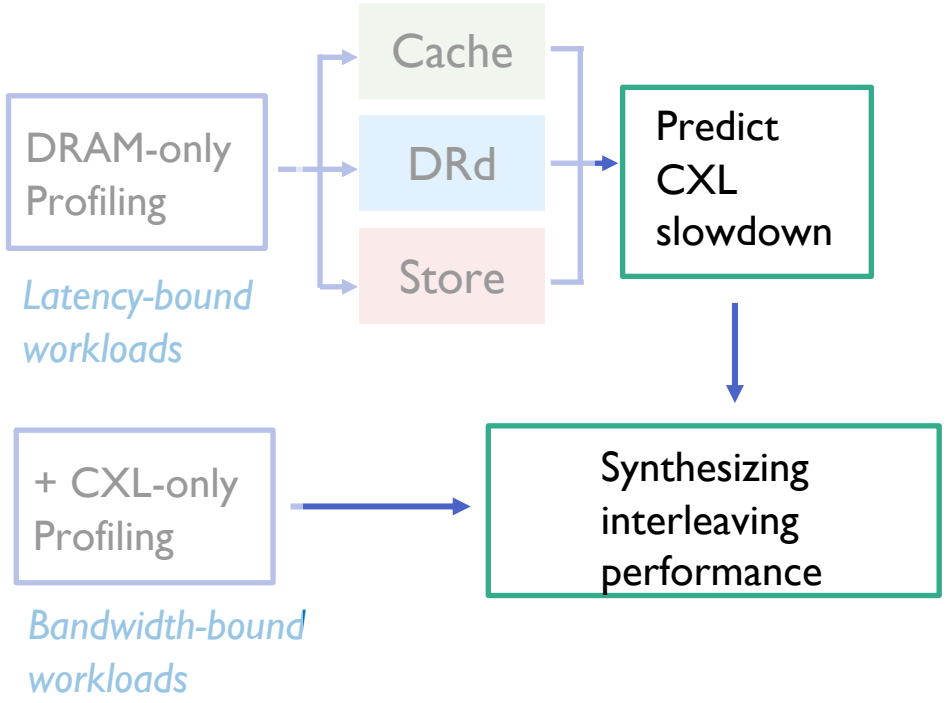


DRAM/CXL software interleaving

Round-robin allocation M:N (DRAM:CXL)



CAMP: Causal Analytical Memory Prediction



Workload colocation:
Identifies latency-tolerant workloads safely placed on CXL

+12% performance improvement vs. MPKI guided placement

Best-shot interleaving:
Predicts the optimal interleaving ratio for bandwidth-bound workloads

+21% performance vs. state-of-the-art tiering systems

Outline

Overview

Slowdown Prediction

Synthesizing Interleaving Performance

Use Cases

Predict CXL Slowdown from Local

$$\text{Slowdown} = \frac{\Delta c}{c}$$

$$S_{Cache} + S_{DRd} + S_{Store}$$

$$\frac{\Delta \text{stalls}_{Cache}}{c} \qquad \frac{\Delta \text{stalls}_{DRd}}{c} \qquad \frac{\Delta \text{stalls}_{Store}}{c}$$

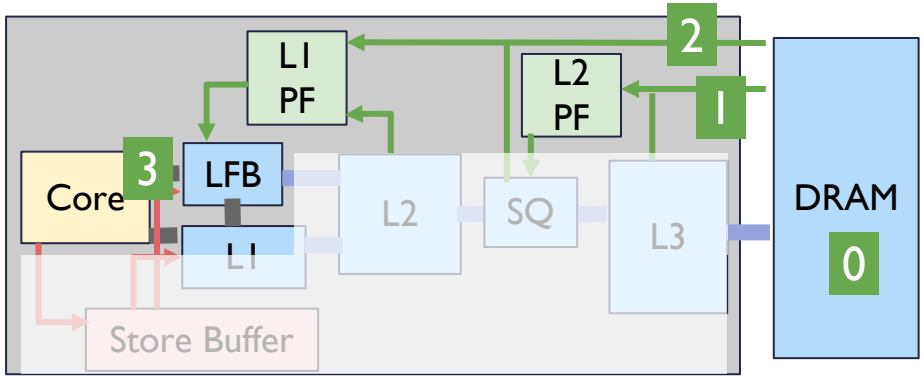
Melody [ASPLOS'25]

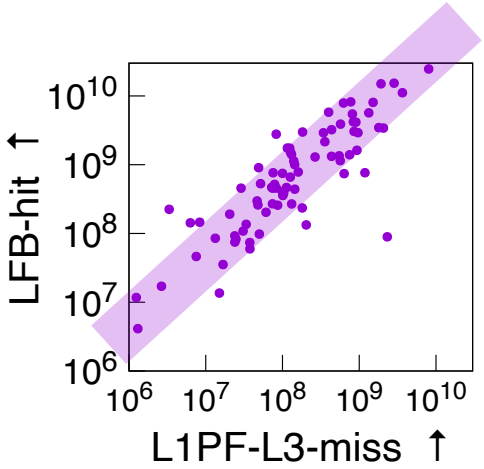
Predict Δ from local measurement?

Microarchitectural Pressure Points

S_{Cache}	+	S_{DRd}	+	S_{Store}
Line Fill Buffer (LFB)		Memory-Level Parallelism (MLP)		Store Buffer
Prefetching		Latency		
Delayed prefetches affect data access to Line Fill Buffer (LFB)		The change of latency and MLP affects the increased DRd stalls		Store Buffer is the bottleneck when intensive writes are issued

Slowdown from Prefetching (S_{Cache} Reasoning)





Each point demotes one workload

0 Memory latency ↑

Prefetching timelineness ↓

1 L2PF-L3-miss ↓

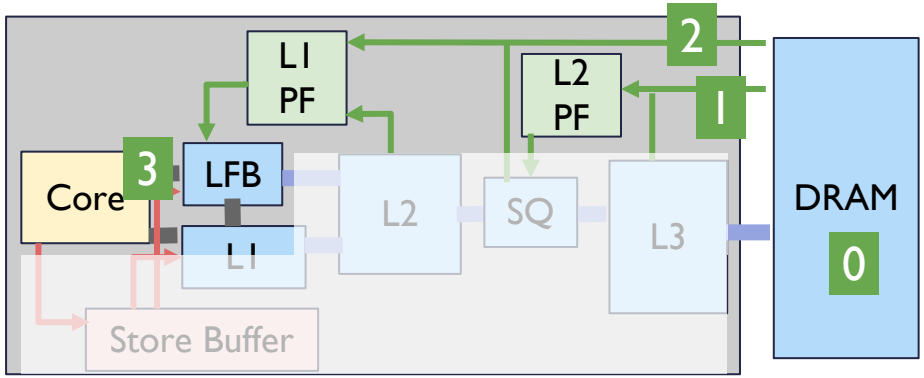
2 L1PF-L3-miss ↑

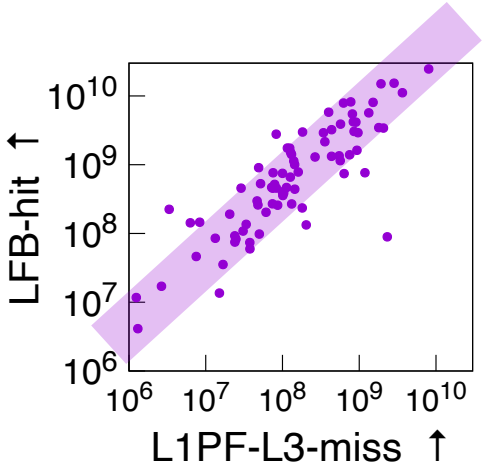
3 LFB hit ↑ L1 hit ↓

The prefetched data previously served from L1 resides on LFB.

The prefetched data from DRAM to LFB is delayed by increased memory latency

Slowdown from Prefetching (S_{Cache} Reasoning)





Each point demotes one workload

0 Memory latency ↑

Prefetching timelineness ↓

1 L2PF-L3-miss ↓

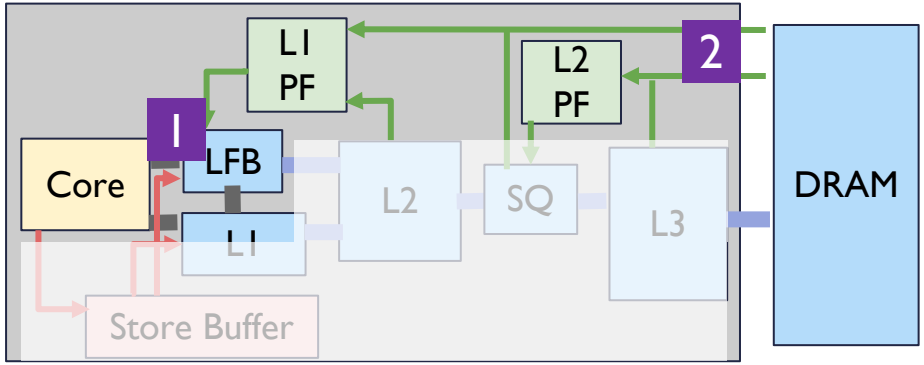
2 L1PF-L3-miss ↑

3 LFB hit ↑ L1 hit ↓

Demand reads on caches are delayed due to late prefetches

4 Stalls on cache ↑

Slowdown from Prefetching (S_{Cache} Prediction)



How to derive the predictors?

1 LFB hit ratio ($R_{LFB\text{-}hit}$):
Workload reliance on data access on LFB

2 Prefetch from memory (R_{Mem}):
The fraction of LFB allocations attributable to memory prefetches

$$S_{Cache} = k \times R_{LFB\text{-}hit} \times R_{Mem} \times \left(\frac{\Delta stalls_{Cache}}{c}\right)$$

Slowdown from Demand Reads (S_{DRd})

$$S_{DRd} = \frac{\Delta stalls_{DRd}}{c} \approx \frac{\Delta C}{c}$$

Memory active cycles

# of data requests

Latency

$$C = \frac{N \times L}{MLP}$$

Memory-Level Parallelism

$$S_{DRd} \approx V \times \frac{C_{DRAM}}{c}$$

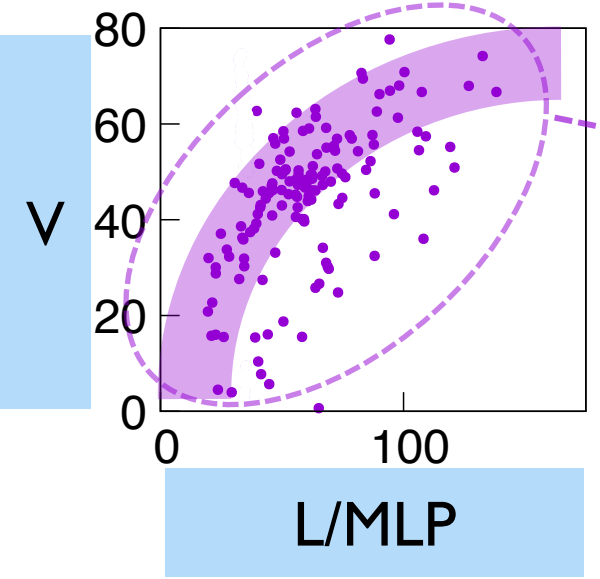
The variance of latency and MLP

Normalized C (measured on local)

Slowdown from Demand Reads (S_{DRd})

$$S_{DRd} \approx V \times \frac{C_{DRAM}}{c}$$

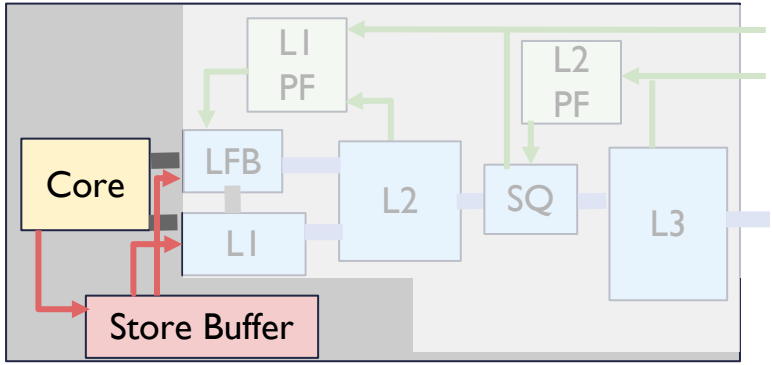
How to model V from L and MLP on local measurement?



$$S_{DRd} \approx V \times \frac{C_{DRAM}}{c}$$

$$S_{DRd} \approx k \times \frac{1}{p + q \cdot \frac{MLP_{DRAM}}{L_{DRAM}}} \times \frac{s_{LLC}}{c}$$

Slowdown from Stores (S_{Store})



s_{SB}: the stall cycles when store buffer is full due to the intensive writes

$$S_{\text{Store}} \approx k \times \frac{s_{\text{SB}}}{c}$$

Intensive writes issued to Store Buffer

CPU stalls incur when Store Buffer is full

Increased memory latency -> more stalls on Store Buffer

Implementation

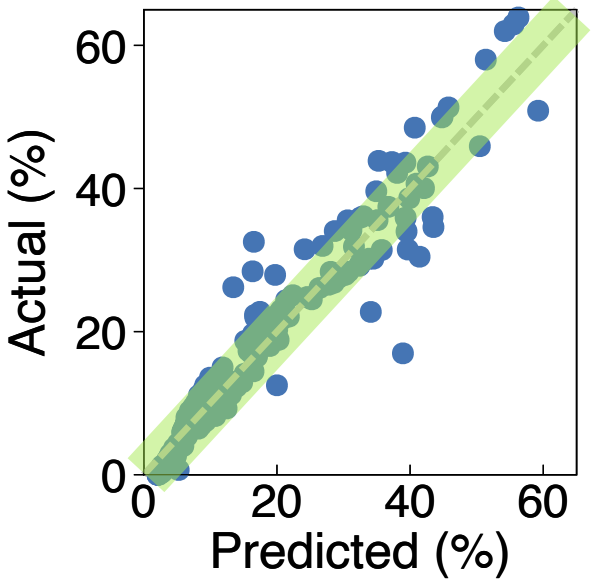
Each sub-slowdown is modeled as k x f(metrics)

Counter mapping	Map the abstracted metrics to specific Intel PMU counters
One-time calibration	Use microbenchmarks to determine k and other hardware dependent constants
Run-time prediction	Feed measured PMU counters into the DRAM-run to compute the predicted performance

CAMP Prediction Model is Accurate

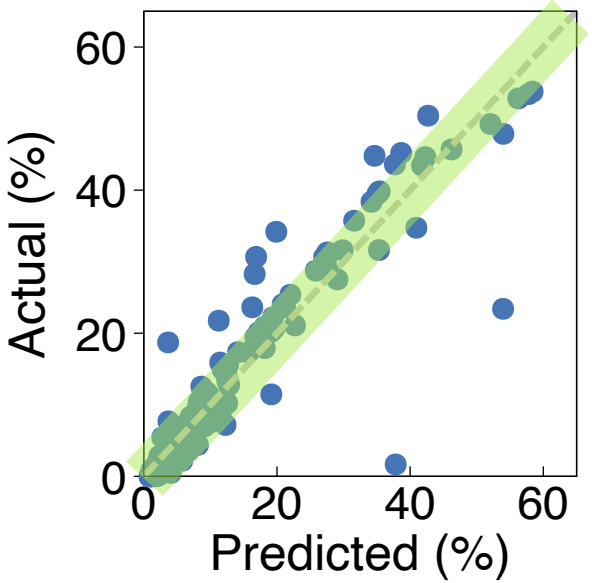
Apply the models to workload performance prediction

NUMA Correlation=0.97



88.4% workloads ≤ 5% abs. error

CXL Correlation=0.94



92.4% workloads ≤ 5% abs. error

265 workloads

- *CPU SPEC*
- *PARSEC*
- *GAPBS*
- *PBBS*
- *XSBench*
- *Phoronix*
- *Redis*
- *Spark*
- *VoltDB*
- *MLPerf*
- *Llama*
- *GPT-2*
- *DLRM*

Outline

Overview

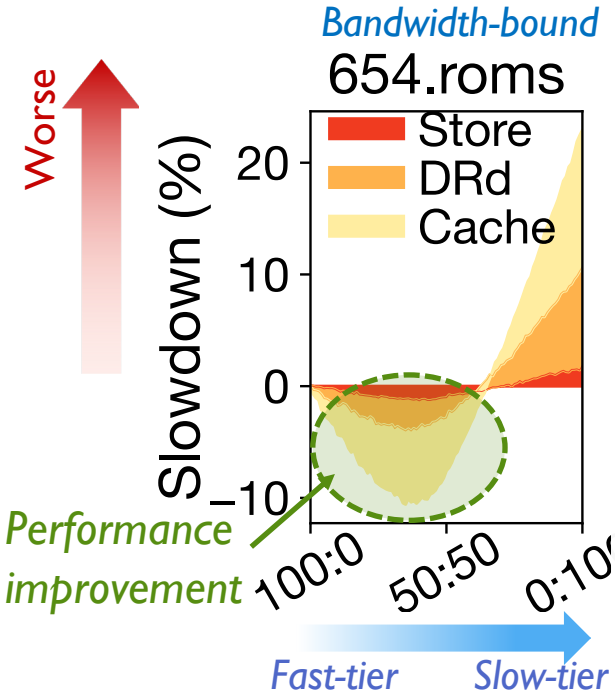
Slowdown Prediction

Synthesizing Interleaving Performance

Use Cases

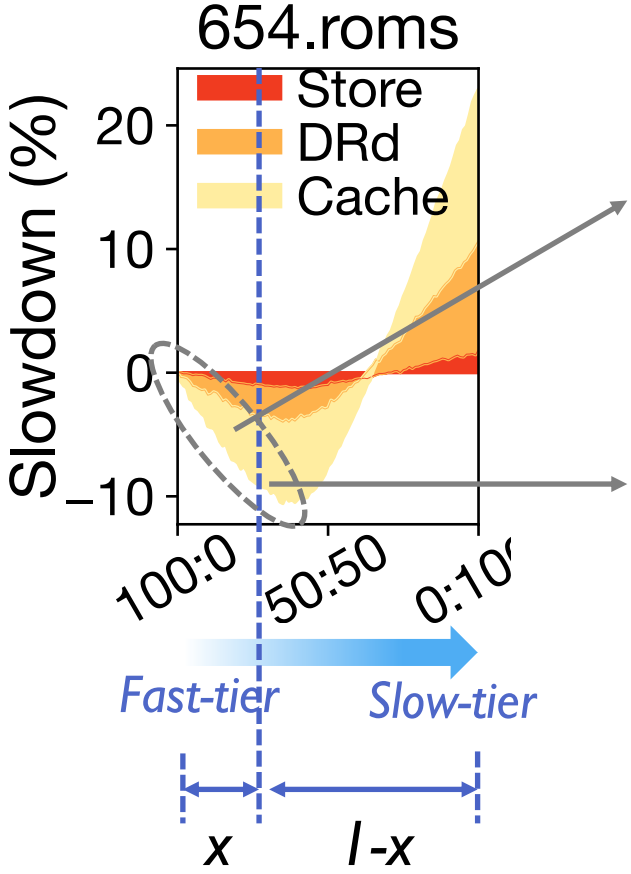
Interleaving Performance Spectrum

How is the continuous spectrum of performance under interleaving?



The performance slowdown under interleaving can still be attributed to 3 sources

Challenge



Allocating x of total memory footprint to a memory node does *not* lead to *proportionable* CPU cycles (or stalled cycles) with respect to the value of x.

What will be its performance at any given x?

Model for Demand Read (DRd)

$$S_{DRd} = \frac{\Delta stalls_{DRd}}{c} \approx \frac{\Delta C}{c}$$

Memory active cycles

# of data requests

Latency

$$C = \frac{N \times L}{MLP}$$

Memory-Level Parallelism

The variance of C is determined by how N, L and MLP change with x

Model for Demand Read (DRd)

# of data requests

$$C = \frac{N \times L}{MLP}$$

The # data requests to a memory node is proportional to x

$$N_x \propto x$$

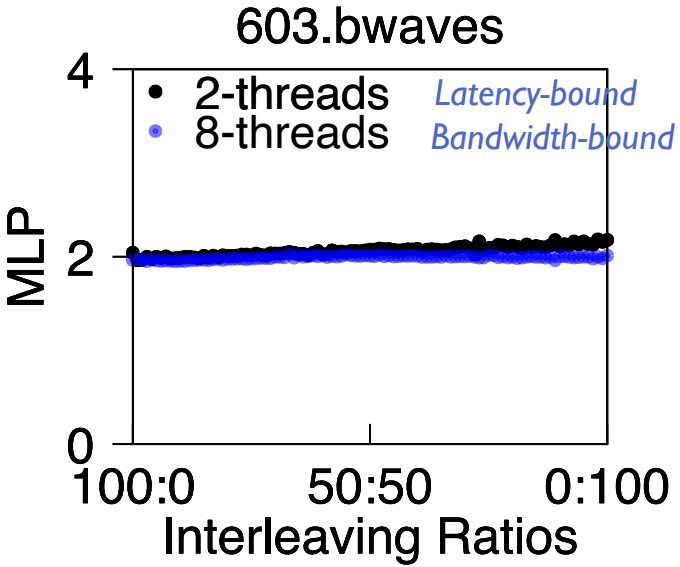
Model for Demand Read (DRd)

$$C = \frac{N \times L}{MLP}$$

Memory-Level Parallelism

MLP's variance is minimal no matter latency-bound or bandwidth-bound

MLP_x -> constant



Model for Demand Read (DRd)

$$C = \frac{N \times L}{MLP}$$

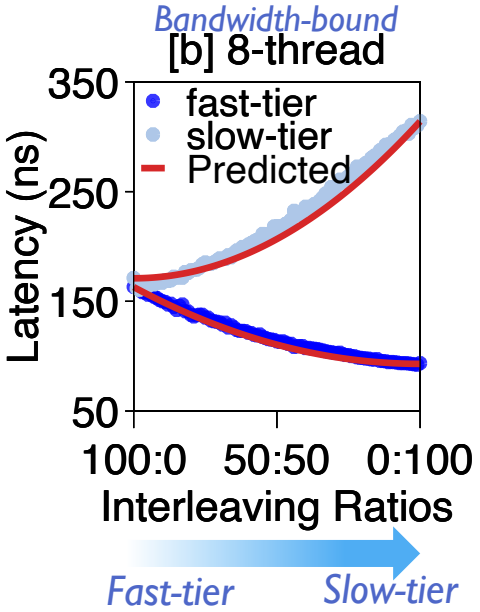
Latency

Latency changes as a quadratic curve

Loaded latency

$$L_x = L_{idle} + (L_{full} - L_{idle}) * x^2$$

Unloaded latency



$L_{idle} \approx L_{full}$ -> Latency maintains constant as x changes

Model for Demand Read (DRd)

$$S_{DRd} = \frac{\Delta stalls_{DRd}}{c} \approx \frac{\Delta C}{c}$$

Memory active cycles

# of data requests

$N_x \propto x$

$$C = \frac{N \times L}{MLP}$$

Latency

$L_x = L_{idle} + (L_{full} - L_{idle}) * x^2$

Memory-Level Parallelism

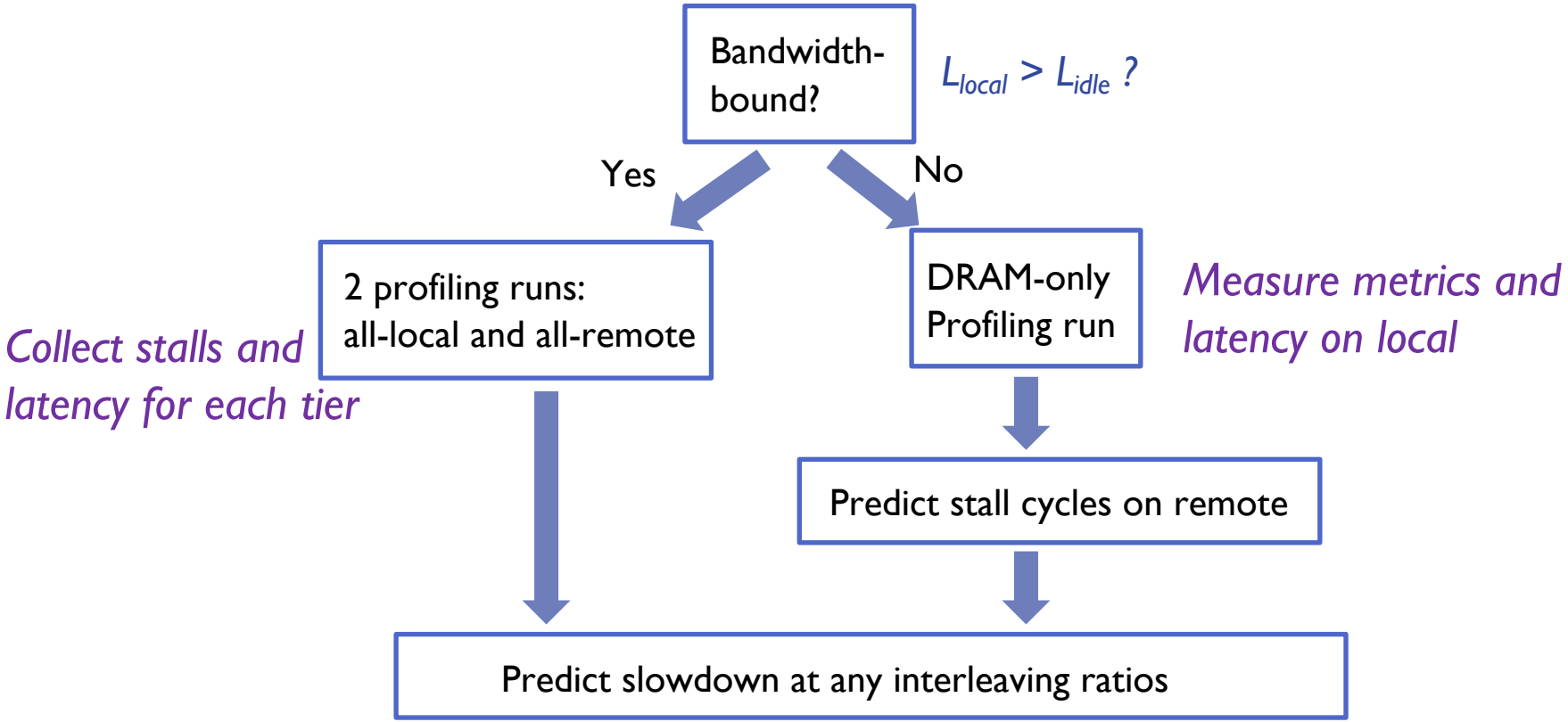
MLP_x -> constant

The variance of C can be computed by x, L_{full} and L_{idle}

$$\mathcal{M}(x') = \frac{x' \cdot [L_{\text{idle}} + (L_{\text{full}} - L_{\text{idle}}) \cdot x'^2]}{L_{\text{full}}}$$

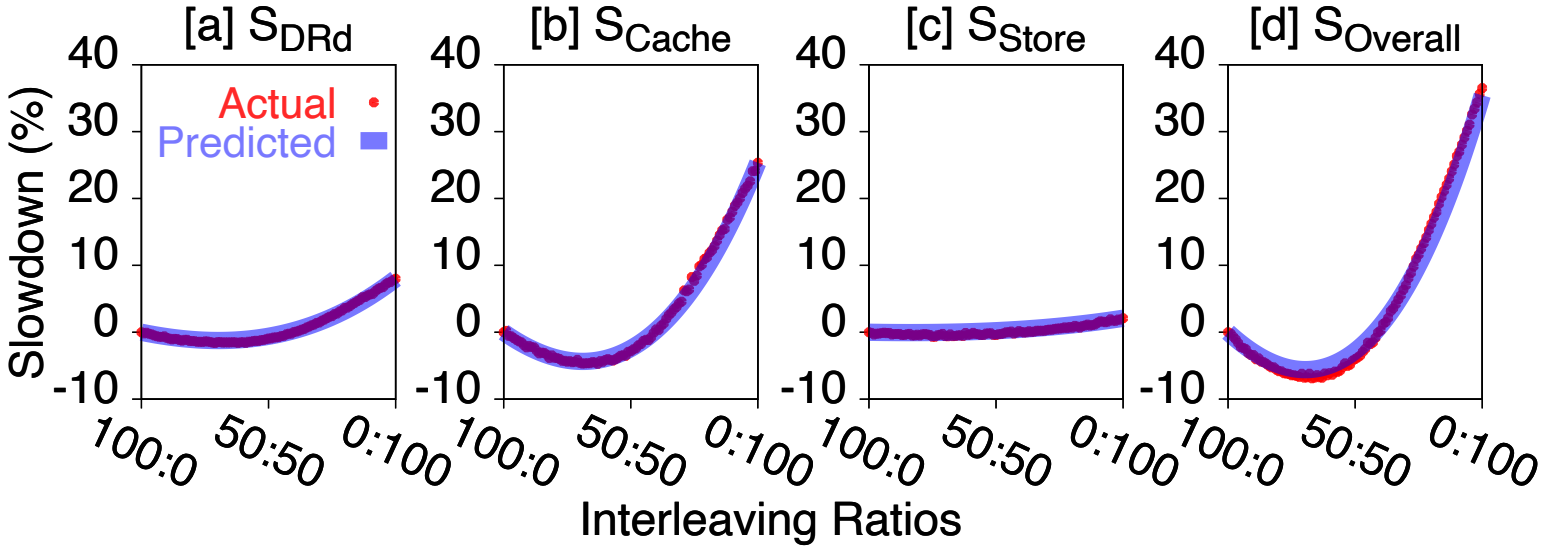
More in the paper: complete model

CAMP Interleaving Model Workflow



Interleaving Model Accuracy

Example: 8-thread 603.bwaves



Interleaving slowdown at any given ratio can be predicted accurately

Outline

Overview

Slowdown Prediction

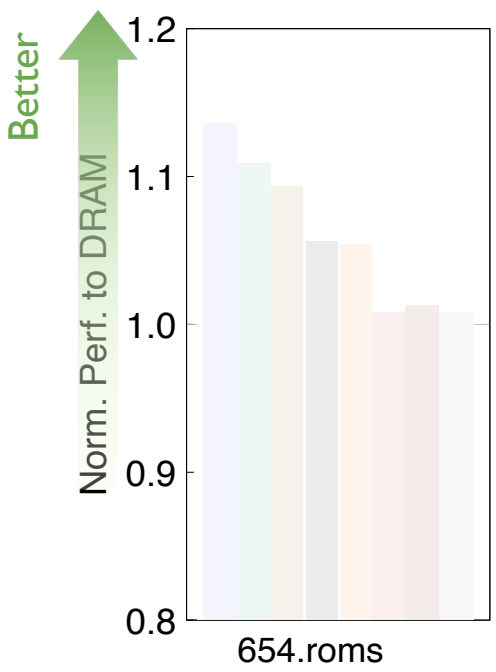
Synthesizing Interleaving Preformance

Use Cases

Best-shot Interleaving

Predict the optimal interleaving ratio for bandwidth-bound workloads

Compare with existing tiering polices (654.roms), fast-slow ratio: 4:1



Outperforms Caption and Interleave 1-1 by 3% and 4%

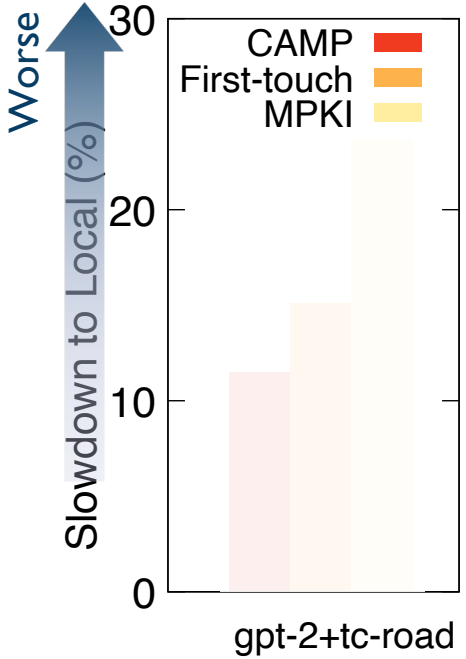
Outperforms first-touch by 8% due to maximize the aggregate bandwidth even though the fast-tier is not fully used

Outperforms tiering policies by up to 13% due to additional page migration overhead

More in the paper: other workloads

Workload Colocation

Colocate latency-bound workloads (example: gpt-2 and tc-road)



+12% performance vs. MPKI guided placement

Summary

CXL memory performance prediction from causal based analysis

Interleaving performance prediction

Use cases

- "Best-shot" interleaving
- workload colocation

Paper Code

https://github.com/MoatLab/CAMP

Thank you! Questions?

Performance Predictability in Heterogeneous Memory

Jinshu Liu
Virginia Tech
Blacksburg, USA

Hanchen Xu
Virginia Tech
Blacksburg, USA

Daniel S. Berger
Microsoft and University of Washington
Redmond, USA

Marcos K. Aguilera
NVIDIA
Santa Clara, USA

Huaicheng Li
Virginia Tech
Blacksburg, USA

Abstract

Heterogeneous memory combining DRAM and CXL exhibits variable performance, yet existing metrics correlate weakly with actual slowdown. We present Camp, a principled framework for predicting CXL-induced slowdown. Our key insight is that a DRAM run (plus a CXL run for bandwidth-bound workloads) exposes the causal microarchitectural pressure points where CXL latency translates into additional processor stall cycles. Camp captures these signals using 12 performance counters to analytically decompose slowdown into three orthogonal components: demand reads, cache/prefetching, and stores. Camp also introduces a closed-form model for software-based weighted interleaving that predicts performance across DRAM–CXL ratios. Across 265 workloads on NUMA and three CXL devices, Camp achieves 91–97% prediction accuracy within 10% absolute error. We demonstrate that these models enable practical system policies, including "Best-shot" interleaving and colocated workload placement, improving performance by up to 21% and 23% over existing tiering and colocation approaches.

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

Keywords: CXL Memory, Modeling, Prediction, Interleaving

ACM Reference Format:
Jinshu Liu, Hanchen Xu, Daniel S. Berger, Marcos K. Aguilera, and Huaicheng Li. 2026. Performance Predictability in Heterogeneous Memory. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '26), March 22–26, 2026, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3779212.3790201

1 Introduction

Modern servers increasingly adopt heterogeneous memory architectures, combining fast local DRAM with larger, slower tiers such as NUMA and Compute Express Link (CXL) attached memory. While this disaggregation expands capacity cost-effectively, it introduces significant sensitivity in application performance. Misplacing latency-sensitive data onto slower tiers can trigger severe pipeline stalls, waste scarce DRAM bandwidth, and violate Service Level Objectives (SLOs) [18, 34, 36, 37, 46].

Effective management of these systems therefore requires *predictive visibility:* the ability to quantify exactly how a placement decision will impact application slowdown *before* that decision is enacted. Without accurate predictors, cloud operators are forced to overprovision DRAM to minimize risk, while runtimes and operating systems rely on reactive migration loops that waste resources correcting placement errors after performance has already degraded.

Despite extensive prior work, a gap remains between identifying performance signals and predicting slowdown. Profiling tools identify architectural events correlated with performance [26, 30, 36, 40, 56], but stop short of quantitative slowdown forecasts. Runtime systems employ heuristics, such as access frequency, LLC misses, latency, or stall cycles to guide tiered placement [24, 27, 38, 45, 51, 59]. However, these approaches are inherently *reactive* (detecting degradation post facto) or *descriptive* (attributing observed degradation to causes). For example, Melody [36] provides a robust framework for decomposing slowdown into components, but it is an *attribution* tool: it requires execution on both DRAM and CXL to explain the past. Similarly, SoarAlto [38] uses Memory-Level Parallelism (MLP) as a reactive metric to model demand-read-induced slowdown for tiering decisions, but does not offer a forward-looking model to predict overall slowdown or synthesize optimal interleaving ratios a priori.

This work introduces Camp[1], a framework that bridges this gap by transforming *offline attribution* into *prediction.* We address a fundamental question: *Can we predict how a workload will perform on CXL or under weighted interleaving across DRAM/CXL [16] using intrinsic workload signatures?*