Resource-Complementary Scheduling on Datacenter Shared Servers

Yibo Yan [1] Zixuan Peng [2] Ramesh Govindan [1] Seo Jin Park [1]

[1] USC [2] UW-Madison

USC Viterbi Thomas Lord Department of Computer Science

nsdi '26

Ongoing Work **Want Feedback!**

1 The Problem

Tension: Resource Utilization ↔ Co-location Interference

State-of-the-art schedulers (like Caladan[1]) mitigate this reactively:

➪ Throllting best effort (BE) app upon contention

➪ Protect latency-critical (LC) apps

We explore: proactively pairing using non-conflicting resources: cores, cache lines, and nemory bandwidth.

> **KEY INSIGHT**
>
> *Slowdown caused by microarchitectural interference can be systematically predicted by modeling memory-stall cycles from hardware counters.*

2 The RCS Model

Resource-complementary Scheduling (RCS) model:

➪ Predicts task slowdown under a forcasted system bandwidth load

➪ Given task microarchitectural characteristics

Overview: How to predict the slowdown?

Step 1: Pred Mem Stall Δ%

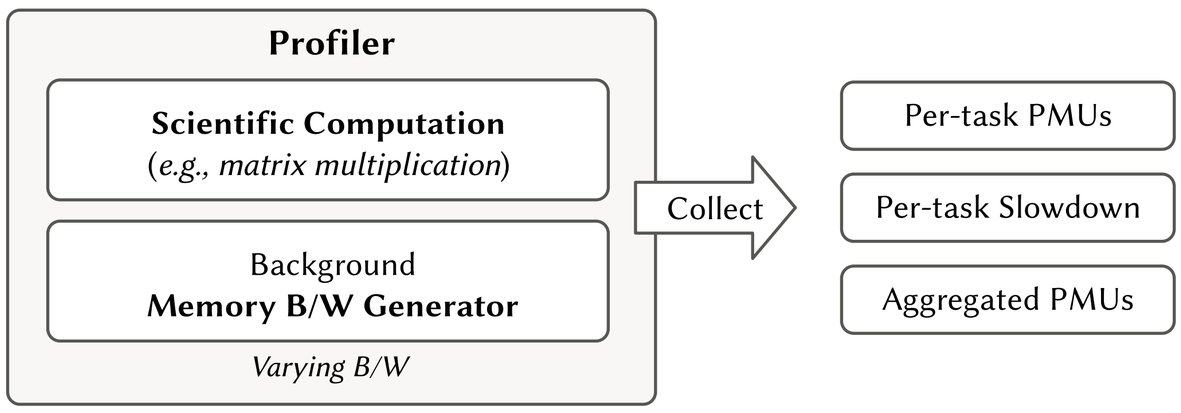
Step 2: Pred Slowdown via Mem Stall Δ%

How many additional cycles stalled due to memory access under interference (ratio over the total cycles)

Insight: Mem stall Δ% shows a strong linearity with slowdown.

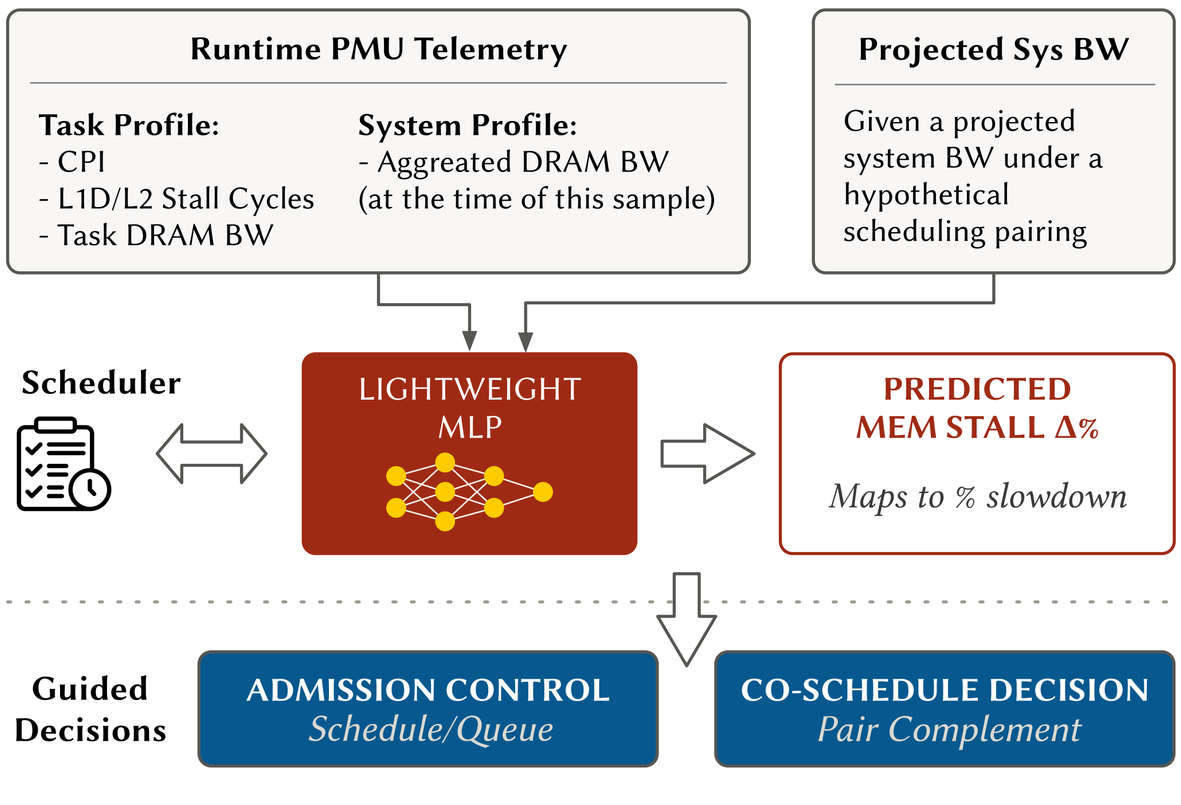
Approach

◇ Build the model: *Offline Microarchitectural Profiling*



Use offline data to train a lightweight MLP model to predict mem stall Δ%

◇ Use the model: *Online Decision Making*



3 Memory Stall Δ% Prediction Accuracy

Trained on PolyBench[3] (30 scientific computation) + 5 additional CPU-bound computations (e.g., prime number and floating-point calculation).

Each computation:

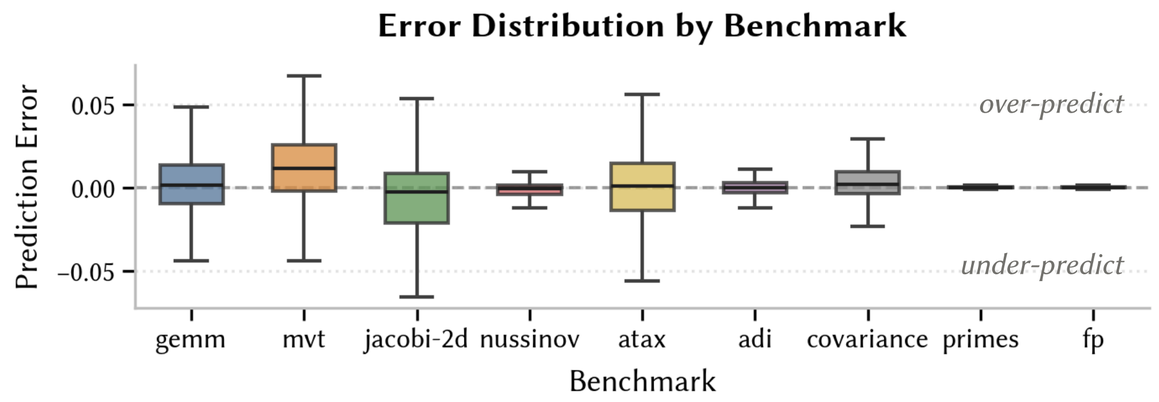
➪ 7 sizes of working set (fits into L1 ⟶ has go to DRAM).

➪ 12 interference levels (no interference ⟶ BW saturation).

➪ Sampled 20 data-point for each (computation × size × interference) pair.

Total corpus size: 35 × 7 × 12 × 20 = 58,800

Accuracy Statistics	R^2	RMSE	MSE	Prediction Error mostly within ± 5%
	0.97	0.023	0.0005	



Challenge: Same computation may have entirely different resource usage profile under different working-set sizes.

Results: Our model generalizes well in both:

➪ different computations and

➪ different working-set sizes for the same computation.

Compute-Bound: gemm, primes, fp **Memory-Bound**: mvt, jacobi-2d, atax

Mixed: nussinov, adi, covariance (compute-bound at large working-set)

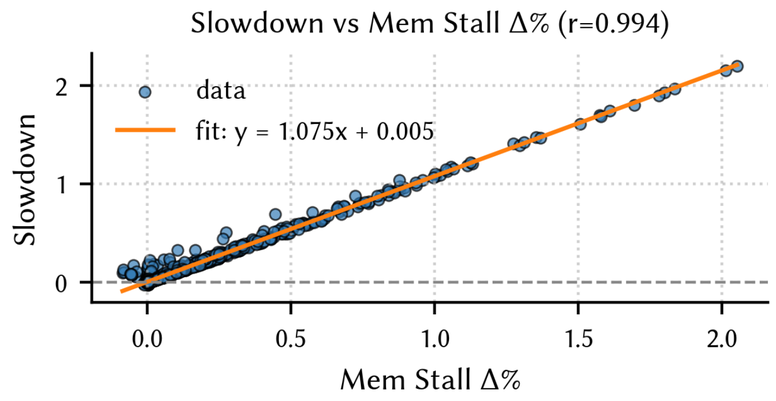
◇ Model prediction latency

Batch prediction: 25ms (8306 records)

Single prediction: 107 us

Possible for ms-scale scheduling.

4 Slowdown via Memory Stall Δ%



Strong linearity
Strong correlation

Easy to map from predicted mem stall Δ% to predicted slowdown

5 Ongoing Work

Integrating RCS models into user-level scheduling frameworks, such as Caladan[1] and Skyloft[2].

Open Challenges:

➪ How to use this model efficiently to support millisecond-scale scheduling? Each prediction still takes ~100us.

➪ How to make this more generalized to different systems?

[1] Joshua Fried, Zhenyuan Ruan, Amy Ousterhout, and Adam Belay. Caladan: Mitigating interference at microsecond timescales. OSDI'20.

[2] Yuekai Jia, Kaifu Tian, Yuyang You, Yu Chen, and Kang Chen. Skyloft: A general high-efficient scheduling framework in user space. SOSP'24.

[3] Matthias Reisinger. MatthiasJReisinger/PolyBenchC4.2.1, March 2026.