Pond: CXL-Based Memory Pooling Systems for Cloud Platforms

Huaicheng Li, *Daniel S. Berger*, *Lisa Hsu*, Daniel Ernst, *Pantea Zardoshti*, *Stanko Novakovic*, *Monish Shah*, *Samir Rajadnya*, *Scott Lee*, *Ishwar Agarwal*, Mark D. Hill, *Marcus Fontoura*, *Ricardo Bianchini*

ASPLOS'23

VIRGINIA TECH

Microsoft

UNIVERSITY of WASHINGTON

WISCONSIN

The Public Cloud Memory Setting

- Cloud providers targets
  - ***Performance*** comparable to on-premise datacenters
  - Competitive hardware ***cost***
- Memory allocation gold standard
  - VM memory is ***statically pinned*** to the ***same NUMA node***
- DRAM is costly
  - ***~50%*** **of the server cost** for Azure!

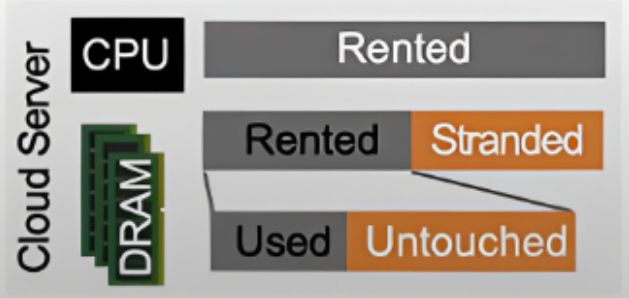
The Public Cloud Memory Setting

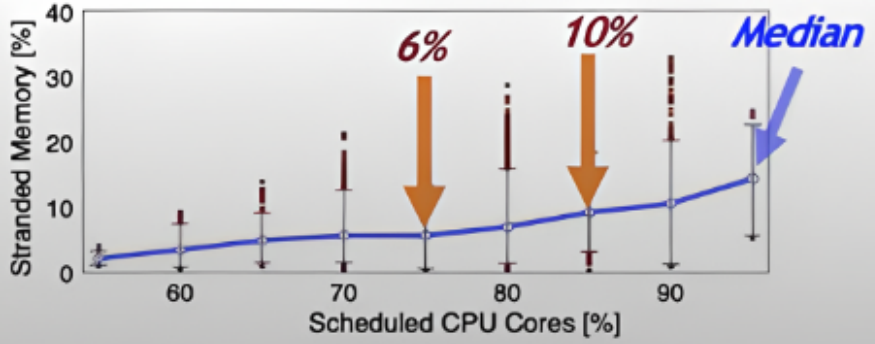
- Cloud providers targets
  - Performance comparable to on-premise datacenters
  - Excellent performance for opaque VMs at a competitive hardware cost
- Memory allocation gold standard
  - VM memory is statically allocated and pinned to the same NUMA node
- DRAM is costly due to its poor scaling properties
  - ~50% of the server cost for Azure!

Stranded and Untouched Cloud Memory

- Memory stranding
  - No free CPU cores but memory left
  - Up to 25% stranded memory at 95th percentile

- Untouched memory due to overprovisioning
  - 45% of untouched memory for half of the VMs





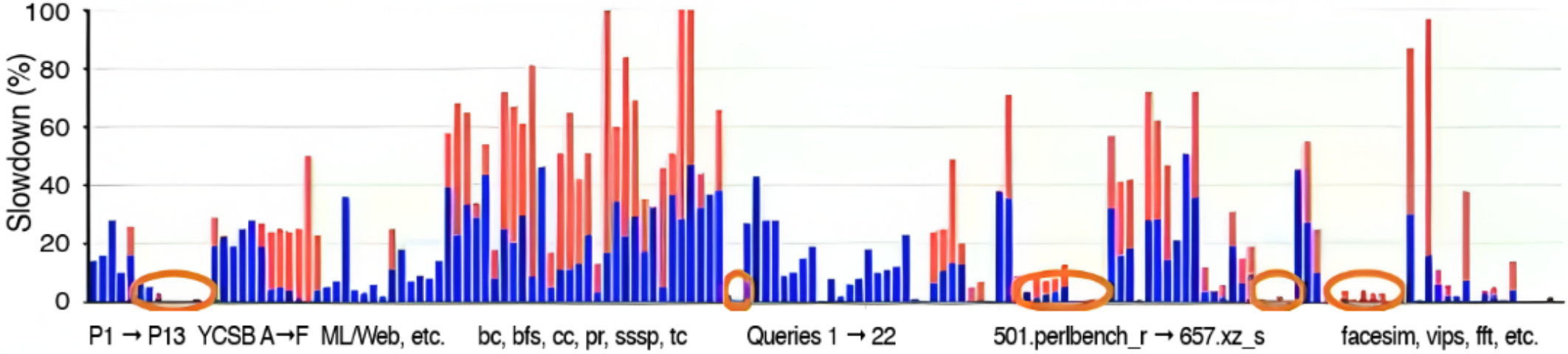
Pooling via CXL: Opportunities and Challenges

- CXL enables practical and performant pooling ☺
  - *Load/Store* access over PCIe 5.0 ("*CXL.mem*" protocol)
  - More practical than RDMA-based disaggregation designs
- CXL has *higher access latency than local DRAM* ☹
  - CPU-less node with **additional 70~90ns (~2x)**
  - CXL switches are slow and will add more latencies
  - ***Latency-sensitive workloads will suffer from CXL latencies***

Local DRAM (~90ns) + CXL (70~90ns)

CXL Performance Impact

Approximated CXL latencies: 142ns (182%), and 255ns (222%)



158 workloads: Proprietary, Redis, VoltDB, Spark, GAPBS, TPC-H, SPEC CPU 2017, etc.

(a) A small fraction of workloads are *not* sensitive to CXL latencies

(b) ~60% of the workloads see more than 5% slowdowns

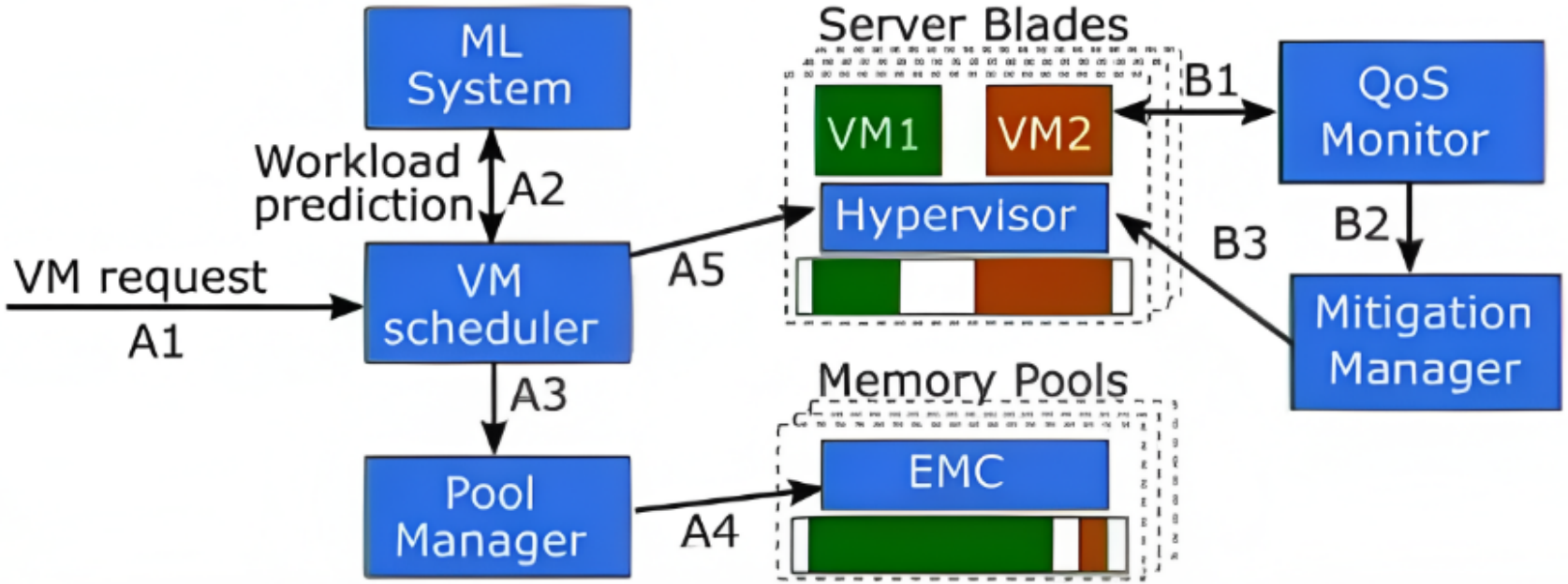
(c) Latency-sensitive workloads see *bigger* impact under higher CXL latencies

How to pool stranded and untouched memory via CXL for efficiency without sacrificing (much) performance at scale?

e.g., 95% of NUMA-local VM performance

Idea: *Predict the amount of VM memory that can be safely allocated from the pool while satisfying the performance requirement via QoS monitoring.*

Pond: A Full-Stack Memory Pooling System



Pond contributions:

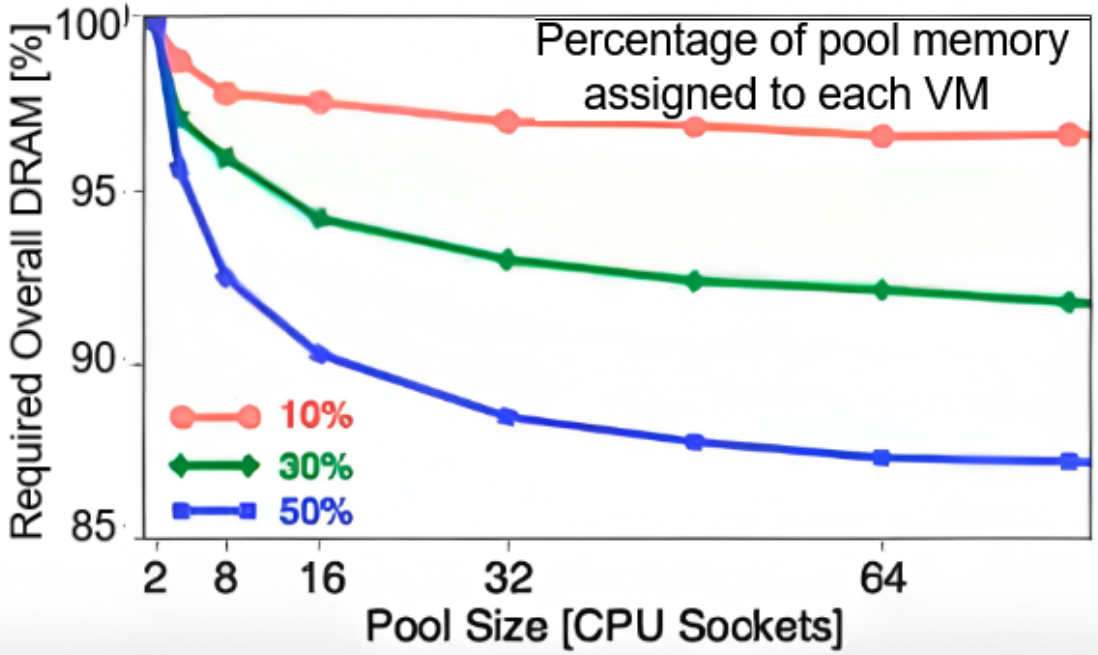
Hardware, system software, and distributed system layers to manage pooled memory

Pond benefits:

Reduce DRAM needs by 7% with a small pool → 3.5% reduction in cloud server cost

- Background & Motivation
- Pond Design
  - Overview
  - Memory pool scope
  - zNUMA
  - Prediction-assisted VM memory allocation
- Evaluation
- Conclusion

Pond Small Low Latency Pool

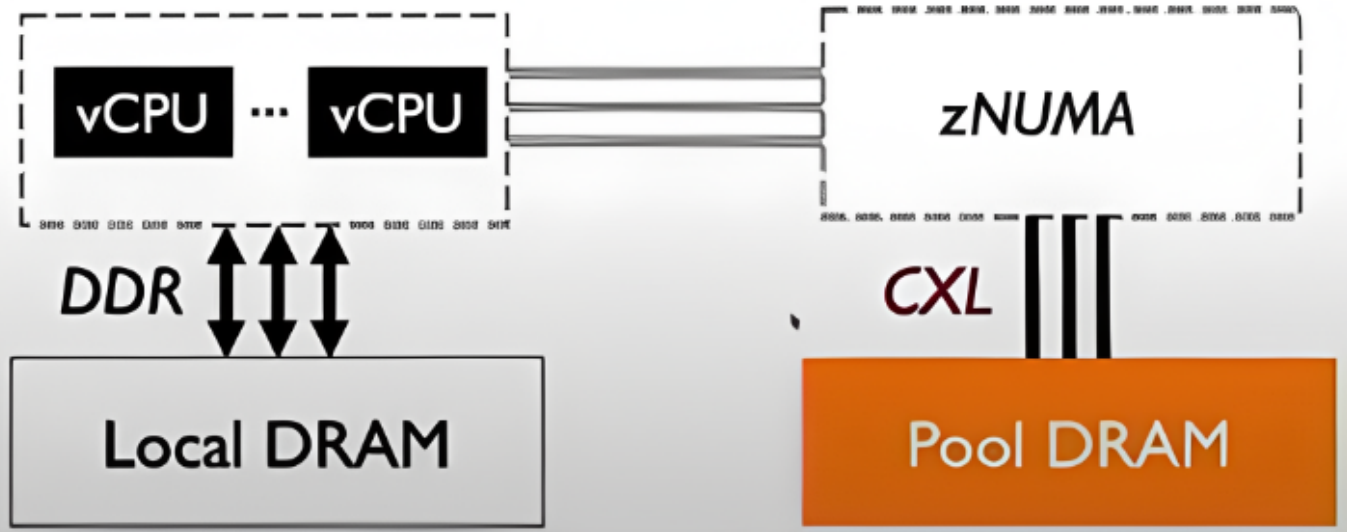


Small pools are effective!

While larger pools get diminishing returns.

CXL Memory as a zNUMA Node to the VMs

- *Zero-core NUMA (zNUMA)*
- Funneling VM memory accesses by reusing existing OS memory management schemes (local-memory preference)
- No spilling under correct predictions



Pond Prediction-based VM Memory Provisioning

if (workload latency insensitive)

Entire pool/CXL DRAM

else if (no untouched memory)

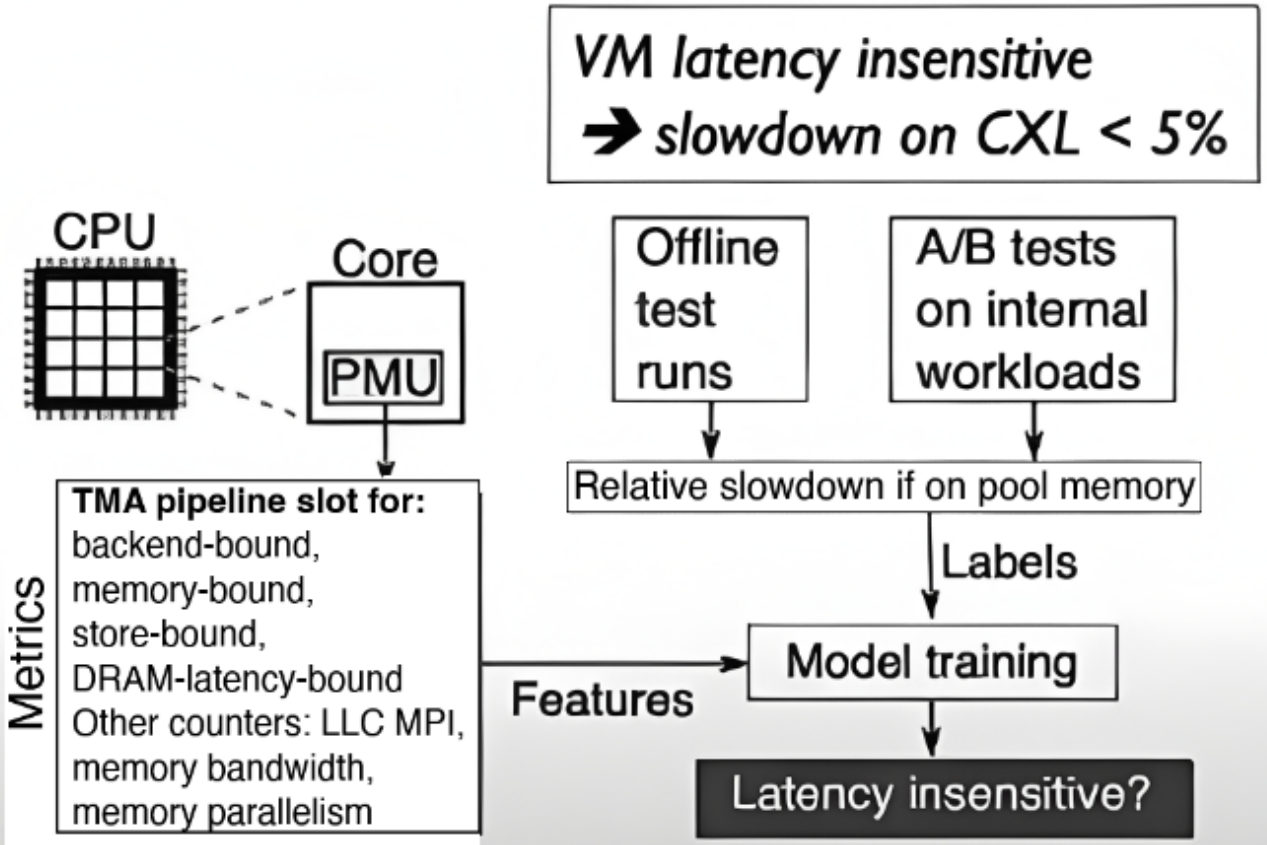
Entire local DRAM

else

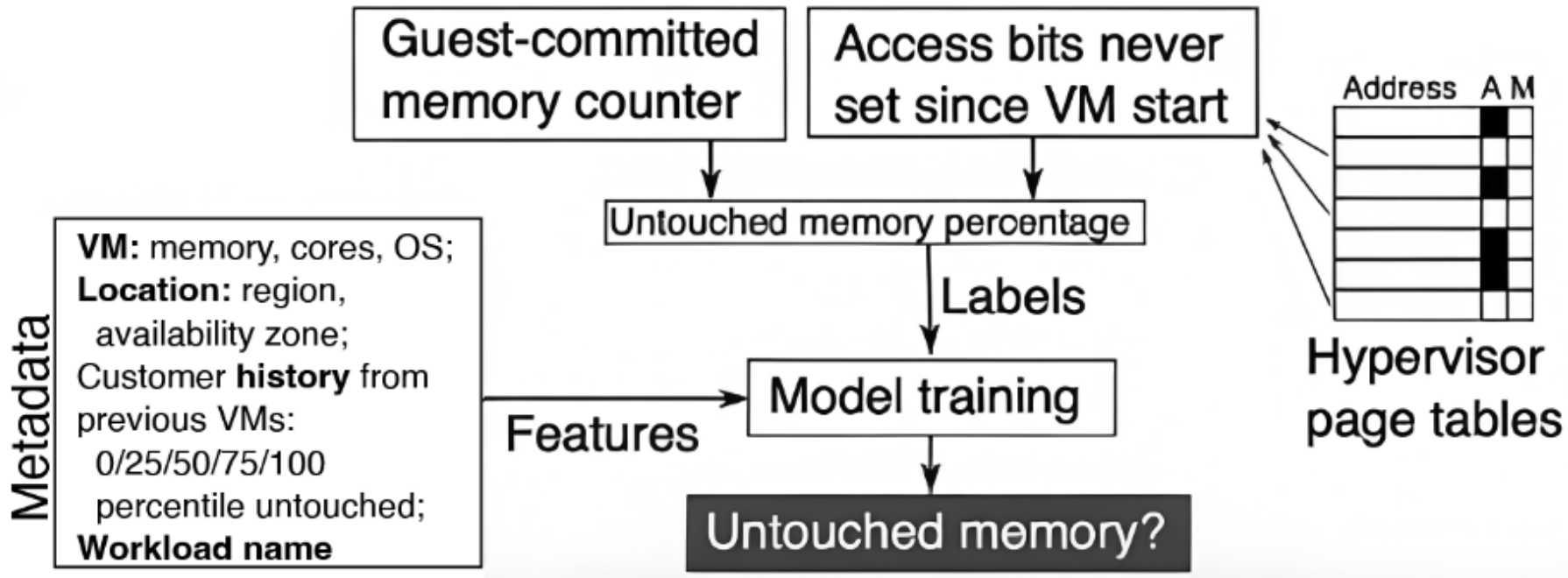
zNUMA: Pool DRAM = Untouched

Latency Sensitivity Prediction

Features from opaque VMs
➔ *existing HW counters*

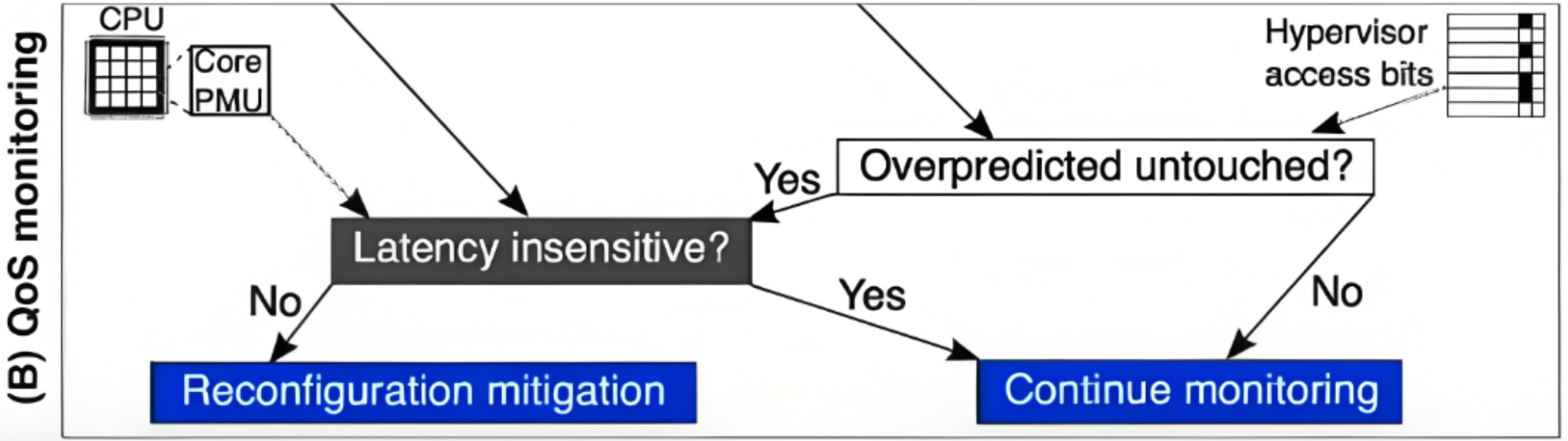


Untouched Memory Prediction



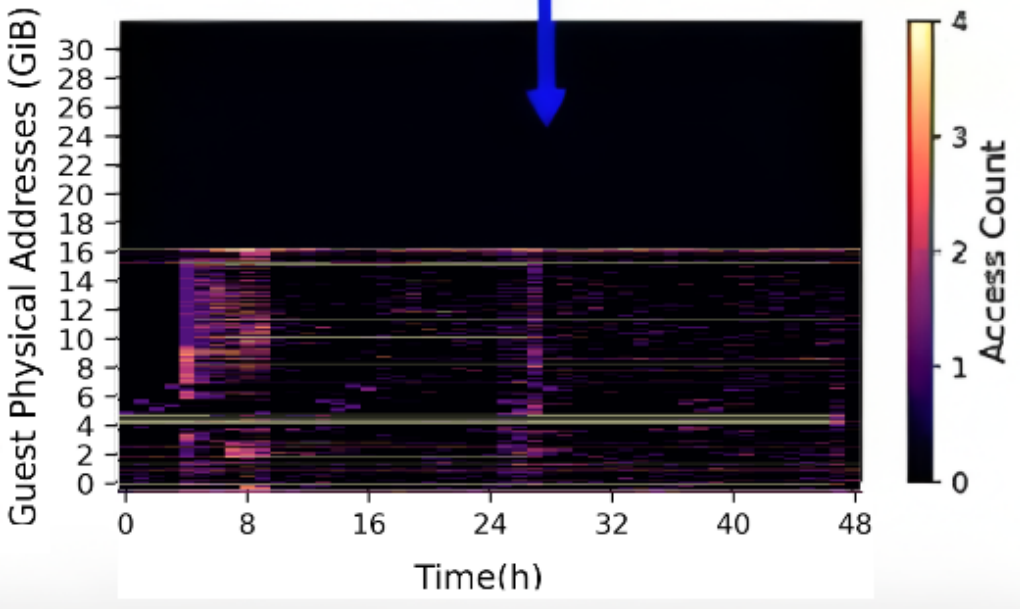
Prediction target: the amount of untouched memory (GB)

Misprediction Handling



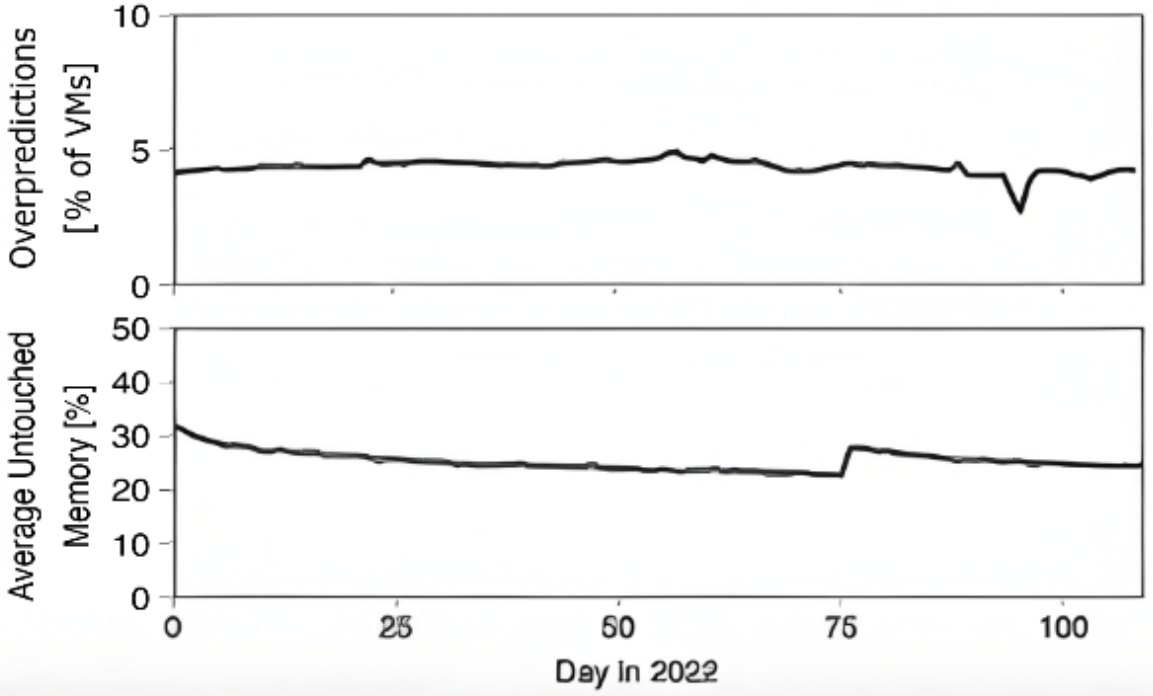
zNUMA Effectiveness

Workloads	zNUMA traffic
Video	0.25%
Database	0.06%
KV store	0.11%
Analytics	0.38%



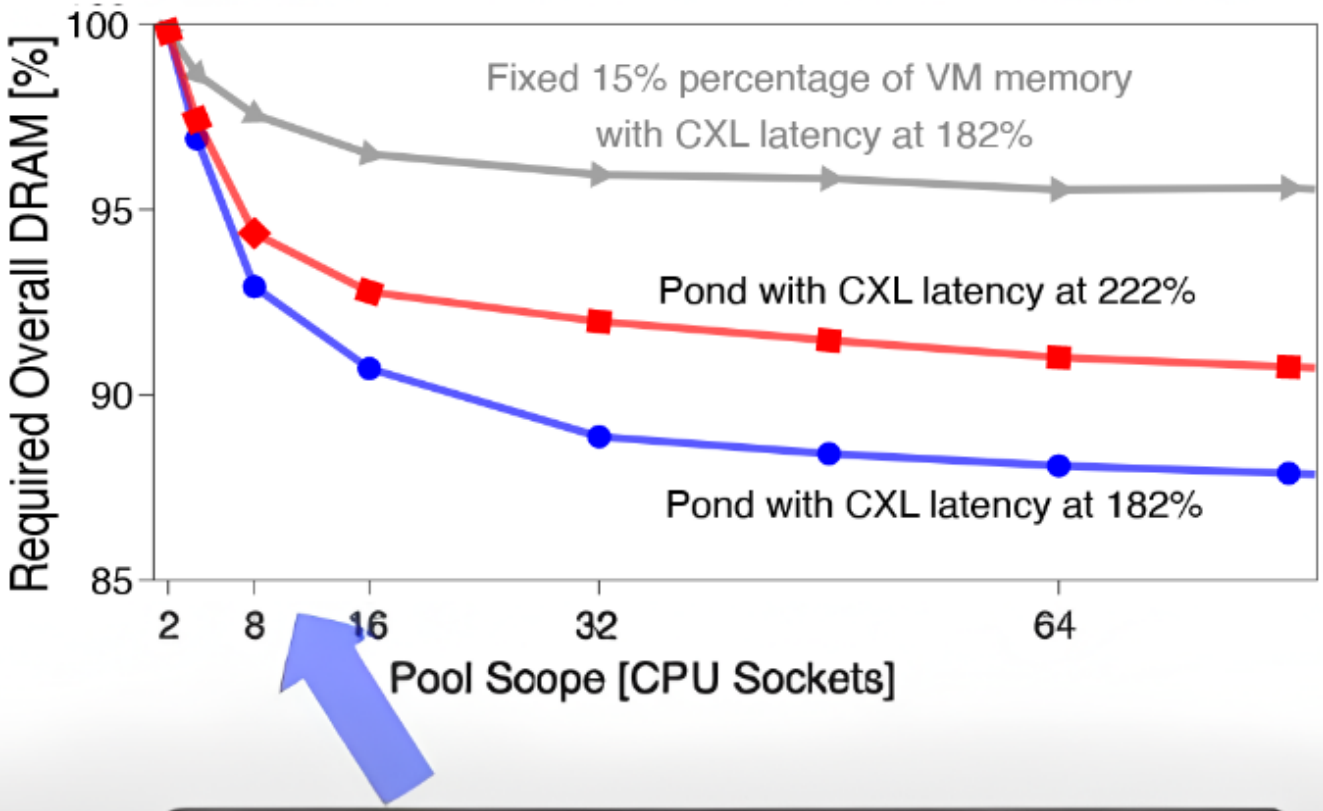
zNUMA is effective for correct predictions

Pond Prediction Model Accuracy



Pond prediction model identifies 25% of untouched memory while only overpredicting 4% of VMs

Pond End-to-end Memory Savings



8-16 socket pool, 7-9% DRAM savings

Configured to target <5% slowdown for 98% of VMs

Pond Summary

Applicable for locally attached CXL

Feasible hardware implementation

Close to local DRAM performance

7-9% DRAM savings by pooling
~25% untouched memory