Pond: CXL-Based Memory Pooling Systems for Cloud Platforms

Huaicheng Li[1], Daniel S. Berger[2,3], Lisa Hsu, Daniel Ernst[2], Pantea Zardoshti[2], Stanko Novakovic, Monish Shah, Samir Rajadnya[2], Scott Lee[2], Ishwar Agarwal, Mark D. Hill[2,4], Marcus Fontoura, Ricardo Bianchini[2]

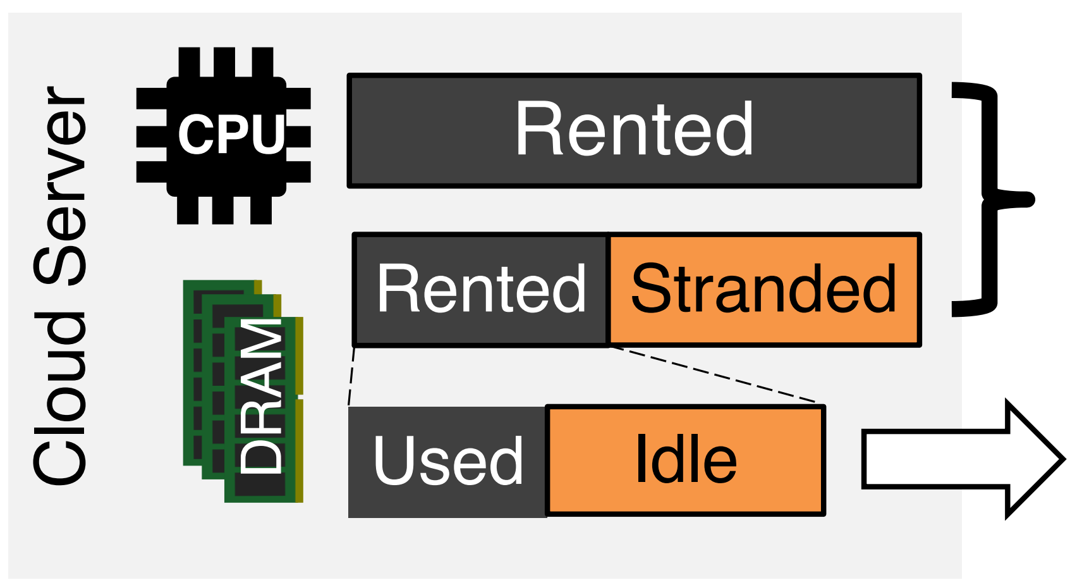
1 VIRGINIA TECH 2 Microsoft 3 UNIVERSITY of WASHINGTON 4 WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Public clouds spend ~50% on memory & much is wasted.
Pond pooling with fast CXL saves 7-9% memory.

The Need for Memory Pooling

(1). DRAM is a major server **cost**: Azure (**50%**)

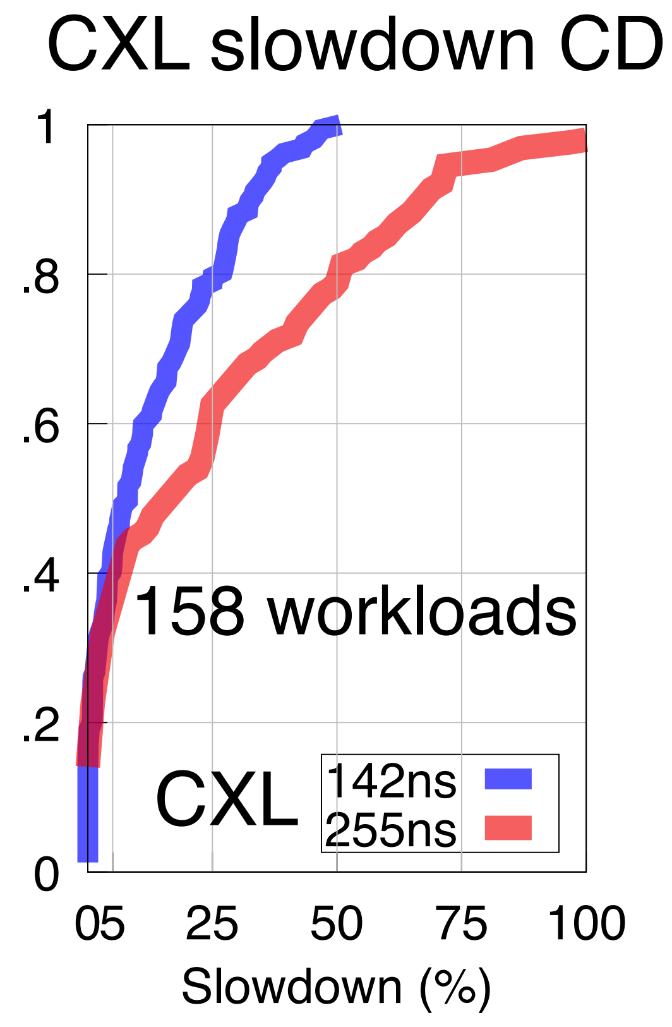
(2). Memory stranding and untouched memory

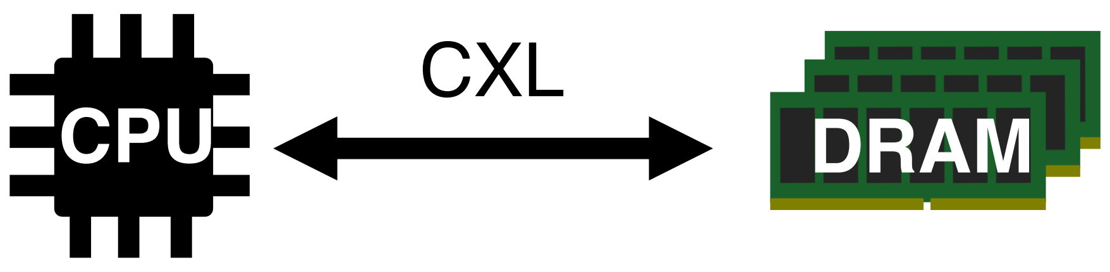


Up to **25%** stranded memory

Half of the VMs have **50%** of unused memory

Naive CXL Pooling is Inefficient





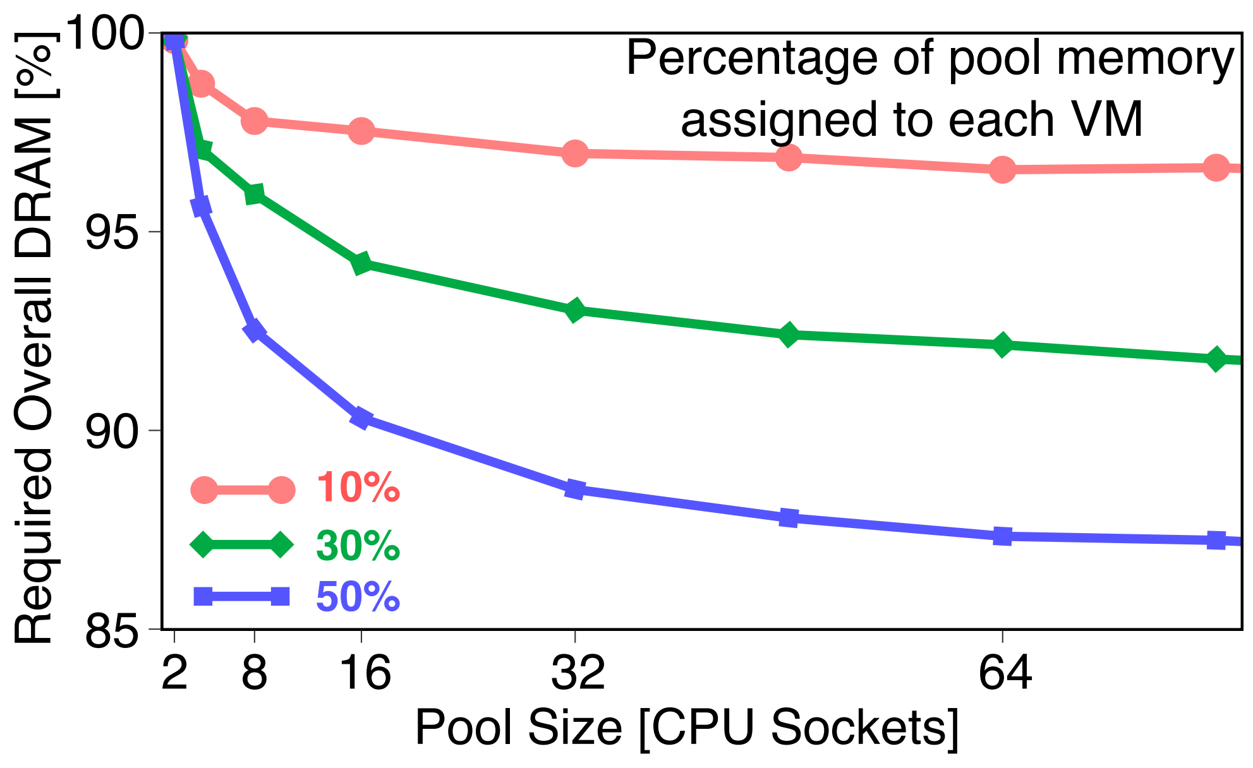
❑ **CXL latency:** one extra NUMA hop

❑ Workloads suffer from significant performance slowdowns under CXL

How to pool stranded memory via CXL while targeting NUMA-local performance?

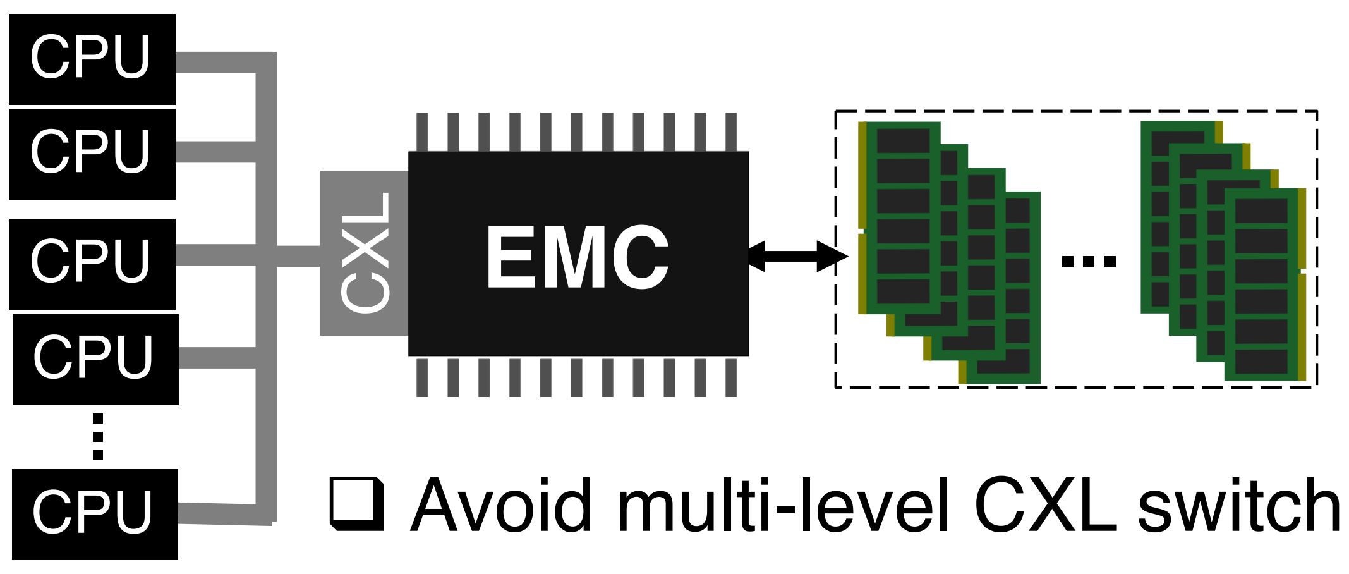
Pond: An End-to-End CXL-Based Pooling Design for Datacenters

(1). A small low-latency memory pool design



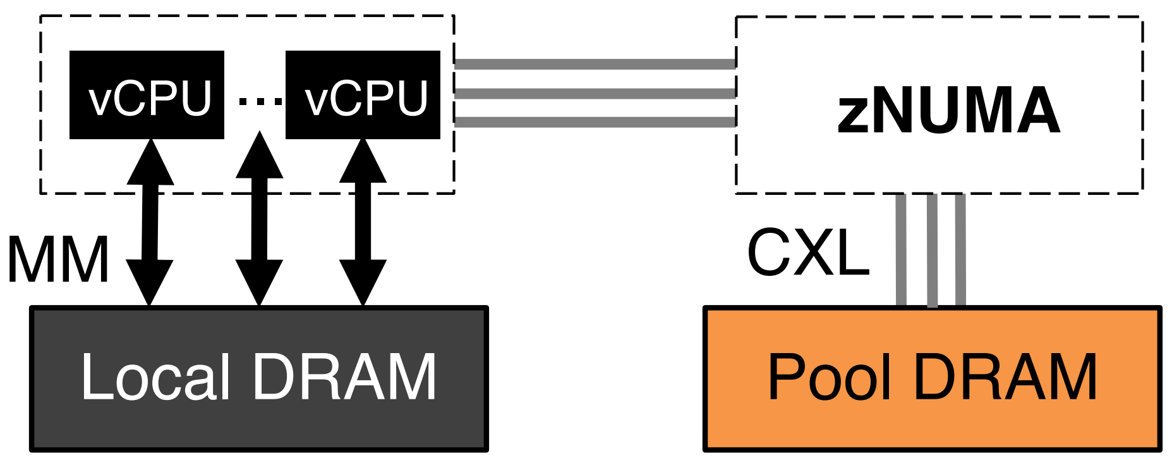
❑ **A small (8-16 sockets) pool is enough!**

(2). External memory controller (EMC)

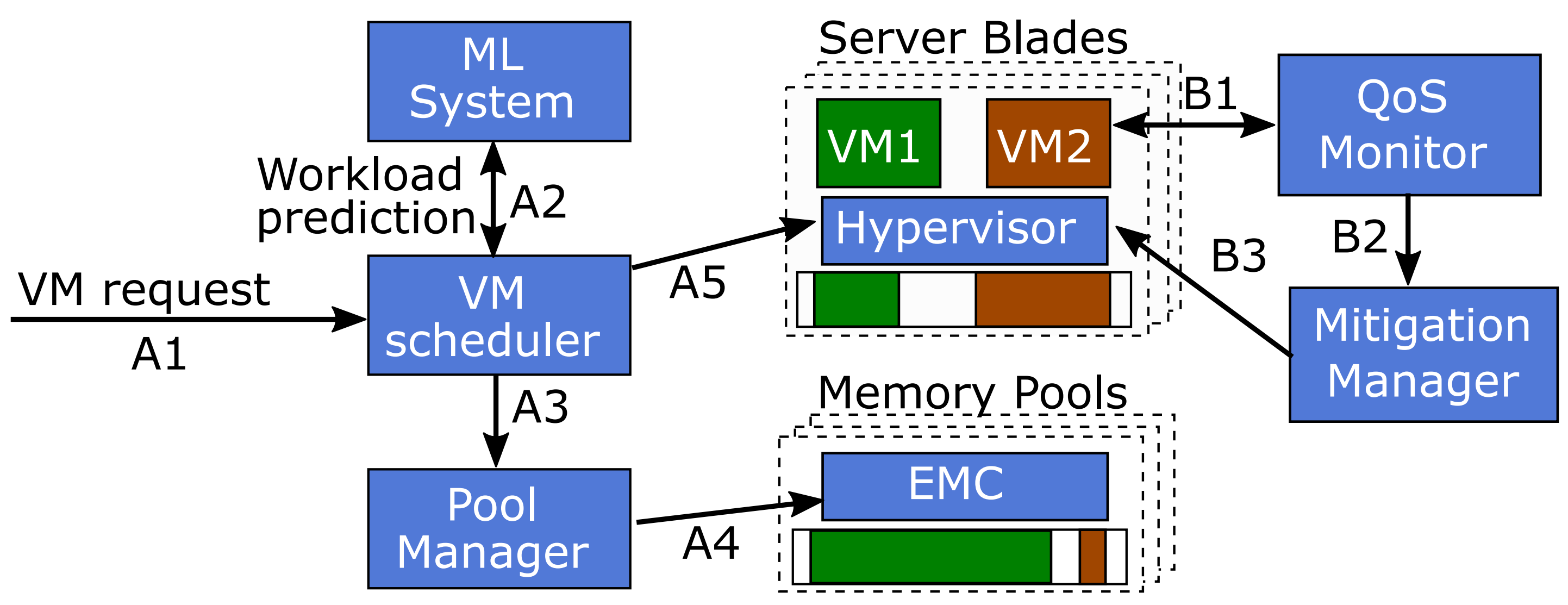


❑ Avoid multi-level CXL switch latencies

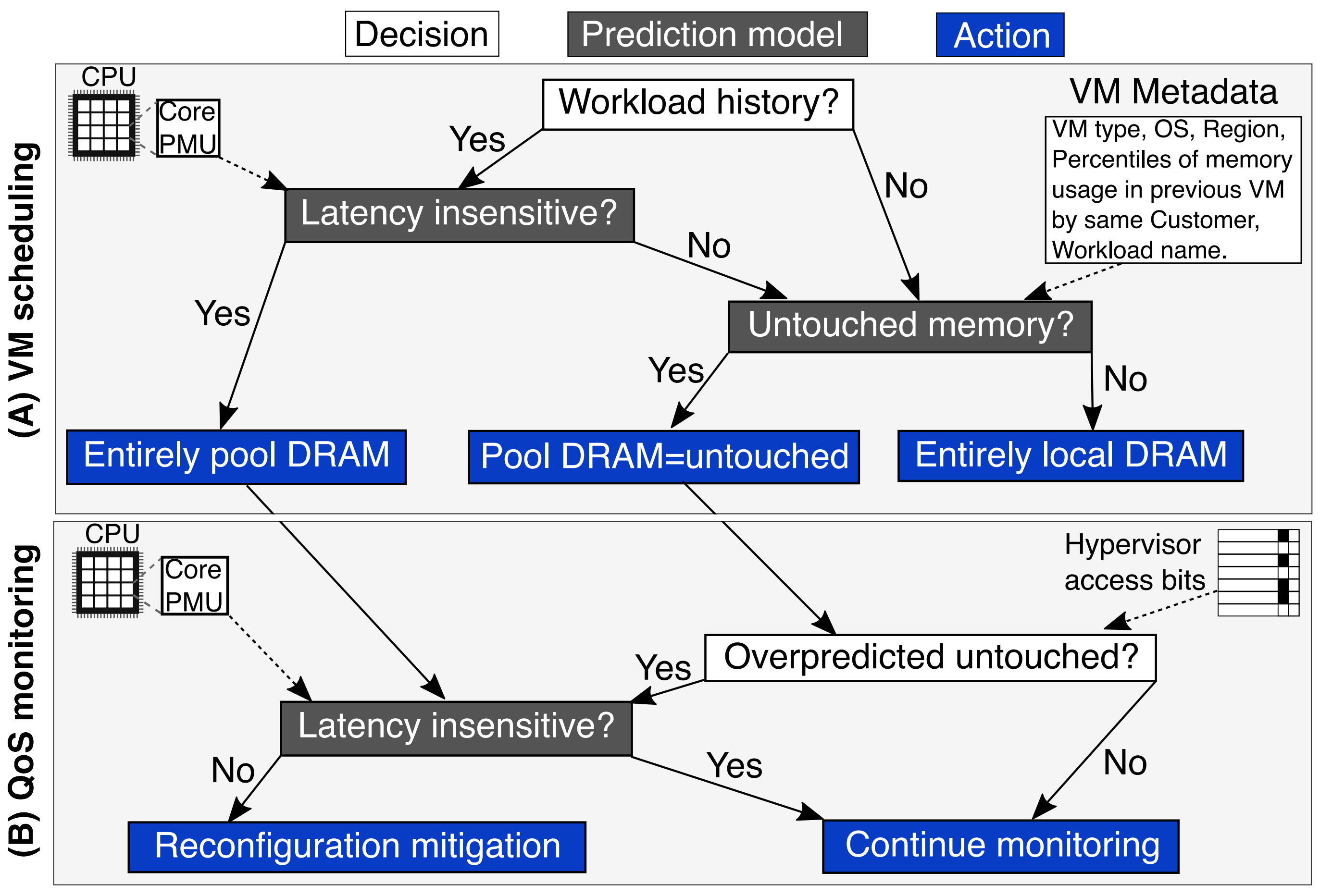
(3). zNUMA: zero-core NUMA for VMs



(4). Pond control plane



(5). Pond prediction models

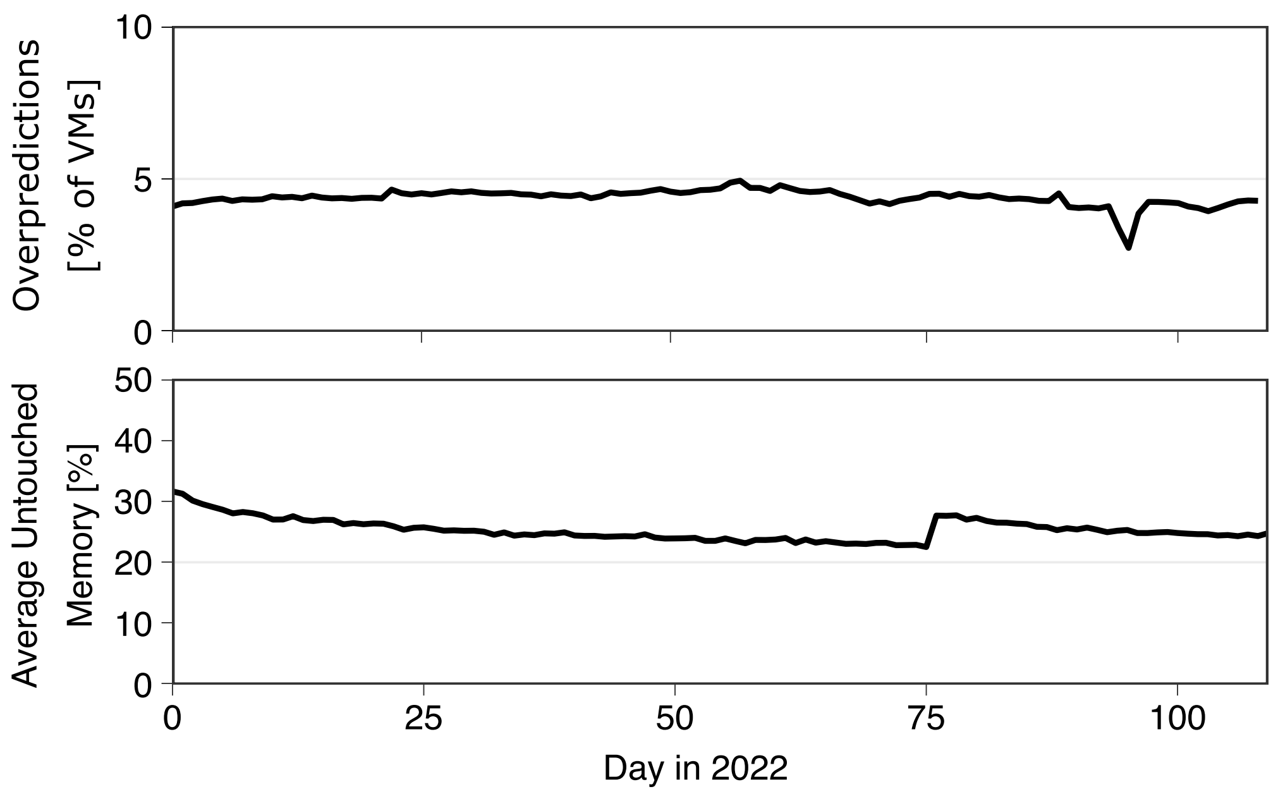


Pond Design is Effective in Saving 7-9% DRAM Needs

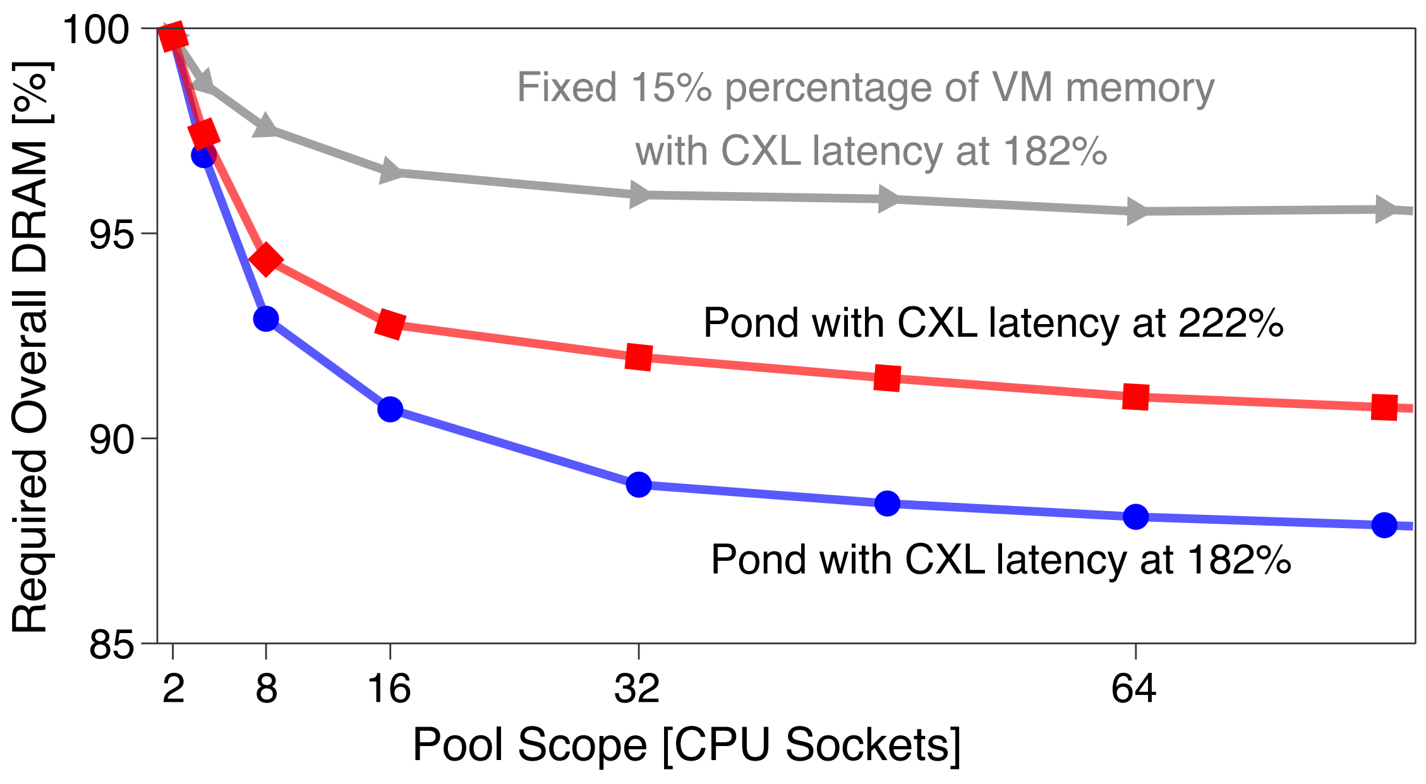
(1). zNUMA is effective

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

(2). Pond overpredicts 4% of VMs



(3). Pond saves 7-9% DRAM



Appears in the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'23)