I/O Coordination for Better Resource Sharing

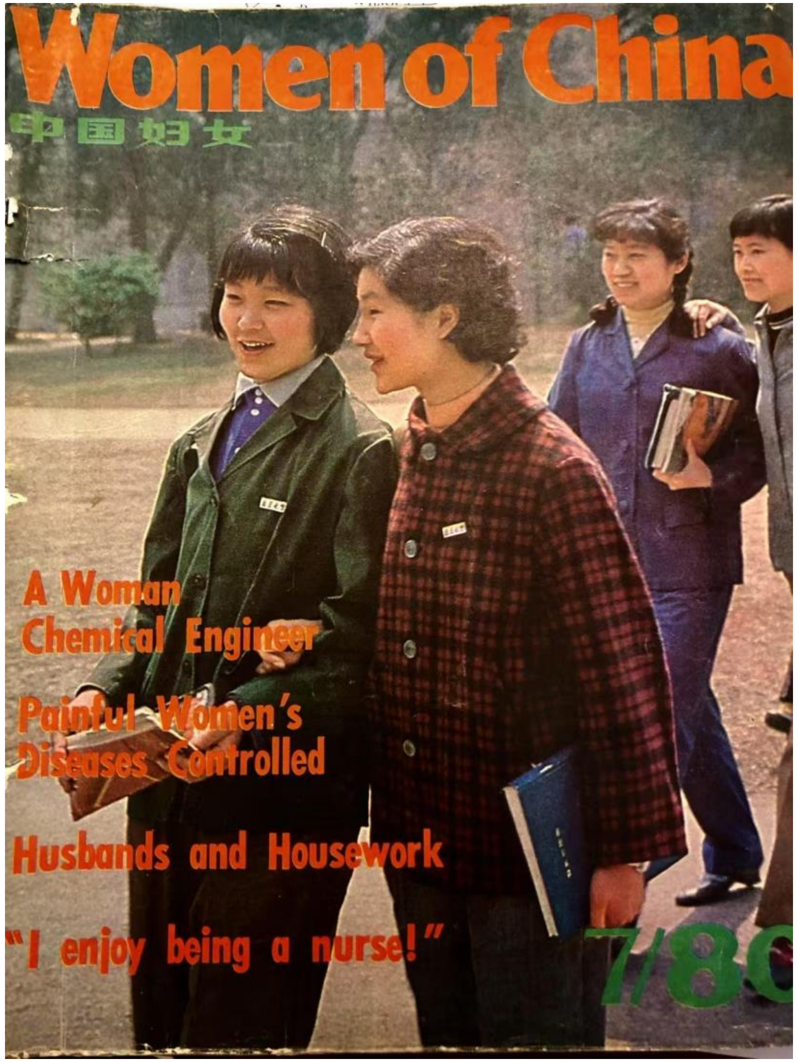
From HPC to AI Storage

Xiaosong Ma

Department of Computer Science, MBZUAI

HotStorage 2025

Decades Ago



Women of China

中国妇女

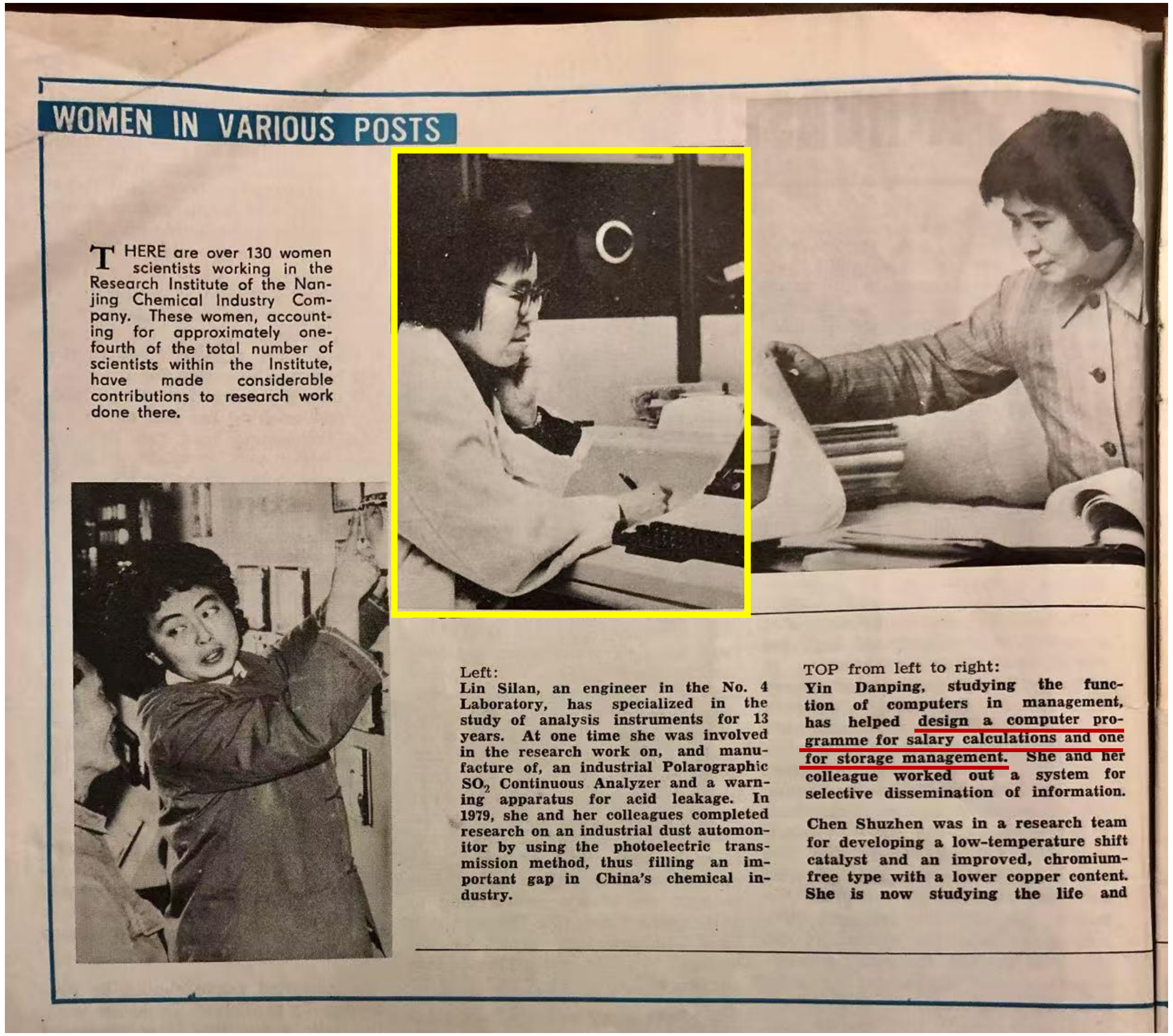
A Woman Chemical Engineer

Painful Women's Diseases Controlled

Husbands and Housework

"I enjoy being a nurse!"

7/80



WOMEN IN VARIOUS POSTS

THERE are over 130 women scientists working in the Research Institute of the Nanjing Chemical Industry Company. These women, accounting for approximately one-fourth of the total number of scientists within the Institute, have made considerable contributions to research work done there.

Left:
Lin Silan, an engineer in the No. 4 Laboratory, has specialized in the study of analysis instruments for 13 years. At one time she was involved in the research work on, and manufacture of, an industrial Polarographic SO_2 Continuous Analyzer and a warning apparatus for acid leakage. In 1979, she and her colleagues completed research on an industrial dust automonitor by using the photoelectric transmission method, thus filling an important gap in China's chemical industry.

TOP from left to right:
Yin Danping, studying the function of computers in management, has helped design a computer programme for salary calculations and one for storage management. She and her colleague worked out a system for selective dissemination of information.

Chen Shuzhen was in a research team for developing a low-temperature shift catalyst and an improved, chromium-free type with a lower copper content. She is now studying the life and

Compute Resources Shared Then

First data center – 1950s

Source: https://opticalcloudinfra.com/index.php/what-why-and-how/short-data-center-history

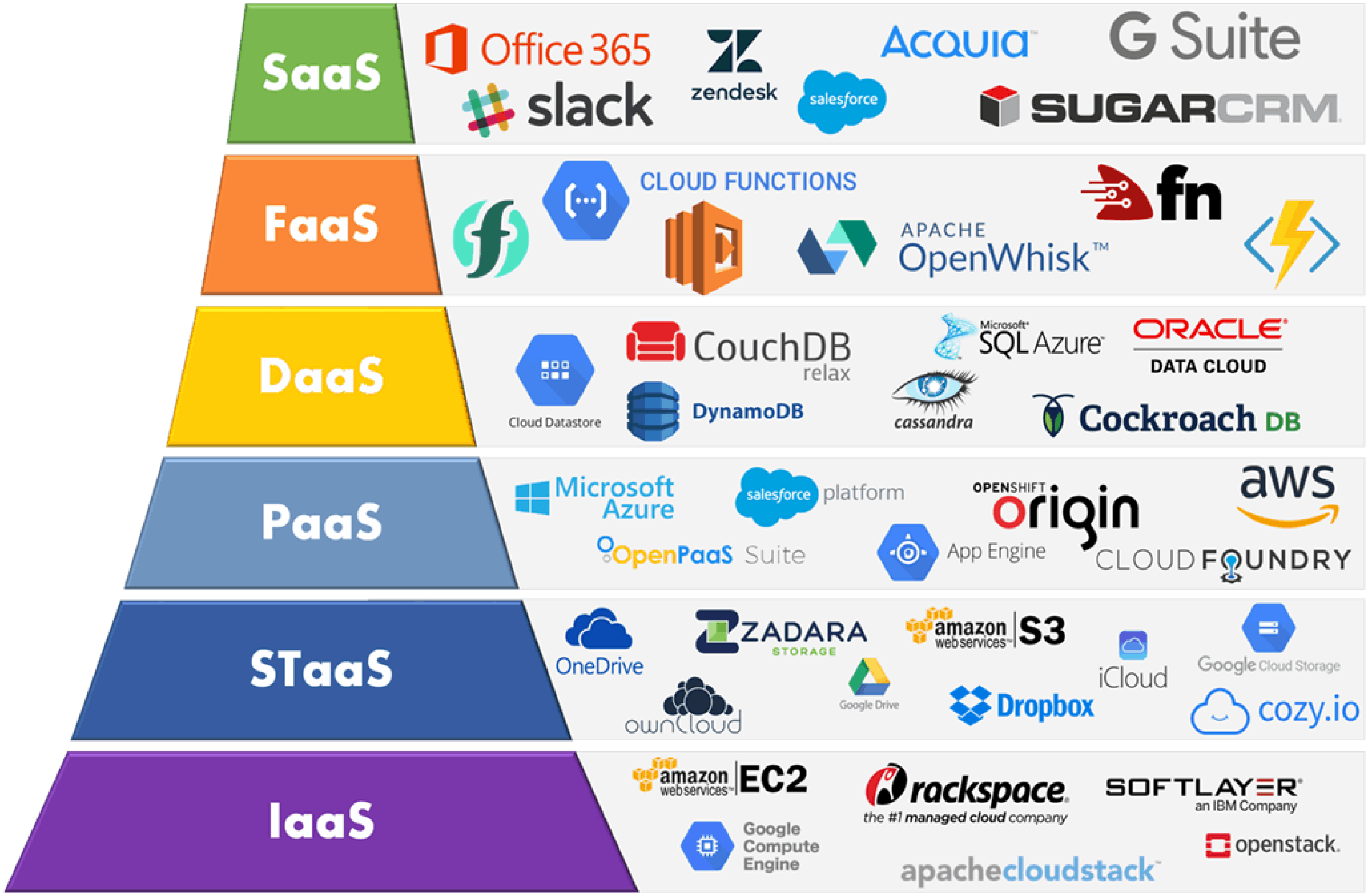
And Again Now

Supercomputers

Source: Oak Ridge National Lab

Data centers

Source: https://www.crn.com/news/data-center/google-unveils-new-750m-data-center-as-part-of-9-5b-goal



Cloud platforms and services

Source: https://kinsta.com/blog/go`ogle-cloud-vs-aws/

Sharing-Friendly Hardware Platforms

❑ Multi-core processors

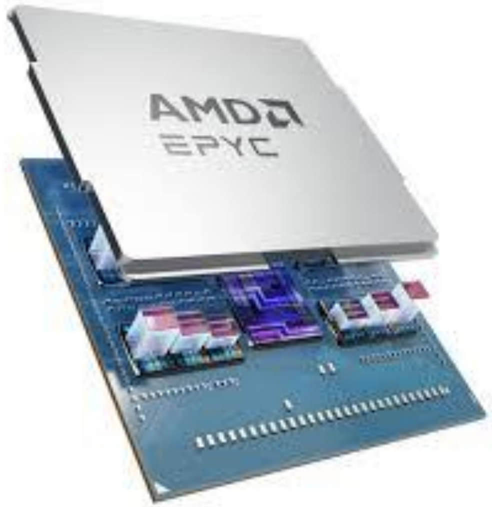
- ➢ Dozens of cores
- ➢ CPU caches optimized for multi-tenancy (e.g., large L2 caches)
- ➢ TBs of DRAM space
- ➢ Mechanisms for inter-core resource allocation (cache ways, memory BW)

❑ Powerful interconnect

- ➢ Fast network connections (e.g., up to 100Gbps at AWS)
- ➢ Smart NICs/DPUs offloading computation tasks

❑ High-capacity storage

- ➢ NVMe SSDs offering space, bandwidth, and IOPS for sharing



AMD EPYC 9654

Source: AMD

Implications of Pervasive Resource Sharing

❑ Programs to run

- ➢ on unknown/changing hardware
- ➢ with unknown/changing neighbors

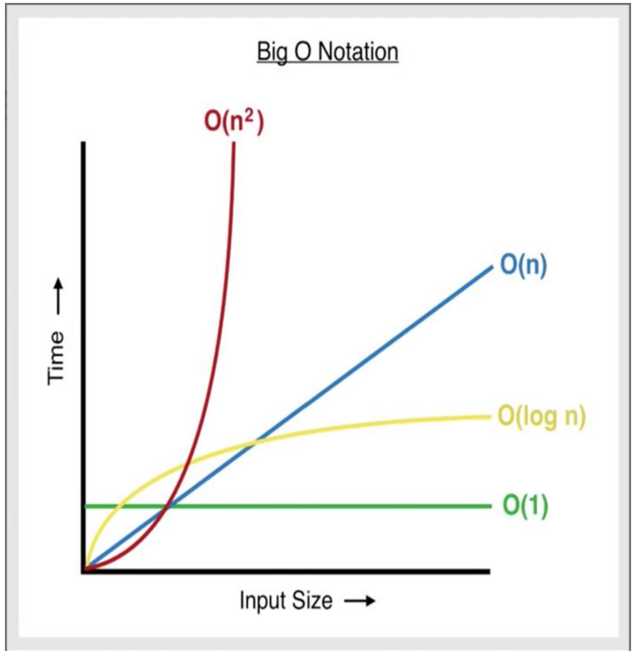
□ Major challenge: performance portability

□ Desirable for applications/services to

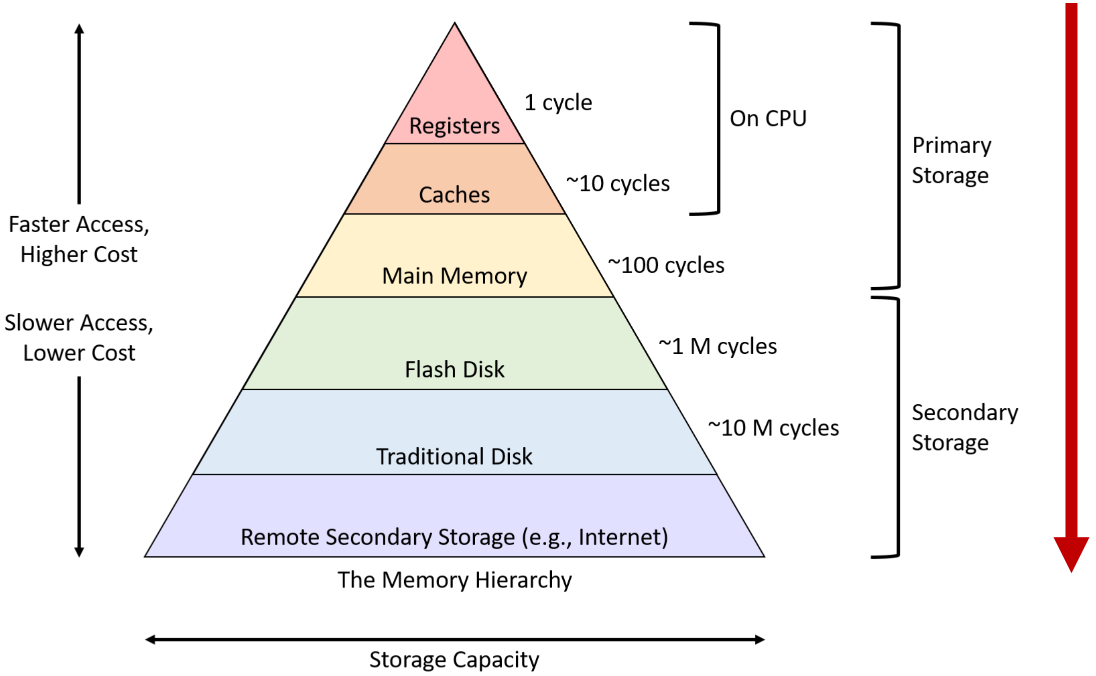
- ➢ **retain (optimized) performance across platforms**
- ➢ **achieve hardware potential**

Challenge Lies in Storage Hierarchy

- Computation logic more “portable”
  - Instruction execution easier in scheduling and isolation
  - Current server processors w. sizable per-core resources (e.g., L1+L2 cache)
- Data access path deeper and more complex



Larger, cheaper, slower, less random-friendly



The Memory Hierarchy

(https://medium.com/dataseries/a-quick-primer-on-big-o-notation-c99ccc7ddbae)

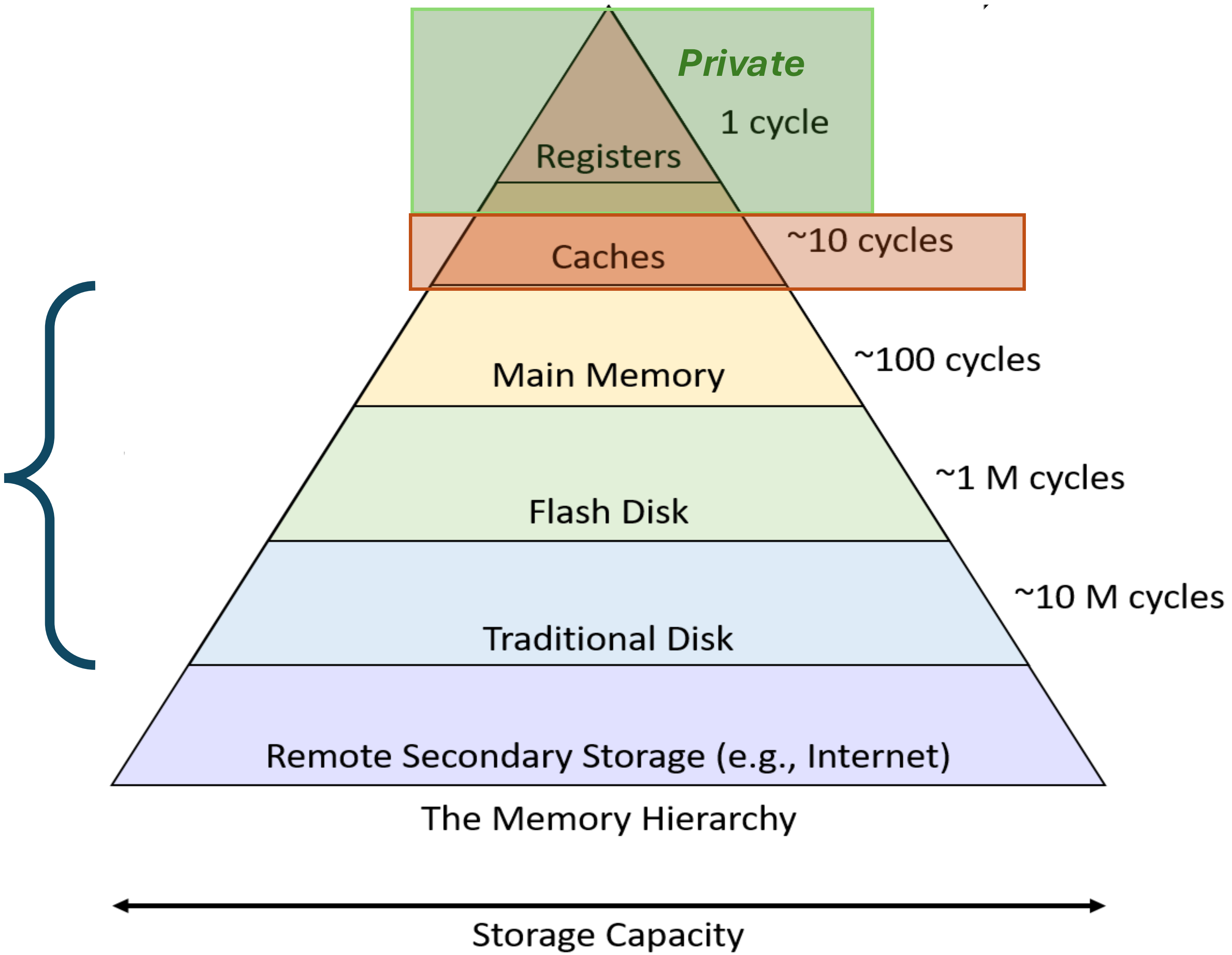
(https://diveintosystems.org/book/C11-MemHierarchy/mem_hierarchy.html)

Storage and I/O Not Efficiently Shared

- ❑ Major factor leading to storage frustrations and wastes
  - ➢ From single server to data-centers/supercomputers

- ❑ In this talk
  - ➢ Sample related problems and solutions in our past research
  - ➢ Extended I/O hierarchy for AI workloads

Challenging Layer 1: Shared Cache



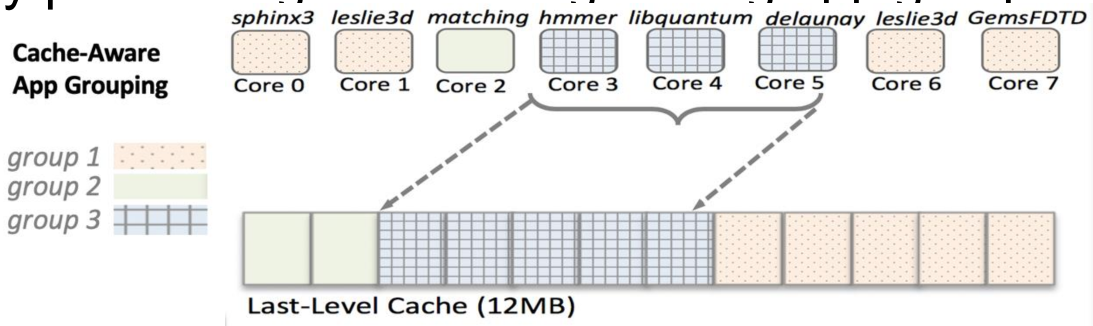
Mature on-demand provisioning on cloud (Space, BW, IOPS)

The Memory Hierarchy

Cache Partitioning in Commodity Multicores

- ✓ Current processors offer hardware cache-partitioning support (Intel CAT)
- Partitioning last-level cache among co-running apps reduces interference ➔ improves system performance

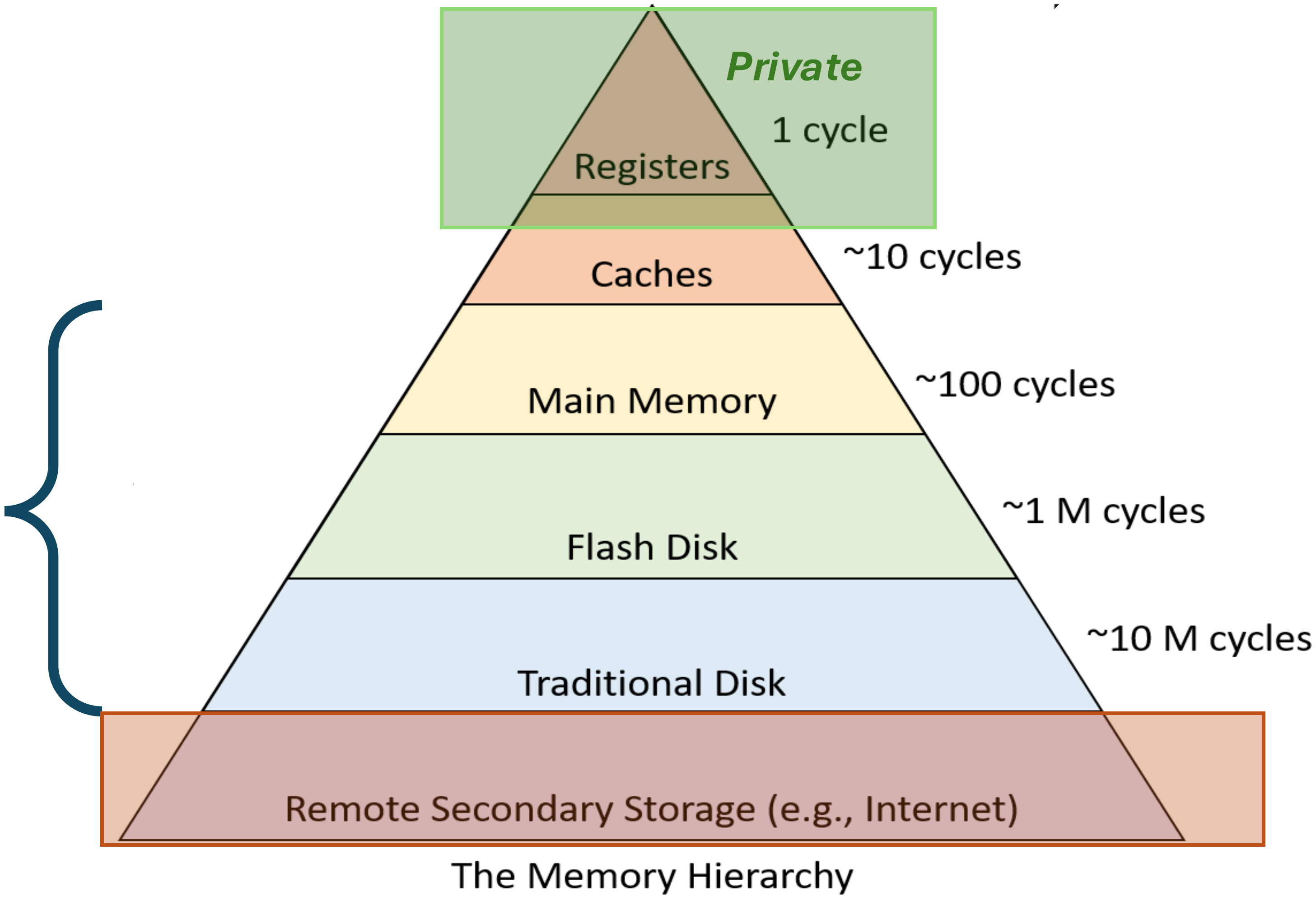
- ***Kpart*** [HPCA ‘18]
- ✕ Two key challenges limit usability of CAT
  - Current hardware implements coarse-grained way-partitioning ➔ hurts system performance!
  - Lacks hardware monitoring units to collect cache-profiling data
- Solution: hybrid way partitioning and sharing among app groups



  - Significant performance gain on real hardware (avg 24%, max 79%)

Challenging Layer 2: Supercomputer Shared Storage

Mature on-demand provisioning on cloud (Space, BW, IOPS)



The Memory Hierarchy

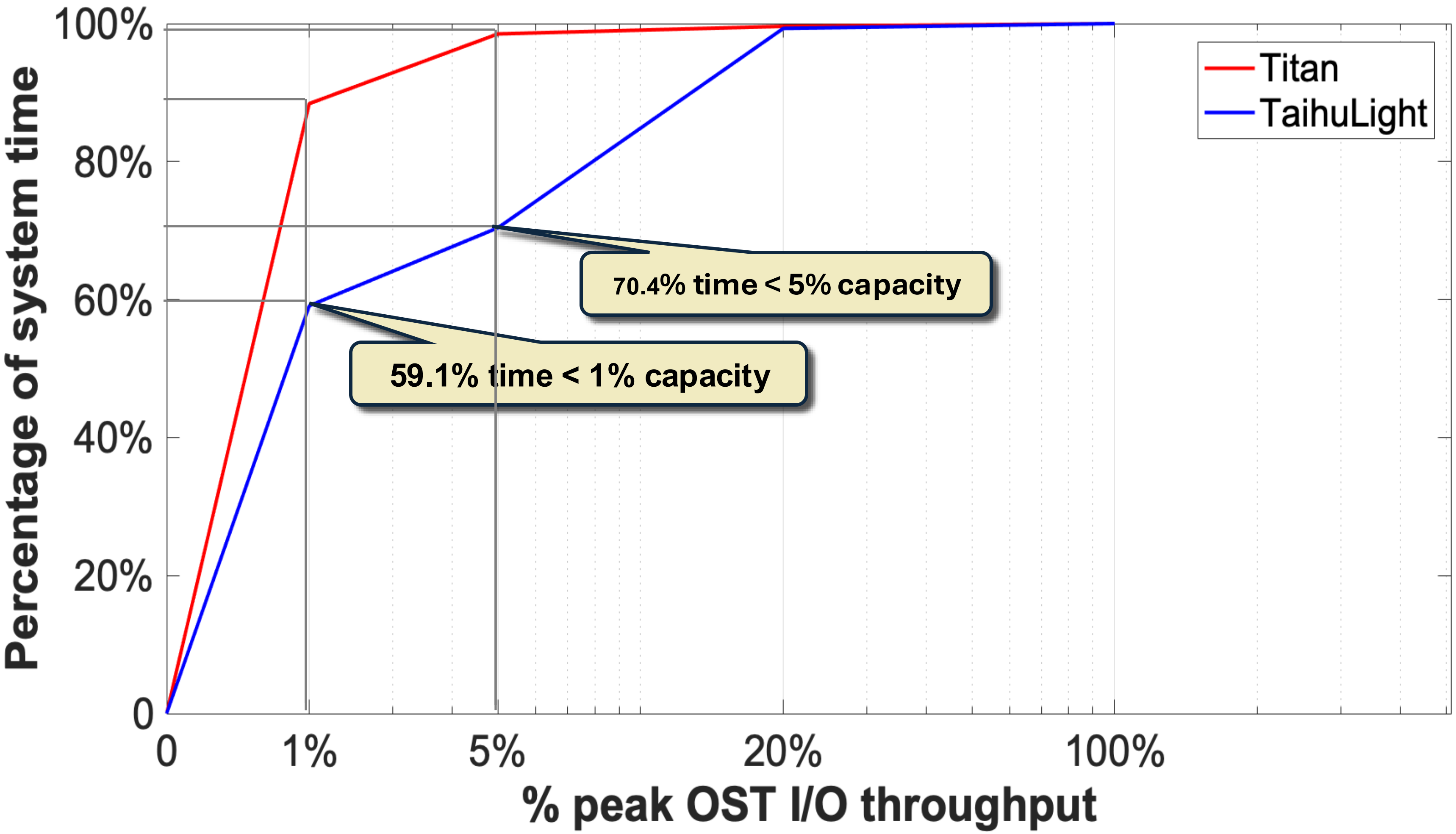
Supercomputer Storage: Not Better Shared

- ❑ Observation from supercomputer I/O profiling ([FAST14], [SC16], [NSDI19])
  - ➢ Significant inter-job interference -> inconsistent I/O performance
  - ➢ Vast majority of HPC jobs non-I/O intensive -> overall low I/O resource utilization

Name	Value
Total number of logged jobs	181,969
Unique applications identified	9,998
Initial I/O-intensive candidates	95
Candidates passing scope checking	67
Candidates passing minimum support	**42**
User-verified candidates	8

Job I/O statistics based on ORNL Titan supercomputer, 2015

Bottleneck, Contention Point, and Under-Utilization



70.4% time < 5% capacity

59.1% time < 1% capacity

End-to-end Supercomputer I/O Monitoring [NSDI19]

- ❑ Understand HPC I/O for designing future systems/applications
  - ➢ Lightweight end-to-end I/O resource monitoring

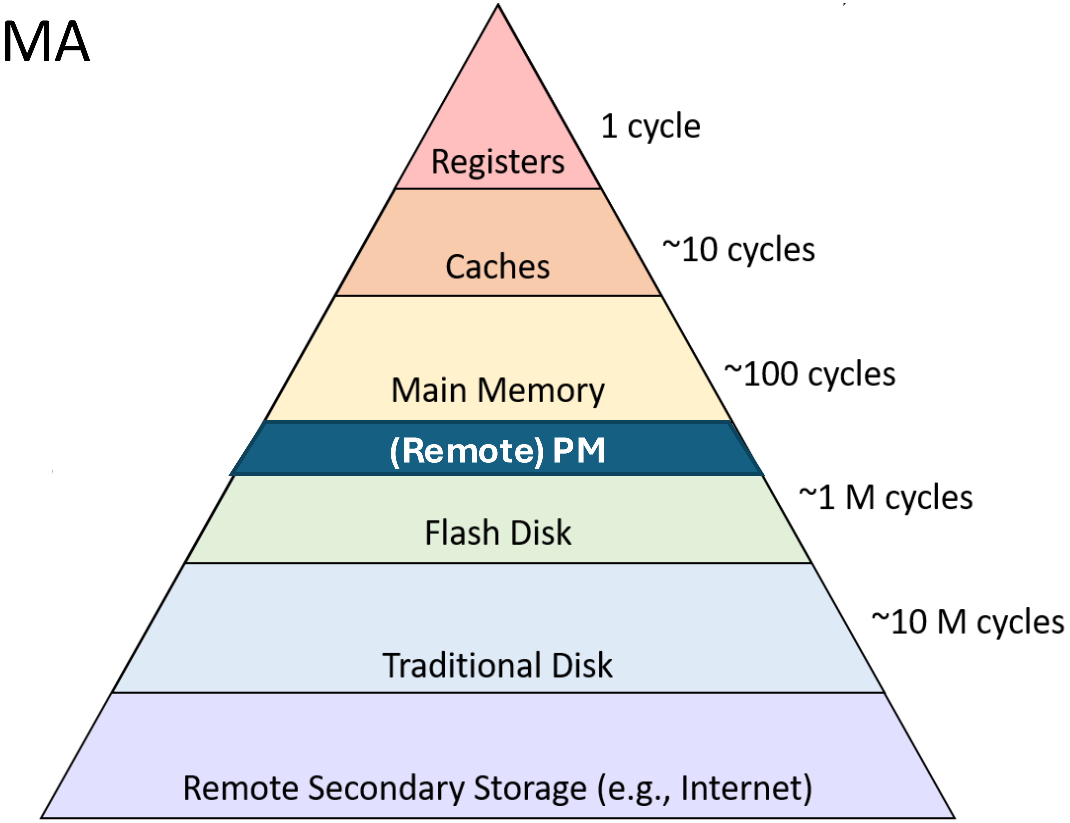
- ❑ Deployed at TaihuLight
  - ➢ No user effort required
  - ➢ Code and monitoring data released: ***https://github.com/Beaconsys/Beacon***

- ❑ Findings based on 18-month monitoring on production platform
  - ➢ Wide-spread adoption of inefficient I/O modes
    - Lowing both **application performance** and **hardware utilization**
  - ➢ System anomalies and their behaviors (echoing findings from datacenters)
  - ➢ Obscure design/configuration problems, e.g., forwarding layer cache thrashing
  - ➢ Significant forwarding node load imbalance => Application-aware I/O forwarding [FAST19]

Another Layer in Pyramid: Remote PM

❑ Persistent memory disaggregation

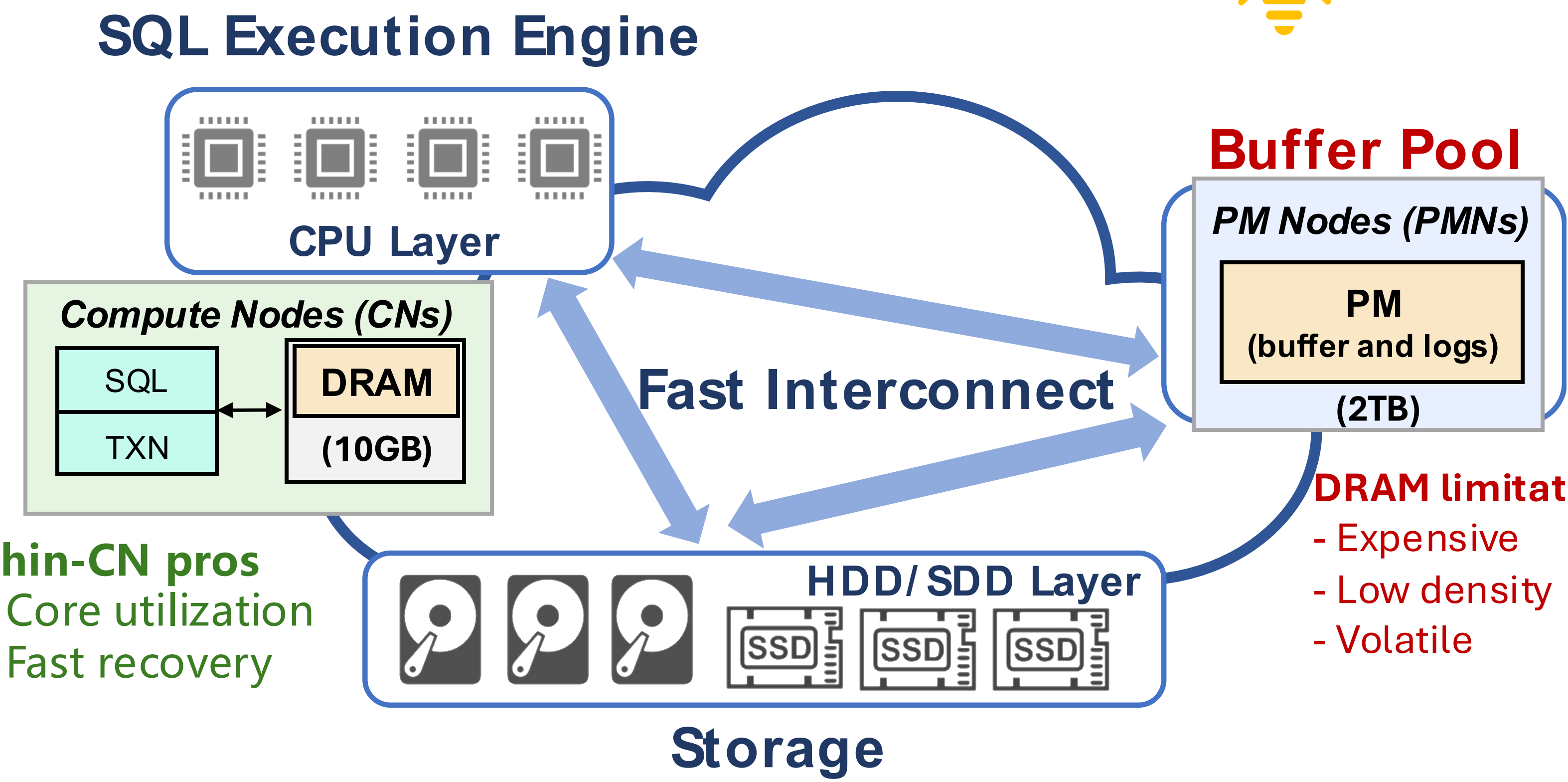
- ➢ Faster than local SSDs with RDMA
- ➢ Enables
  - large memory buffer
  - lean compute nodes



The Memory Hierarchy

Cloud-Native DB on Disaggregated PM

- Distributed RDBMS [ASPLOS23, VLDB25]



SQL Execution Engine

CPU Layer

Compute Nodes (CNs)

SQL

TXN

DRAM

(10GB)

Fast Interconnect

Buffer Pool

PM Nodes (PMNs)

PM

(buffer and logs)

(2TB)

HDD/SDD Layer

SSD

SSD

SSD

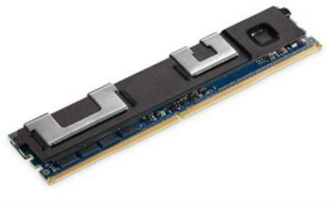
Storage

Thin-CN pros

- Core utilization
- Fast recovery

DRAM limitations

- Expensive
- Low density
- Volatile



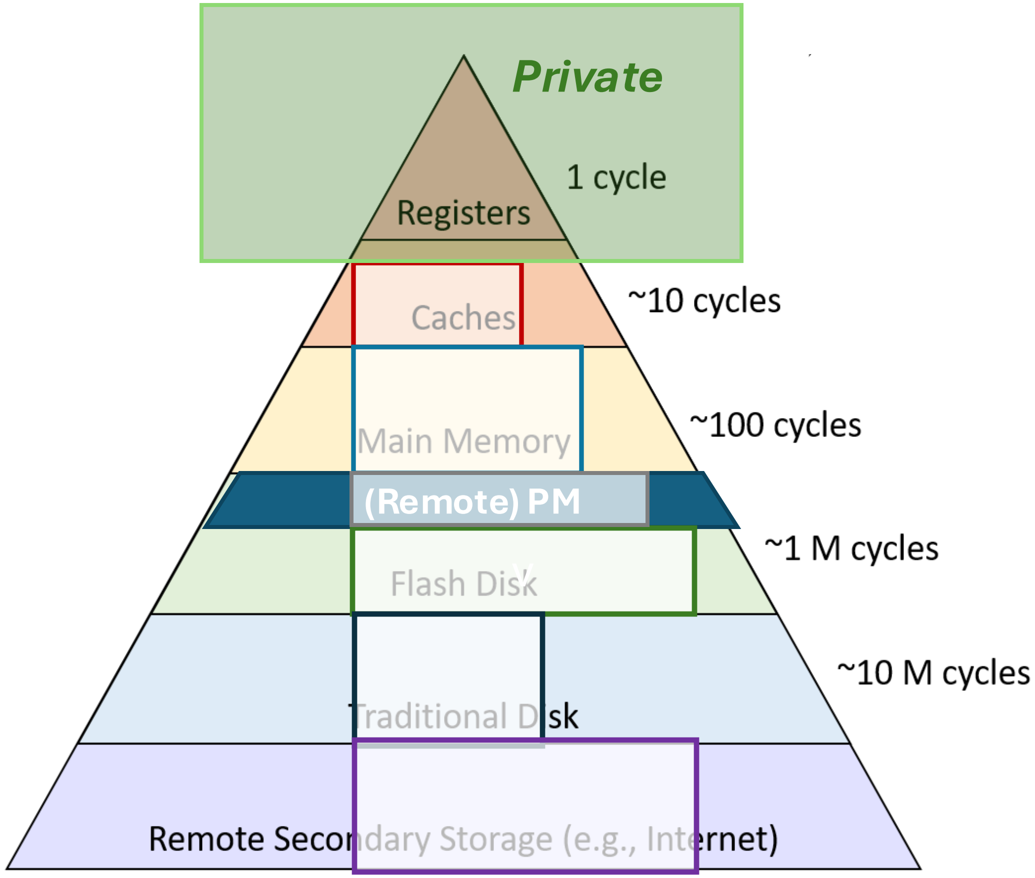
Optane PM

XL-FLASH

All-around features:

- Performance close DRAM
- Yet cheaper
- High density
- Byte-addressable
- Non-volatile

“Storage Container” All the Way



The Memory Hierarchy

Within single workload: how to better use allocated hardware?

Sample Memory/Cache Performance

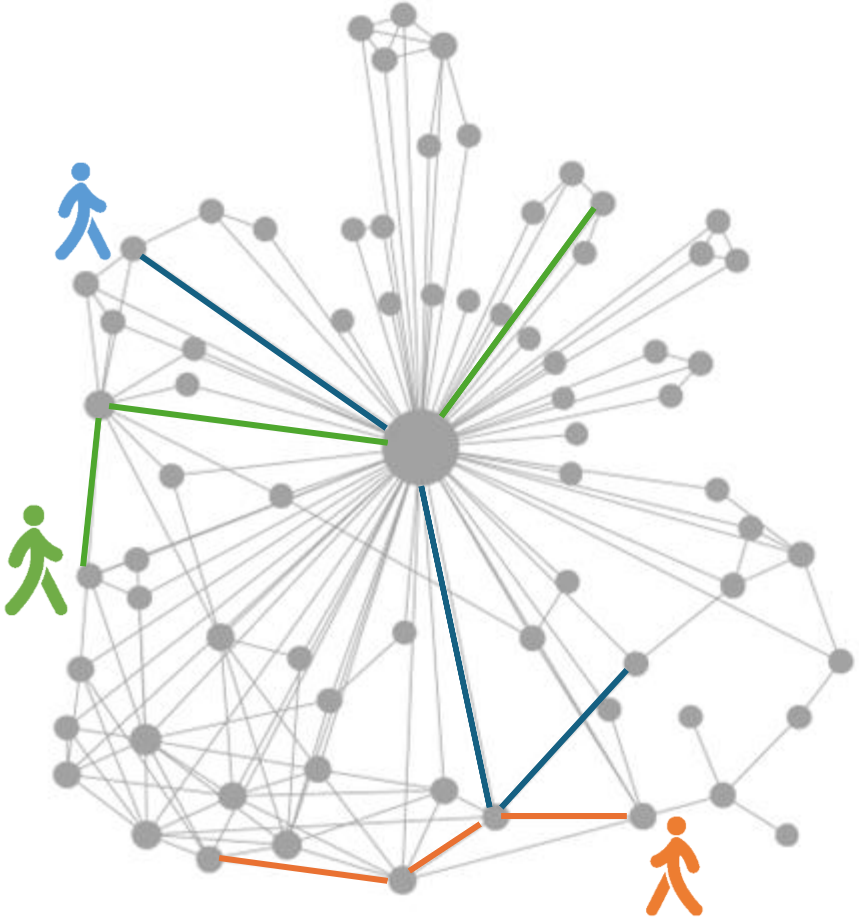
Location	L1 cache	L2 cache	L3 cache	Local mem	Remote mem
Sequential read	0.42ns	0.41ns	0.44ns	0.76ns	1.51ns
Random Read	0.77ns	0.95ns	2.60ns	18.35ns	24.35ns
Pointer-chasing	1.69ns	5.26ns	19.26ns	116.90ns	194.26ns

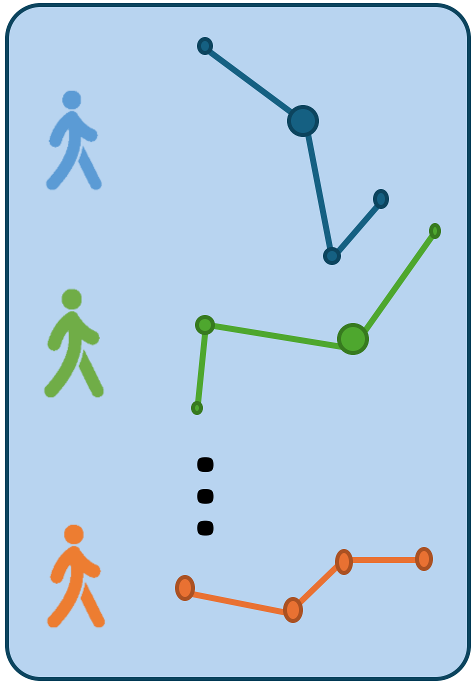
4-byte read latency at different cache/DRAM layers

- □ Sequential reads quite cheap, and relatively uniform
  - ➢ Even across NUMA node (remote memory)
- □ Random reads slower, with wider distribution
  - ➢ Large gap between L3 and DRAM
  - ➢ Pointer-chasing especially costly: even in L3

High-Concurrency Scenario 1: Graph Random Walk

- Problem definition
  - Input: graph, set of walkers placed at starting vertices
  - Each walker walks around
    - By randomly selecting an edge to follow
    - For given number of steps or till given termination condition
  - Output
    - Computation during walk, and/or
    - Set of walk paths



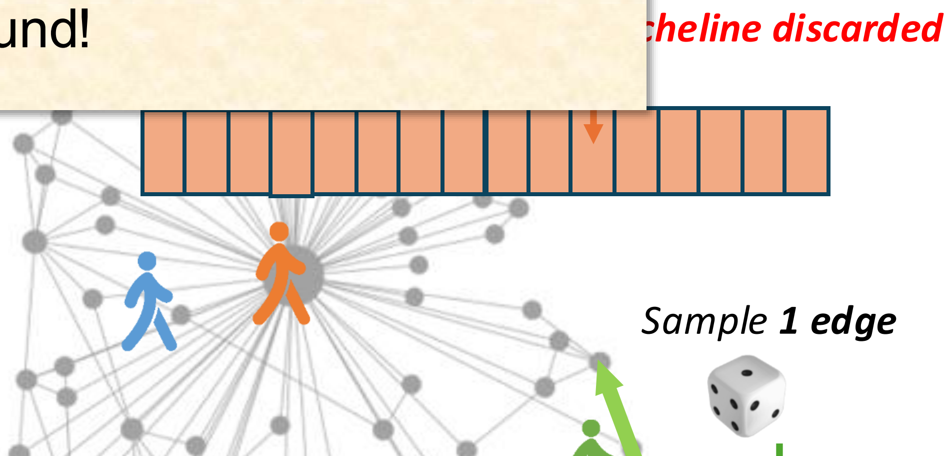


Challenge: Slow Random Memory Accesses

- Walke
  - Fo

> Random walk doesn’t mean **random memory accesses**
> - Plenty of temporal and spatial locality around!

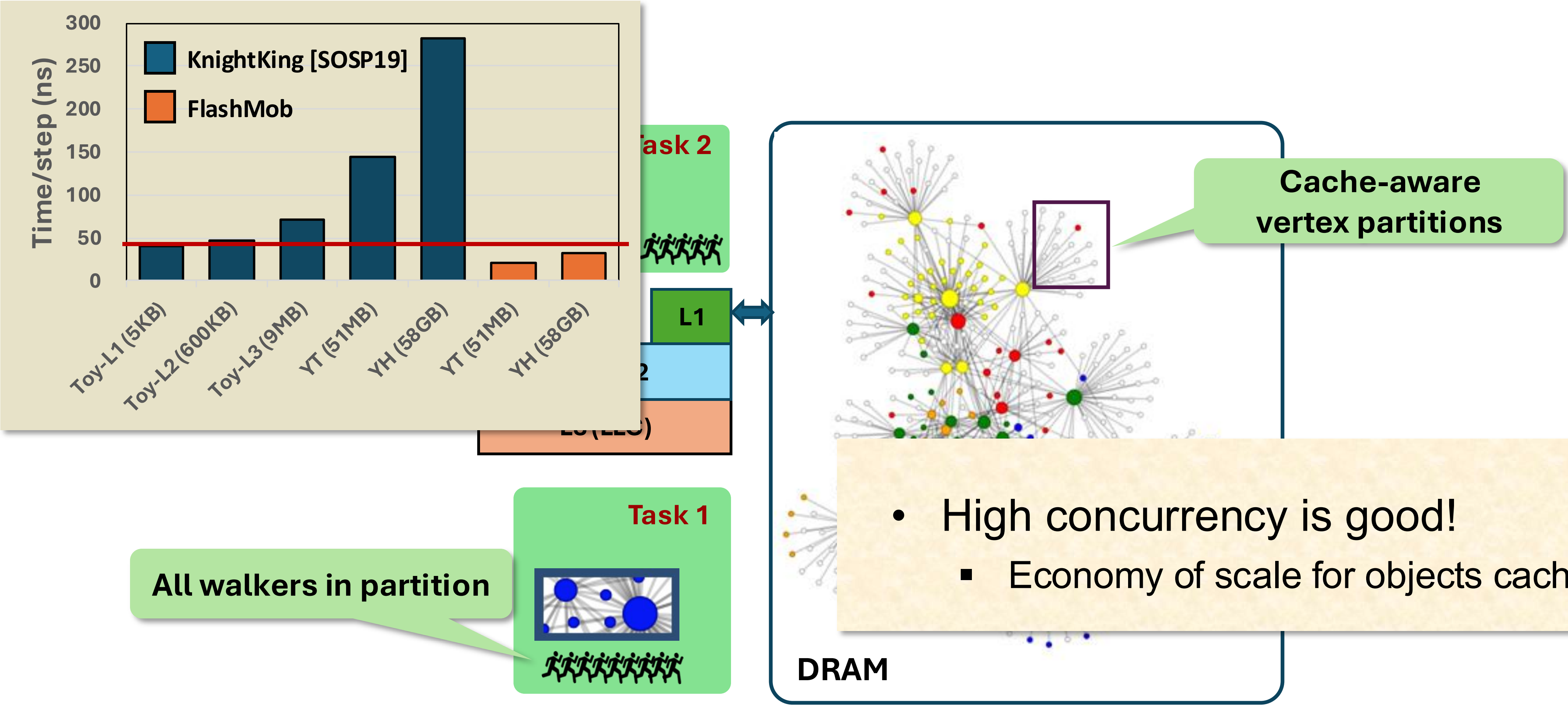
- Probl
  - Accept randomness in memory accesses
  - Process walkers in turn, wherever they are
    - High data dependency (pointer chasing)
    - Low utilization of loaded cache lines



Location	L1 cache	L2 cache	L3 cache	Local mem	Remote mem
Sequential read	0.42ns	0.41ns	0.44ns	0.76ns	1.51ns
Random Read	0.77ns	0.95ns	2.60ns	18.35ns	24.35ns
Pointer-chasing	1.69ns	5.26ns	19.26ns	116.90ns	194.26ns

Access latency at different cache/DRAM layers

FlashMob [SOSP21]: Cache-speed Graph Random Walk



Cache-aware vertex partitions

All walkers in partition

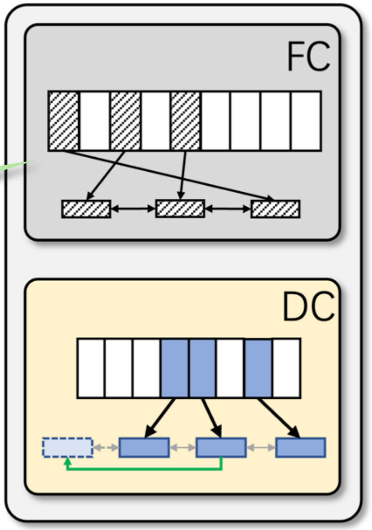
- High concurrency is good!
  - Economy of scale for objects cached

High-Concurrency Scenario 2: Memory Caching

- New challenges due to high-performance hardware
  - Faster storage: 1000x to 10x latency gap from DRAM
  - Scalability to high core counts

- High concurrency is bad!
  - Write contention on cache hits

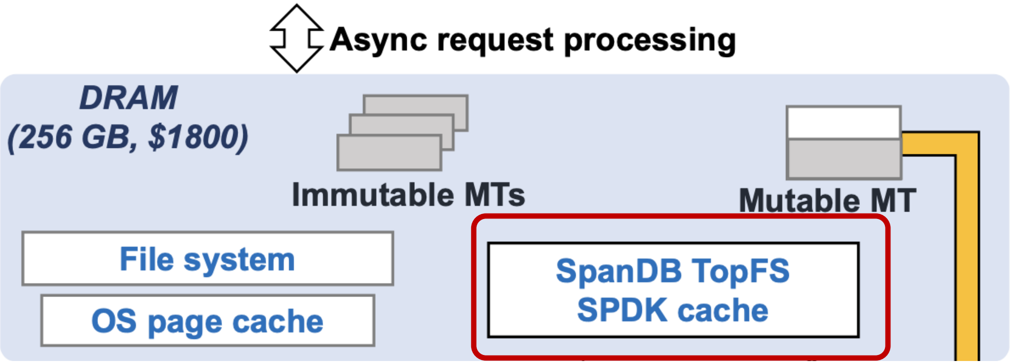
- FrozenHot [EuroSys ’23]
  - Speeding up hit path by removing cache management

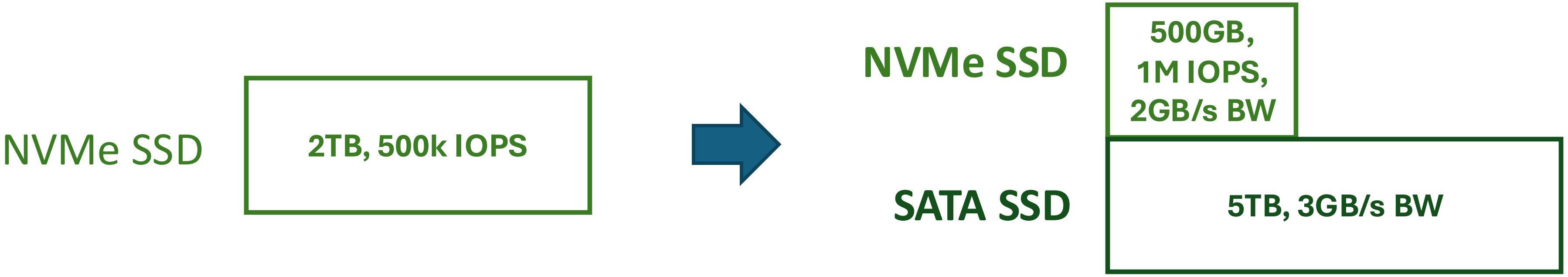


Most hits on frozen cache (FC)
- **No list management**
- **No locks**
- **Faster hash table**

High-Concurrency Scenario 3: KV Stores

- ❑ SpanDB [FAST21]
  - ➢ Implemented within RocksDB
- ❑ Distributing LSM-tree based KV data
  - ➢ Large and slow disk for capacity
  - ➢ Small and fast disk for speed
- ❑ Automatic tree layer placement
  - ➢ Adaptive to partition sizes and workloads
  - ➢ Allow hybrid storage for cost-effectiveness




NVMe SSD

2TB, 500k IOPS

NVMe SSD

500GB, 1M IOPS, 2GB/s BW

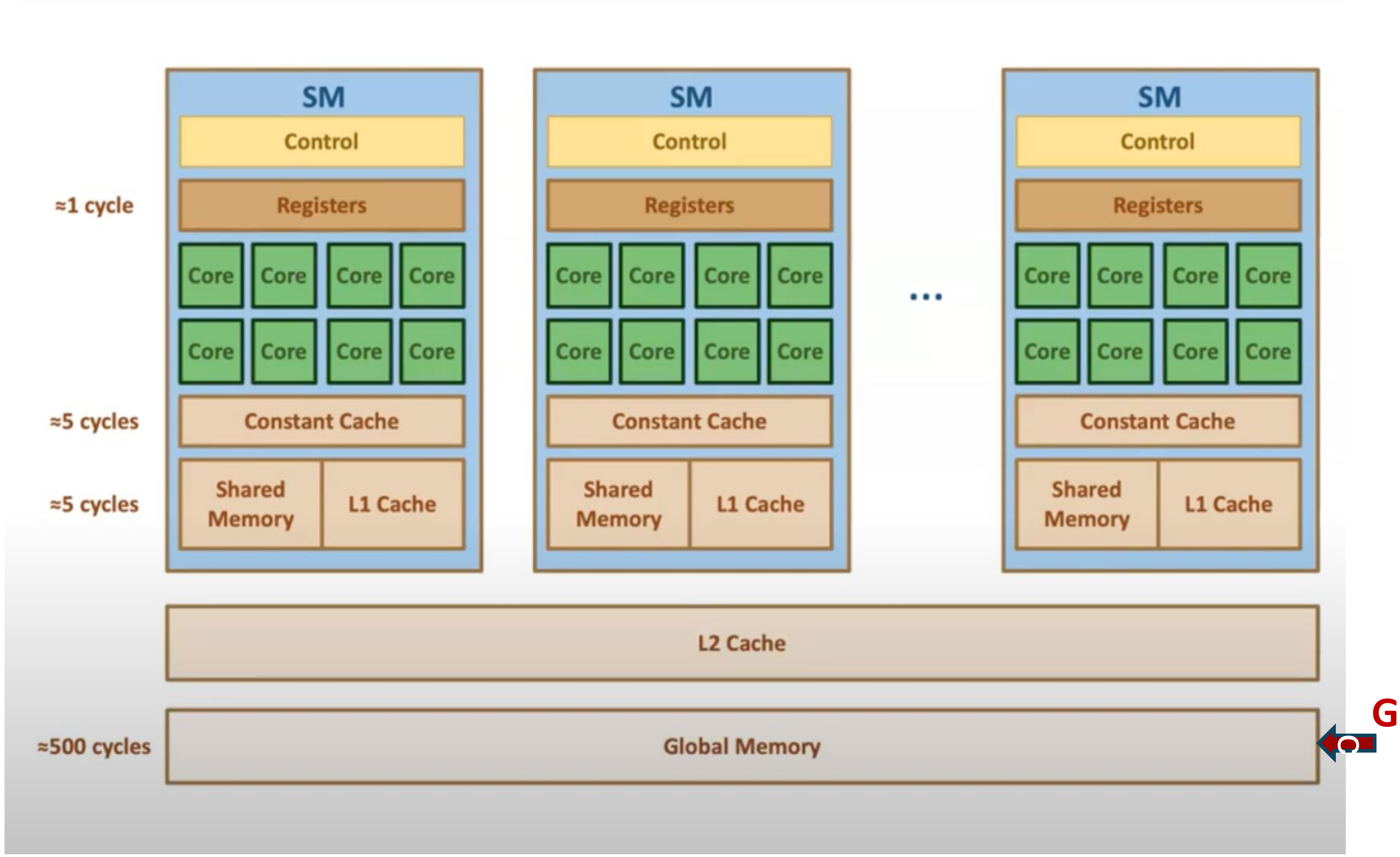
SATA SSD

5TB, 3GB/s BW

Storage and I/O Not Efficiently Shared

- ❑ Major factor leading to storage frustrations and wastes
  - ➢ From single server to data-centers/supercomputers
- ❑ In this talk
  - ➢ Sample related problems and solutions in our past research
  - ➢ Extended I/O hierarchy for AI workloads

AI and LLM Age: GPU as a Supercomputer



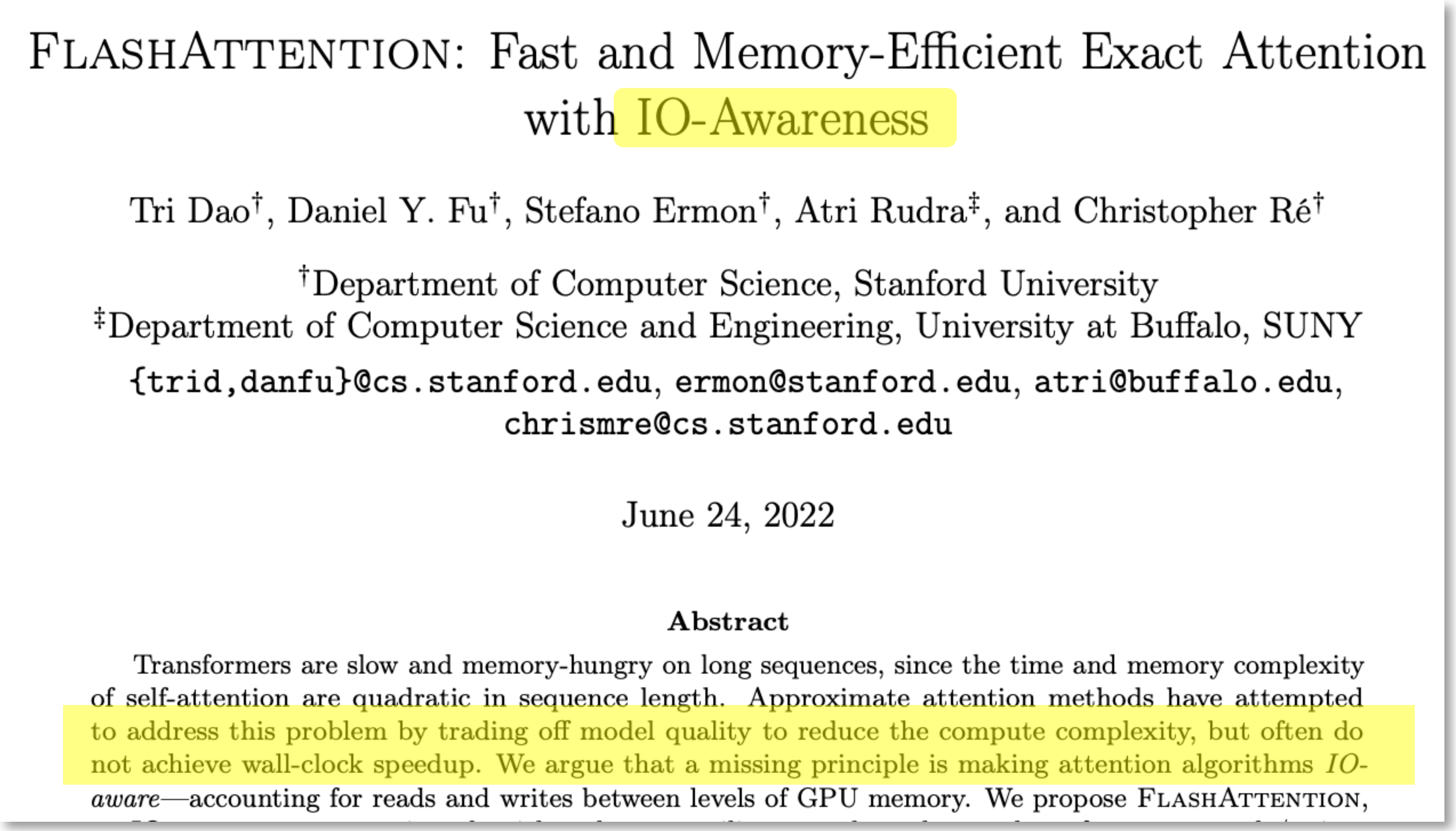
Global memory (HBM)
Much larger/slower

NVIDIA H100 Memory Hierarchy

Credit: Izzat El Hajj, HetSys Course: Lecture 4: GPU Memory Hierarchy (Spring 2023), Onur Mutlu Lectures

GPU Global Memory the New Disk?

- ❑ FlashAttention: optimization targeting long sequences
  - ➢ Adopted by major LLM frameworks: PyTorch, Megatron, DeepSeek …



FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao[†], Daniel Y. Fu[†], Stefano Ermon[†], Atri Rudra[‡], and Christopher Ré[†]

[†]Department of Computer Science, Stanford University
[‡]Department of Computer Science and Engineering, University at Buffalo, SUNY

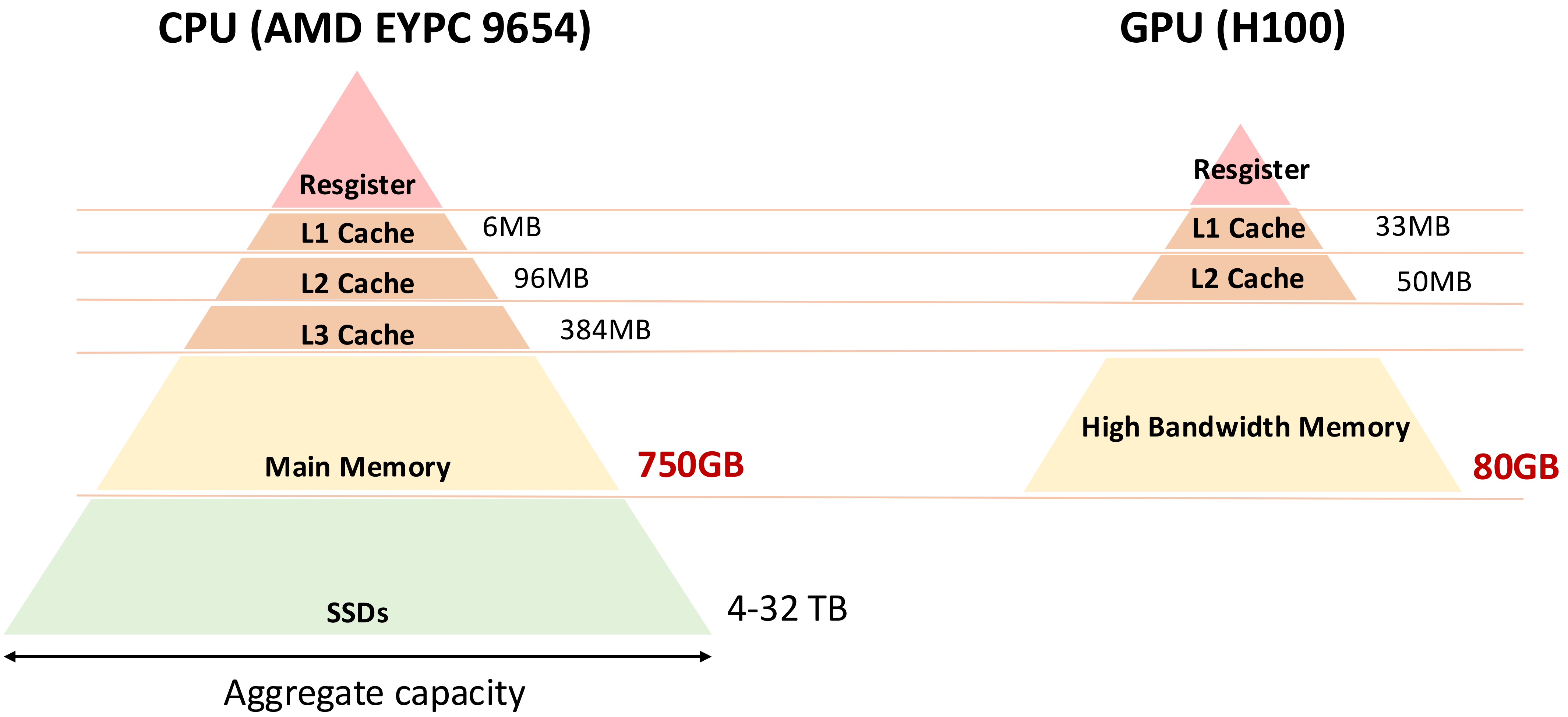
{trid,danfu}@cs.stanford.edu, ermon@stanford.edu, atri@buffalo.edu,
chrismre@cs.stanford.edu

June 24, 2022

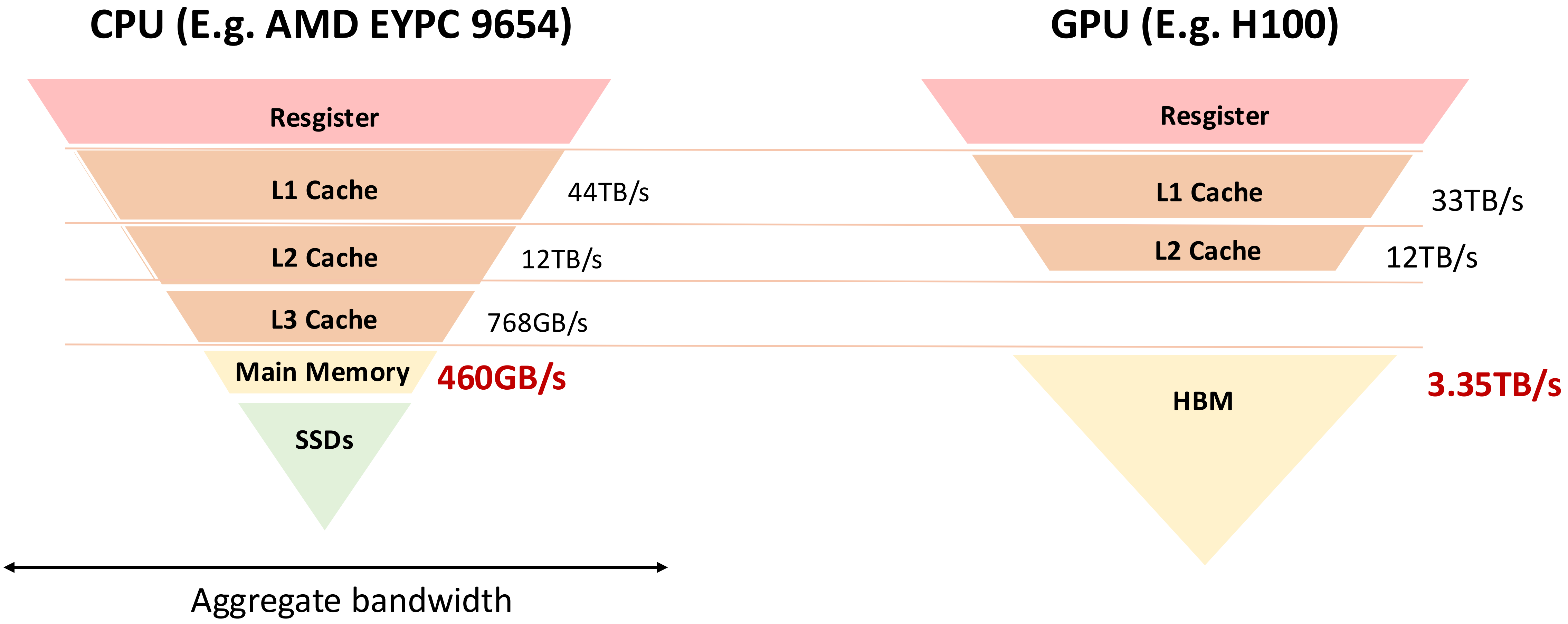
Abstract

Transformers are slow and memory-hungry on long sequences, since the time and memory complexity of self-attention are quadratic in sequence length. Approximate attention methods have attempted to address this problem by trading off model quality to reduce the compute complexity, but often do not achieve wall-clock speedup. We argue that a missing principle is making attention algorithms *IO-aware*—accounting for reads and writes between levels of GPU memory. We propose FlashAttention,

CPU/GPU-Side Storage Hierarchy (Capacity)



CPU/GPU-Side Storage Hierarchy (Bandwidth)



“I/O Characteristics” of Transformer Components

Operator	Workload	Access type	Bottleneck
Normalization	Training & inference	Balanced R-W	Memory
GEMM	Training & inference	Mainly reads	Compute → Memory
Attention	Training & inference	Mainly reads	See table below
Dropout	Training	Balanced R-W	Memory
Activation Function	Training & inference	Balanced R-W	Compute → Memory

Attn sequence length	Workload	Bound
Long	Training & prefill	Compute → Memory
Short	Training & prefill	Compute → Memory
Long	**Decode**	**Memory**
Short	Decode	Compute → Memory

Good News: Storage-Friendly Access Patterns

❑ Large, sequential, read-heavy accesses

- ➢ Little random reads, relatively light writes
- ➢ Regular, predictable, collaborative data streaming
- ➢ **Data content/precision could be manipulated!**

❑ Many storage/HPC tricks apply

- ➢ Prefetching
- ➢ Tiling/Tiering

Bad News: High Efficiency Demands Large Space

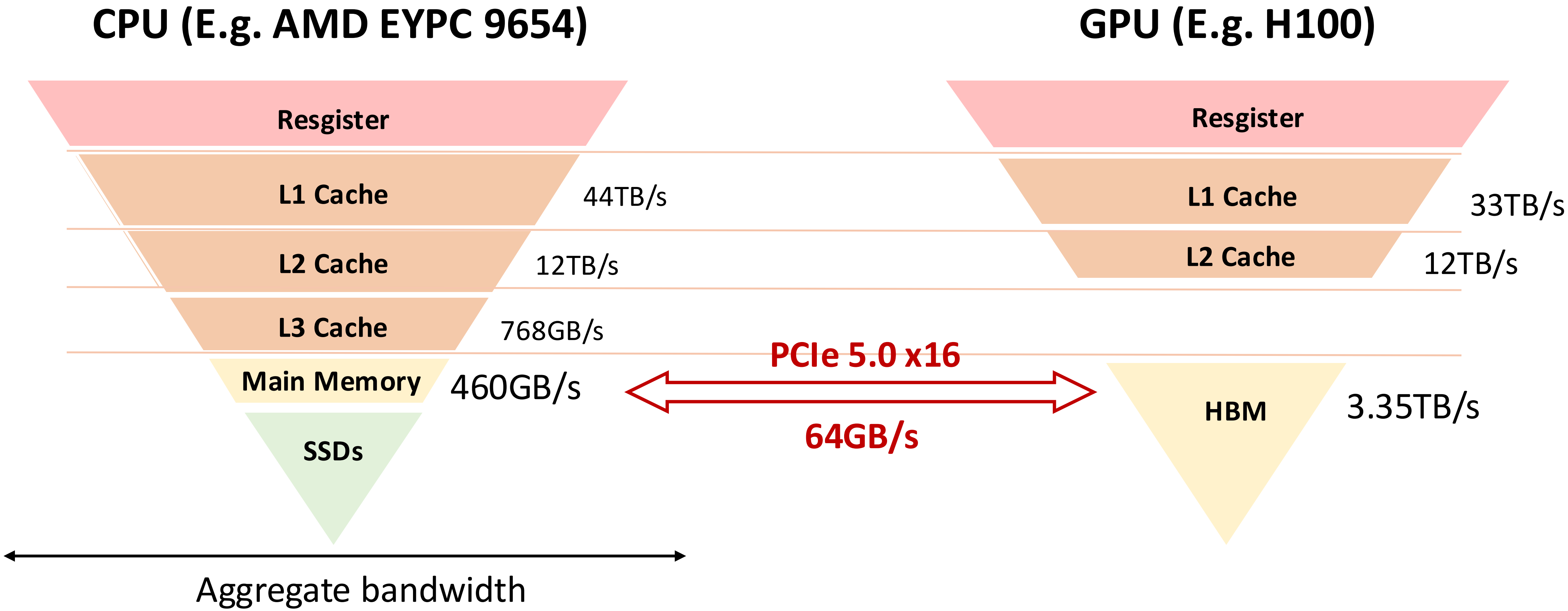
❑ GPU MFU (Model FLOPS Utilization) relies on batch size

- ➢ Larger batches -> higher parallelism, more data reuse
- ➢ Current leading frameworks get <50% of GPU peak TFLOPS

❑ Batch size limited by HBM size

- ➢ Especially w. **long-sequence attention** in **decoding**

Can CPU-Side Memory Help?



Recent Work on Reducing KV-Cache I/O Demands

- ❑ Parameter/KV Cache offloading
  - ➢ FlexGen [ICML23]
  - ➢ MoE-Lightning [ASPLOS25]
- ❑ KV Cache Compression
  - ➢ Keyformer [MLsys24]
- ❑ Quantization
  - ➢ ZipCache [NeurlPS24]
- ❑ Window attention
  - ➢ StreamingLLM [ICLR24]

Closing Remarks

- ❑ **Shared nature makes storage challenging and interesting**
  - ➢ Contention and interference, but also higher throughput and utilization
  - ➢ Joint CPU-GPU storage hierarchy creates more scenarios for sharing/coordination
    - HBM too small to saturate GPU cores, too fast for DRAM to stream

- ❑ **Education also challenged by new modes of knowledge sharing**
  - ➢ ***AI practitioners need to know systems basics***
  - ➢ ***CS students need to retain focus/courage in system building***