

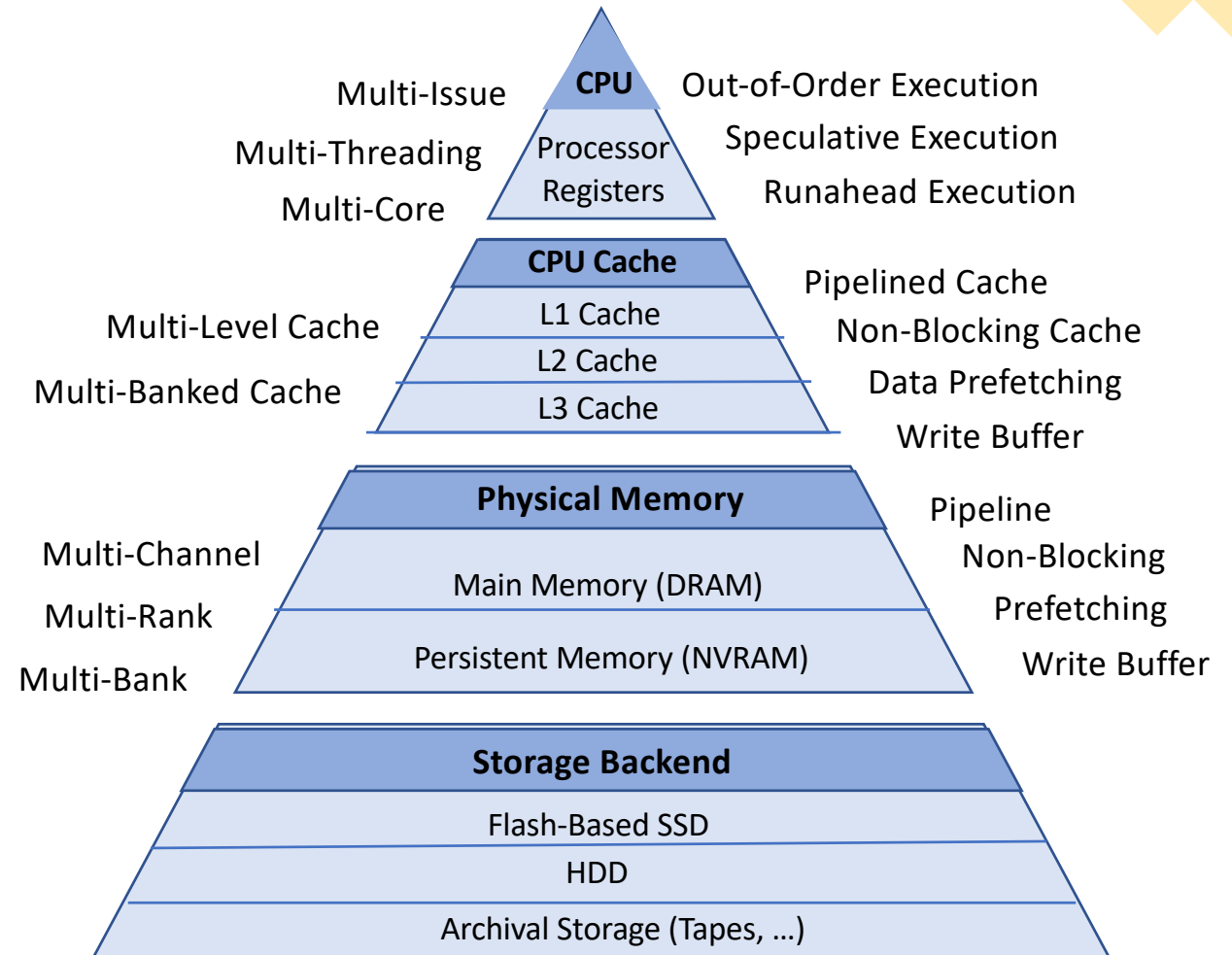
A Unified Framework for Performance Analysis and Optimization of Memory Systems

Jason Liu, Florida International University
Xian-He Sun, Illinois Institute of Technology

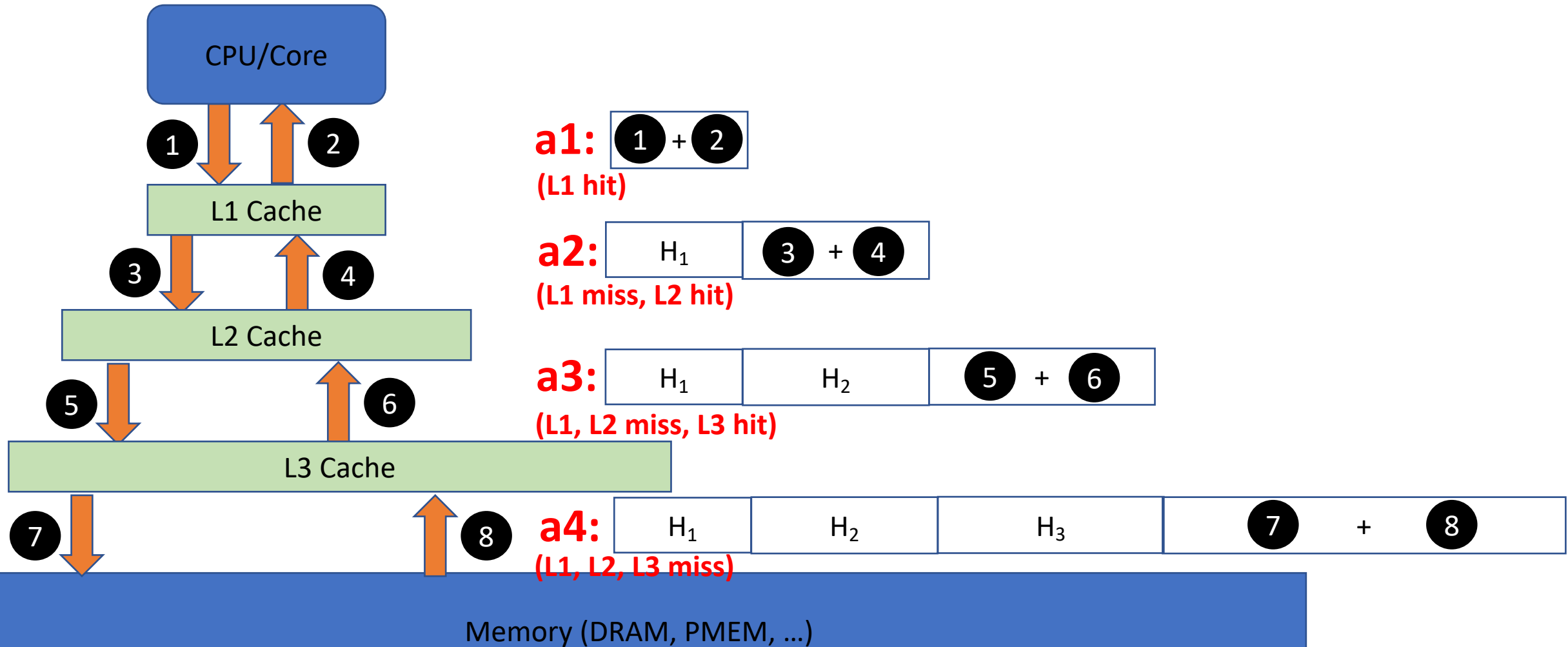
ModSim 2021: Workshop on Modeling & Simulation of Systems and Applications, Oct 5-8, 2021

Memory Models for Performance Analysis and Optimization

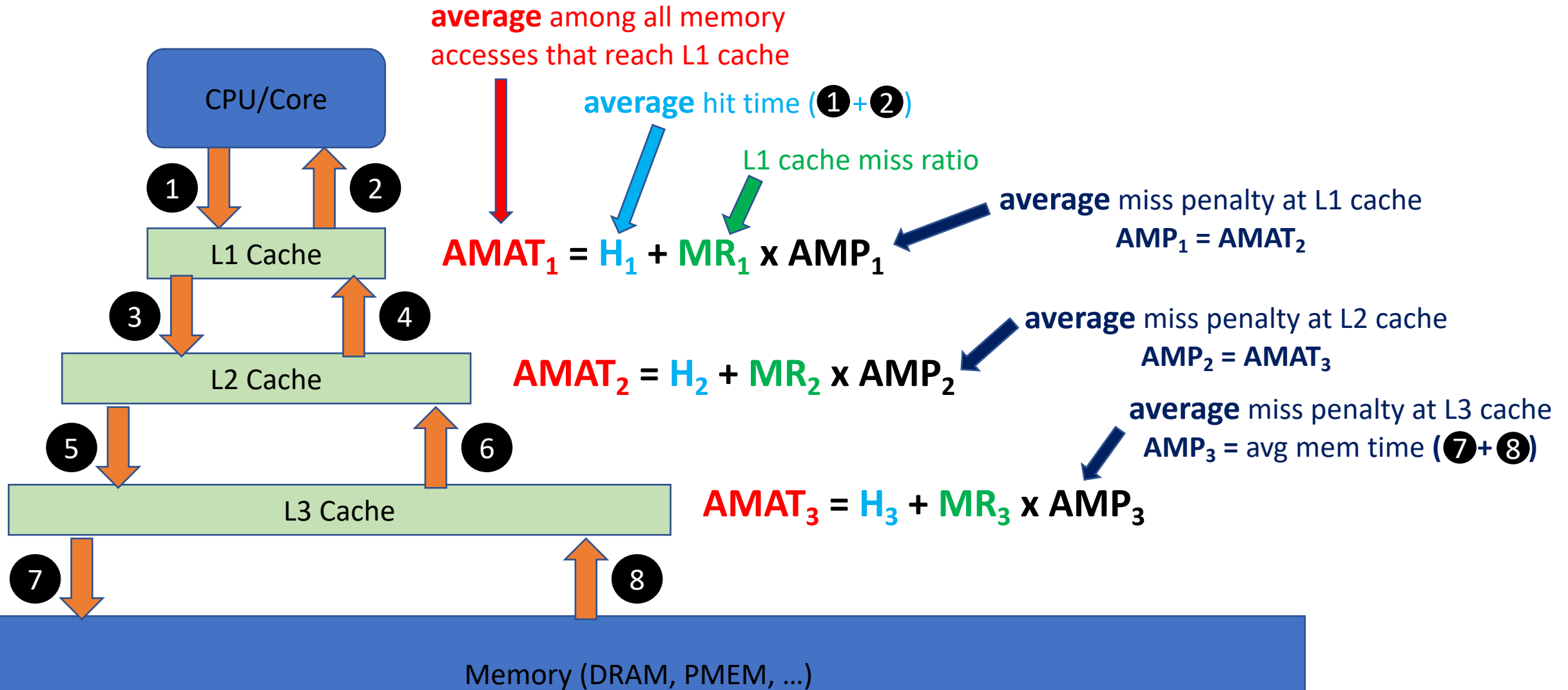
- “Memory wall”, the gap between CPU and memory performance, has been increasing ever since
- Three outstanding issues with modern memory systems:
 - “Scale-up” with deeper hierarchy
 - “Scale-out” with access latency and data demand
 - Heterogeneity in memory technology
- Memory models are essential for evaluating design alternatives and identifying performance bottlenecks



Memory Access Time



Average Memory Access Time (AMAT)



Concurrent Average Memory Access Time (C-AMAT)

- AMAT model failed to consider concurrency
- C-AMAT (concurrent AMAT) extends AMAT to account for concurrency:
$$C\text{-AMAT} = H/C_H + \rho_M \cdot pAMP/C_M$$
$$C\text{-AMAT}(l) = H(l)/C_H(l) + \rho_m(l) \cdot \kappa(l) \cdot C\text{-AMAT}(l+1)$$
- APC (Accesses Per memory active Cycle) is a reciprocal of C-AMAT: memory access rates (measurable in practice) as opposed to memory access time
- LPM (Layered Performance Matching) is a method of matching the data request rate and supply rate in a memory hierarchy for performance optimization

➤ X.-H. Sun and D. Wang, "Concurrent Average Memory Access Time", *IEEE Computers*, vol. 47, no. 5, pp. 74-80, May 2014.

➤ X.-H. Sun and D. Wang, "APC: A Performance Metric of Memory Systems", *ACM SIGMETRICS Performance Evaluation Review*, Volume 40, Issue 2, 2012.

➤ Y. Liu and X. Sun. "LPM: A Systematic Methodology for Concurrent Data Access Pattern Optimization from a Matching Perspective", *IEEE Transactions on Parallel and Distributed Systems*, 30(11):2478–2493, Nov 2019.

A Memory-Centric View

The original C-AMAT derivation was focused on individual memory accesses, and somehow difficult to envision the effect of concurrency at a specific cache/memory device and across the memory hierarchy

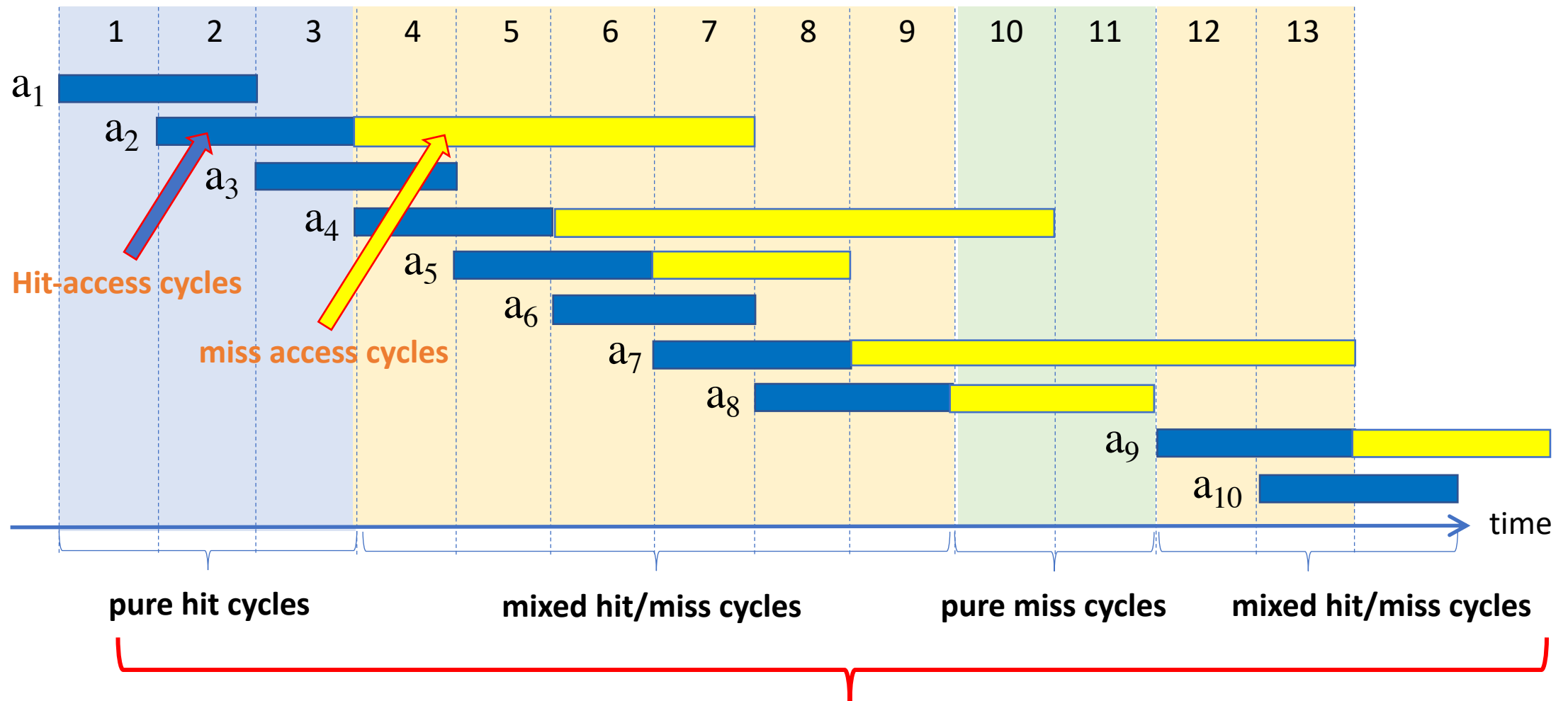
A memory-centric view focuses on specific cache/memory device as memory accesses land on the device and how they overlap and interact with one another

It provides a time synchronized view across the memory hierarchy and makes it easier to realize recursive definitions

A Memory-Centric View

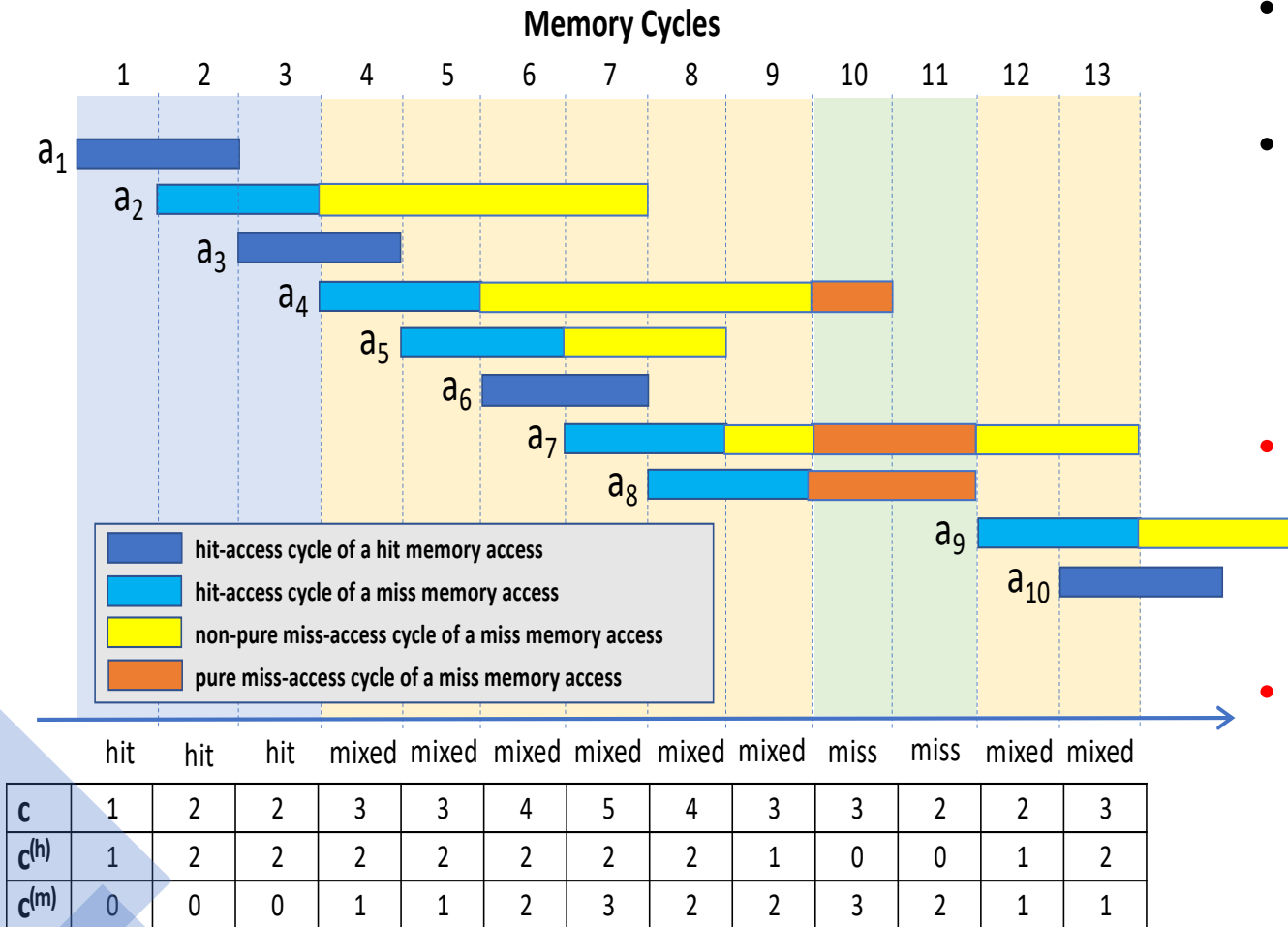
- The time at each cache (in terms of CPU cycles) can be divided into four categories depending on how memory accesses overlap:
 - \mathcal{E} is the set of memory inactive cycles: $e = |\mathcal{E}|$
 - \mathcal{H} is the set of pure hit cycles: $h = |\mathcal{H}|$
 - \mathcal{M} is the set of pure miss cycles: $m = |\mathcal{M}|$
 - \mathcal{X} is the set of mixed hit/miss cycles: $x = |\mathcal{X}|$
- Total cycles: $n = e + h + m + x$
- The last three are collectively called memory active cycles (Ω)
 - $\omega = h + m + x$

Time Measured in CPU Cycles at the Cache Device



where ω is the number of memory active cycles ($h+m+x$) and α is the number of memory accesses

Memory Access Concurrency



- Let $c_i^{(h)}$ be the **hit concurrency**, the number of overlapped hit-access cycles at CPU cycle i
- Let $c_i^{(m)}$ be the **miss concurrency**, the number of overlapped miss-access cycles at CPU cycle i
- Let c_i be the **memory access concurrency** (i.e., overlapped memory accesses) at CPU cycle i

$$c_i = c_i^{(h)} + c_i^{(m)}, \quad (1 \leq i \leq n)$$

- **Average hit concurrency:**

$$C_H = \frac{1}{h+x} \sum_{i \in H \cup X} c_i^{(h)} = \frac{1}{h+x} \sum_{i=1}^n c_i^{(h)}$$

- **Average pure miss concurrency:**

$$C_M = \frac{1}{m} \sum_{i \in M} c_i^{(m)}$$

- **Average memory access concurrency:**

$$C = \frac{1}{\omega} \sum_{i \in \Omega} c_i = \frac{1}{\omega} \sum_{i=1}^n c_i$$

Results – Easy Proofs

- **C-AMAT = AMAT / C**, where C is average memory access concurrency
- C-AMAT can be expressed by extending AMAT formulation:

$$C\text{-AMAT} = \frac{H}{C_H} + \rho_M \times \frac{pAMP}{C_M}$$

where ρ_M is the pure miss ratio and $pAMP$ is the average pure miss penalty

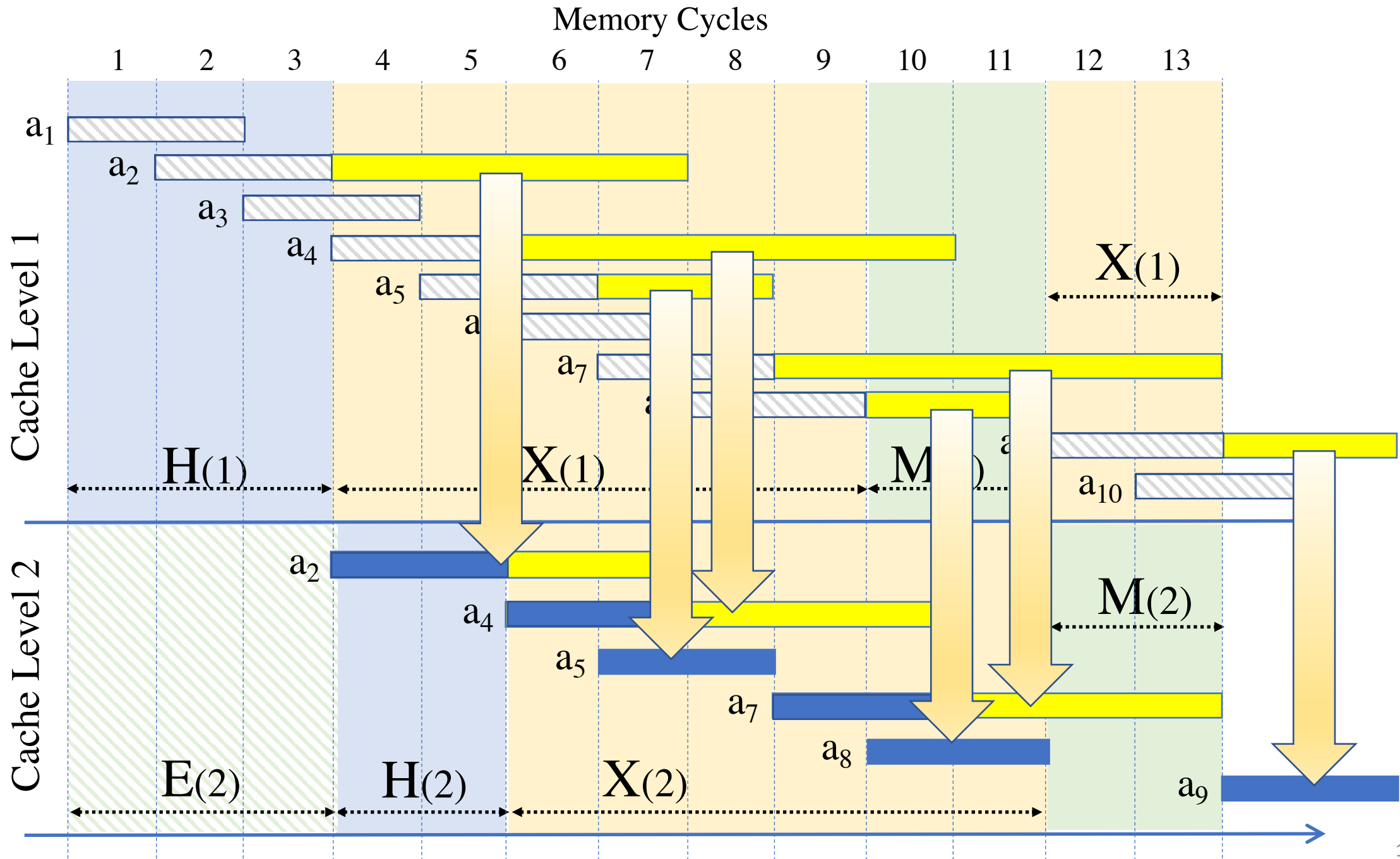
$$pAMP = \frac{\sum_{i \in M} c_i^{(m)}}{\alpha_M}$$

- Memory Stall Time (MST) per memory access:

$$MST = m / \alpha = \mu \times \kappa \times C\text{-AMAT}$$

where

$$\mu = \frac{m + x}{\omega}, \quad \kappa = \frac{m}{m + x}$$



Memory Hierarchy

- Three observations between cache level l and cache level $l+1$
 1. Memory accesses at next level are misses from the previous level

$$\alpha(l+1) = \alpha_m(l) = \alpha(l) \times \rho_m(l)$$

2. Only miss access cycles propagate to the next level

$$c_i(l+1) = c_i^{(m)}(l)$$

3. Pure hit cycles become memory inactive cycles at the next level

$$\begin{aligned}\omega(l+1) &= \omega(l) - h(l) = m(l) + x(l) \\ &= \left(\frac{m(l) + x(l)}{\omega(l)} \right) \times \omega(l) = \mu(l) \times \omega(l)\end{aligned}$$

Recursive Definitions of C-AMAT

- First recursive definition:


$$C-AMAT(l) = \frac{H(l)}{C_H(l)} + \rho_m(l) \times \kappa(l) \times C-AMAT(l+1)$$

$$\text{where } \kappa(l) = \frac{\rho_M(l)}{\rho_m(l)} \times \frac{pAMP(l)}{AMP(l)} \times \frac{C_m(l)}{C_M(l)} = \frac{m(l)}{m(l) + x(l)}$$

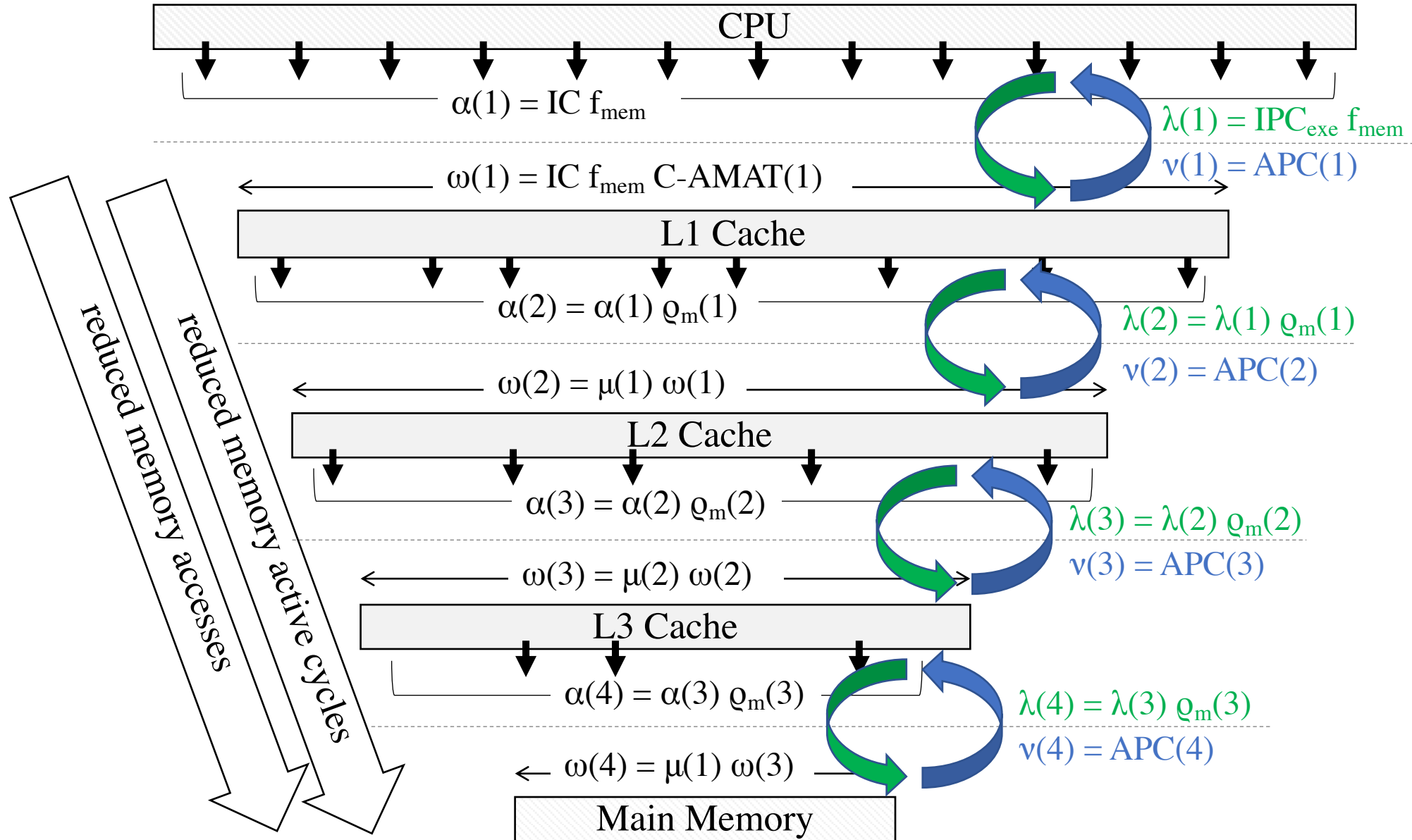
- Second recursive definition:

$$C-AMAT(l) = \frac{\rho_m(l)}{\mu(l)} \times C-AMAT(l+1).$$

$$\text{where } \mu(l) = \frac{m(l) + x(l)}{\omega(l)}$$


$$C-AMAT(l) = C-AMAT(1) \times \prod_{i=1}^{l-1} \frac{\mu(i)}{\rho_m(i)}$$

Layered Performance Matching



Recursive Definition of Matching Ratio

- Matching ratio is defined to be the ratio between the request rate and the supply rate:

$$LPMR(l) = \frac{\lambda(l)}{\nu(l)} = IPC_{\text{exe}} \times f_{\text{mem}} \times C\text{-}AMAT(1) \times \prod_{i=1}^{l-1} \mu(i)$$

- Let Δ be the ratio of memory access time over compute time

$$LPMR(l) = \frac{\Delta}{\mu(1) \times \kappa(1)} \times \prod_{i=1}^{l-1} \mu(i).$$

Performance Optimization as Recursive Process

• From
$$LPMR(l) = \frac{\Delta}{\mu(1) \times \kappa(1)} \times \prod_{i=1}^{l-1} \mu(i).$$

$l \nearrow LPMR(l) \searrow$

$LPMR(1) = \frac{\Delta}{\mu(1) \times \kappa(1)},$

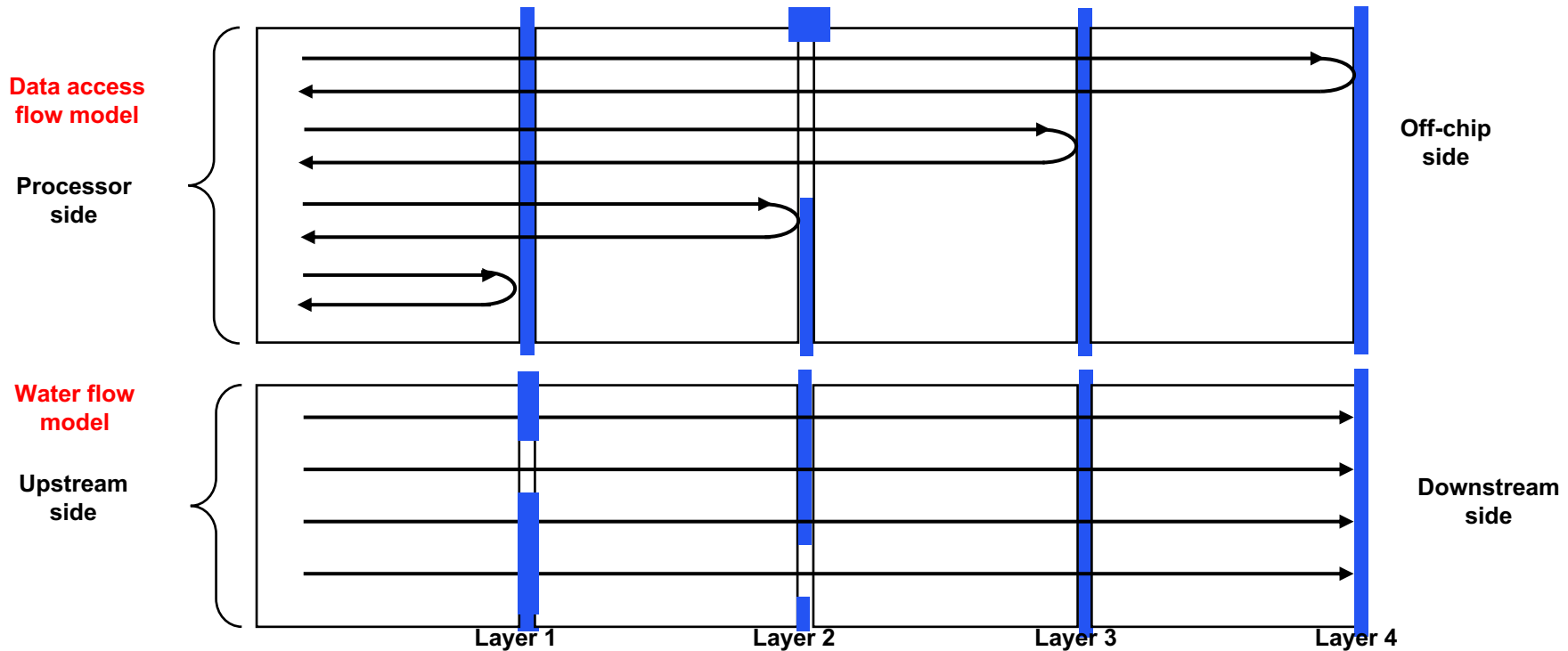
$LPMR(2) = \frac{\Delta}{\kappa(1)},$

$LPMR(3) = \frac{\mu(2) \times \Delta}{\kappa(1)}.$

- If **LPMR(1)** is below the target, all good.
- If **LPMR(1)** is above target, and **LPMR(2)** is below target, then we should optimize L1 cache.
- If both **LPMR(1)** and **LPMR(2)** are above target, and **LPMR(3)** is below target, then we should optimize L1 and L2 caches.
- Otherwise, all three cache levels need to be optimized.

Memory Sluice Gate Theory

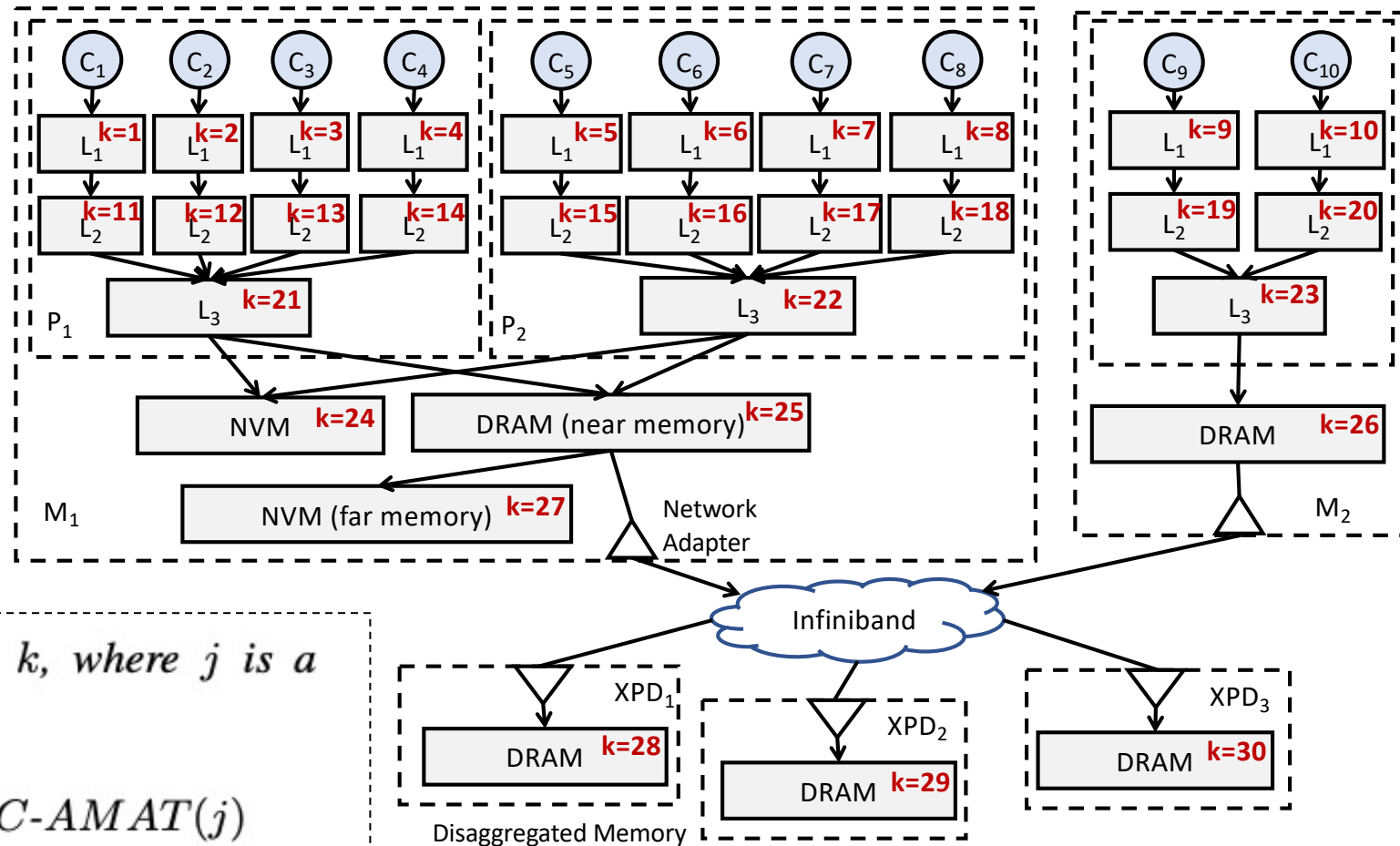
Sluice Gate Theorem: *If a memory system can match an application's data access requirement for any matching parameter $T_1 > 0$, then this memory system has removed the memory wall effect for this application.*



X.-H. Sun and Y.-H. Liu, "Utilizing Concurrency Data Access: A New Theory," in Proc. of LCPC2016, Sept. 2016, New York, USA

Generalized Model with Merging and Splitting (Ongoing Work)

- A memory system can be viewed as a multitree
 - Multitree is a DAG in which “a set of nodes reachable from any node induces a tree”



Theorem 1. For any memory units j and k , where j is a predecessor of k , i.e., $j \in \mathbb{P}(k)$, we have:

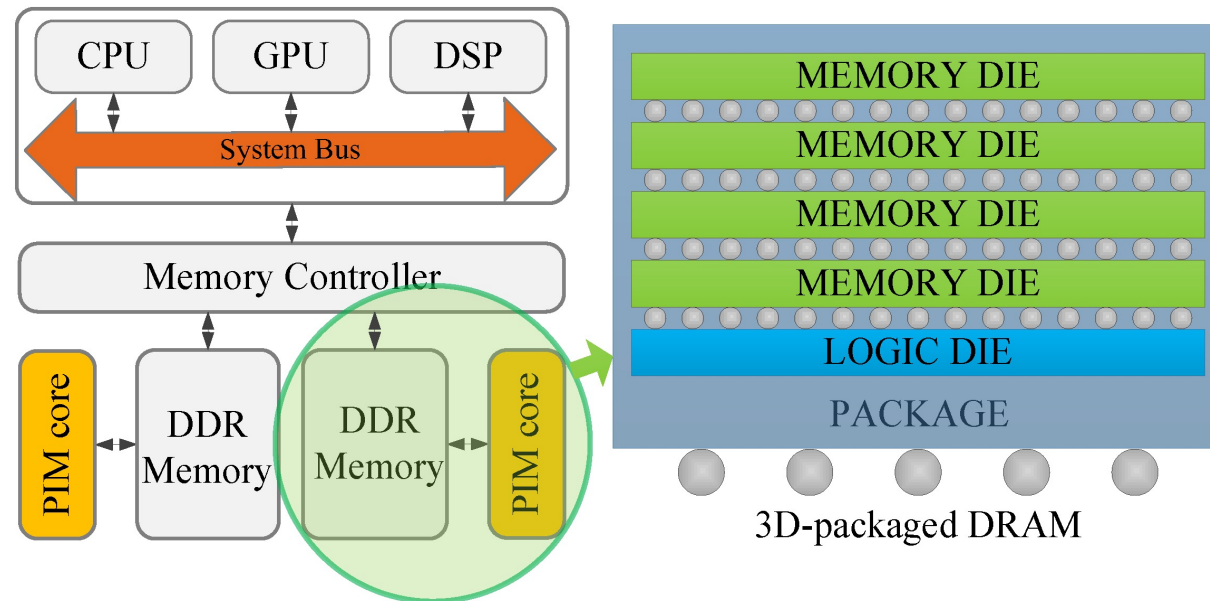
$$C-AMAT(k) = \frac{\psi_j(k) \cdot \eta_k(j) \cdot \mu_k(j)}{\psi_k(j) \cdot \eta_j(k) \cdot \rho_k(j)} \cdot C-AMAT(j)$$

Applications of the Theoretical Model

- Two questions:
 1. How to use the theoretical results in actual architecture design?
 2. How does it benefit architecture and system research community?
- Examples:
 - Multicore architecture (shared LLC)
 - GPU memory performance
 - Disaggregated memory systems (DMS)
- Design and implement simulators using the unified model to measure C-AMAT and LPM effects

PIM: A Part of the Sluice Gate Consideration

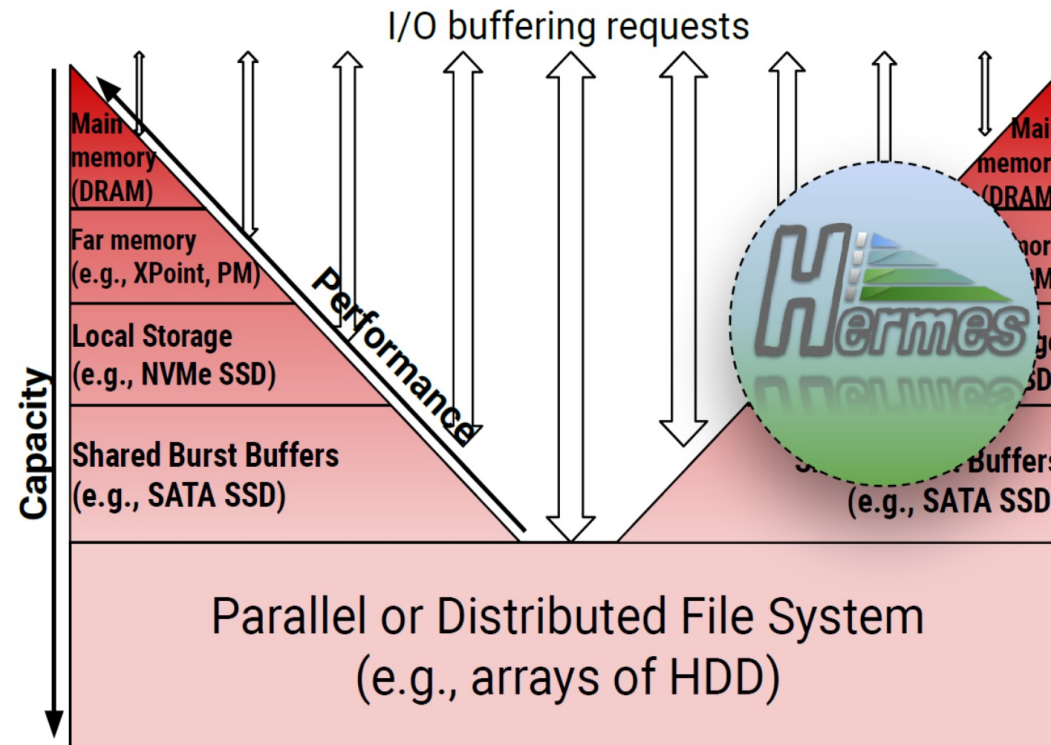
- PIM conduct computing at the memory side and reduce data movement
- PIM is less powerful and when can help is an issue
- When to use PIM is part of the Sluice Gate theory
- Similar discussions for NDP (near data processing) and ISP (in storage processing)



L Yan, M. Zhang, R. Wang, X. Chen, X. Zou, X. Lu, H Han, and X.-H. Sun, "CoPIM: A Concurrency-aware PIM Workload Offloading Architecture for Graph Applications," in the proceedings of the 2021 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED2021), July 26 – 28, 2021 (Virtual)

Hermes: A Multi-tiered I/O Buffering System

- Selective cache, concurrent, matching
- Independent management of each tier



A. Kougkas, H. Devarajan, and X.-H. Sun, "I/O Acceleration via Multi-Tiered Data Buffering and Prefetching," Journal of Computer Science and Technology, vol. 35, no. 1, pp. 92-120, Jan. 2020

Conclusion and Future Work

The unified framework establishes a coherent and enhanced math foundation for several existing performance models: C-AMAT, APC, LPM.

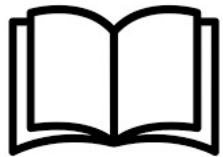
Limitations

- Cache write-backs
- Averages vs. transient behaviors
- Measurement and decision making in real time

Ongoing and future work

- Multicore, GPU, disaggregate memory
- Evaluating cache bypass
- Evaluating the impact of writebacks

Thank you!



Jason Liu, Pedro Espina, and Xian-He Sun. **A Study on Modeling and Optimization of Memory Systems.** *J. Comput. Sci. Technol.* **36**, 71–89 (2021). <https://doi.org/10.1007/s11390-021-0771-8>



We thank our sponsor, National Science Foundation, for the support of this work via grants CCF-2008000 and CCF-2008907.