Memory, New Techs and Bandwidth Challenges for Future HPC-Systems

- Jeffrey Vetter
  Oak Ridge National Laboratory
- Thorsten Hoefler
  ETH Zurich
- Ron Brightwell
  Sandia National Laboratories
- Xian-He Sun (Session Chair)
  Illinois Institute of Technology

Deep Memory-Storage Hierarchy and Pace-Matching Data Access

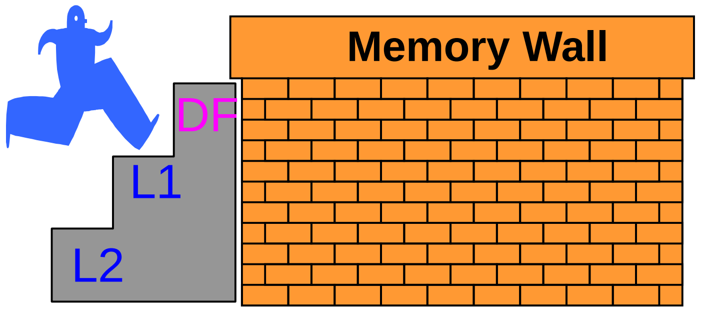
Xian-He Sun

Illinois Institute of Technology
sun@iit.edu

A Invited Talk at ISC2019

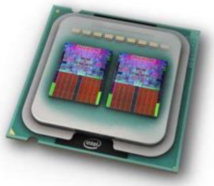
Hot Issues

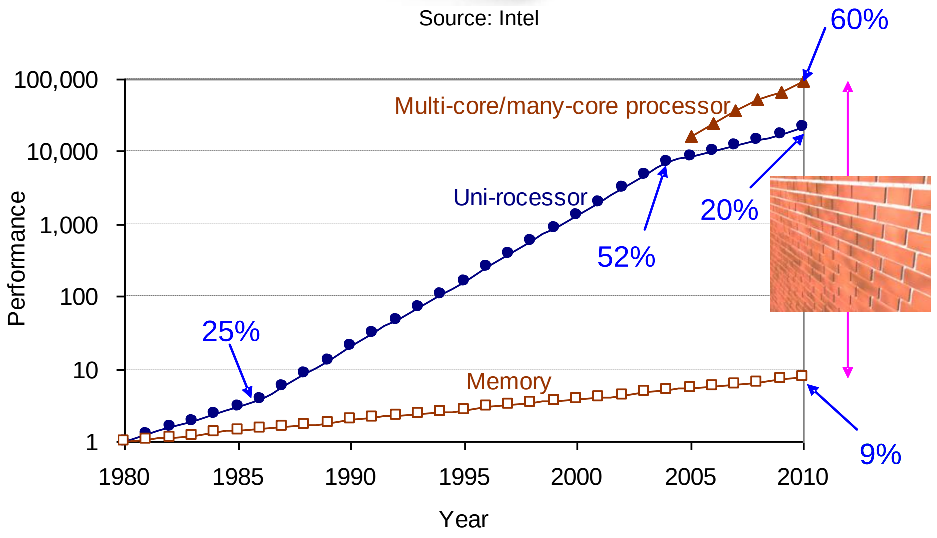
- Big Data
- *High Performance and Could Computing*
- AI and Deep Learning

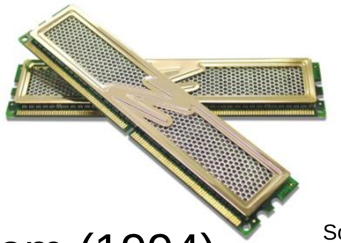


The Memory-wall Problem

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004
  - Aggregate multi-core/many-core processor performance even higher since 2004
- Memory: ~9% per year
  - Storage: ~6% per year
- Processor-memory speed gap keeps increasing

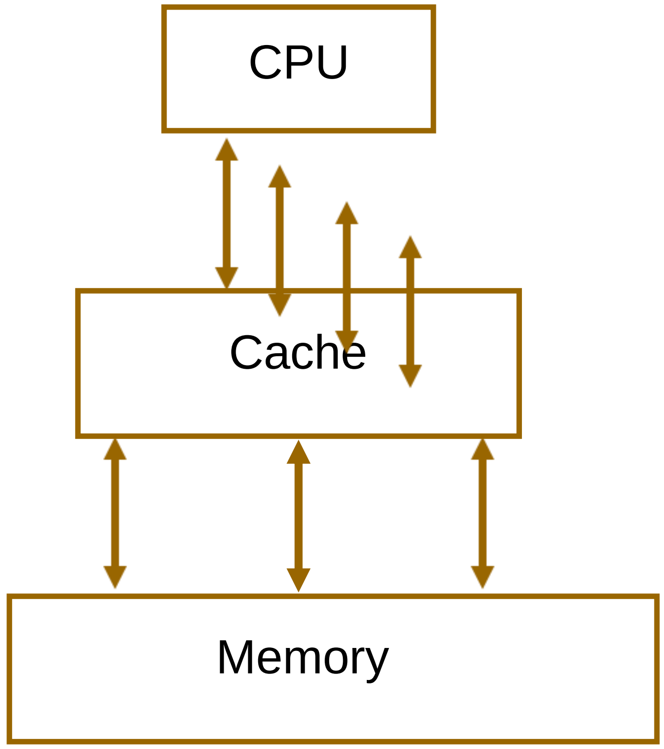

Source: Intel




Source: OCZ

Memory-bounded speedup (1990), Memory wall problem (1994)

Solution: Memory Hierarchy & Concurrency



Multi-core
Multi-threading
Multi-issue

Out-of-order Execution
Speculative Execution
Runahead Execution

Multi-banked Cache
Multi-level Cache

Pipelined Cache
Non-blocking Cache
Data Prefetching
Write buffer

Multi-channel
Multi-rank
Multi-bank

Pipeline
Non-blocking
Prefetching
Write buffer

Input-Output (I/O)

Parallel File System

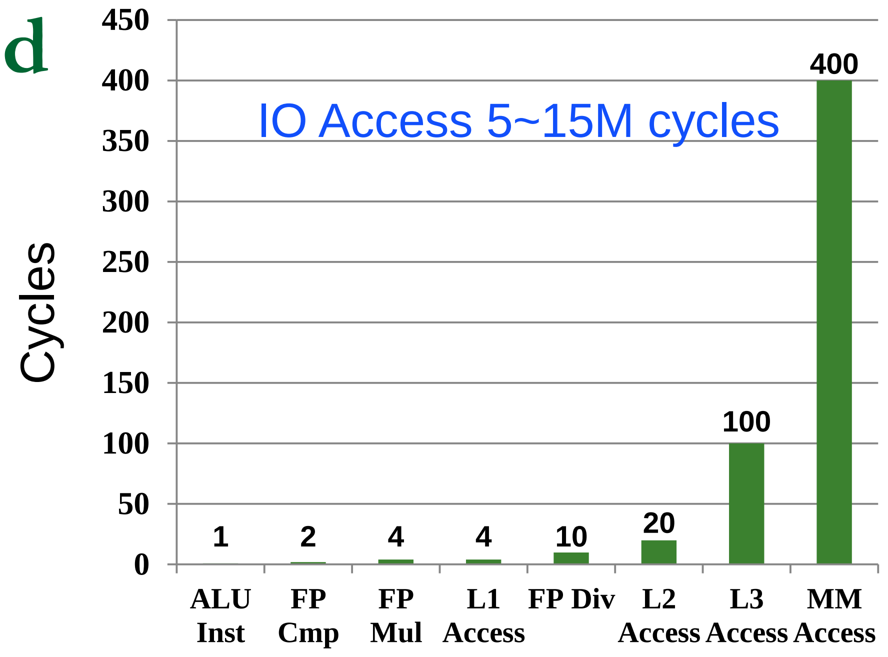
Disks

Assumption of Current Solutions

- Memory Hierarchy: Locality
- Concurrence: Data access pattern
  - Data stream

Extremely Unbalanced Operation Latency

Performances vary largely



Existing Memory Metrics

- Miss Rate(MR)
  - {the number of miss memory accesses} over {the number of total memory accesses}
- Misses Per Kilo-Instructions(MPKI)
  - {the number of miss memory accesses} over {the number of total committed Instructions × 1000}
- Average Miss Penalty(AMP)
  - {the summary of single miss latency} over {the number of miss memory accesses}
- Average Memory Access Time (AMAT)
  - **AMAT = Hit time + MR × AMP**
- Flaw of Existing Metrics
  - Focus on a single component or
  - A single memory access

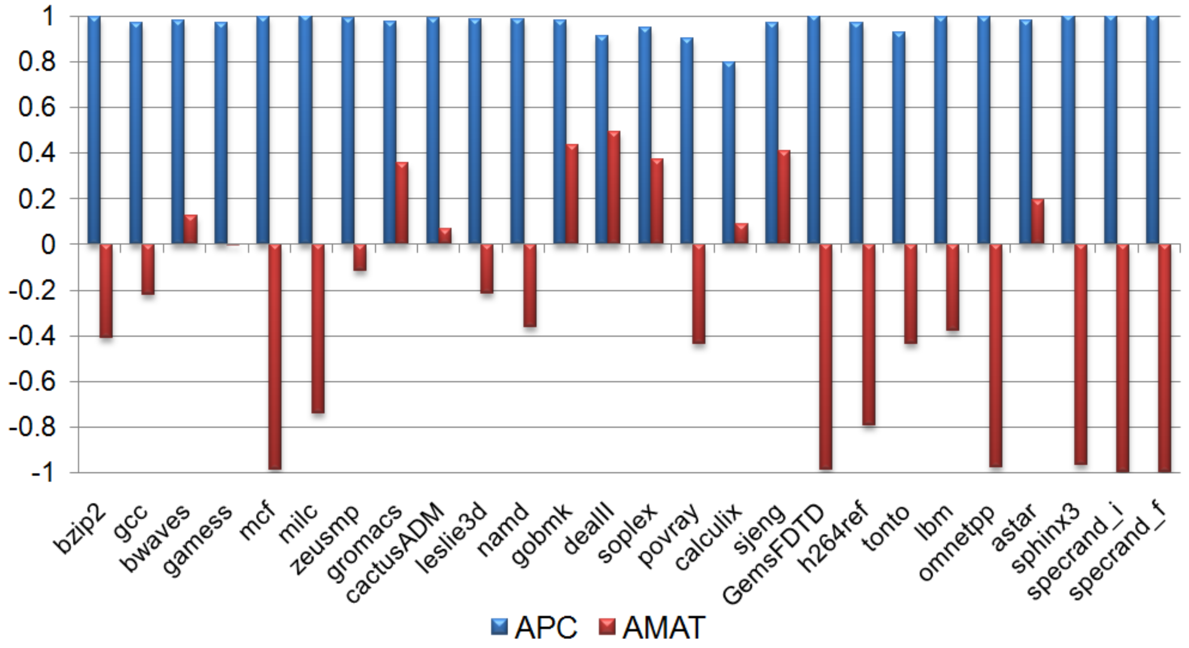
Missing memory parallelism/concurrency

APC: a concurrent measurement from memory side

- Access Per memory active Cycle (APC)
  - APC = A/T
- APC is measured as the number of memory accesses per memory active cycle or Access Per Memory Active Cycle (APMAC)
  - Measure T based on *memory (active) cycle*
  - Measure A based on the *overlapping mode*
- Benefits of APC
  - Separate memory evaluation from CPU evaluation
  - **Each memory level** has its own APC value

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

APC & IPC: Changing Cache Parallelism



- Changing the number of MSHR entries (1→2→10→16)
- APC still has the dominant correlation, with average value of 0.9656
- AMAT does not correlate with IPC for most applications
  - APC record the CPU blocked cycles by MSHR cycles
  - AMAT cannot records block cycles, it only measure the issued memory requests

D. Wang, X.-H. Sun "Memory Access Cycle and the Measurement of Memory Systems", IEEE Transactions on Computers, vol. 63, no. 7, pp. 1626-1639, July.2014

Concurrent-AMAT: step to optimization

- The traditional AMAT(Average Memory Access Time) :

  AMAT = HitCycle + MR$\times$AMP

- MR is the miss rate of cache accesses; and AMP is the average miss penalty

- **Concurrent-AMAT (C-AMAT)**:

  $$\text{C-AMAT} = \text{HitCycle}/C_H + p\text{MR} \times p\text{AMP}/C_M = 1/\text{APC}$$

- C_H is the hit concurrency; C_M is the *pure* miss concurrency
- *p*MR and *p*AMP are *pure* miss rate and average *pure* miss penalty
- A pure miss is a miss containing at least one cycle which does not have any hit activity

X.-H. Sun and D. Wang, "Concurrent Average Memory Access Time", in ***IEEE Computers***, vol. 47, no. 5, pp. 74-80,May 2014.(IIT Technical Report, IIT/CS-SCS-2012-05)

Recursive in Memory Hierarchy

- AMAT is recursive
  - AMAT = $HitCycle_1 + MR_1 \times AMP_1$

    Where $AMP_1 = (HitCycle_2 + MR_2 \times AMP_2)$
  - AMAT = $HitCycle + MR \times (H2 + MR2 \times (H3 + MR3 \times AMP3))$

- C-AMAT is also recursive

$$C\text{-}AMAT_1 = \frac{H_1}{C_{H_1}} + MR_1 \times \kappa_1 \times C\text{-}AMAT_2$$

Where

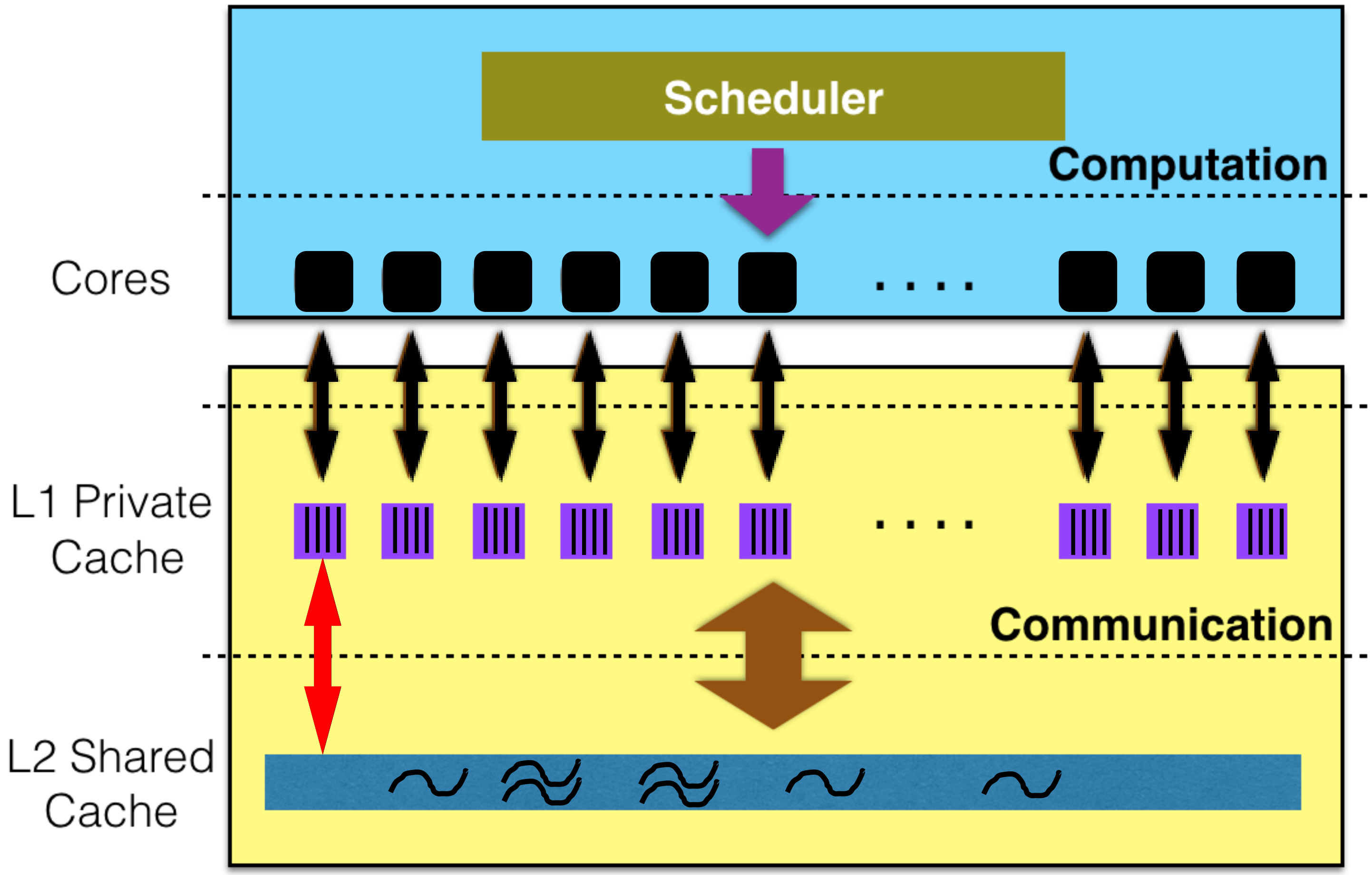
$$C\text{-}AMAT_2 = \frac{H_2}{C_{H_2}} + pMR_2 \times \frac{pAMP_2}{C_{M_2}}$$

$$\kappa_1 = \frac{pMR_1}{MR_1} \times \frac{pAMP_1}{AMP_1} \times \frac{C_{m_1}}{C_{M_1}}$$

With Clear Physical Meaning

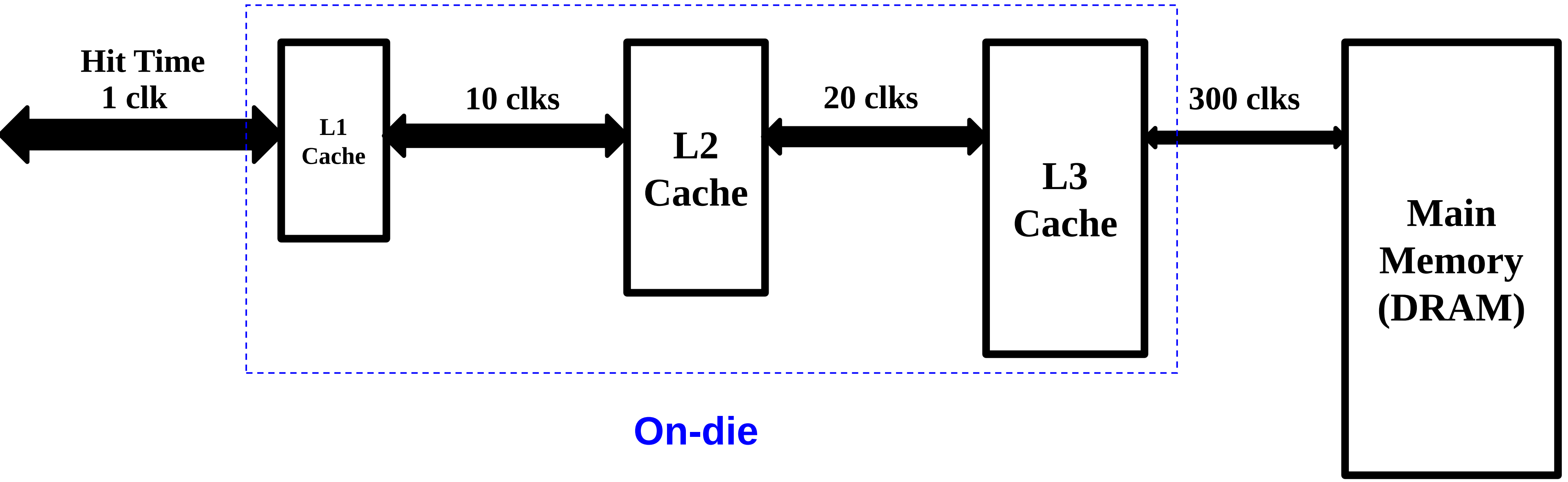
X.-H. Sun, "Concurrent-AMAT: a mathematical model for Big Data access," HPC-Magazine, May 12, 2014

C-AMAT in Multicore Environments



Separation in shared environments

Data Access Time: AMAT

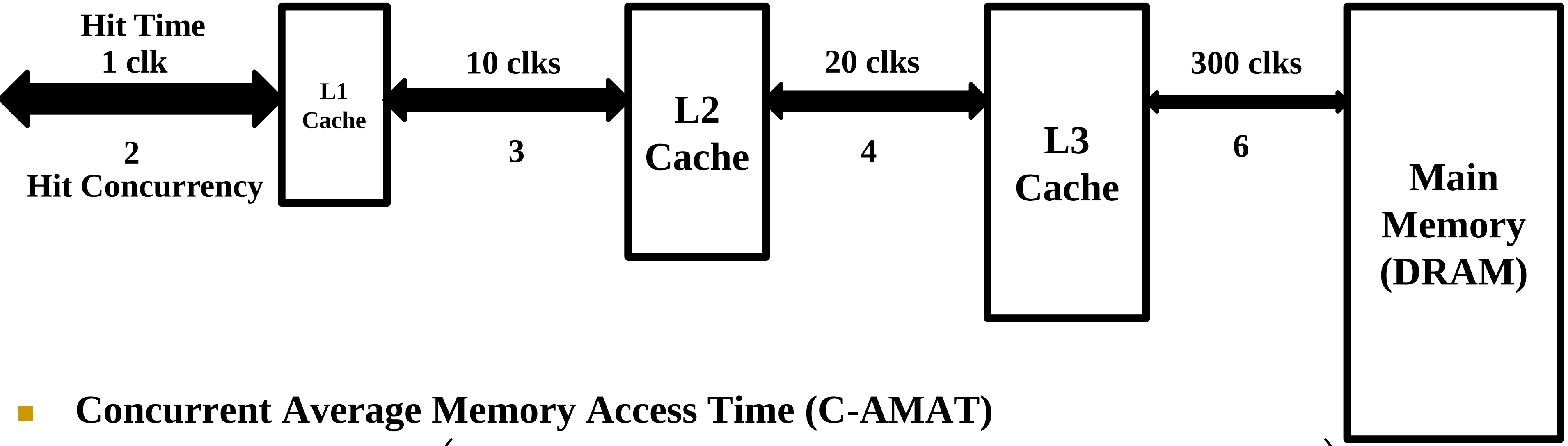


- Average Memory Access Time (AMAT)

  $= T_{hit}(L1) + Miss\%(L1)^* (T_{hit}(L2) + Miss\%(L2)^* (T_{hit}(L3) + Miss\%(L3)^*T(memory)))$
- Example: (Latency as shown above)
  - *Miss rate: L1=10%, L2=5%, L3=1%* (*Be careful miss rate definition*)
  - AMAT

    = 2.115

Data Access Time: C-AMAT



- **Concurrent Average Memory Access Time (C-AMAT)**

$$= \frac{H_1}{C_{H_1}} + MR_1 \times \kappa_1 \times \left(\frac{H_2}{C_{H_2}} + MR_2 \times \kappa_2 \times \left(\frac{H_3}{C_{H_3}} + MR_3 \times \kappa_3 \times \frac{H_{Mem}}{C_{H_{Mem}}} \right) \right)$$

- **Example**
  - *Miss Rate: L1=10%, L2=5%, L3=1%*
  - *κ: L1=0.56, L2=0.6, L3=0.8*
  - *C-AMAT≈0.696*

pMR, pAMP, AMP, C_M, C_m: L1=7%, 10, 10, 5, 4
L2=3%, 60, 40, 9, 6
L3=0.8%, 400, 300, 16, 12

Technique Impact Analysis (with C-AMAT)

Classes	Items	IssueRatio	MR	pMR	AMP	pAMP	C_H	C_M	AMAT	C-AMAT$_{stall}$
Hardware techniques	Pipelined cache access	+		⊕	−	⊕	⊕		−	⊕
	Non-blocking caches	+		⊕		⊕		⊕		⊕
	Multi-banked caches	+		⊕		⊕	⊕	⊕		⊕
	Large IW & ROB, Runahead	+		⊕		⊕	⊕	⊕		⊕
	SMT	+	−		−	⊕	⊕	⊕	−	⊕
Compiler techniques	Loop Interchange		+	⊕					+	⊕
	Matrices blocking		+	⊕					+	⊕
	Data and control dependency related optimization						⊕	⊕		⊕
Application techniques	Copy data into local scalar variables and operate on local copies		+	⊕	+	⊕			+	⊕
	Vectorize the code		+	⊕	+	⊕			+	⊕
	Split structs into hot and cold parts, where the hot part has a pointer to the cold part		+	⊕	+	⊕			+	⊕

+ or ⊕ means that the technique improves the factor, − means hurts the factor, and blank means it has no necessary impact. These notions are used in the same manner as that of Hennessy and Patterson [6].

- + means from AMAT (included by C-AMAT too), ⊕ means from C-AMAT
- C-AMAT unifies the combined impact of locality and concurrency, and makes concurrency contribution measureable

What does C-AMAT says?

Optimal = ~~Optimal Locality~~ ?

Optimal = ~~Optimal Locality + Optimal Concurrence~~ ?

What Does C-AMAT Say?

- C-AMAT is an extension of AMAT to consider concurrency
  - C-AMAT can be measured at **each layer** with **APC**
- C-AMAT is data-centric thinking
  - Data access is as important as computing
- **High locality may hurt performance**
  - The Pure Miss concept
- **Balance** locality, concurrency, overlapping with C-AMAT
- C-AMAT uniquely integrates the **joint impact** of locality, concurrency, and overlapping for optimization (analysis and measurement)

Application: Memory stall time *(the performance we care)*

Traditional AMAT model

$$CPU\text{-}time = IC \times (CPI_{exe} + \underbrace{f_{mem} \times AMAT}_{\text{Memory stall time}}) \times Cycle\text{-}time$$

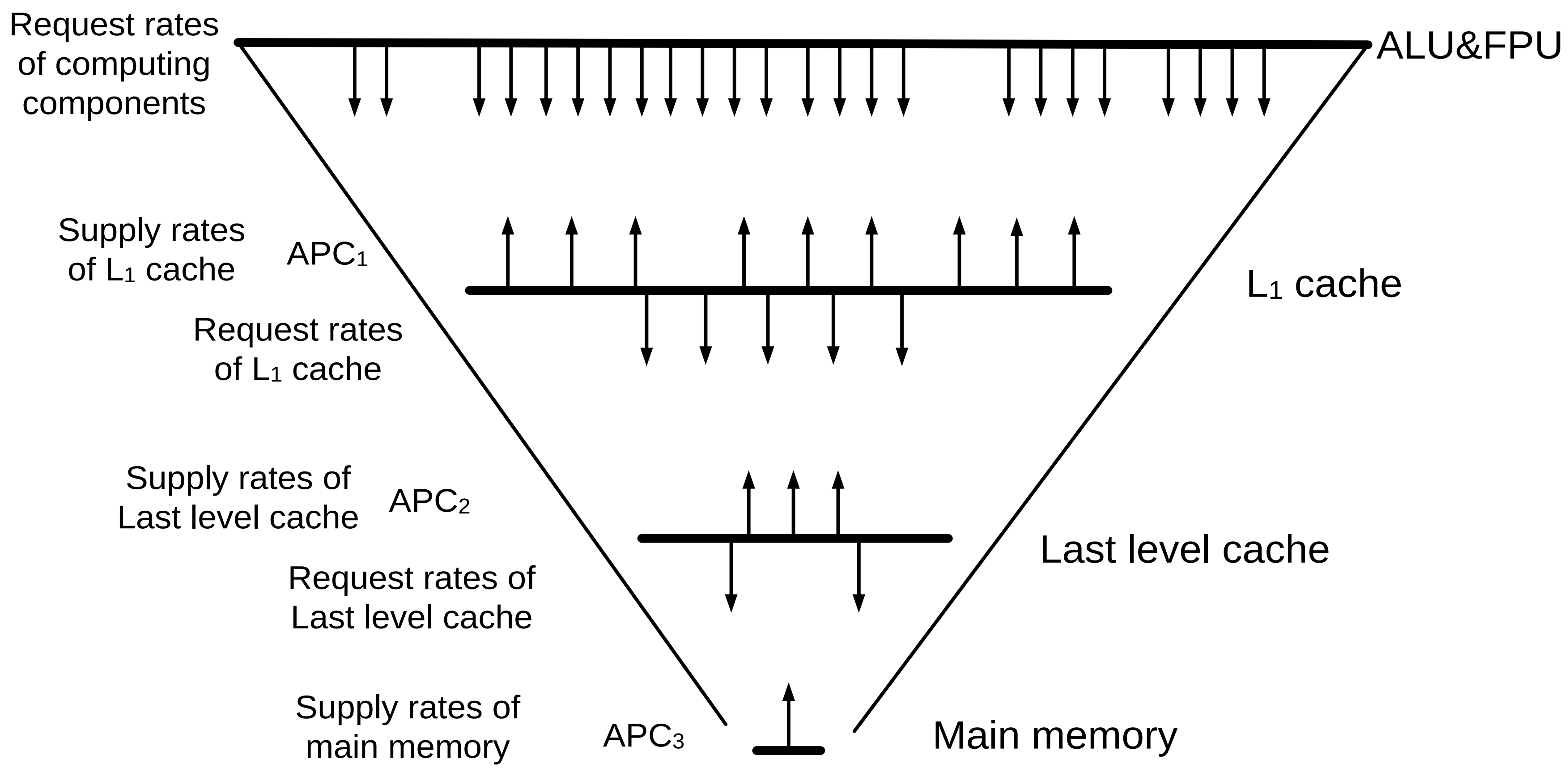
New C-AMAT model

$$Exec-time = IC \times (CPI_{exe} + \underbrace{f_{mem} \times C-AMAT \times (1-overlapRatio_{c\text{-}m})}_{\text{Memory stall time}}) \times cycle-time$$

$$CPU\text{-}time = IC \times \left(CPI_{exe} + \underbrace{f_{mem} \times \frac{pMR \times pAMP}{C_M}}_{\text{Memory stall time}}\right) \times Cycle\text{-}time$$

Only pure miss will cause processor stall, and the penalty is formulated here

Application: Layered Performance Matching


Request rates of computing components
ALU&FPU
Supply rates of L1 cache
APC1
L1 cache
Request rates of L1 cache
Supply rates of Last level cache
APC2
Last level cache
Request rates of Last level cache
Supply rates of main memory
APC3
Main memory

Yu-Hang Liu, Xian-He Sun, "*LPM: Concurrency-driven Layered Performance Matching*," in ICPP2015, Beijing, China, Sept. 2015.

Quantify Mismatching: with C-AMAT

$$LPMR_1 = \frac{IPC_{exe} \times f_{mem}}{APC}$$

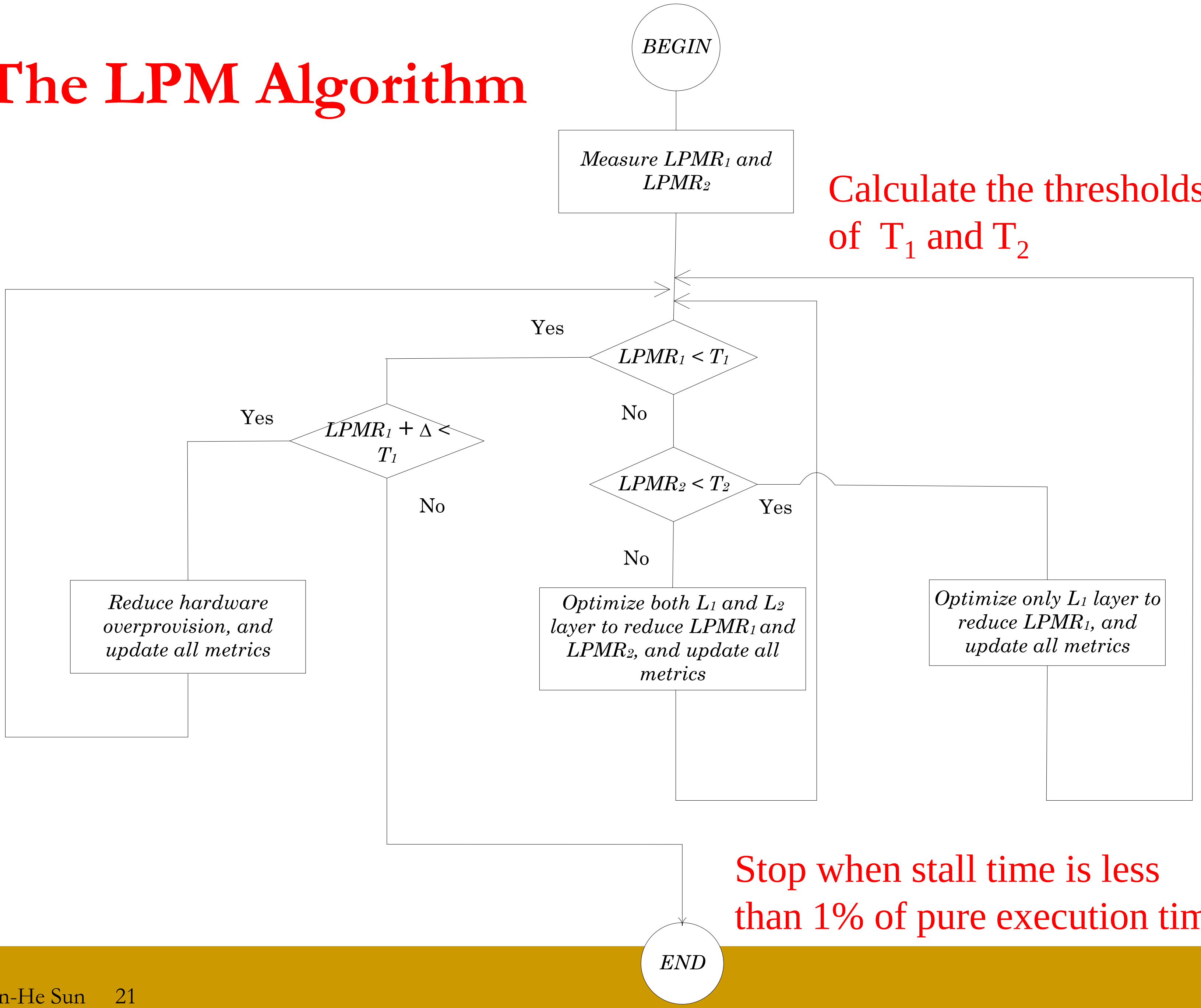
$$LPMR_2 = \frac{IPC_{exe} \times f_{mem} \times MR_1}{APC_2}$$

$$LPMR_3 = \frac{IPC_{exe} \times f_{mem} \times MR_1 \times MR_2}{APC_3}$$

- C-AMAT measures the request and supply at each layer
- C-AMAT can increase supply with effective concurrency
- Mismatch ratio directly determines memory stall time

Y. Liu, X.-H. Sun. "LPM: A Systematic Methodology for Concurrent Data Access Pattern Optimization from a Matching Perspective," IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS), June 2019

The LPM Algorithm


BEGIN
Measure $LPMR_1$ and $LPMR_2$
$LPMR_1 < T_1$
Yes
No
$LPMR_1 + \Delta < T_1$
Yes
No
$LPMR_2 < T_2$
Yes
No
Reduce hardware overprovision, and update all metrics
Optimize both L_1 and L_2 layer to reduce $LPMR_1$ and $LPMR_2$, and update all metrics
Optimize only L_1 layer to reduce $LPMR_1$, and update all metrics
END

Calculate the thresholds of T_1 and T_2

Stop when stall time is less than 1% of pure execution time

C-AMAT in Action

New C-AMAT model

$$CPU\text{−}time = IC \times \left(CPI_{exe} + \underline{f_{mem} \times \frac{pMR \times pAMP}{C_M}} \right) \times Cycle\text{−}time$$

Memory stall time

Only pure miss will cause processor stall, and the penalty is formulated here

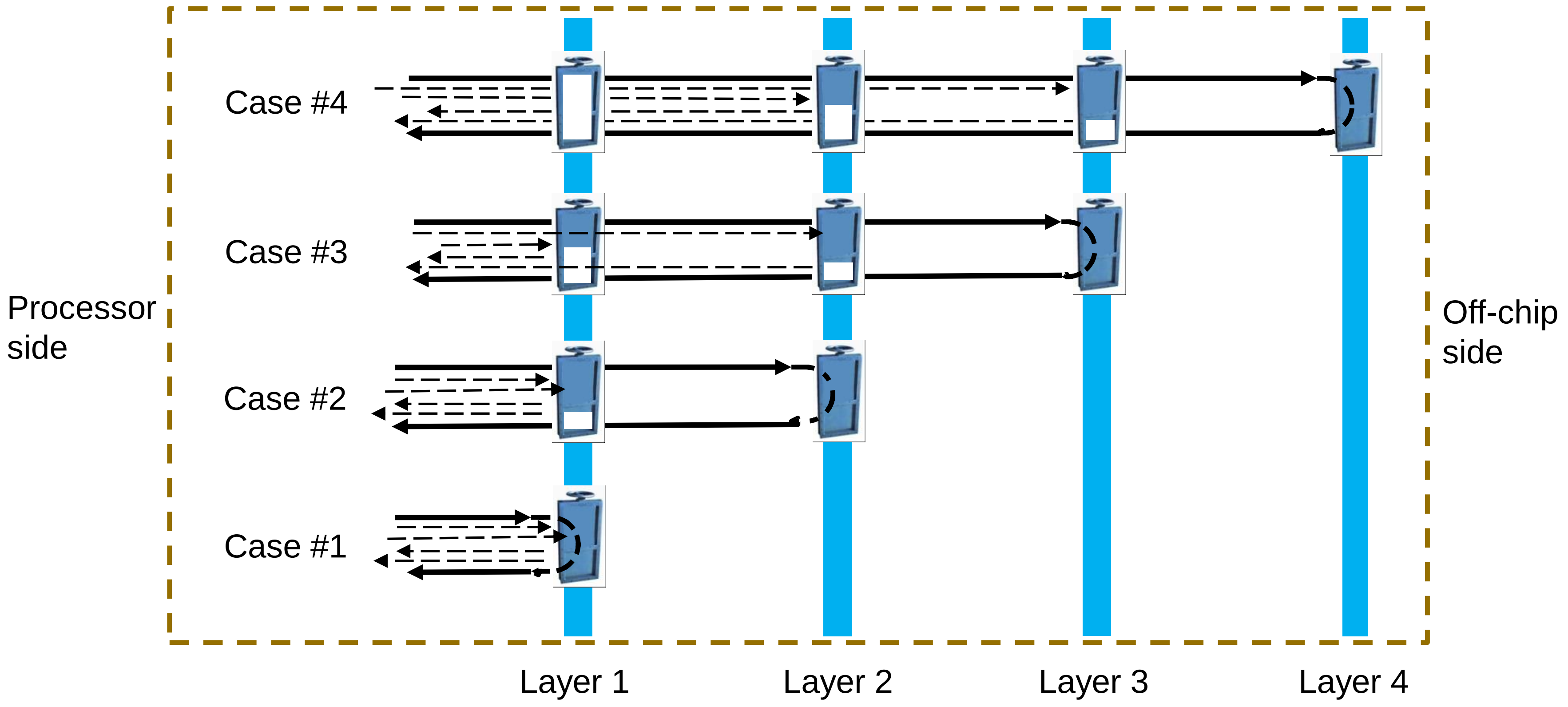
The Relation of LPMR and Stall time

$$CPU - time = IC \times CPI_{exe} \times \underline{(1 + \kappa_1 \times LPMR_2)} \times Cycle - time$$

Memory stall time

Pace Matching Data Access（搏动数据获取）

No delay data access (*using C-AMAT as the gate to guide and LPM as the global controller for in-situ optimization*)



Sun, Xian-He, and Yu-Hang Liu. "Utilizing Concurrency: A New Theory for Memory Wall." In International Workshop on Languages and Compilers for Parallel Computing, pp. 18-23. Springer, Cham, 2016.

Case study I: Eliminate memory-wall impact

LPM Optimization on Reconfigurable Architecture: $T_1 = 1.52$, $T_2 = 2.14$

Configuration		*A*	*B*	*C*	*D*	*E*
Sluice Width	Pipeline issue width	4	4	6	8	8
	IW size	32	64	64	128	96
	ROB size	32	64	64	128	96
	L_1 cache port number	1	1	2	4	4
	MSHR numbers	4	8	16	16	16
	L_2 cache interleaving	4	8	8	8	8
Mismatching degree	$LPMR_1$	8.1	6.2	2.1	1.2	1.4
	$LPMR_2$	9.6	9.3	3.1	1.6	1.9

Increased data access performance for more than **150 times** with the LPM algorithm

Case I Discussion

- GEM5 & DRAMSim2 are integrated with added C-AMAT component
  - 410.bwaves benchmark from SPEC CPU 2006

Memory-wall Removed !!!

- *Stall* time was > 60%, optimized to **< 1%**
  - **Stall time reduction** (memory performance improvement) is **150 times**
    Execution time speedup **2.5** (100/40)
  - If beginning is 70%, then speedup is 230 times (0.7/0.003)
  - If beginning is 90%, then speedup is **900 times** (0.9/0.001)

- The stall time reduction
  - Application dependent
  - Including computing and data access overlapping
  - LPM can be used in **task scheduling** in a heterogeneous environment
  - Can be used to determine the optimal number of layers

Memory Sluice Gate Theory

- It is mathematically **correct**, but under the assumptions
  - The application has sufficient data locality & concurrency
  - The system has sufficient hardware to support the data locality & concurrency
- The architecture needs to be **elastic**
  - Even for a given application may have different data access patterns
- It is **a framework** for solving the memory-wall problem
  - Do not need to wait for technology improvement
  - Guide technology improvements

Y. Liu, X.-H. Sun. "LPM: A Systematic Methodology for Concurrent Data Access Pattern Optimization from a Matching Perspective," IEEE Transactions on Parallel and Distributed Systems (TPDS), June 2019.

Application of Sluice-Gate Pace Matching

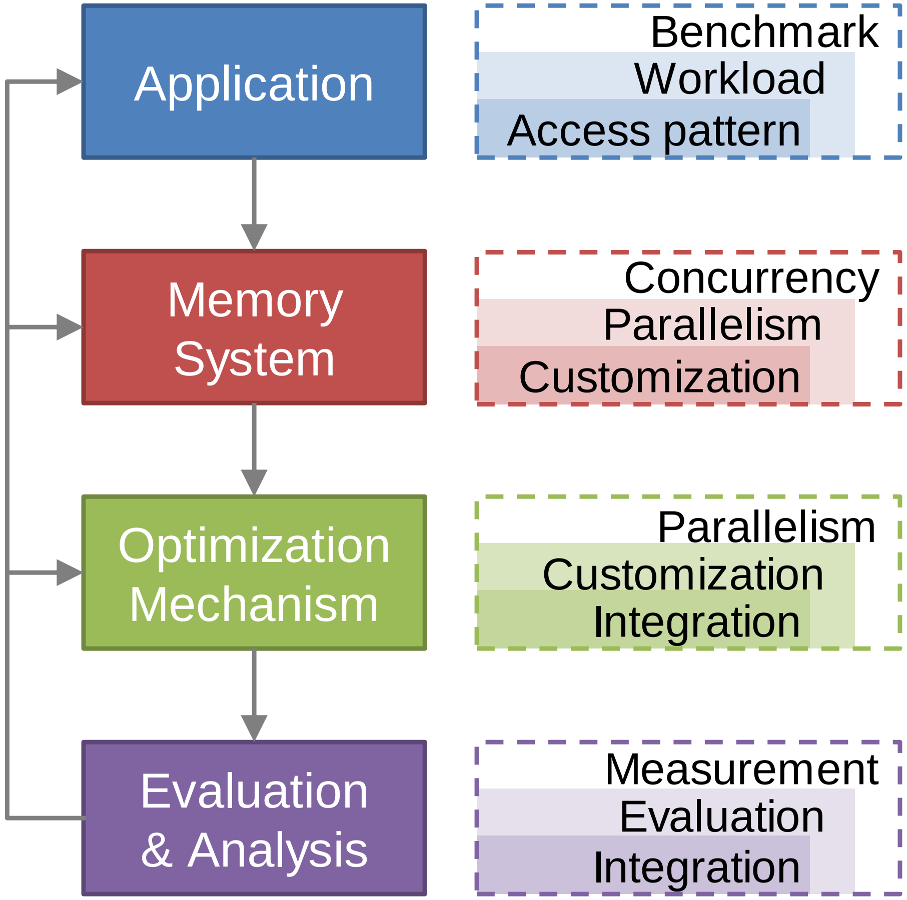
- Architecture Design and Configuration
  - Co-Design for data intensive computing
  - FPGA, ASIC, GPU utilization
- System Design and Optimization
  - Deep memory hierarchy
  - Data concurrency considered scheduling and optimization
  - Compiler technology
- Algorithm Design and Optimization
  - Explore data concurrency
  - Memory-centric programming

- **File system is the last level of memory**

Y-H Liu & Xian-He Sun, "*C^2-bound: A Capacity and Concurrency driven Analytical Model for Manycore Design*," in Proc. of the ACM/IEEE SC'15, Austin, USA, Nov. 2015.

Current Work: Deep Memory-Storage Hierarchy

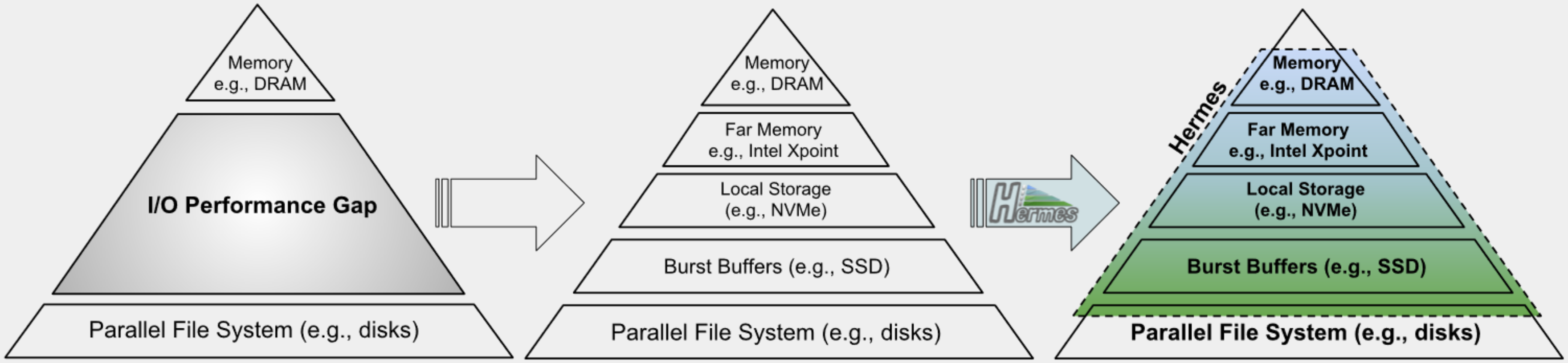
- Application-aware I/O optimization (HDF5)
- Smart, selective, multi-layers, software-hardware, memory-IO
- (Dynamic) Customized optimization
- Following the C-AMAT memory and path-matchng model



Kougkas, A., H. Devarajan, and X.-H. Sun. "Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system," in Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (ACM HPDC), pp. 219-230, ACM, 2018.

Hermes Overview

- A new, multi-tiered, distributed caching platform that:
  - Enables, manages, and supervises I/O operations in the Deep Memory and Storage Hierarchy (DMSH).
  - Offers selective and dynamic layered data placement/replacement
  - Is modular, extensible, and performance-oriented.
  - Supports a wide variety of applications (scientific, BigData, etc.,).


Memory e.g., DRAM
I/O Performance Gap
Parallel File System (e.g., disks)
Memory e.g., DRAM
Far Memory e.g., Intel Xpoint
Local Storage (e.g., NVMe)
Burst Buffers (e.g., SSD)
Parallel File System (e.g., disks)
Hermes
Hermes
Memory e.g., DRAM
Far Memory e.g., Intel Xpoint
Local Storage (e.g., NVMe)
Burst Buffers (e.g., SSD)
Parallel File System (e.g., disks)

Conclusion

- Big data is asking us to rethinking of memory system design
- C-AMAT, LPM, & sluice gate data transfer are new thoughts to meet the needs
- Hermes is a system which breaks the rank of memory & storage
- More challenges and opportunities toward the data-centric system design

TOO MANY THINGS NEED TO DO
FROM HARDWARE TO SOFTWARE