Memory Performance and Optimization

- Locality
- Concurrency
- AMAT
- C-AMAT

Class Schedule

- Review Nov. 7, 2019
- Nov. 12, Regular class, Exam Monday, Nov. 14, 2019
- The week of Nov. 18, no class, individual prepare term project
- Term project presentation: Monday Nov. 25 during and after class hour., Tuesday, Nov. 26 (back up)
- Term report due: Nov. 26

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)**:

  $$C\text{-}AMAT = HitCycle/C_H + pMR \times pAMP/C_M = 1/APC$$

- 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 $AMP_1 = (HitCycle_2 + MR_2 \times AMP_2)$

- 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

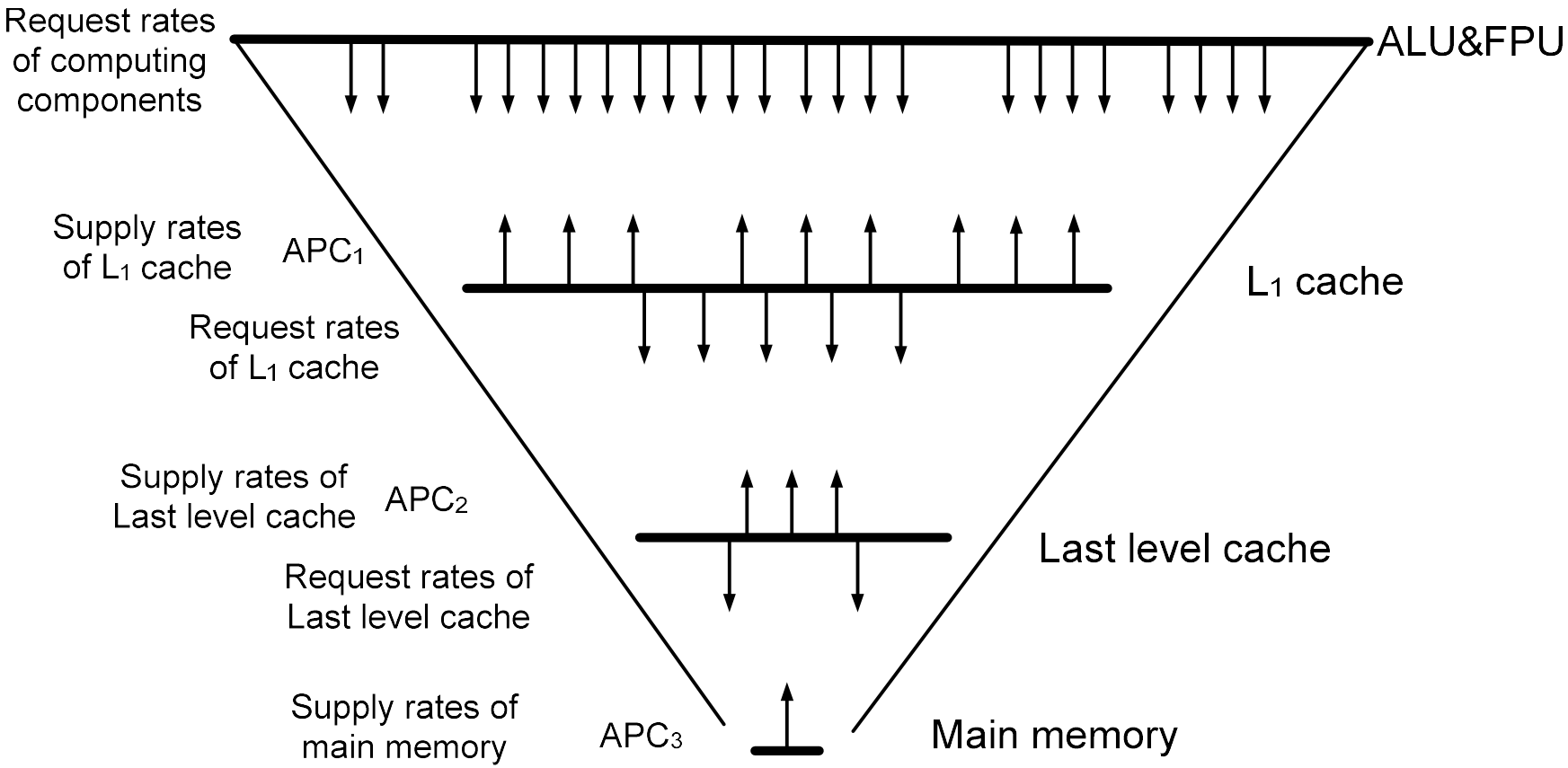
$$C\text{-}AMAT_2 = \frac{H_2}{C_{H_2}} + pMR_2 \times \frac{pAMP_2}{C_{M_2}}$$

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

With Clear Physical Meaning

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

Application: Layered Performance Matching



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

Quantify Mismatching: with C-AMAT

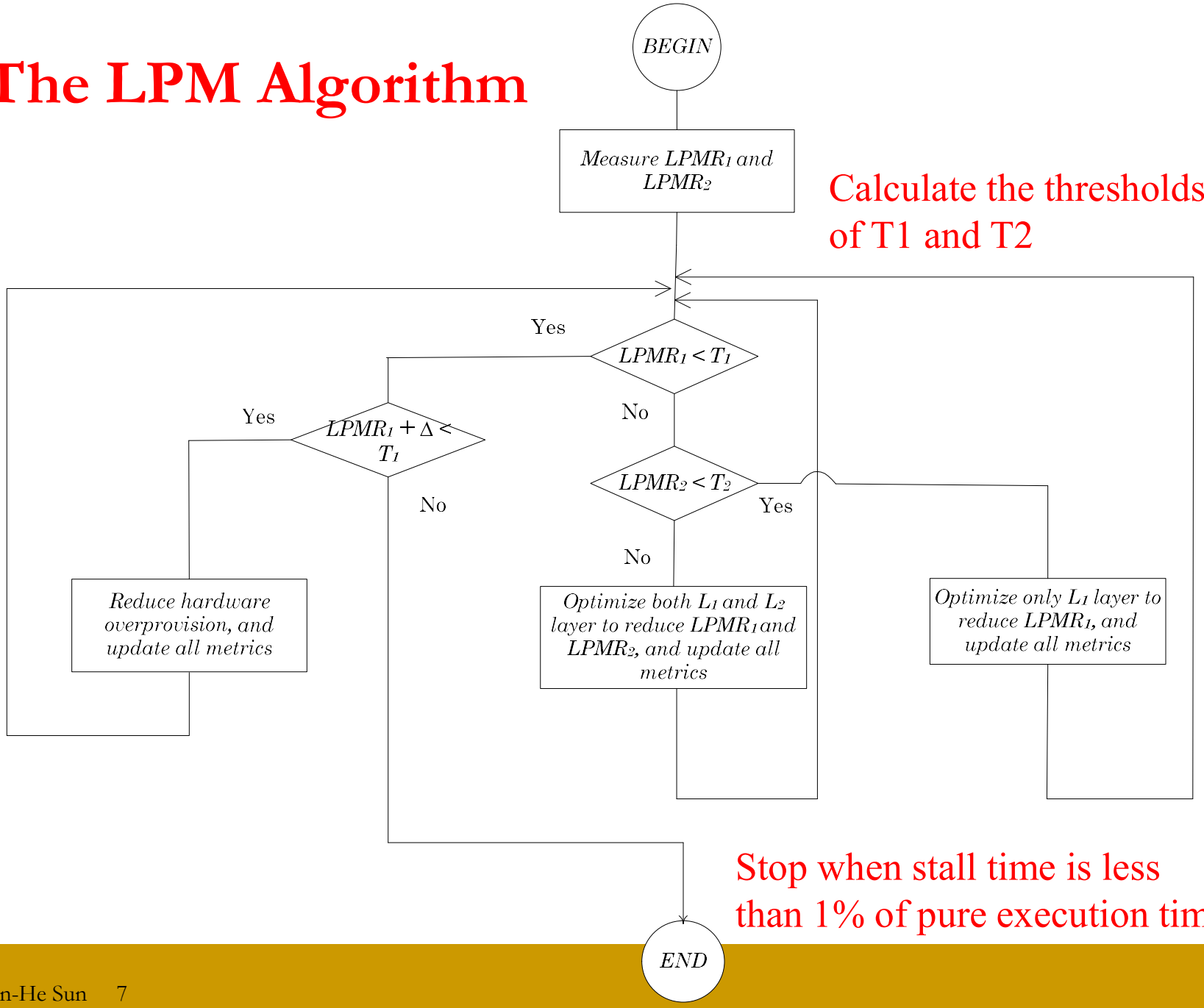
$$LPMR_1 = \frac{IPC_{exe} \times f_{mem}}{APC_1}$$

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

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

- 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

The LPM Algorithm

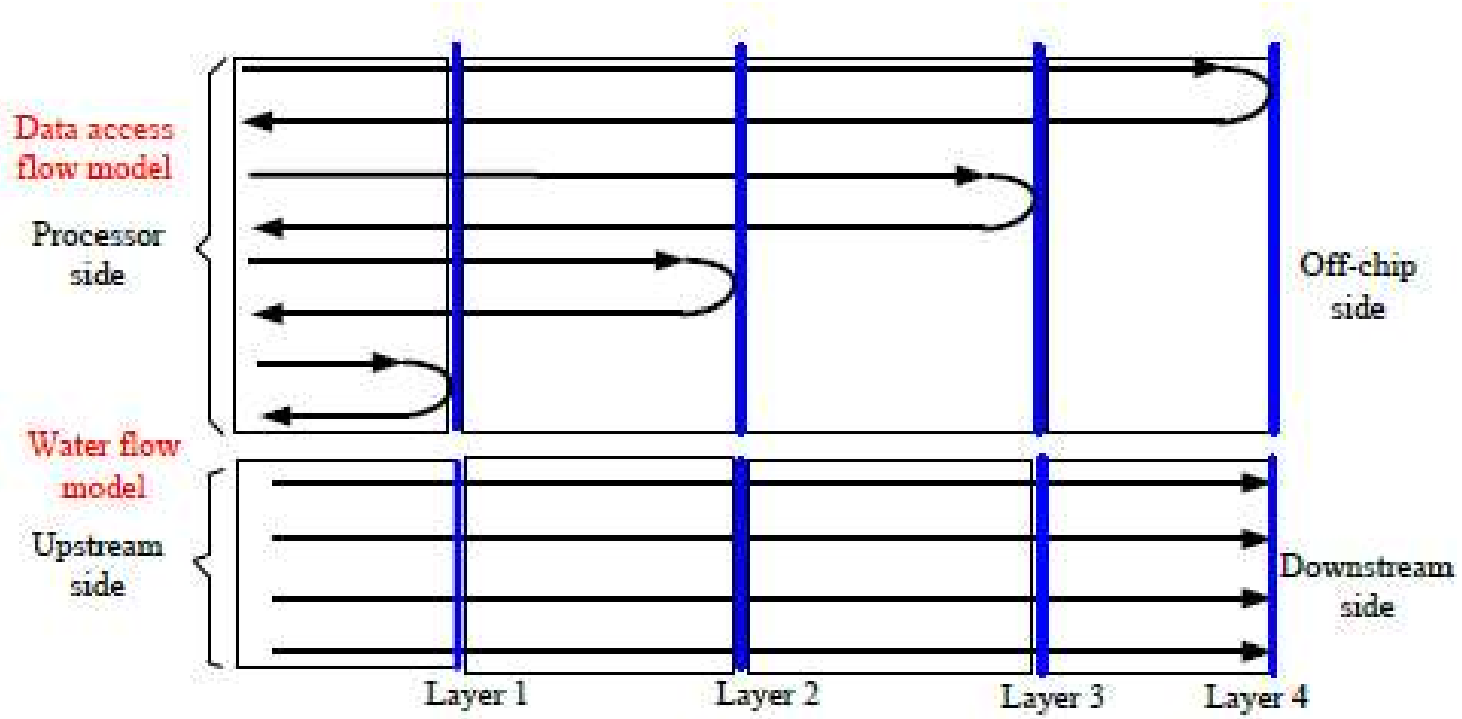


Calculate the thresholds of T1 and T2

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

A Billion Dollars Question

- Data transfer in a memory hierarchy is staged
  - Locality can **reduce** the request at each stage
  - Concurrency can **increase** the supply at each stage
- Can we optimize locality/concurrency to **find a match**, therefore, remove the memory-wall impact?



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.*

***Concurrency Match Theorem**: If an application meets the following condition at each memory layer, then the LPM algorithm can find a performance matching for the application for any matching threshold $T_1 > 0$.*

$$\left.\begin{array}{c} high\ hit\ concurrency \\ small\ hit\ time \end{array}\right\} \text{AND} \left\{\begin{array}{c} high\ pure\ miss\ concurrency \\ low\ pure\ miss\ ratio \\ low\ pure\ miss\ penalty \end{array}\right.$$

With efforts, we can find a match

X.-H. Sun, Y. Liu "Memory Sluice Gate Theory: Have we found a solution for memory wall", IIT/CS-SCS2016-1, March, 2016

Potential Benefits

- Data stall time is 50% to 70% of the total run-time
- That is, data stall time is 1X to 2.3X of computing time
- If stall time is reduced to 1%, compared to the 70% stall time, memory performance improves two hundred and thirty (230) times
- With LPM, our case studies show, the condition of data stall time is less than 1% of computing time is achievable

Why LPM is feasible

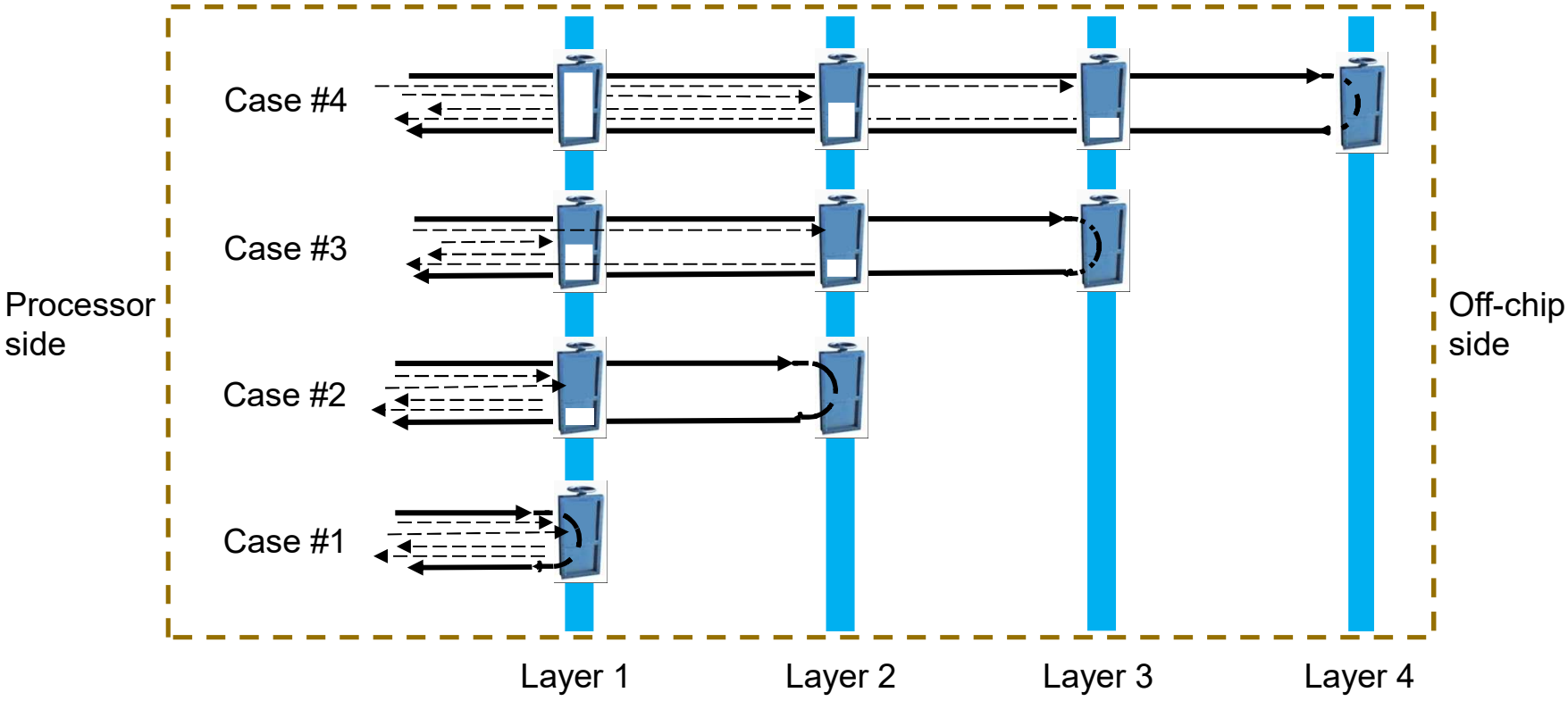
- Memory concurrency can mask data access delay
  - Hit-hit (all the data accesses are hits)
  - Hit-miss (or miss-hit, that is, at least a hit exists with misses)
  - Miss-miss (all data accesses are misses, named pure miss)
- Hit-hit increases data access bandwidth
- Miss-miss, named pure misses, will cause data stall
  - Should reduce pure misses
  - If we cannot avoid them, let them occur at the same time
- Hit-miss is the most interesting one
  - Can hide the penalties of the misses

Why LPM is feasible

- Application has data access patterns
- Configurable hardware become available
- Heterogeneous multi-core become available
- Application-aware software scheduling and partitioning become available
- With a set of lightweight counters, we are able to identify data access patterns and deploy proper optimization techniques

Sluice Gate Theory（存储水闸理论）

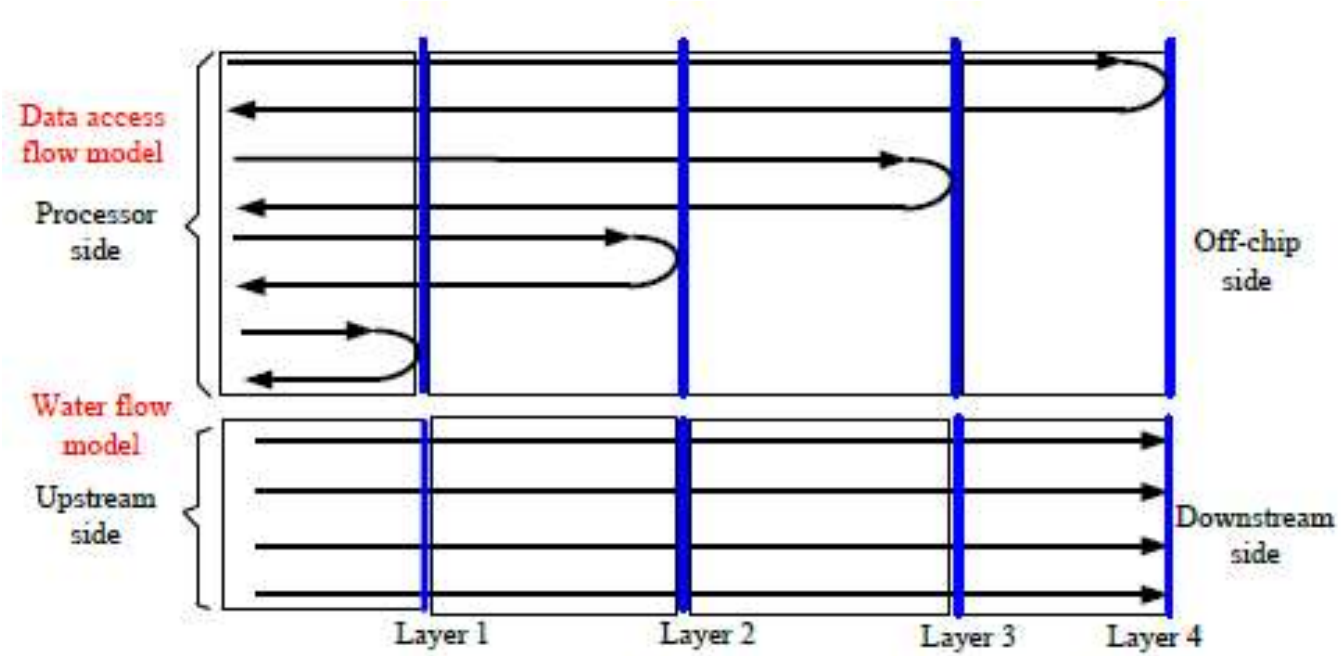
Performance of different layers should match (**using C-AMAT as a gate to guide and control the data flow**)



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

Pace Matching Data Transfer（搏动数据传输）

- C-AMAT is **the sluice gate calculator**
- **Match** request/supply during data transfer
  - Eliminate memory wall impact
- Just like heart bit or pace maker for blood transfer
- Sluice Gate Theory says we can find the match



Comments:

- Pace Data Transfer is our final product to solve the data transfer problem
- Theoretically, Sluice Gate Theory says we can do it. In practice?

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

- *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

Are you saying

you have solved the memory-wall problem?

Memory Sluice Gate Theory

- It is mathematically **correct**, but under the assumptions
  - The application has sufficient data concurrency
  - The system has sufficient hardware to support the data 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

The Missing Link

- C-AMAT finds local miss matches
- Layered Performance finds the miss matching layer
- But, LPM and C-AMAT do not tell you how to reduce C-AMAT at each layer
  - How to optimize memory system and reduce C-AMAT with locality, concurrency, overlapping, and the balance between them
  - We have some new tools, but not a automatic solution yet

Possible Ways to Match

- Reduce request
  - Improve locality, computing, etc
- Improve supply
  - Improve data access concurrency , buffer, technology, etc
- Mask the difference
  - Overlapping computing with data access delay (pure miss), prefetch, etc.

Hardware technology, compiler technology, application algorithm design, system scheduling

Technique Impact Analysis (Original)

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined cache access	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Banked caches		+			+	1	Used in L2 of both i7 and Cortex-A8
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware.
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs

Figure 2.11 Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity. Although generally a technique helps only one factor, prefetching can reduce misses if done sufficiently early; if not, it can reduce miss penalty. + means that the technique improves the factor, – means it hurts that factor, and blank means it has no impact. The complexity measure is subjective, with 0 being the easiest and 3 being a challenge.

Figure 2.11 on page 96 in Hennessy & Patterson's latest book

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

Challenging Questions

- C-AMAT can find local miss matches
- Can we use C-AMAT measurement, including its parameters, recurrence expressions, to **quickly find a local match**?

- C-AMAT can identify and unify all the three performance contributors: locality, concurrency, and overlping
- How to reduce C-AMAT with a **balanced locality, concurrency, and overlapping**?

Matching from Another Angle (power consumption)

$$LPMR = \frac{IPC_{exe} \times f_{mem}}{APC} \quad (1)$$

- **Assume**:

  $Cycle_{CPU} = 2$ ns

  $Cycle_{mem} = 8$ ns

  $f_{\text{mem}} = 20\%$

  $IPC_{exe} = 2.5$

  $APC = 1$

- **Then**:
  - In 8 ns, there is 1 memory cycle, and the data supply rate is:

    $$APC * N\_Cycle_{mem} = 1$$

  - In 8 ns, there is 4 cpu cycle, and the data request rate is:

    $$IPC_{exe} * N\_Cycle_{CPU} * f_{\text{mem}} = 2$$

- **So**:
  - The data supply rate does not match the data request rate. We can decrease the CPU *frequency* (increase CPU cycle) to adjust the data request rate. For example, increase the CPU cycle from 2 ns to 4 ns, the N_Cycle_{CPU} would be 2, so:

    $$\boldsymbol{IPC_{exe} * N\_Cycle_{CPU} * f_{\text{mem}} = 1}$$

  - And the data request rate matches with the data supply rate.

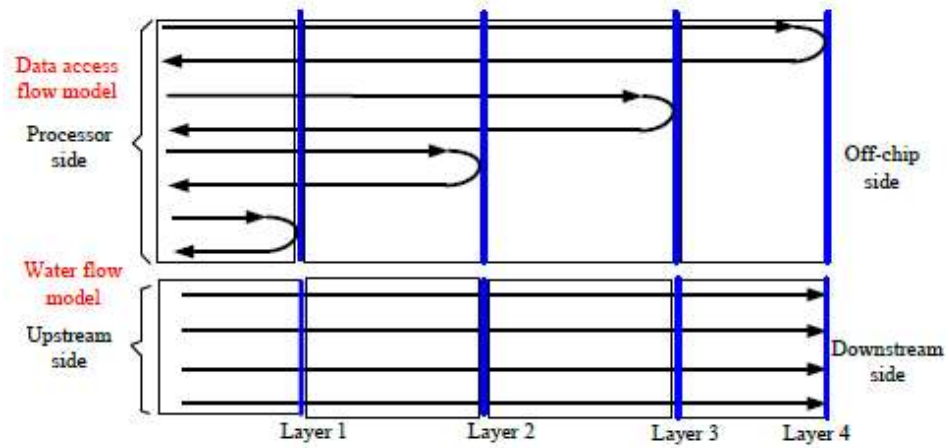
Memory Sluice Gate Theory

- It is mathematically **correct**, but under the assumptions
  - The application has sufficient data concurrency
  - The system has sufficient hardware to support the data 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

But: my computer has a fixed hardware & not elastic ?

Memory Sluice Gate Theory for Data Transfer

- C-AMAT is **the sluice gate calculator**
- **Match** request/supply (Case I)
  - Eliminate memory wall impact
- Optimization (Case II & III, IV & V)
  - Know data access concurrence (Case II)
  - Does not know data access concurrence (Case III)
  - Heterogeneous locality (Case IV)
  - Heterogeneous data transfer time (Case V)



Contribution of Sluice Gate Theory

- The Concept of Sluice
  - Extend the conventional memory hierarchy to include concurrence
  - A data-centric view, require dynamic support
  - Built to mask the gap of performance
- The Concept of Gate
  - Focus on match at each of the sluice stages
  - Balanced optimization of locality, concurrence, and overlapping
  - A unique **Calculator** for optimization/match
- Eliminate memory wall effect is theoretically and practically feasible
  - Require engineering solutions
- **A system solution** for solving the memory wall problem

Related Work

- The traditional AMAT(Average Memory Access Time) :

  $$AMAT = HitCycle + MR \times AMP$$

- MLP (Memory Level Parallelism)
  - The number of outstanding cache misses that can be generated and executed in an overlapped manner
- **C-AMAT=*AMAT/MLP***
  - Assuming each off-chip memory access has a constant latency, say m cycles, then each memory access will count *m* times when calculating different MLP(t), so we have APC=MLP / m
  - The m is really AMAT and C-AMAT is 1/A*PC*
  - *AMAT measures the locality, MLP measures the concurrence, but there is no overlapping*

Dawei Wang and Xian-He Sun, "APC: A Novel Memory Metric and Measurement Methodology for Modern Memory System," IEEE Transactions on Computers, vol. 63, no. 7, pp. 1626-1639, July. 2014)

Revisit

MEMORY SYSTEM BEHAVIOR

with C-AMAT

Three (or Four) Cs

- Compulsory Misses:
  - The first access to a block is not in the cache, so the block must be brought into the cache
- Capacity Misses:
  - If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved.
- Conflict Misses:
  - If block-placement strategy is set associative or direct mapped, conflict misses will occur because a block can be discarded and later retrieved if too many blocks map to its set. Also called collision misses or interference misses

Fourth kind (with concurrent computing)

- Coherence Misses:
  - in multiprocessor systems and multi-core processors
  - Sometimes are false sharing

C-AMAT miss (with concurrent data access)

- Pure Misses:
  - A miss which at least has one cycle there is no any hit
- Peer Pure Misses
  - Pure misses in a shared memory component which are issued by the same processing element

Cache Performance Optimizations

- Reducing cache hit time
- Increasing cache hit concurrency
- Increasing cache miss concurrency
- Reducing cache pure miss rate (overlapping)

- Reducing cache pure miss penalty
- Prefetching (concurrence)

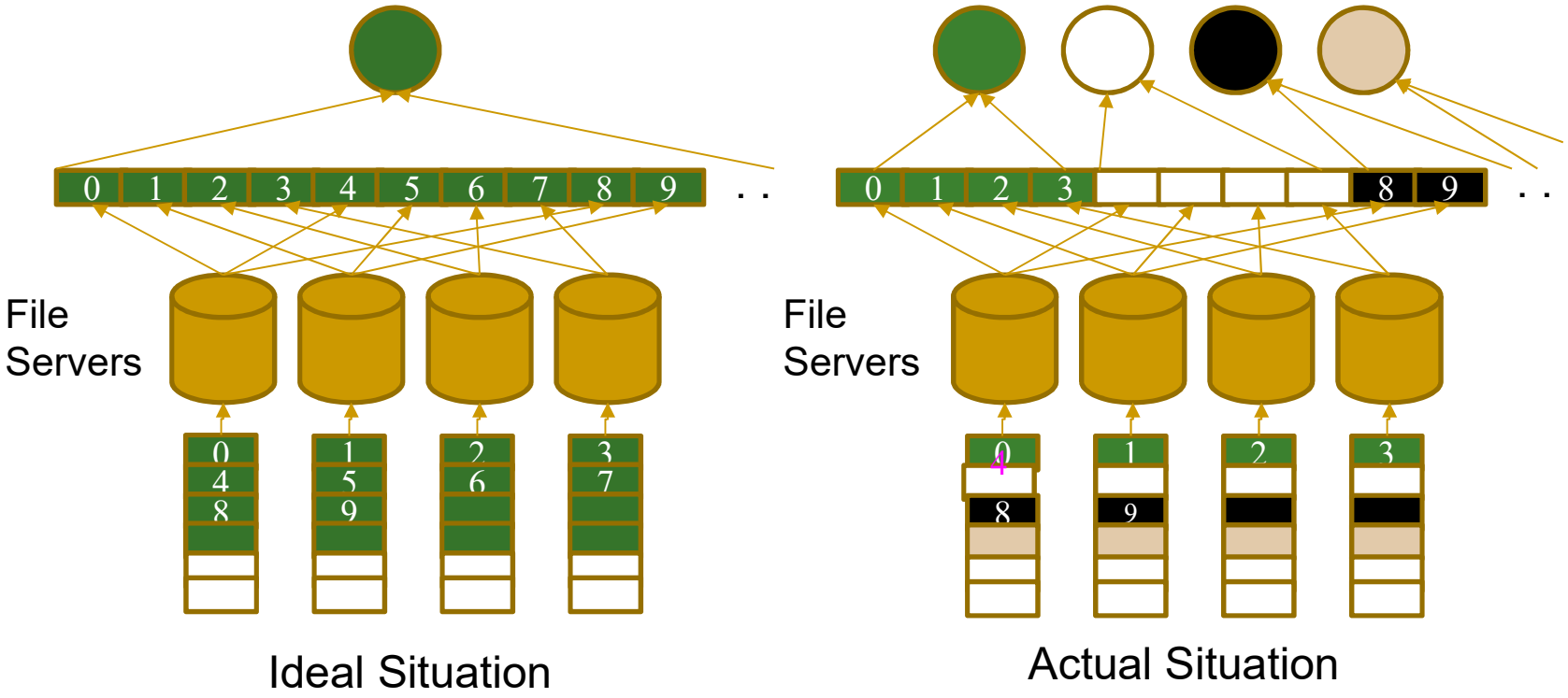
Application of Sluice-Gate Pace Matching

- 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.

Application: Parallel File Systems



- Matching: locality (buffer size) and concurrency
- Heterogeneity: software and hardware
- Integration

IOSIG & IOSIG+: An I/O Characterization Tool

Website: www.cs.iit.edu/~scs/iosig/

Goal: To provide a better understanding of parallel I/O accesses and information to be used for optimization techniques.

Two steps:

Trace collection

- Collects parallel I/O calls of an application
- Does not require any code modification

Trace analysis

- Analyzes the collected information to find data access patterns and C-AMAT parameters
- Handles trace files in any text-based format

Y. Yin, et.al, "Boosting Application-Specific Parallel I/O Optimization Using IOSIG", in Proc. of IEEE/ACM CCGrid, 2012.

IOSIG: from data access to C-AMTA

- Application-aware optimization
- Methods (LPM) and software
- Model (C-AMAT) measurement

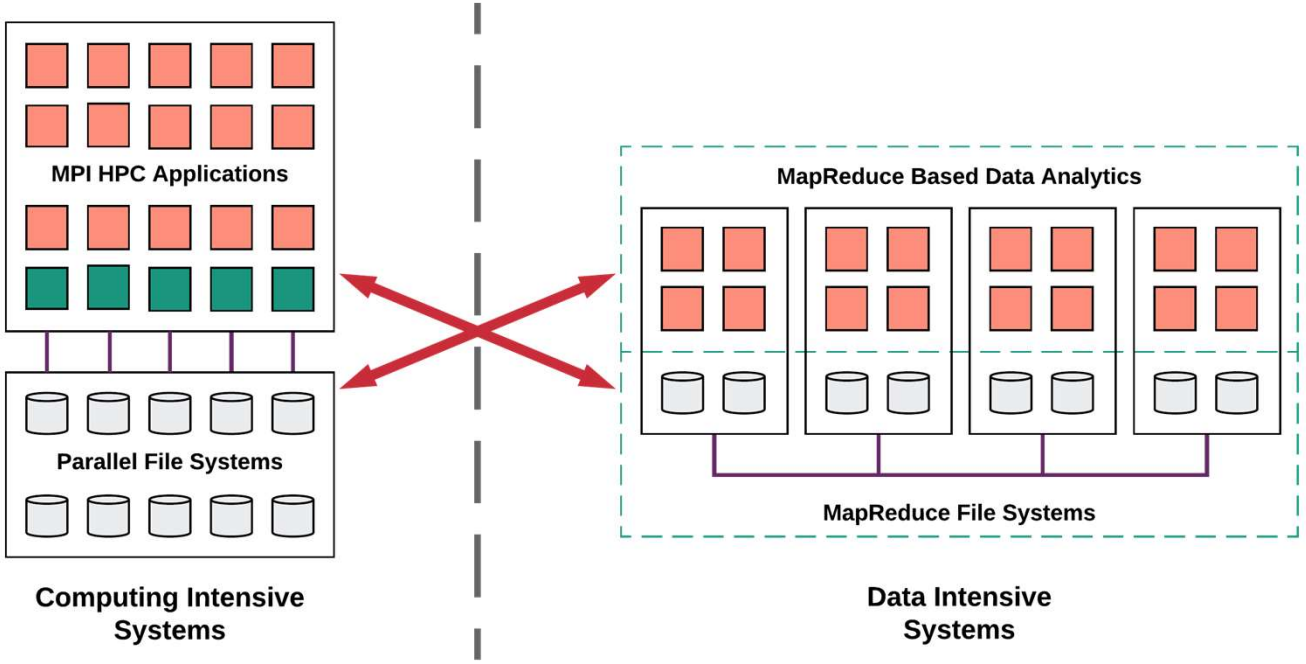
Prefetching	**SC'08**	*Parallel I/O Prefetching Using MPI File Caching and I/O Signatures*
Data Layout	**HPDC11**	*A Cost-intelligent Application-specific Data layout Scheme for Parallel File Systems*
Data Coordination	**SC11**	*Server-Side I/O Coordination for Parallel File Systems*
Data Organization	**PDSW11**	Pattern-aware File Reorganization in MPI-IO
Data Replication	**IPDPS13**	*Layout-Aware Replication Scheme for Parallel I/O Systems*
Cache	**ICDCS14**	S4D-Cache: Smart Selective SSD Cache for Parallel I/O Systems
Heterogeneity	**IPDPS15**	HAS: Heterogeneity-Aware Selective Data Layout Scheme for Parallel File Systems on Hybrid Servers
Integration	**BigData15**	PortHadoop: Support Direct HPC Data Processing in Hadoop
Deep Memory	**HPDC18**	

Data Compression

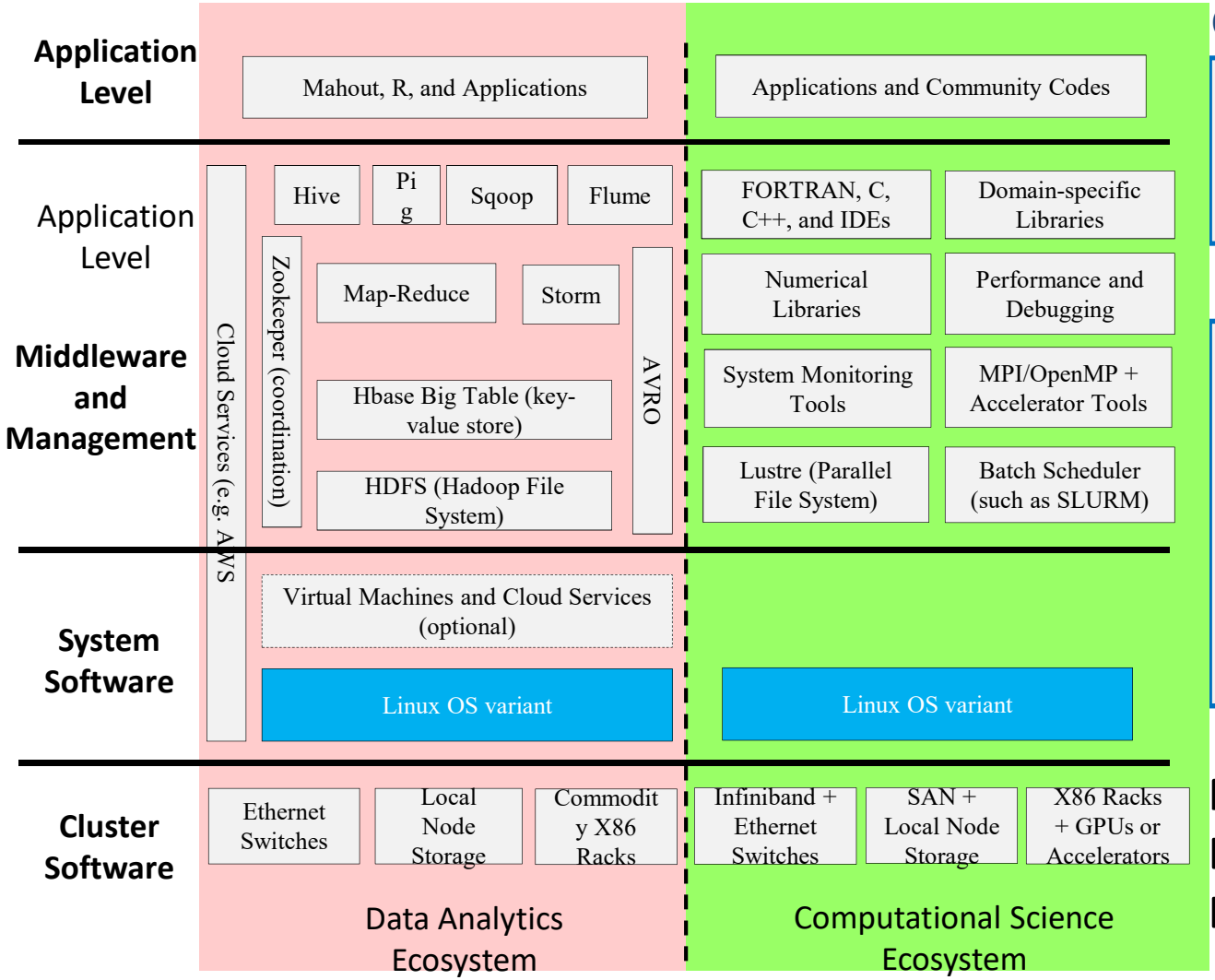
- Website: www.cs.iit.edu/~scs/iosig/

Integrated Data Access System (IDAS)

- Integration of HPC systems and Datacenter/Cloud platform
- NASA Climate Simulation and Earth Science Data Analysis



Data analytics and computing ecosystem compared



Computational Science

FORTRAN,C,C++: languages
PAPI: performance and debugging tool
MPI/OpenMP: multi-core parallel model
SLURM: batch scheduler
Lustre: parallel file system

Data Analytic

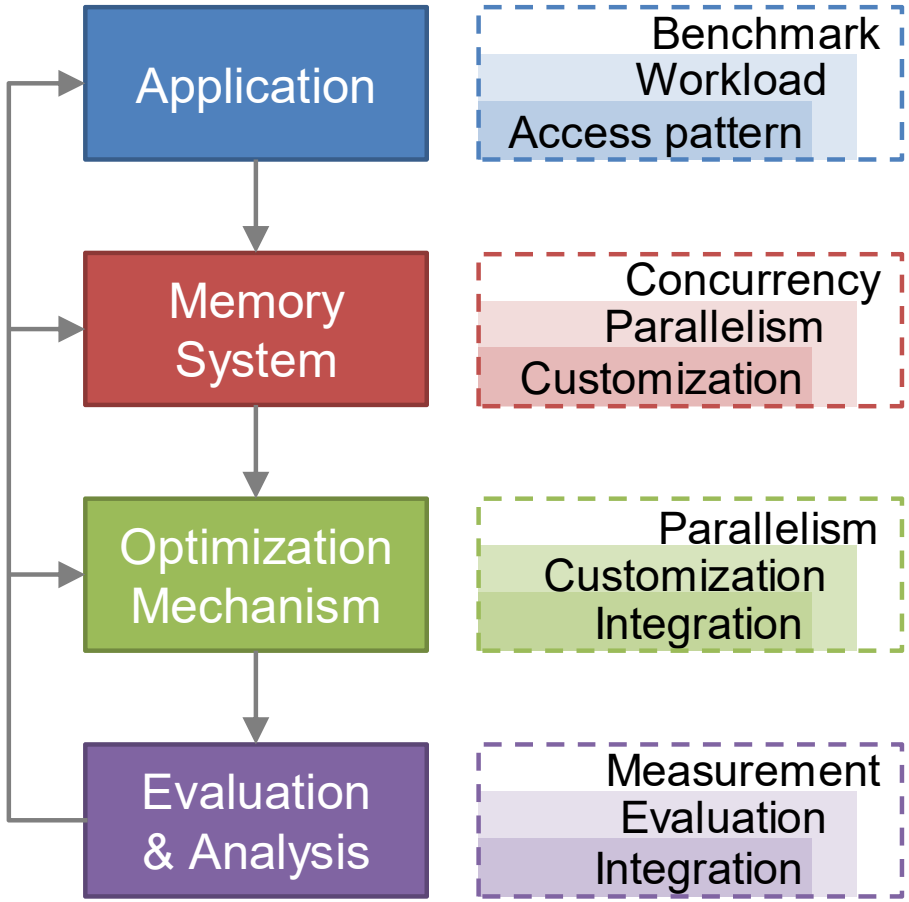
Mahout: machine learning tool
Hive: data warehouse software
Pig: provide high level language for big data
Sqoop: exchange data with traditional database
Flume: log management
Zookeeper: maintaining consistency
Storm: real-time computation system.
Hbase: a distributed, scalable big data store.
AVRO: data serialization system.

NOTE: The Divergence of Big Data and HPC Eco-Systems!

Courtesy by "Exscale Computing and Big Data", DANIEL A. REED AND JACK DONGARRA , CACM July 2015

Current Work: Deep Memory-Storage Hierarchy

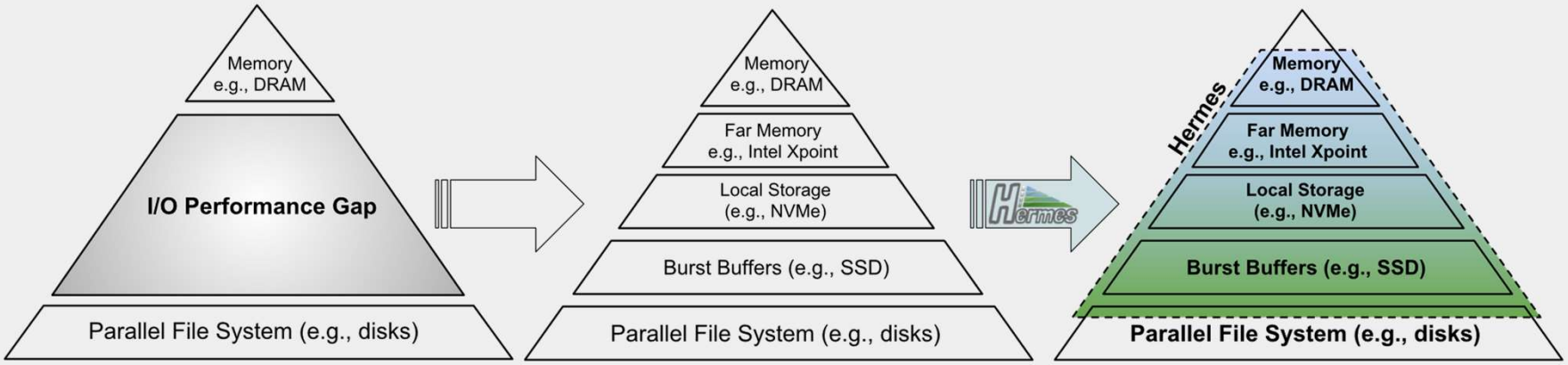
- Application-aware I/O optimization (HDF5)
- Smart, selective, multi-layers, software-hardware, memory-IO
- (Dynamic) Customized optimization
- Following the C-AMAT memory and path-matchng model



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.

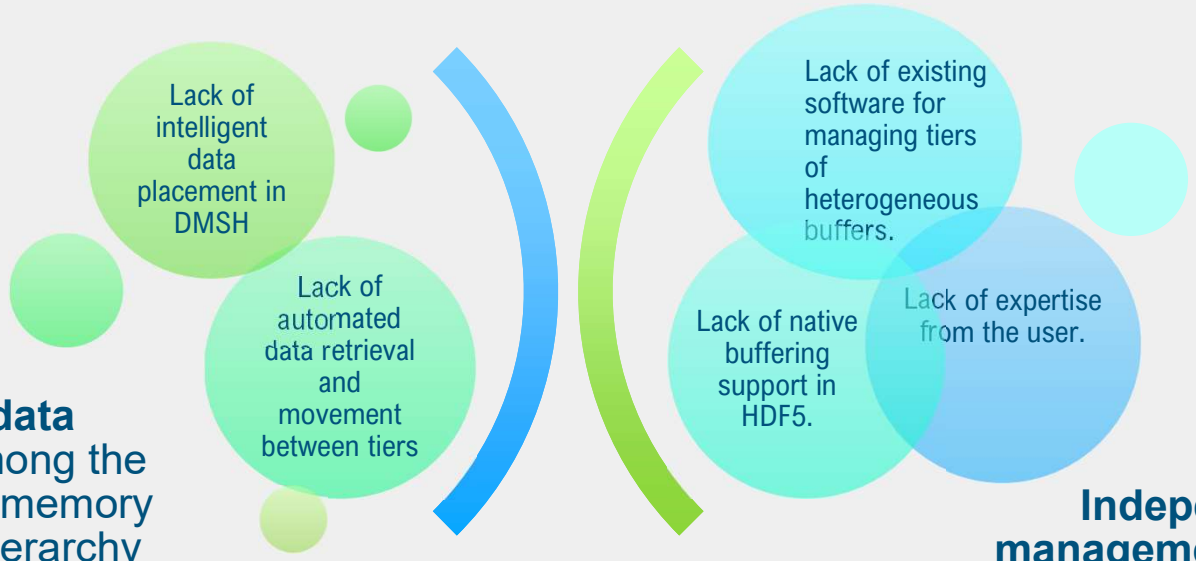
Hermes Overview

- 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.,).



Xian-He Sun, Professor
sun@iit.edu

DMSH Challenges

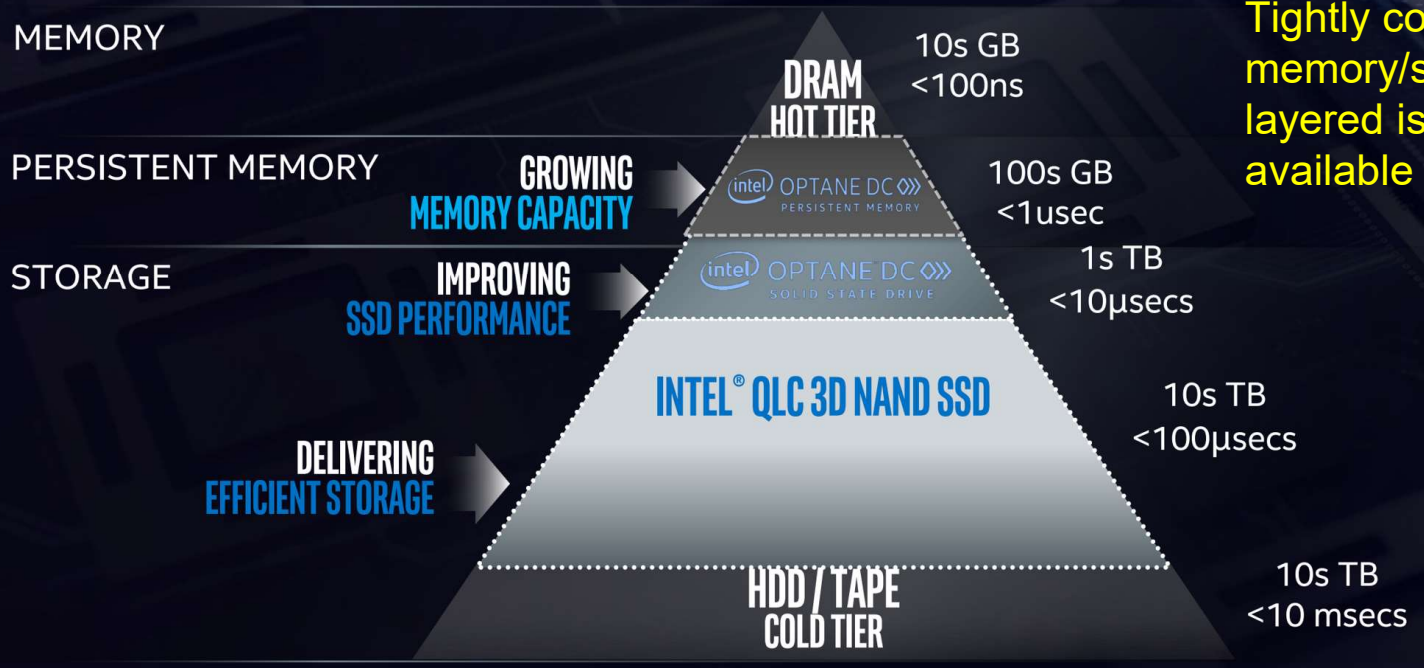


Complex data placement among the tiers of a deep memory and storage hierarchy

Independent management of each tier of the DMSH

Xian-He Sun, Professor
sun@iit.edu

Integrated Smooth Hardware Layers

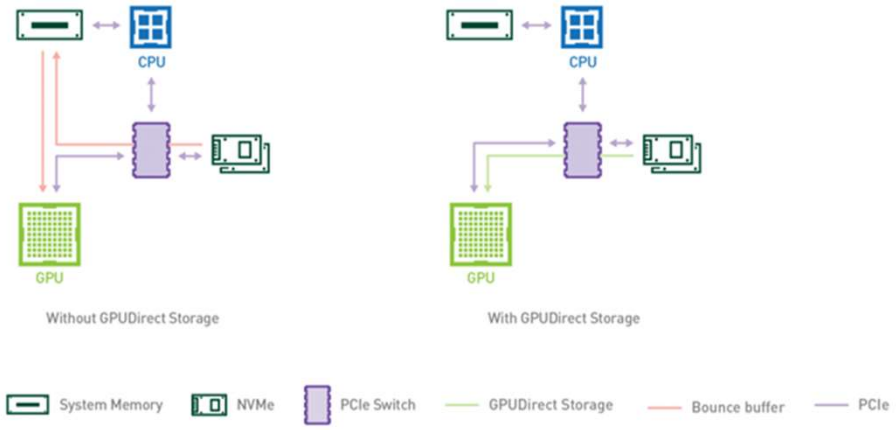


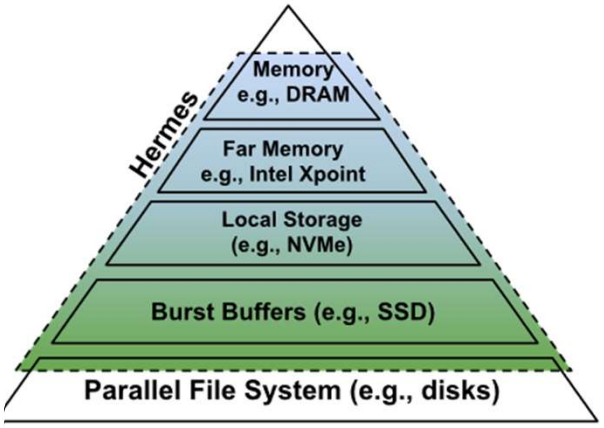
Tightly connected memory/storage layered is becoming available

The Intel Optane Technology for DMSH

Multi-tiered I/O System without Memory

- NVIDIA already support direct GPU I/O access without through memory
  - The same will be true to other accelerators





New Issues

- Is the 10-times slowdown layered design the best layered design?
- Which layer should be private? Which layer should be shared?
- Which layer should be used as buffer, i.e. non-inclusive? Which layer should be used as a level of a memory hierarchy, i.e. inclusive?
- How to utilize dynamic matching under a layered memory-storage hierarchy?
- Making the decision of skip main memory as a decision of matching.
- ….

Conclusion

- Memory **Sluice Gate Theory** is introduced
- Concurrent-AMAT (**C-AMAT**) is the sluice gate calculator
- Matching at sluice gate can remove the memory wall impact
- Matching is Application-aware, a Co-Design process
- Matching needs a rethinking in all aspects, potential is huge
- Start with I/O