The Sluice Gate Theory:
Have we found a solution for memory wall ?

Xian-He Sun
Illinois Institute of Technology
Chicago, Illinois
sun@iit.edu

Keynote, HPC China, Nov. 12, 2015

Outline

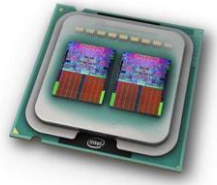
- Rethinking of Memory Systems
- The Concurrent-AMAT (C-AMAT) Model
- Application Case Studies
- The Memory Sluice Gate Theory
- Conclusion

The Surge of Cloud & Big Data

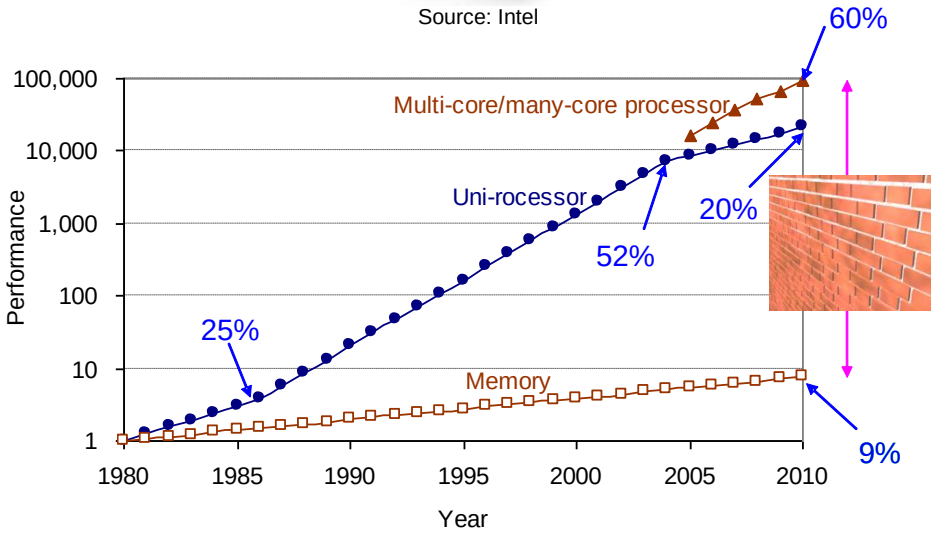
Computing become data intensive

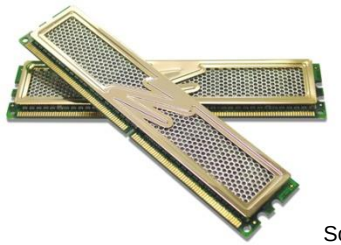
The Memory-wall Problem

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004, ~25% since then
  - New trend: multi-core/many-core architecture
    - Intel TeraFlops chip, 2007
  - Aggregate processor performance much higher
- Memory: ~9% per year
- Processor-memory speed gap keeps increasing



Source: Intel





Source: OCZ

Addressing the HPC Data Challenges

Trends indicate that the "data tsunami" and "memory-wall" will continue

Big-Data problem is a HPC problem:

- **High Performance Data Processing**

  (data-intensive HPC)

Need **rethinking** from **data-centric** view in:

- **Understanding** system, application, and algorithm relevant to data access
- **Optimizing** data access and memory systems
- **Developing** new (memory) system architectures
- **Focus on** the memory-wall problem

Current Solution: Memory Hierarchy

Capacity
Access Time, Bandwidth

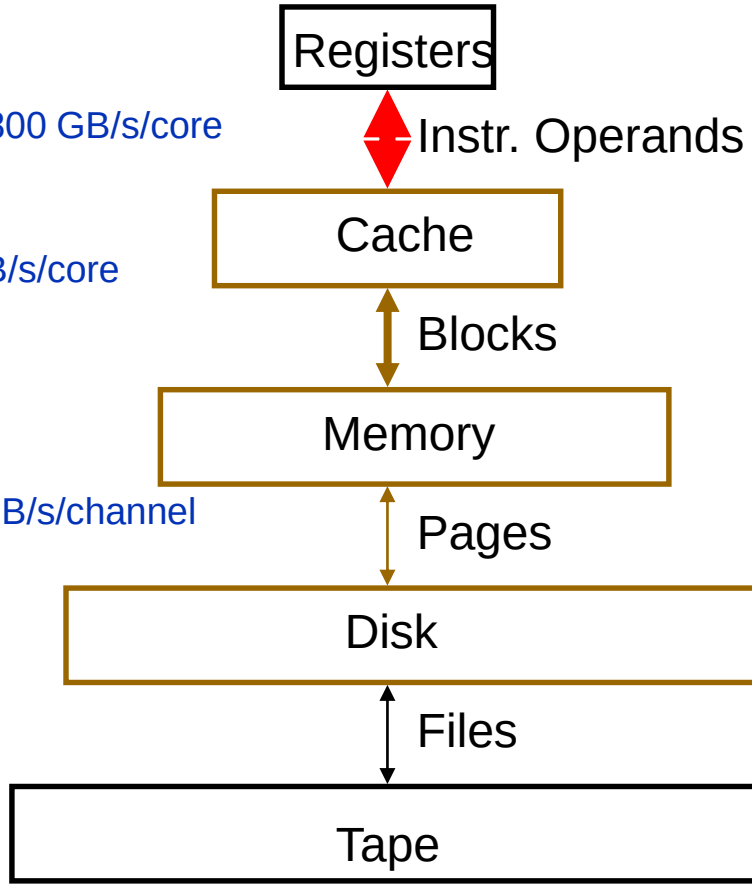
CPU Registers
<8KB
<0.2~0.5 ns, 500~800 GB/s/core

Cache
<50MB
1-10 ns, 50~150GB/s/core

Main Memory
Giga Bytes
50ns-100ns 5~10GB/s/channel

Disk
Tera Bytes, 5 ms
100~300MB/s

Tape
Peta Bytes or
infinite
sec-min



Staging
Xfer Unit

Upper Level
faster

prog./compiler
1-8 bytes

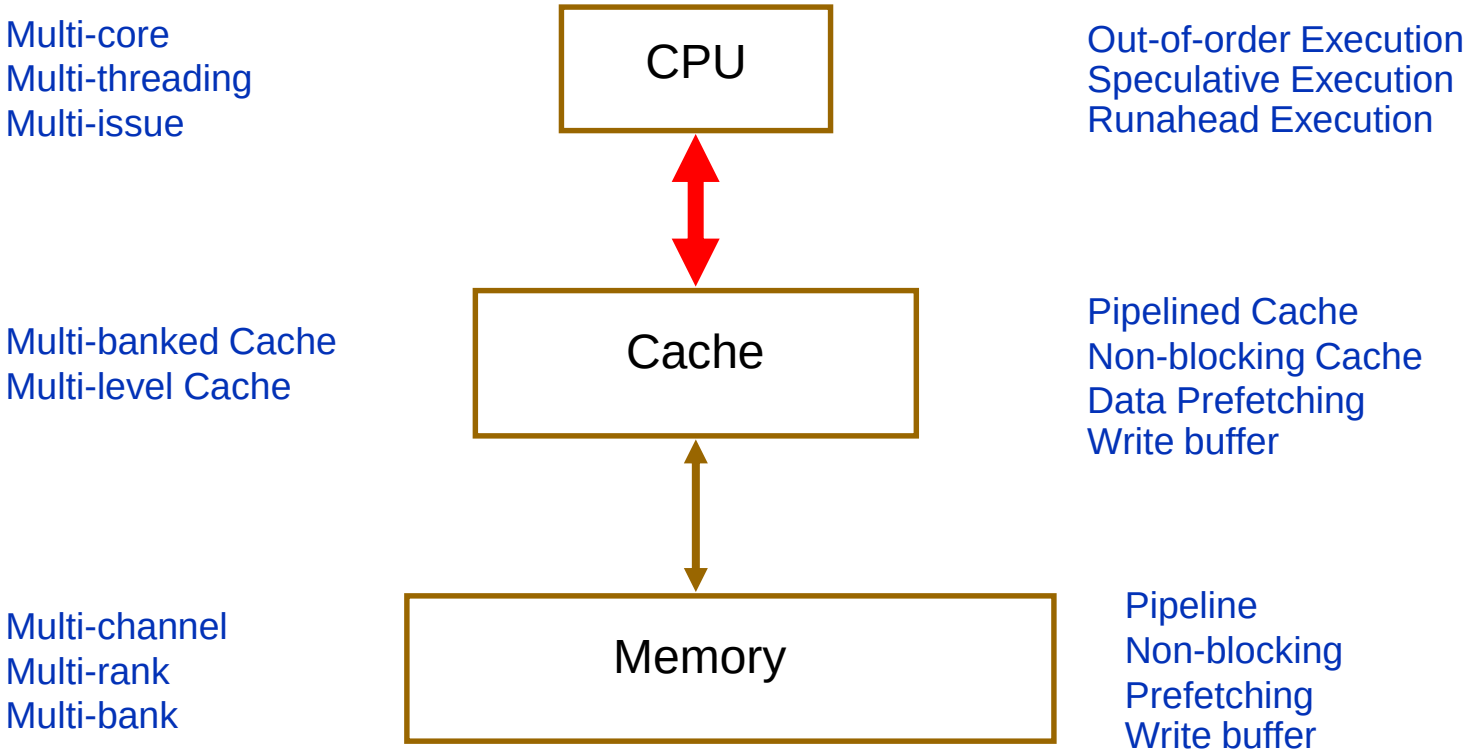
cache cntl
32-128 bytes

OS
4K-4M bytes

user/operator
Mbytes

Larger
Lower Level

Also: (mostly hiding) Memory Concurrency

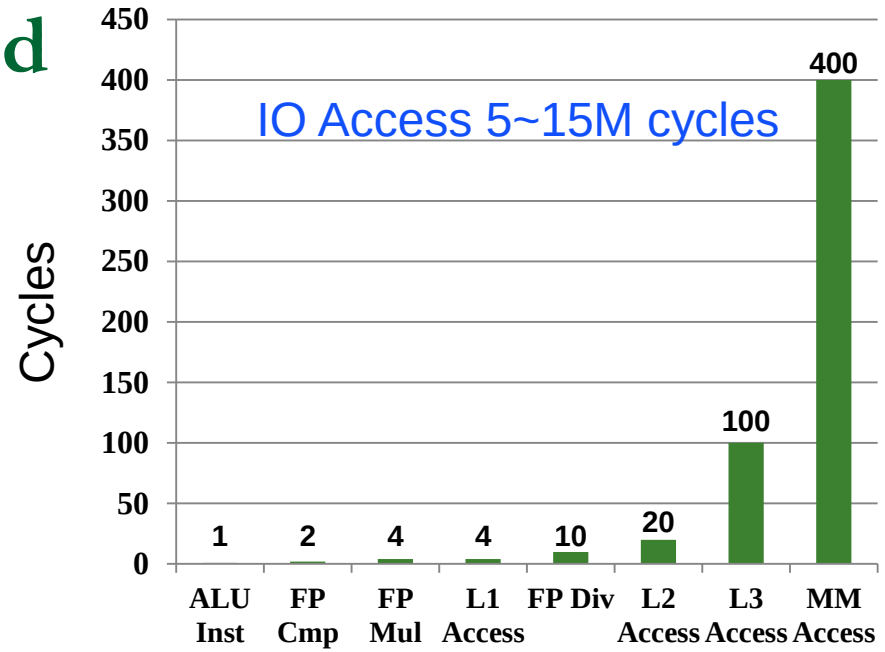


Assumption of Current Solutions

- 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

The Introduction of APC

- Access Per 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)
- Benefits of APC (APMAC)
  - Separate memory evaluation from CPU evaluation
  - Each memory level has its own APC value
  - A better understanding of memory system as a whole, and at each layer
  - A better understanding of the match between computing capacity and memory system performance

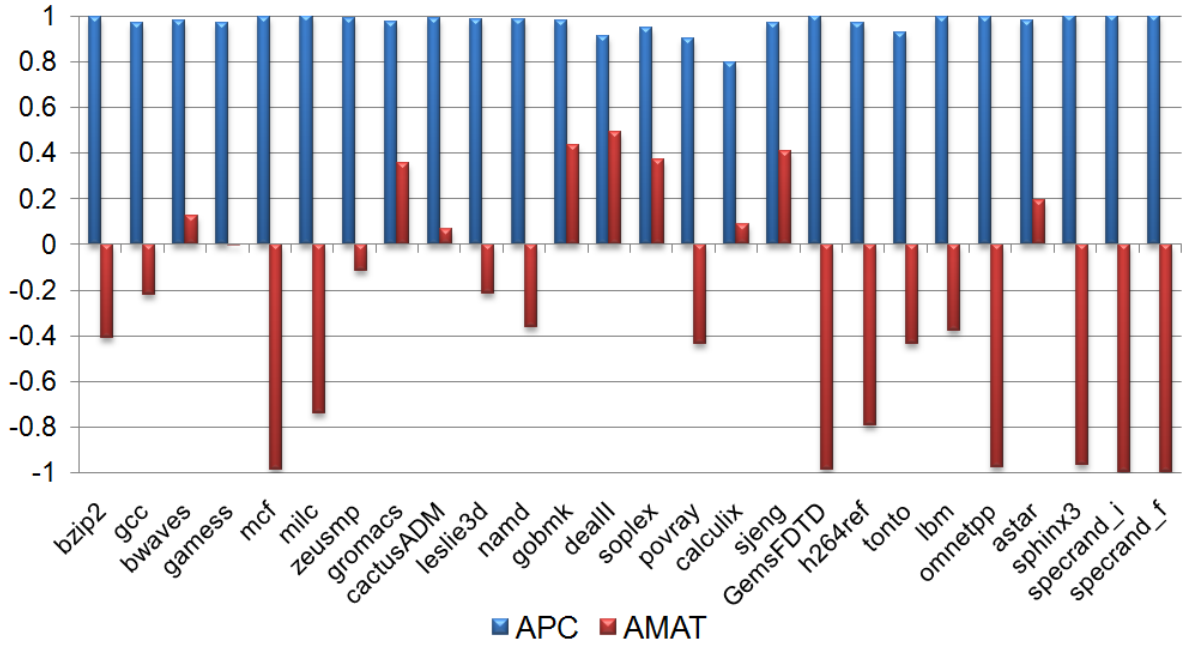
X.-H. Sun and D. Wang, "APC: A Performance Metric of Memory Systems", ACM SIGMETRICS Performance Evaluation Review, Volume 40 , Issue 2, 2012.

APC Measurement

- Measure **T** based on ***memory (active) cycle***
  - Can be measured for ***each layer of a memory hierarchy***
- Measure **A** based on the ***overlapping mode***
  - When there are several memory accesses co-existing during the same clock cycle, *T* only increases by one
- Difficulty in measure memory cycle A & T
  - Hundreds of memory accesses co-exist the memory system
- Hardware cost: one bit
- **Concurrence** and **Data-Centric** *(memory active cycles)* view

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

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

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)

What Does C-AMAT Say?

- C-AMAT is an extension of AMAT to consider concurrency
  - The same as AMAT, if no concurrency present
- C-AMAT introduces the **Pure Miss** concept:
  - Only pure miss causes performance penalty
- **High locality may hurt performance**
  - High locality may lead to pure miss
- **Balance** locality and concurrency with C-AMAT
- C-AMAT uniquely integrates the **joint impact** of locality and concurrency for optimization

Recursive in Memory Hierarchy

- AMAT is recursive
  - $AMAT = HitCycle_1 + MR_1 \times (HitCycle_2 + MR_2 \times AMP_2)$

- C-AMAT is also recursive

$$C\text{-}AMAT_1 = \frac{H_1}{C_{H_1}} + pMR_1 \times \eta_1 \times C\text{-}AMAT_2$$

Where

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

$$\eta_1 = \frac{pAMP_1}{AMP_1} \times \frac{C_{m_1}}{C_{M_1}}$$

With Clear Physical Meaning

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

Impact of C-AMAT

- New dimensions for optimization: **concurrency and balancing**

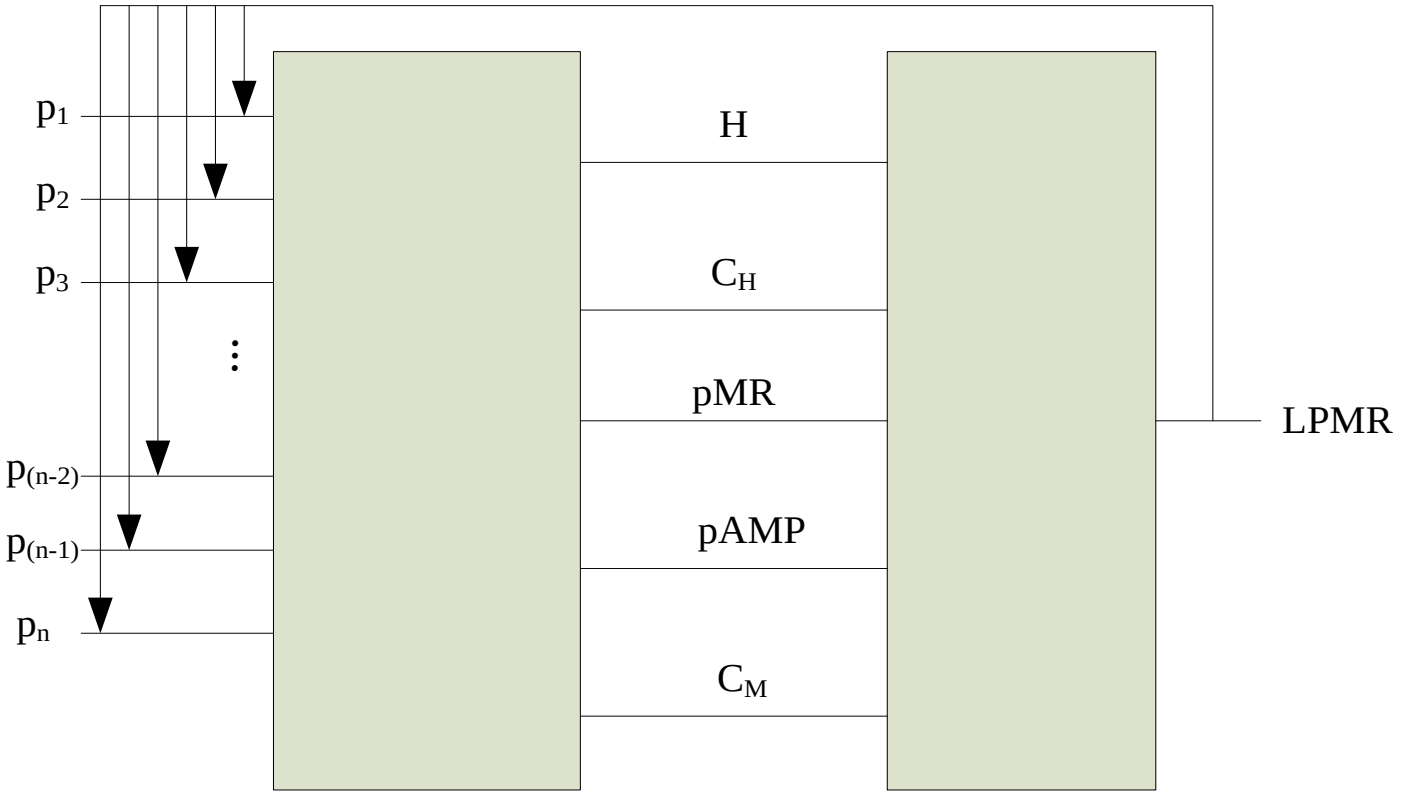
$$\text{C-AMAT} = \text{HitCycle}/C_H + p\text{MR} \times p\text{AMP}/C_M$$

- Can apply at **each layer** of a memory hierarchy
- Existing mechanisms are readily to be extended
  - Every AMAT based optimization has a corresponding C-AMAT extension to include concurrency
- Concurrency as **penalty reducer**: Accurate measure the concurrency contribution

$$\eta_1 = \frac{pAMP_1}{AMP_1} \times \frac{C_{m\ 1}}{C_{M\ 1}}$$

$$C\text{-}AMAT_1 = \frac{H_1}{C_{H_1}} + pMR_1 \times \eta_1 \times C\text{-}AMAT_2$$

Application: Parameters can be measured at runtime



Feedback-based optimization on scheduling and on reconfigurable architecture

Application: Utilizing Memory Concurrency

- Recall C-AMAT is recursive

$$C\text{-}AMAT_1 = \frac{H_1}{C_{H_1}} + pMR_1 \times \eta_1 \times C\text{-}AMAT_2$$

Where

$$C\text{-}AMAT_2 = \frac{H_2}{C_{H_2}} + pMR_2 \times \frac{pAMP_2}{C_{M_2}} \qquad \eta_1 = \frac{pAMP_1}{AMP_1} \times \frac{C_{m_1}}{C_{M_1}}$$

- Rearranging the recursive expressions, we have

$$C\text{-}AMAT_1 = \sum_{i=1}^{n} (a_i \times \frac{H_i}{C_i})$$

$$a_1 = 1$$

$$a_2 = pMR_1 \times \eta_1$$

$$a_3 = pMR_1 \times pMR_2 \times \eta_1 \times \eta_2$$

$$\cdots$$

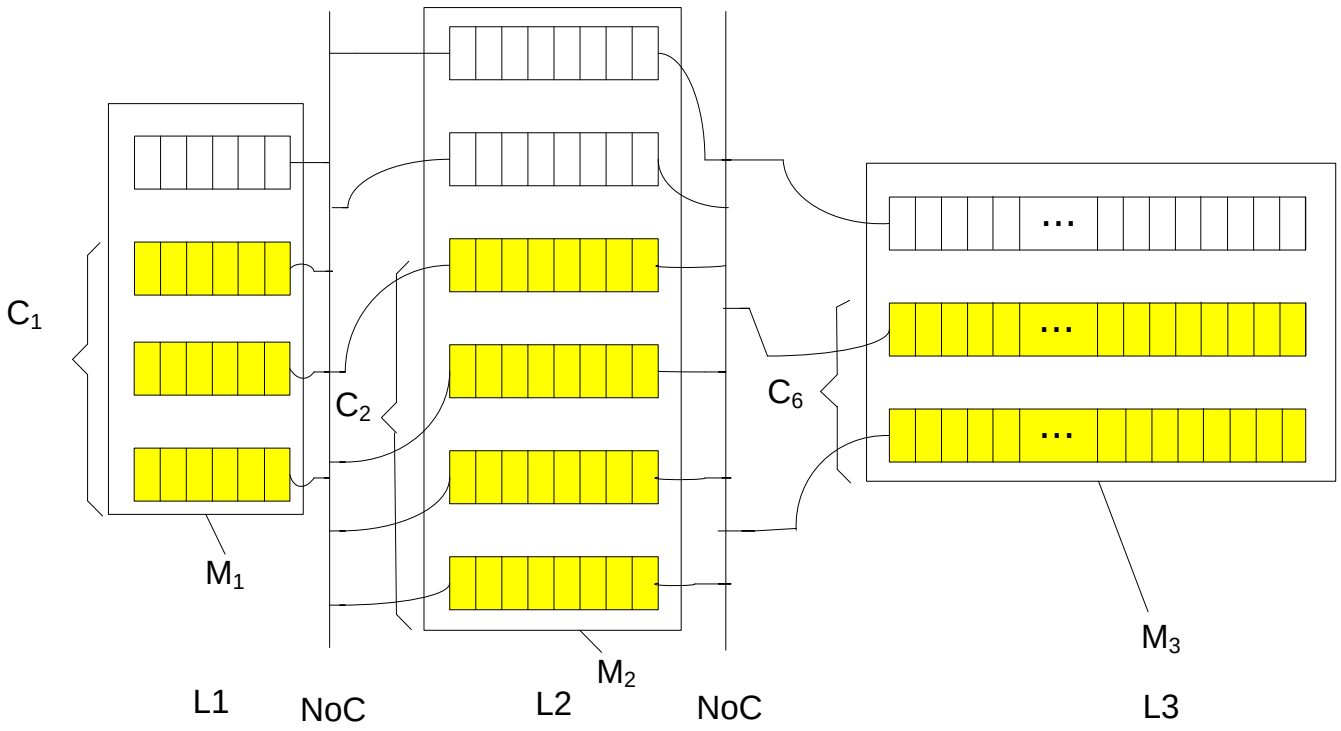
$$a_n = \prod_{i=1}^{n-1} pMR_i \times \prod_{i=1}^{n-1} \eta_i$$

Case I: Utilizing Memory Banks

- Optimize the rearranged C-AMAT under given hardware constraints (an optimization problem for each task)
- Memory concurrency is measured in the number of **memory banks**
- Focus on on-chip caches (memory)
- Transform concurrency optimization into **scheduling**
  - e.g. for L2, task 1 optimal is 18, task 2 optimal is 1, with only 8 hardware memory banks, then the optimal bank scheduling is
  - Task 1 gets 7 and task 2 gets 1
- **Readily to be used**

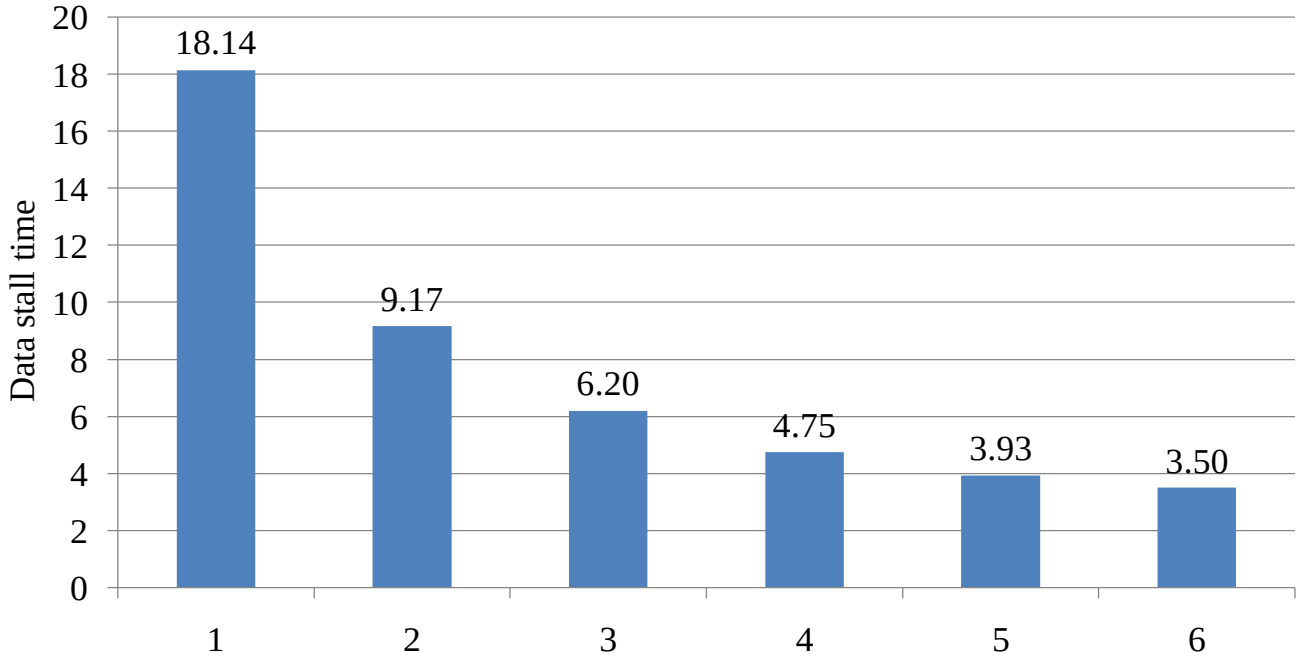
Y. Liu, X.-H. Sun, "Smart-C: Optimizing Memory Concurrency at Each Memory Layer in a Multi-Tasking Environment", IIT/CS-SCS2015-10, Oct., 2015

Case I: The Smart-C Algorithm



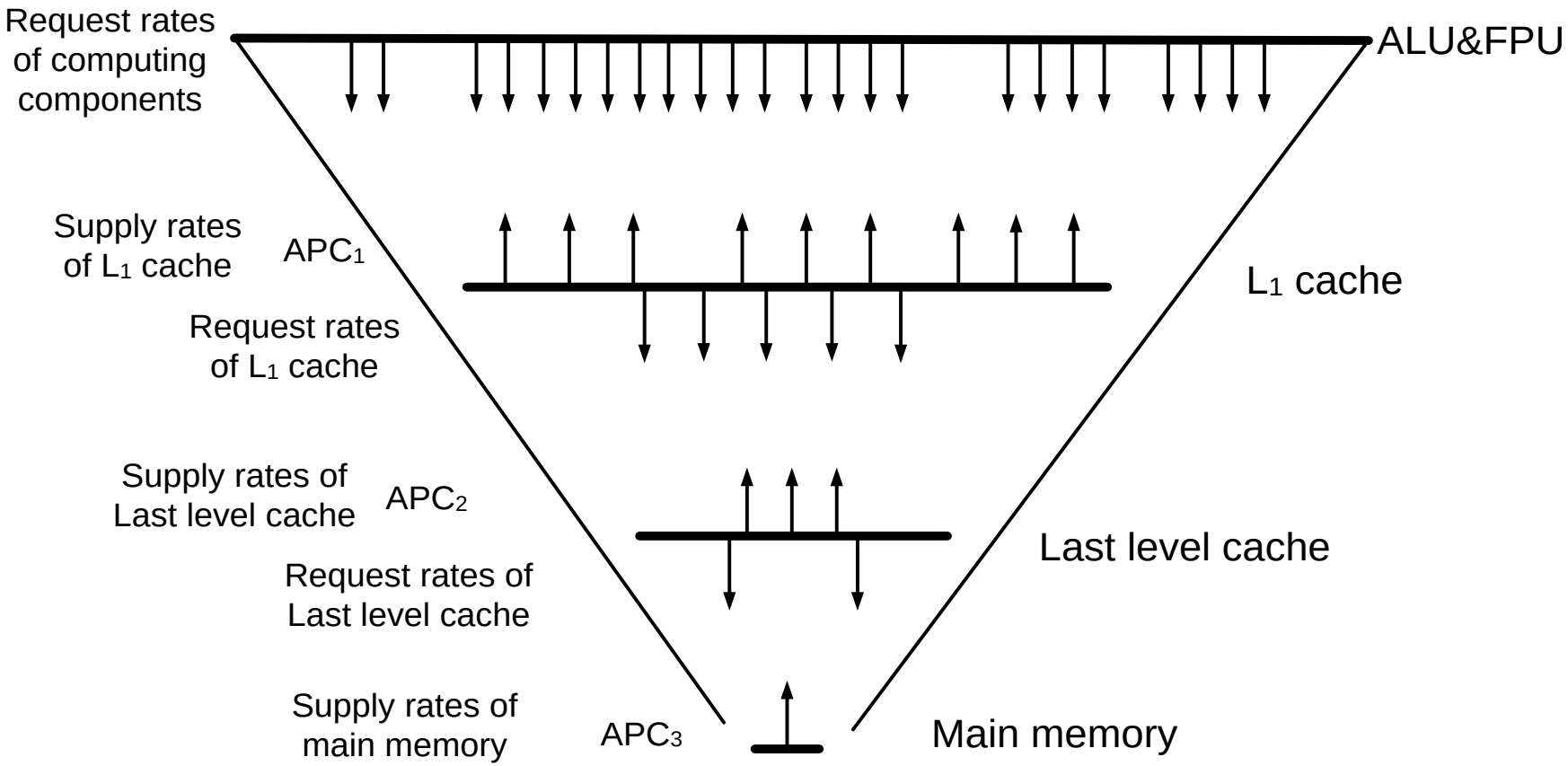
- Memory bank scheduling

Performance Improvement



- On two SPEC CPU 2006 benchmarks “mcf” and “provray”
- Stall time reduction is 5.2 fold

Application: Layered Performance Matching



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

Idea: Match the Request with Supply

$$LPMR(ALU\ \&\ FPU, L_1) = \frac{Request\ rate\ from\ ALU\ \&FPU}{Supply\ rate\ by\ L_1\ cache}$$

$$LPMR(L_1, LLC) = \frac{Request\ rate\ \ from\ L_1\ cache}{Supply\ rate\ by\ \ LLC}$$

$$LPMR(LLC, MM) = \frac{Request\ rate\ \ from\ LLC}{Supply\ rate\ by\ main\ memory}$$

- Match at **each memory layer**
- Adjust the supply performance with concurrency

Quantify Mismatching: with C-AMAT

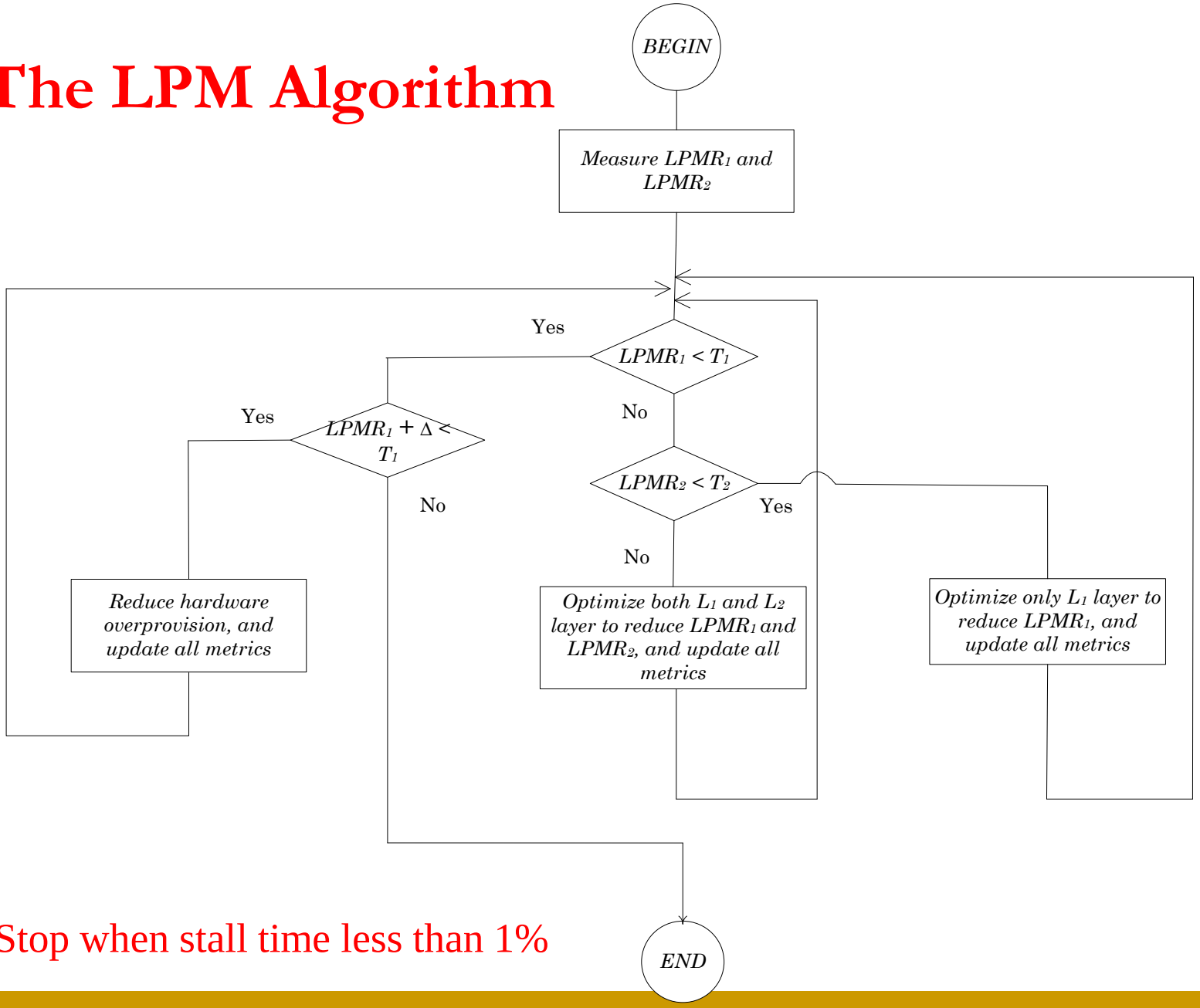
$$LPMR_1 = \frac{IPC_{exe} \times f_{mem}}{APC_1}$$

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

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

- 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



Stop when stall time less than 1%

Case study II: find the best configuration

LPM Optimization on Reconfigurable Architecture

Configuration	A	B	C	D	E
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
$LPMR_1$	16.1	12.8	4.2	2.4	2.8
$LPMR_2$	19.3	18.3	6.1	3.2	5.9
$LPMR_3$	12.6	16.2	11.6	4.6	8.2

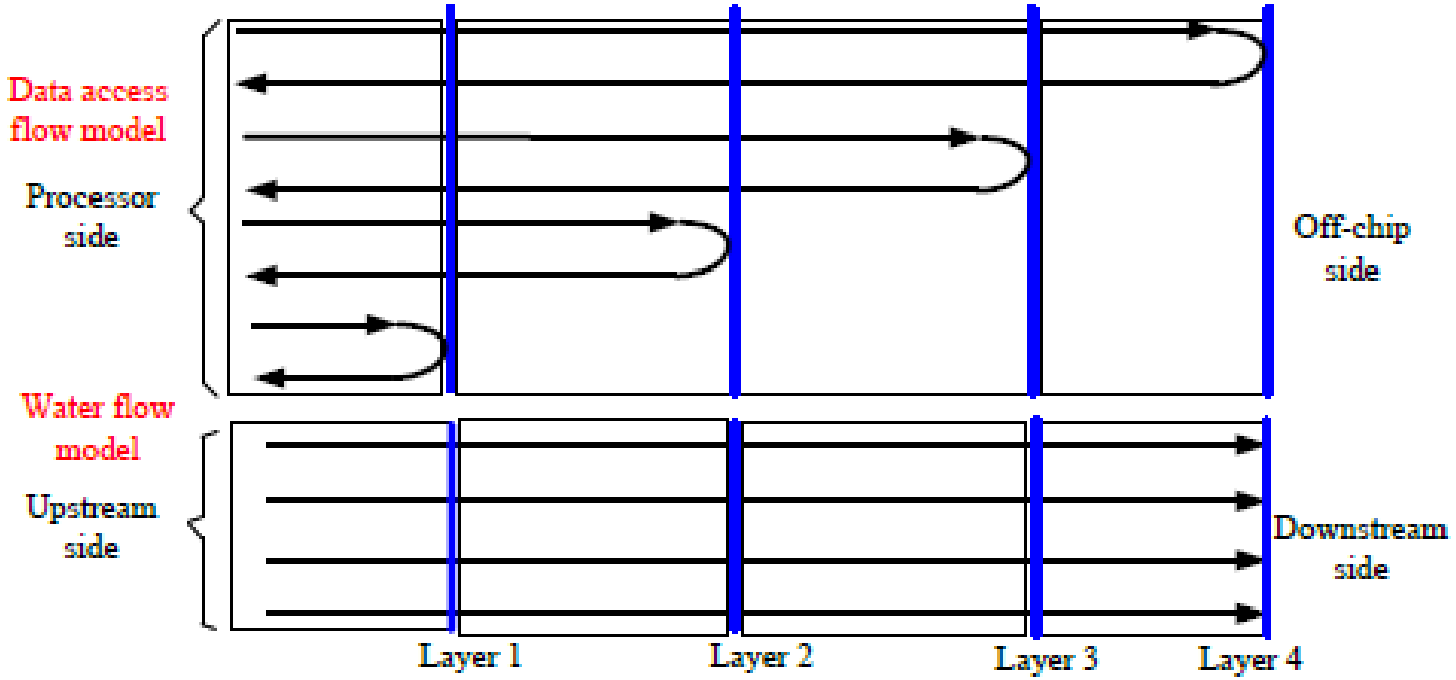
Increased data access performance for more than **150 times** with the LPM algorithm

Case II Discussion

- GEM5 and 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

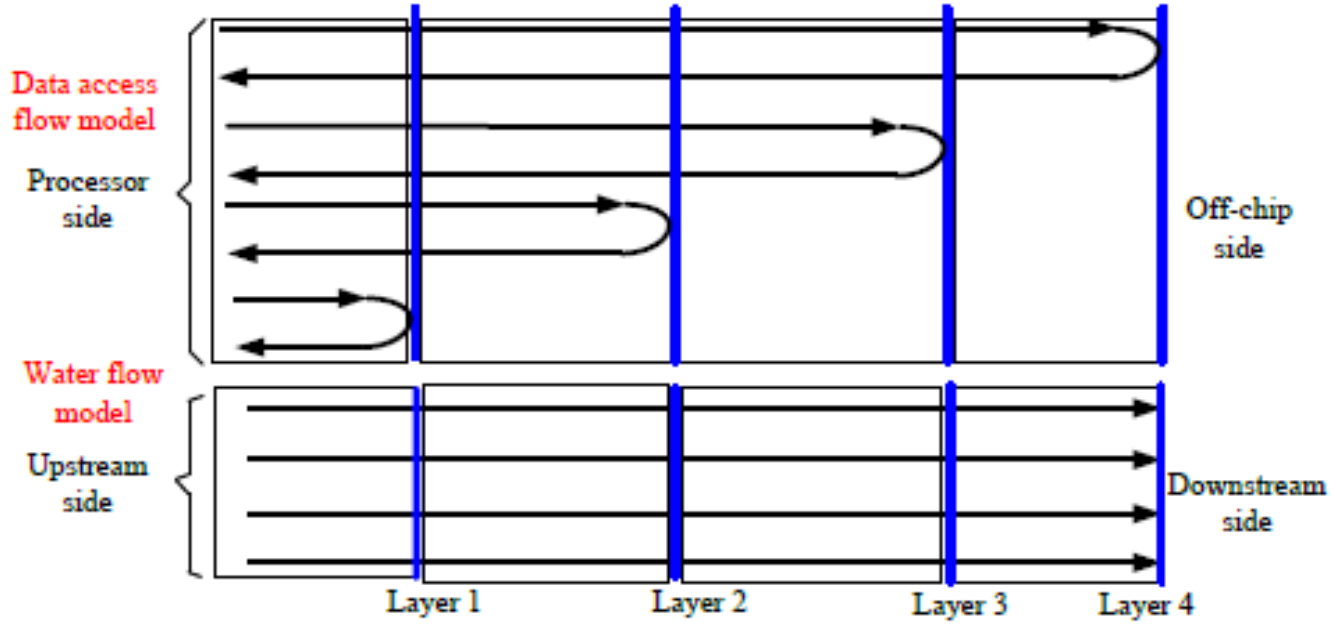
Sluice Gate Theory for Data Transfer

- Data transfer in a memory hierarchy is staged
- Different stages have different capacities
  - Bump and delay at the stage change (gate)
- Not all data go to the next step
- More like water transfer in sluice than water flow in river



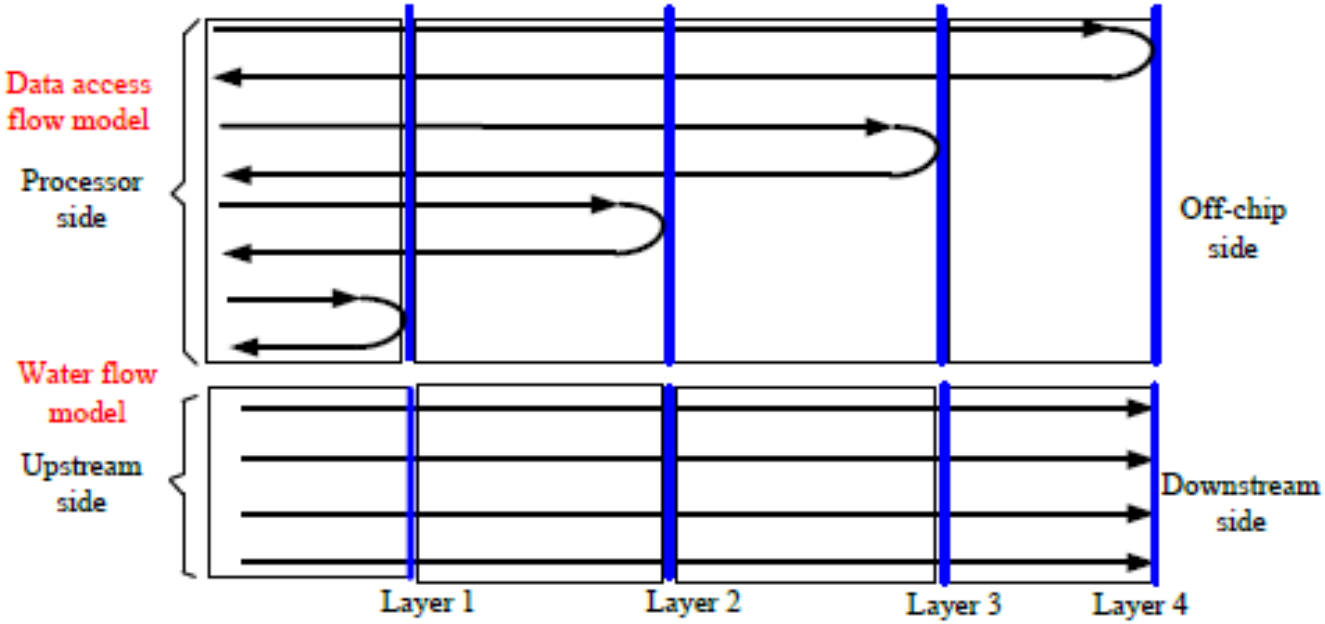
Sluice Gate Theory for Data Transfer

- C-AMAT is **the sluice gate calculator**
- **Match** request/supply at each stage and of the system (Case II)
  - Remove the bump and delay
  - Hardware (software) improvement
- Best effort match under a given hardware configuration (Case I)
  - Utilizing the underlying hardware



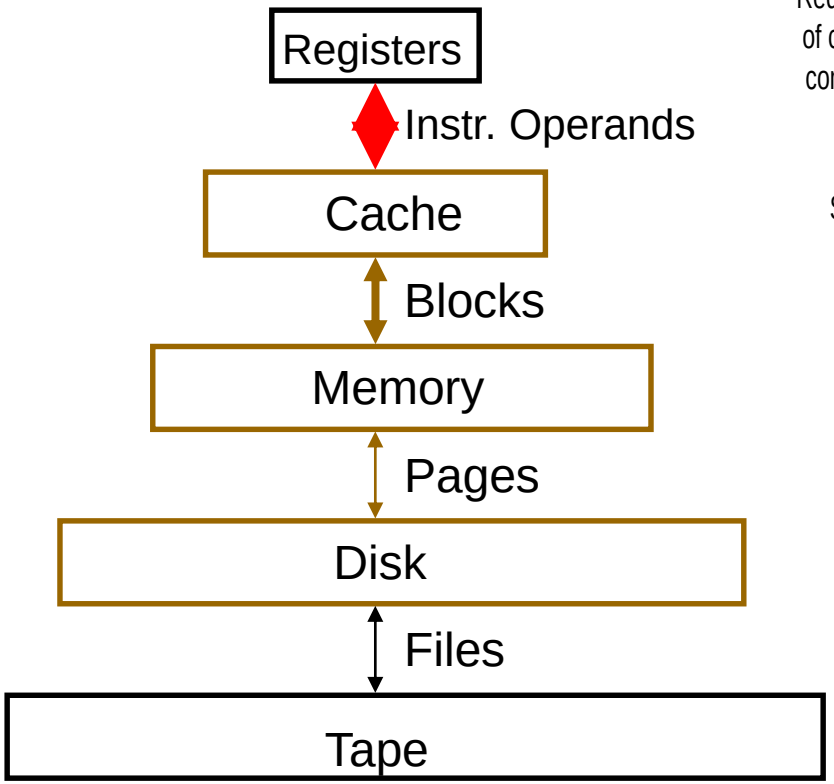
The Sluice Gate Theory

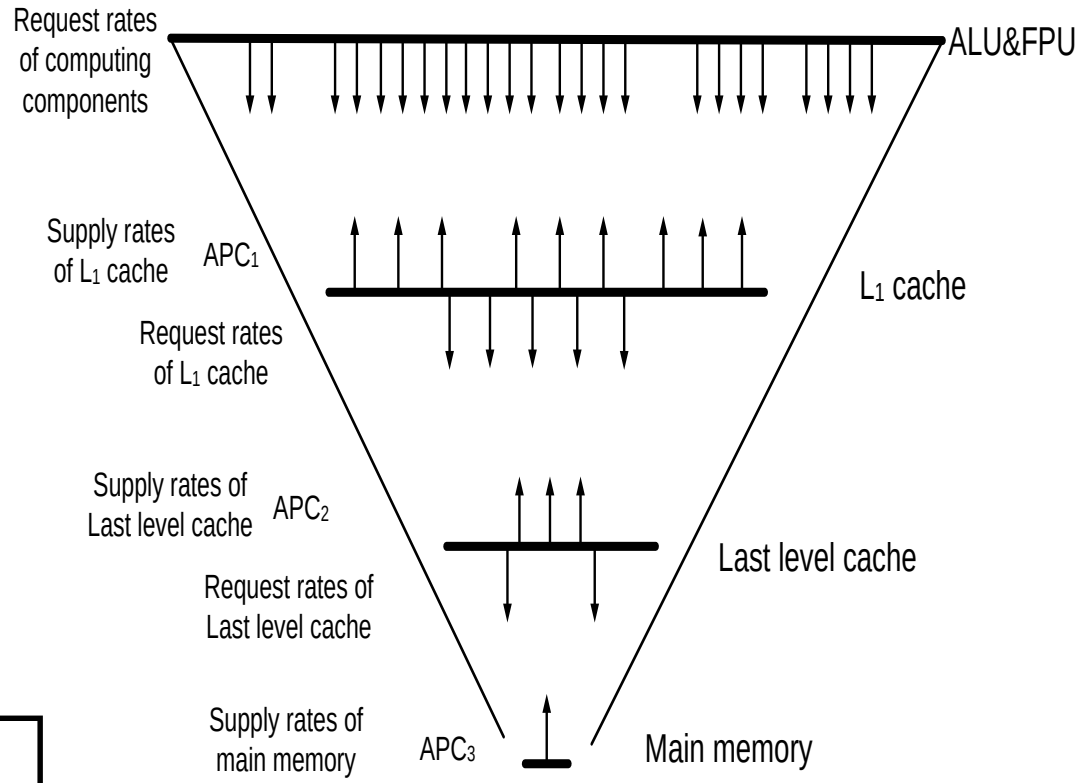
With the C-AMAT sluice gate calculator, sufficient hardware resources and software efforts, the data transfer in a memory hierarchy can be **Matched** at each memory layer for a given application



The Sluice Gate Theory: Match

- The Pyramid, up-side-down Pyramid, sluice data transfer, sluice gate calculator, and the sluice data transfer match





The Sluice Gate Theory

Are you saying

you have solved the memory-wall problem?

The Sluice Gate Theory

- It is a hypothesis, but a reasonable one
  - We have a working example
  - Match can be achieved through many different ways, concurrency is only one of them
- It has a tacit assumption, the architecture is elastic
  - Need to build a general purpose computer
  - Even for a given application may have different data access patterns
- It is **a big step** toward to solve the memory-wall problem
  - Do not need to wait for technology improvement
  - Can guide technology improvement
- The Sluice Gate Theory, as it is claimed, is mathematically **correct**

The Contribution of Sluice Gate Theory

- The Concept of Sluice
  - Memory Sluice is designed to send data to computing
  - It is totally different with the concept of the known dataflow architecture, where data trigger computing
  - Determine high level design choices
- The Concept of Gate
  - Focus on removing and mitigating the performance gap between CPU and memory device during data transfer
  - Sluice is built to mask the gap of performance
- It claims a matching is possible and provides a way of matching and optimization
  - A end-to-end global view for optimization
  - Optimization with two pillars, data locality and concurrency

- **An architectural solution** for solving the memory wall problem

Possible Ways to Match

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

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

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

Conclusion

- The memory **Sluice gate Theory** is introduced
- Concurrent-AMAT (**C-AMAT**) is the sluice gate calculator
- Matching at sluice gate may remove the memory wall impact
- Matching is Application-aware, a Co-Design process
- Matching needs a rethinking in all aspects, potential is huge