Dataflow under the von Neumann Machine: A New Paradigm for Computing Systems

Xian-He Sun

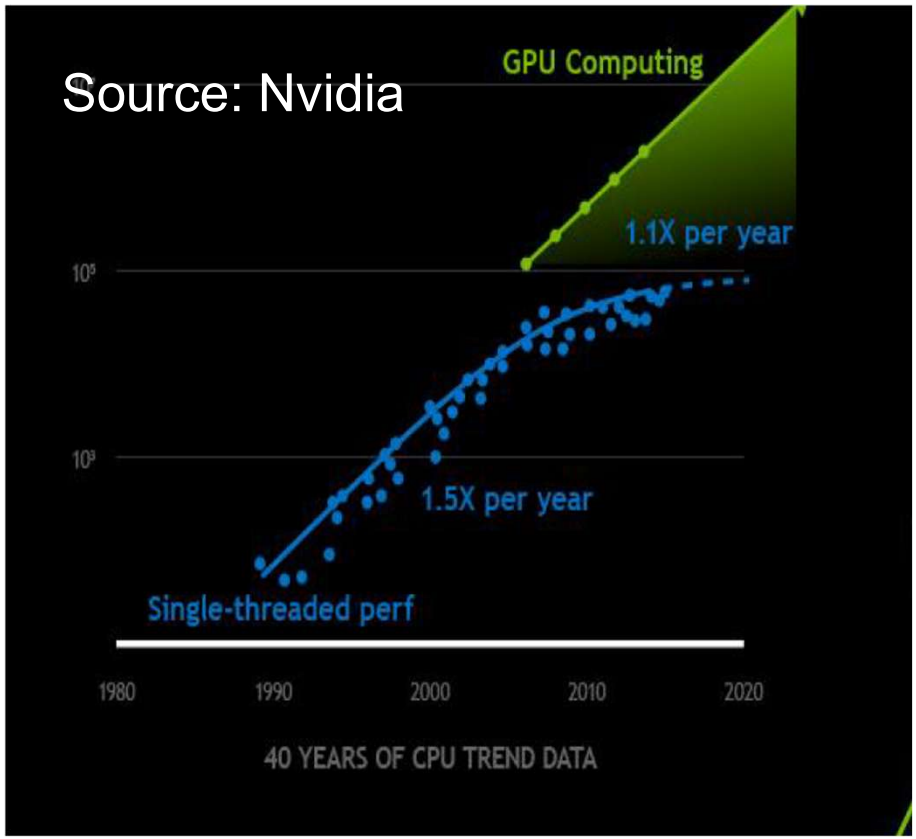
Gnosis Research Center for accelerating data-driven discovery

Illinois Institute of Technology

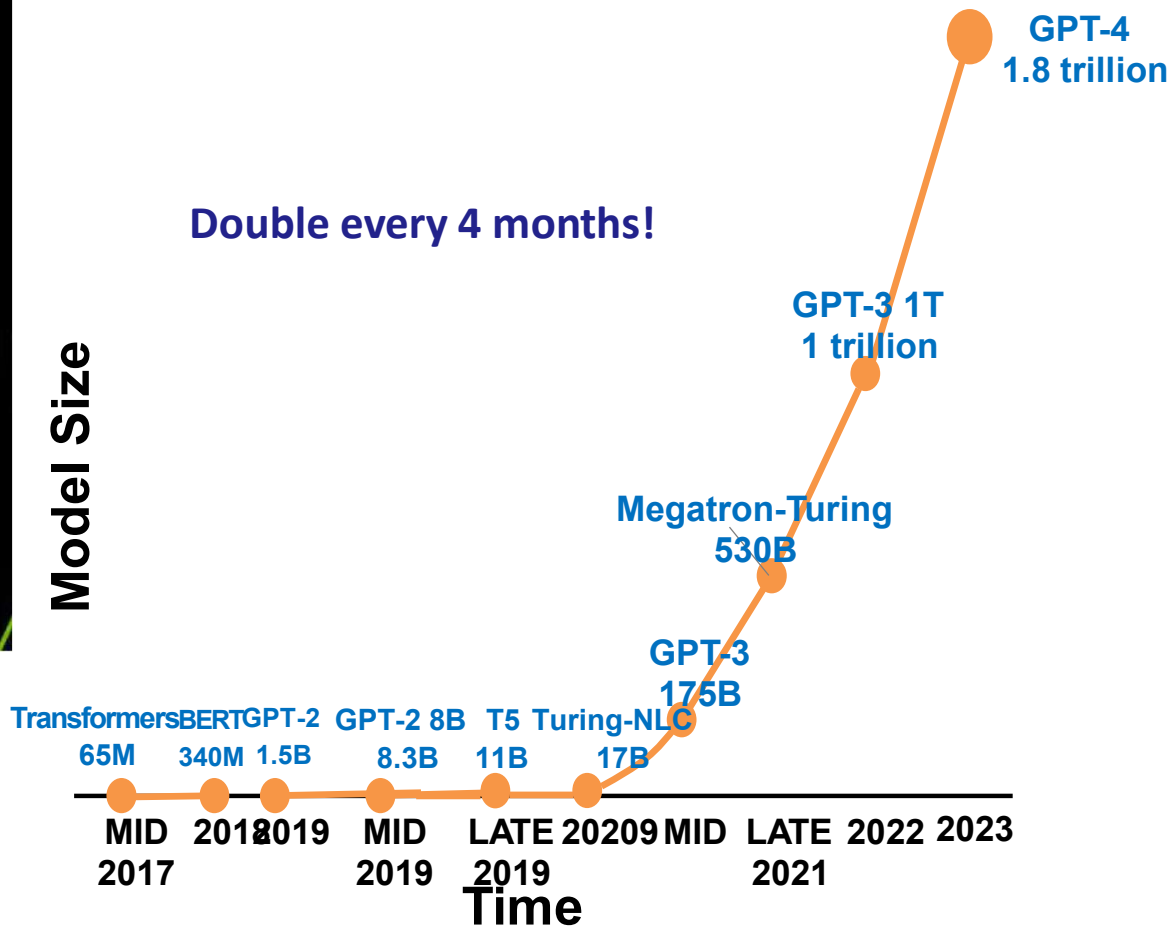
sun@iit.edu

PERMAVOST2025

The Computing & Data Disparity in AI



The new Moors' Law

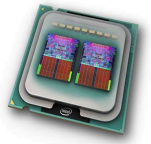


Courtesy of Tuo-Hung (Alex)Hou

Source: Amazon Web Services

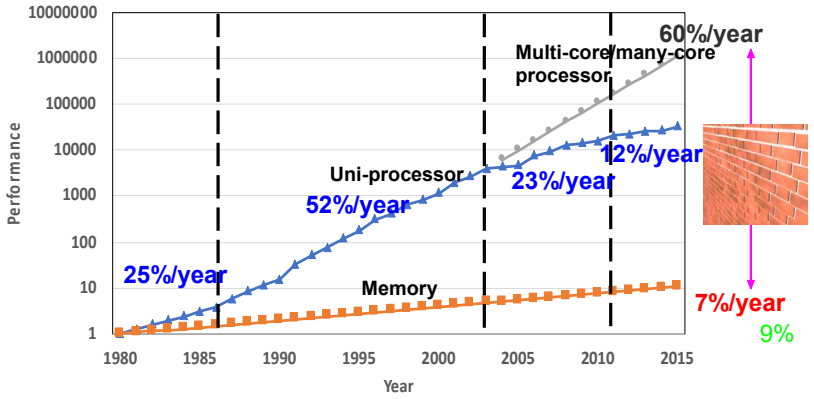
Why Data Access ? The Memory-wall Problem

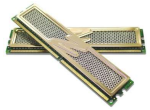
- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004
  - Aggregate multi-core/many-core processor performance even higher since 2004
- Memory: ~7% per year
  - Storage: ~6% per year
- Processor-memory speed gap keeps increasing



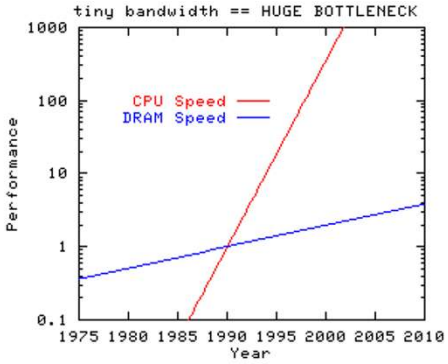
Source: Intel

Leiserson, Charles E., et al. "There's plenty of room at the Top: What will drive computer performance after Moore's law?." *Science* 368.6495 (2020).





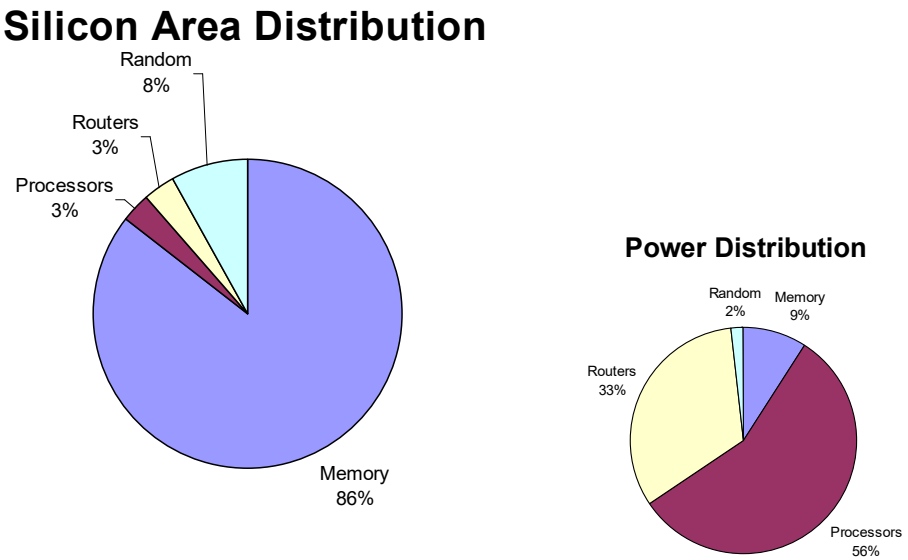
Source: OCZ



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

X.-H. Sun, and L. Ni, "Scalable Problems and **Memory-Bounded Speedup**," Journal of Parallel and Distributed Computing, Vol. 19, 1993.
X.-H. Sun, and L. Ni, "Another View of Parallel Speedup," Proc. of ACM-IEEE Supercomputing'90, NY, Nov. 1990

Impact of Memory-Bound (-Wall)



- Modern microprocessors such as the Pentium Pro, Alpha 21164, Strong Arm SA110, and Longson-3A use 80% or more of their transistors for the on-chip cache

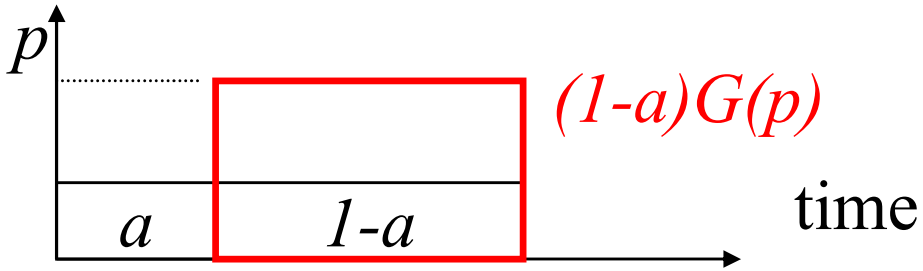
Courtesy of Peter Kogge, UND

- 1989 the first Intel processor with on-chip L1 cache was Intel 486, 8KB size
- 1995 the first Intel processor with on-chip L2 cache was Intel Pentium Pro, 256KB size
- 2003 the first Intel processor with on-chip L3 cache was Intel Itanium 2, 6MB size
- 1980: no cache in micro-processor; 2010: 3-level cache on chip, 4-level cache off chip

Source: Computer Architecture A Quantitative Approach

Memory-Bound & Scaling Function (Sun-Ni's Law)


Xian-He Sun



$$Speedup_{MB} = \frac{Work(p)/Time(p)}{Work(1)/Time(1)} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)G(p)/p} \qquad W^* = G(p*M)$$

Scalability of Multicore

$$\frac{\frac{w_c}{perf(r)} + \frac{w_p'}{m \cdot perf(r)}}{\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)}} = \frac{w_c + m \cdot w_p}{w_c + w_p} = (1 - f') + mf'$$

$$f' = \frac{w_p}{w_c + w_p}$$

Based on Sun-Ni's law **Multicore is scalable**, if data access time is fixed and does not increase with the amount of work & the number of cores

X.-H. Sun and Y. Chen, "Reevaluating Amdahl's Law in the Multicore Era," Journal of Parallel and Distributed Computing, vol. 70, no. 2, pp. 183-188, Feb. 2010

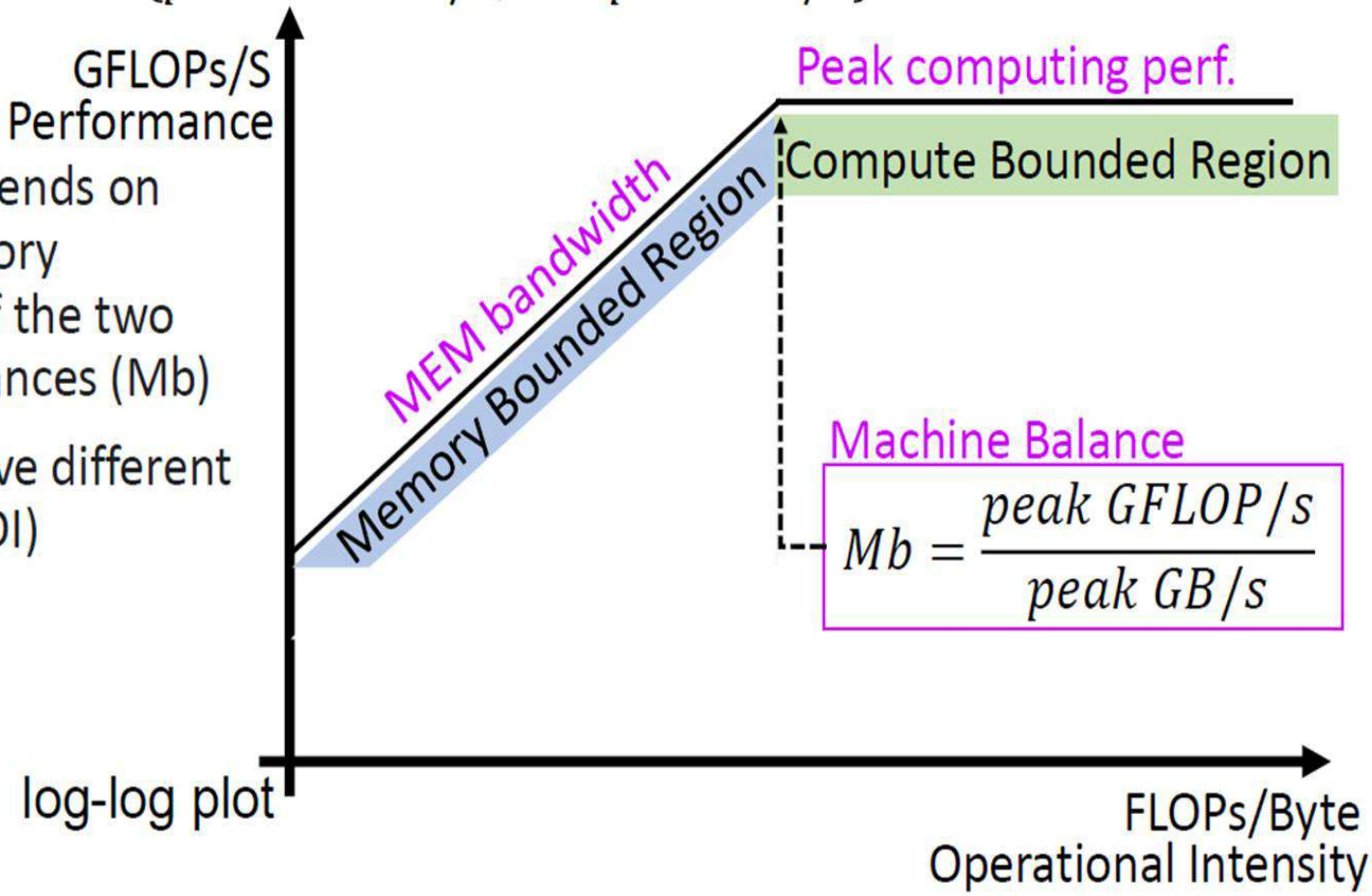
Compute and Memory Performance

Roofline Model:

$$Attainable\ GFLOP/s = min\{peak\ GFLOP/s, OI \cdot peak\ GB/s\}$$

- System Performance depends on both Compute and Memory Performance, the ratio of the two defines the Machine Balances (Mb)
- Different Applications have different Operational Intensities (OI)

$$OI = \frac{Num\ of\ FLOP}{Num\ of\ Byte}$$

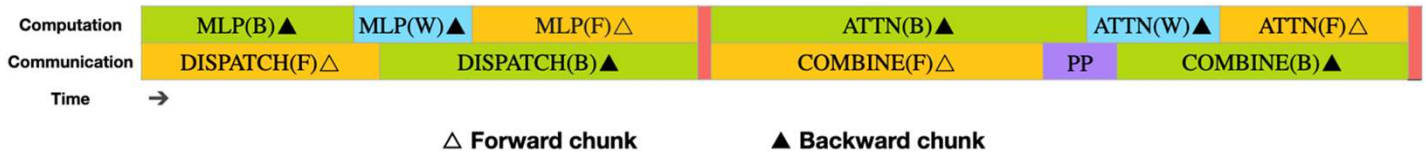


Use the scaling function G(pM) in memory-bound as the Operational Intensities (OI)

Micron E. Confalonieri, IEDM Short Course 2024

The deepseek Example: data-access optimization

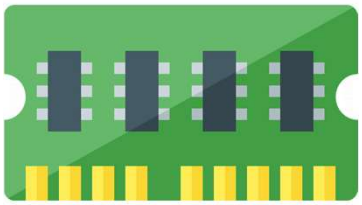
- *DeepSeek MoE uses smaller, fine-grained experts to enhance data parallelism*
- *MoE Expert Routing ensures load balance and prevents parallelism inefficiencies*
- *Optimized pipeline & overlapping hide data movement latency*
- *All-to-All Communication Optimizations mitigate the data movement overheads*
- *Memory-Efficient Training & FP8 Mixed Precision reduce memory requirements and mitigate data movement pressure*
- **Still:** *DeepSeek inference spends **12x more time** on data access than of computing time*



Overlapping strategy for a pair of individual forward and backward chunks

DeepSeek-V3 Technical Report

(Performance) Bound is not just Capacity



Just Buy More DRAM

DRAM is ***very*** expensive

DRAM has a ***very*** high energy cost

Only **20%** of memory used in 20% of time

The **Memory-wall** problem

Larger memory means **longer** searching time

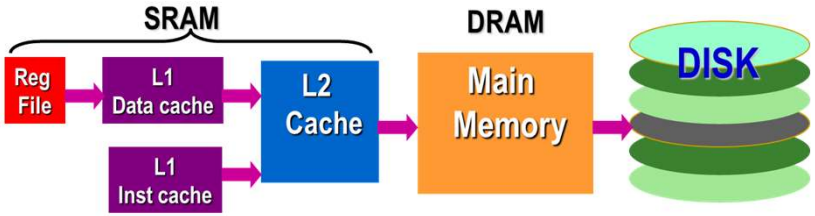
	Price	Capacity	$/GB
DRAM	$106	32GB	$3.30
PMEM	$275	128GB	$2.10
NVMe	$169	2TB	$0.08
SATA SSD	$175	4TB	$0.04
HDD	$369	18TB	$0.02

40x more expensive than NVMe

Simply increasing DRAM is not sustainable!

Existing Solution: ASIC from CPU side

- GPU, DSP, AI Chip
  - GPU is a chip tailored to graphics processing, DSP is for signal processing, and AI chip is designed to do AI tasks
- Limited solution
  - Assume data are on the chip
- Limited application
  - *Computation Accelerator*
  - Please recall our memory-bound results for multicore



The Traditional Approach: Memory Hierarchy

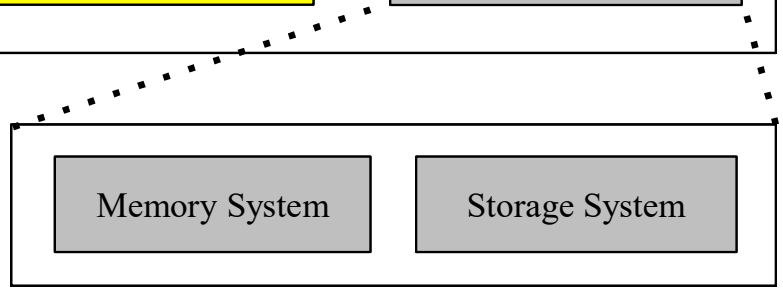
New Solution: PIM chip

- PIM
  - Processing in memory (also called processor in memory) is the integration of a processor with RAM on a single chip.
  - NDP (Near-memory Data Processing)
  - ISP (In-Storage Processing)
- Computer power is weak
  - A full kitchen needs a refrigerator
- Limited application
  - *Data movement reducer*
  - A helper/mitigator

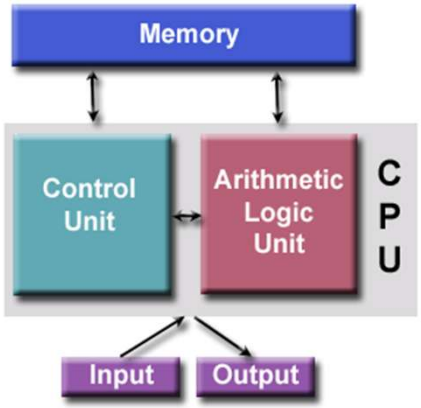
How to use it?

Our Solution: A new component of von Neumann

- Can we make the von Neumann machine **data efficient**?
- **Yes**: focus on data and data access delay (data centric)
- **How**: *Advancing current memory-wall solutions*



Computing System



The Problem is in the Middle

Research

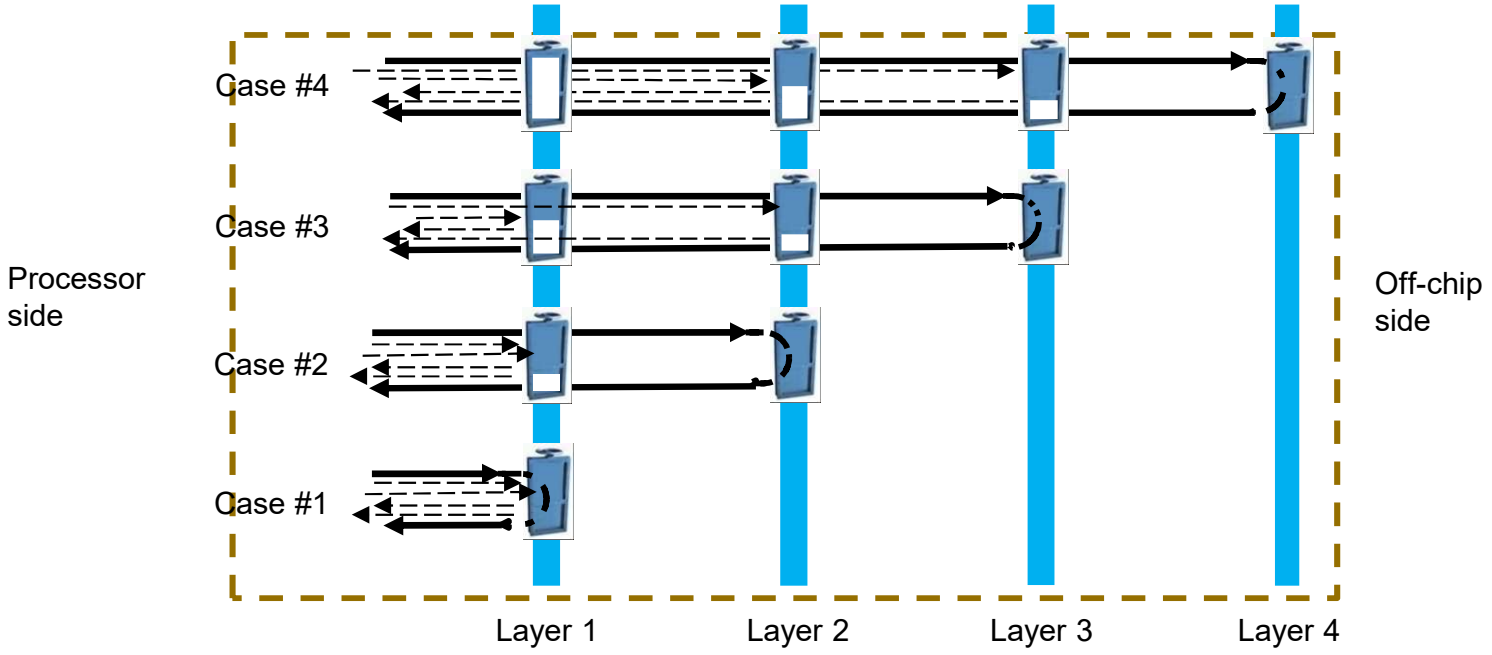
- System Research
- Big Data/AI Systems
- High Performance Computing Data Systems

Dataflow under the von Neumann Machine

Memory Stall Time (MST)

$$\text{CPU.time} = \text{IC} \times (\text{CPI}_{\text{exe}} + \textit{Memory stall time}) \times \text{Cycle.time}$$

Reduce the Memory Stall Time to minimum

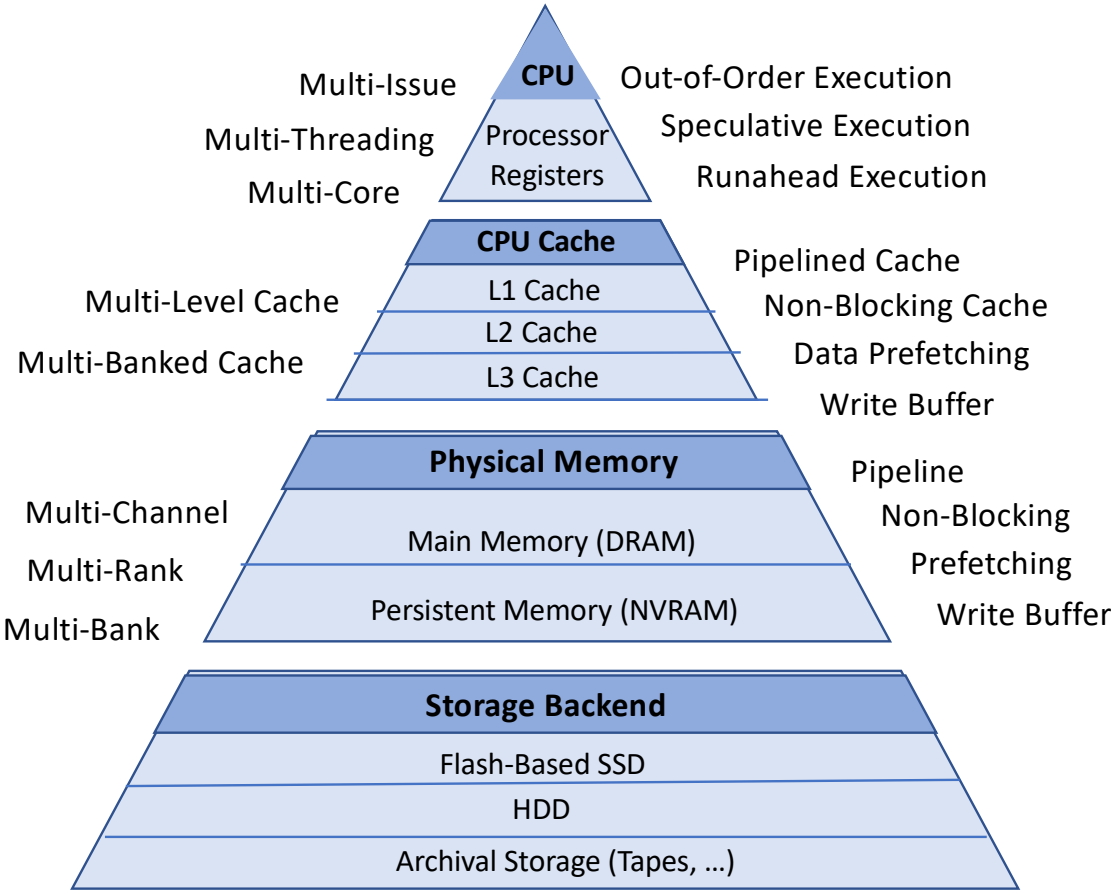


(including in place computing)

Utilize current memory system

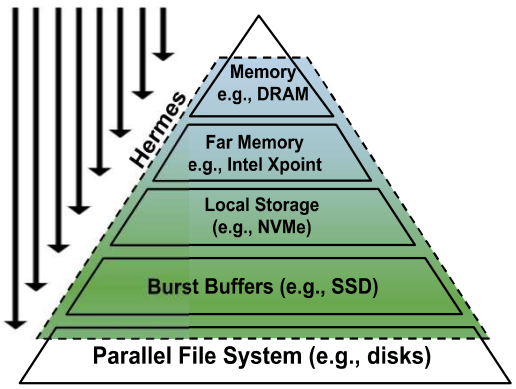
New Disruptive under Existing System:
Integration & Concurrency

- Advanced solution: Deep Memory/IO Hierarchy
  - New technologies and complexities
  - **Locality** and **Concurrency**



Current Deep Memory-Storage Hierarchy
with Concurrence

Our Idea:
Integrated Memory System via Data Access Concurrency



Challenges: Currency, Integration, environment, technology, etc.

Our Idea

C-AMAT: Concurrent Data Access fundamental

$$AMAT = H_1 + MR_1 \times AMP_1$$
Where $AMP_1 = (H_2 + MR_2 \times AMP_2)$

- C-AMAT is Recursive

Where:

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

$$C - AMAT_2 = \frac{H_2}{C_{H_2}} + MR_2 \times \kappa_2 \times C - AMAT_3$$

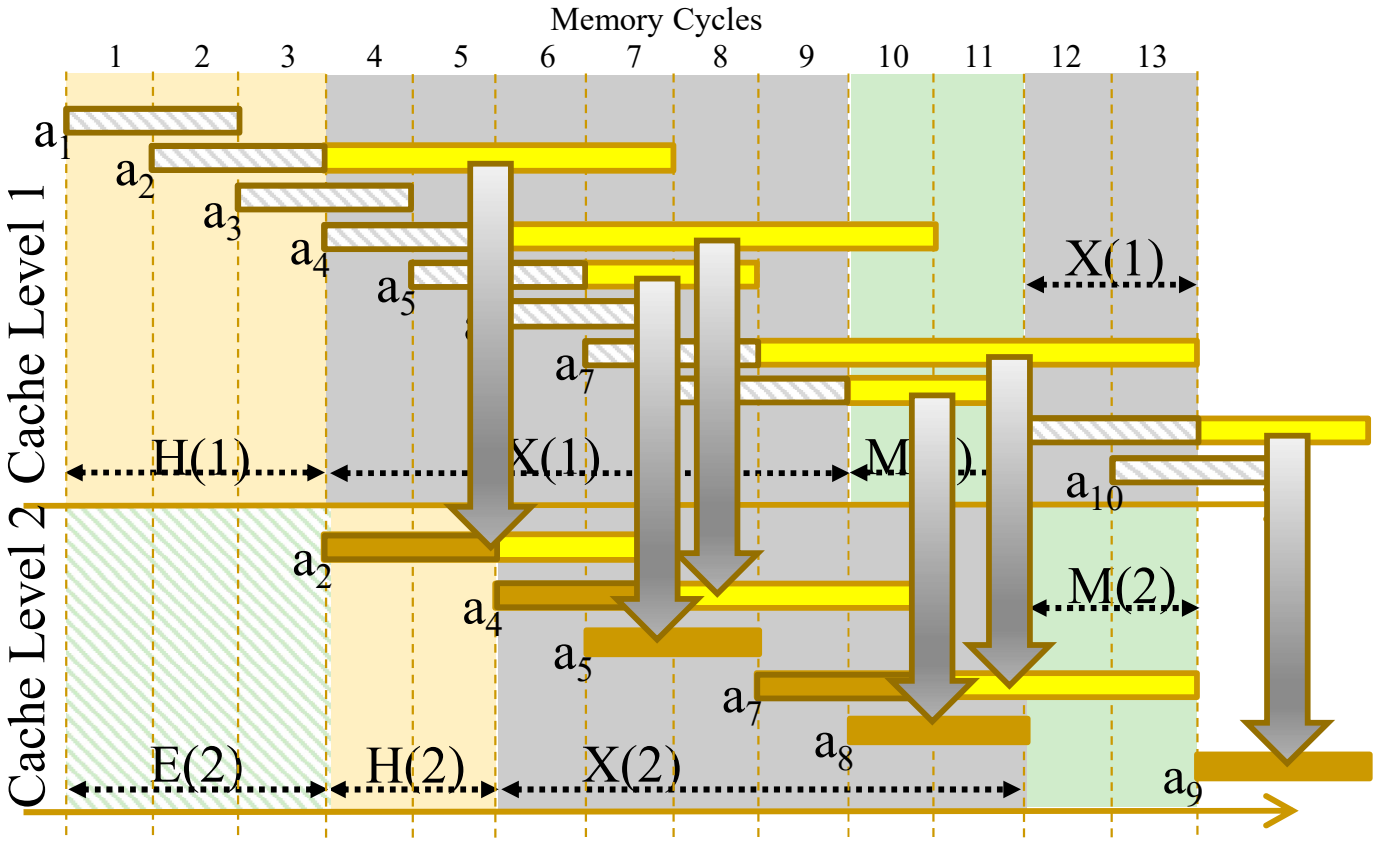
- H: the hit time
- MR: the miss ratio
- C_H : the hit concurrency
- κ: the overlapping ratio (pure miss cycles over miss cycles)
- A pure miss cycle is a miss cycle with no hit

$$\kappa_1 = \frac{pMR_1}{MR_1} \times \frac{pAMP_1}{AMP_1} \times \frac{C_{m\ 1}}{C_{M\ 1}}$$

A lot of Room in data Concurrency

Concurrence & locality are equally important

Sun, Xian-He, and Dawei Wang. "Concurrent average memory access time." IEEE Computer 47, no. 5 (2014): 74-80.



L1 pure misses cause CPU stall

From concurrence-supported locality

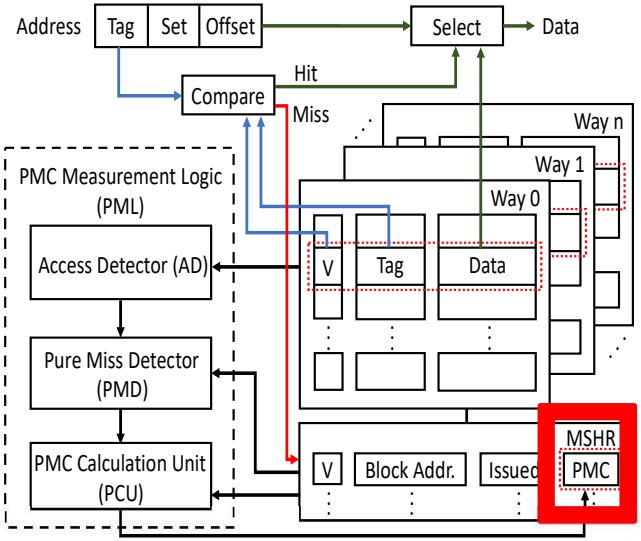
To Concurrence-aware Locality, & Locality-aware concurrence,

J. Liu, P. Espina, & X.-H. Sun, "A Study on Modeling and Optimization of Memory Systems," Journal of Computer Science and Technology (JCST), vol. 36, no. 1, pp. 1-19, January 2021

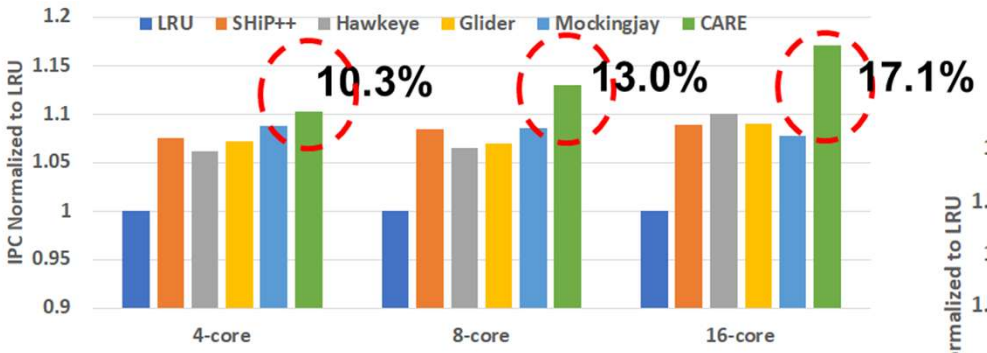
Put in Action: Concurrency-Aware Cache Replacement

Design

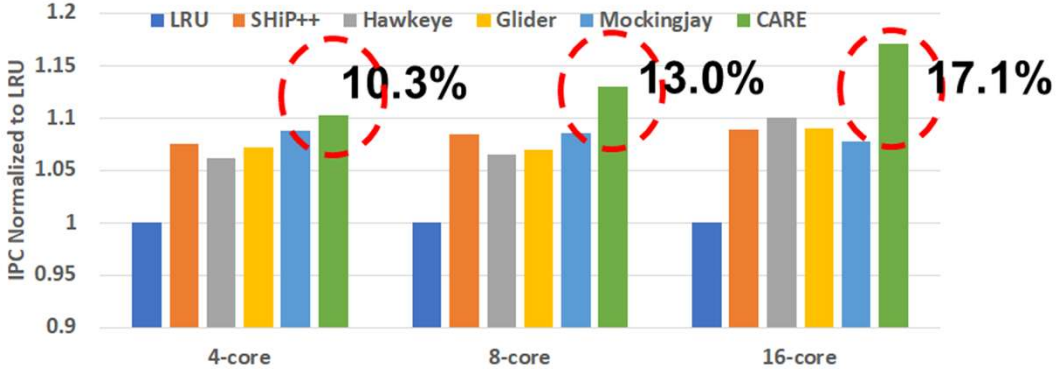
- **Pure Miss Contribution (PMC)**
  - Pure miss is more important than miss
  - Pure miss has weight (pure miss cycles)
  - PMC is the number of pure miss cycles
  - PMC can be measured by monitoring MSHR occupancy
- **PMC is Predictable**
  - PMC values of the misses caused by the same Program Counter (PC) are relatively stable



PMC Measurement Structure



SPEC & GAP Workloads

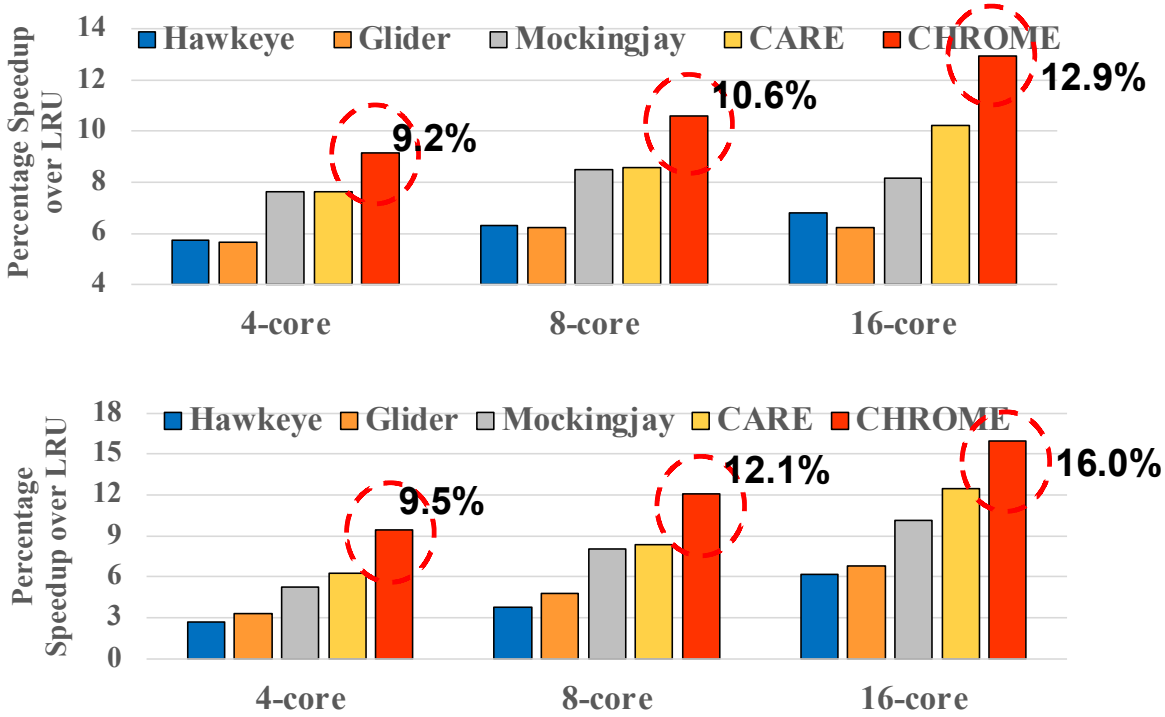


X. Lu, R. Wang, and X.-H. Sun. "CARE: A Concurrency-Aware Enhanced Lightweight Cache Management Framework", in the 29th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2023).

Action: Concurrency-Aware Holistic Cache Management

CHROME:

- Integrates **cache bypassing** and **replacement** with pattern-based **prefetching**
- **Dynamic Online Learning**: for varying workloads and system configurations
- **Multiple Program Features** for understanding of memory access patterns
- **Concurrency-Aware Rewards for concurrent accesses (C-AMAT model)**



CHROME outperforms state-of-the-art cache management schemes

– Effective across both SPEC and GAP memory-intensive workloads

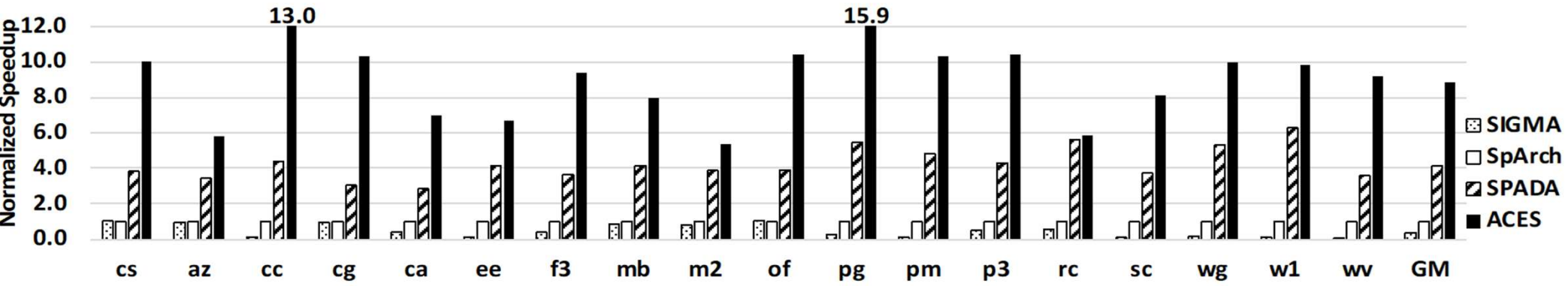
– Demonstrates strong scalability

X. Lu, H. Najafi, J. Liu, and X.-H. Sun, "CHROME: Concurrency-Aware Holistic Cache Management Framework with Online Reinforcement Learning ," in the 30th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2024).

Action: Concurrency-Aware SpMM Accelerator

ACES : Concurrency-Aware SpMM Accelerator

- **Adaptive Execution Flow:** better software concurrency
- **Non-Blocking (NB) Buffer:** better hardware concurrency
- **Concurrency-Aware Cache Replacement:** Considers **Reuse Distance (RD)** and **Fiber Density (FD)**, allows all cache lines of a row to be accessed **concurrently** without any **misses**

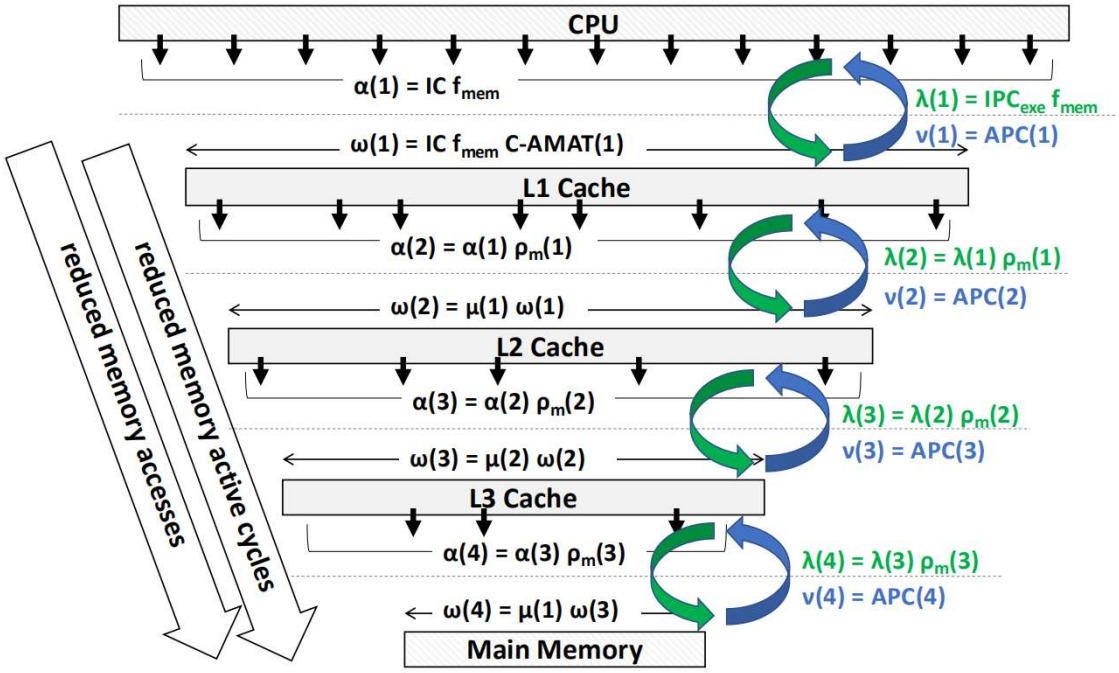


- ACES consistently provides **optimal** performance across **all** workloads
- **25.5×** over SIGMA, **8.9×** over SpArch, and **2.1×** over SPADA
- Non-Blocking (NB) Buffers enable greater concurrency support
- Concurrency-aware cache replacement policies further exploit this hardware-supported parallelism

X. Lu, B. Long, X. Chen, Y. Han, and X.-H. Sun, "ACES: Accelerating Sparse Matrix Multiplication with Adaptive Execution Flow and Concurrency-Aware Cache Optimizations ," in the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2024).

System Optimization: Layered Performance Matching (LPM)

- If each layer's performance is matched, then we get the top layer's performance and the lowest layer's capacity
- The *Matching* ratio values of request and supply at each layer are given and the matching process is well designed & analyzed



Simulatable → Measurable → Controllable → Optimizable

Y. Liu and X. Sun, "LPM: A Systematic Methodology for Concurrent Data Access Pattern Optimization from a Matching Perspective," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 11, pp. 2478-2493, 1 Nov. 2019.

Theoretical Foundation (modeling)

A Matching Example (increase performance)

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

- **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 improve memory *concurrency* by adding memory banks or ports to increase data supply ability to adjust the data supply rate. For example, increase *APC* from 1 to 2, so:

$$\boldsymbol{APC * N\_Cycle_{mem} = 2}$$

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

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.

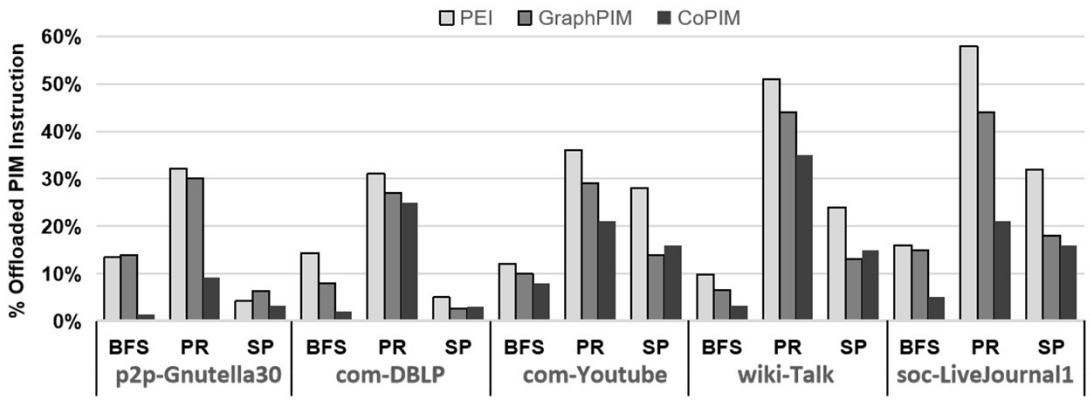
Case Study: Put PIM into the Formula

Memory Stall Time (MST)

$$\text{CPU.time} = \text{IC} \times (\text{CPI}_{\text{exe}} + \textit{Memory stall time}) \times \text{Cycle.time}$$

$$\text{CPU.time} = \text{IC}_{\text{exe}} \times (\text{CPI}_{\text{exe}} + \textit{Memory stall time}) \times \text{Cycle.time} + \text{IC}_{\text{pim}} \times (\text{CPI}_{\text{pim}} + \text{PIM stall time}) \times \text{Cycle.time}_{\text{pim}} - \text{Overlapping}$$

- PIM is a way to reduce request and is a trade-off of computing and MST
- Use Pure Miss as a indicator for offloading
- Result: data movement cost decides where to do the computing



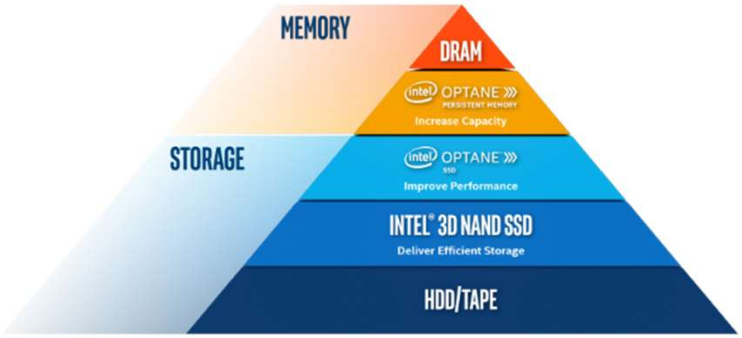
Percentage of offloaded instructions into memory

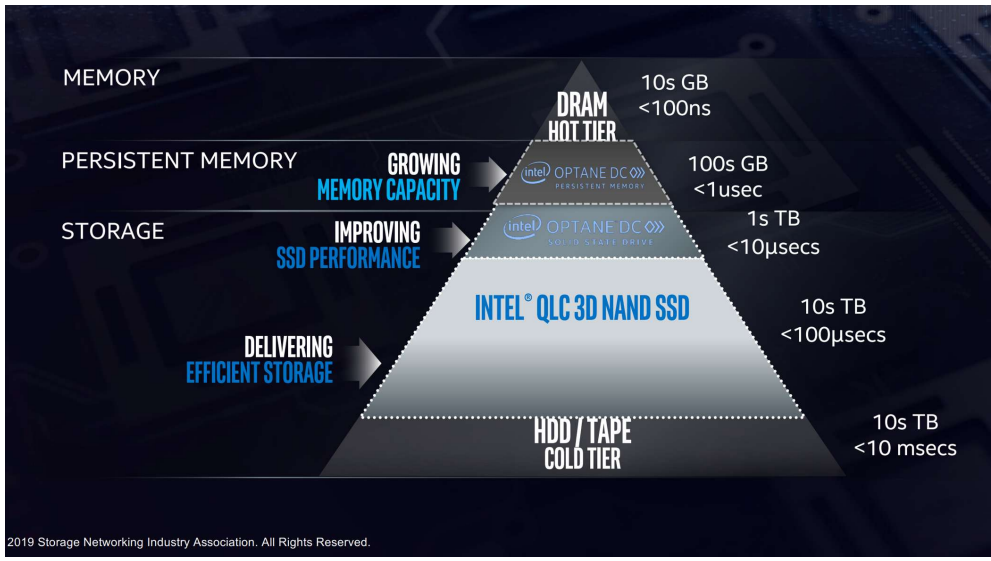
Speedup by **19.5%** than **PEI** with **51.1%** fewer offloaded instructions

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 (ISLPED202), July 26 – 28, 2021

Case Study 4: The Intel Optane Technology

- Manufacture hardware integration (matching)
- Challenges
  - Data access is application dependent
  - Hardware **elasticity**
  - Global optimization
  - OS, etc.
- ***An architecture ahead of time?***

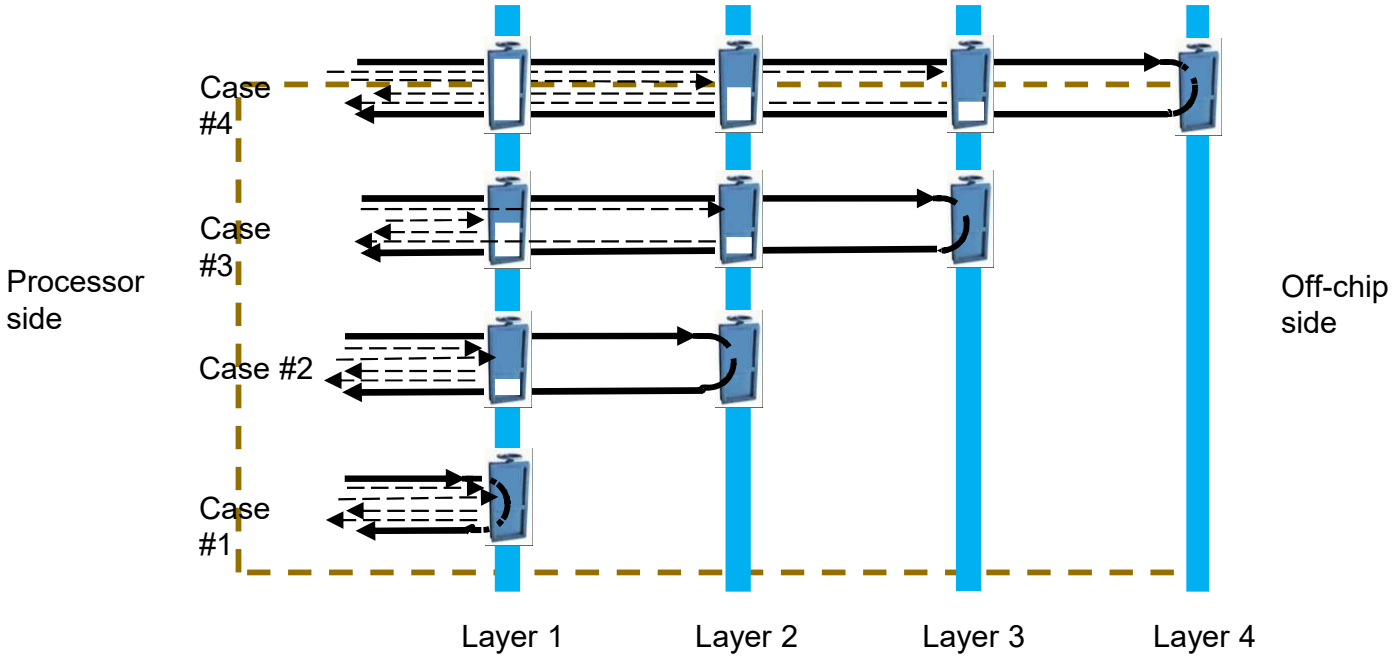




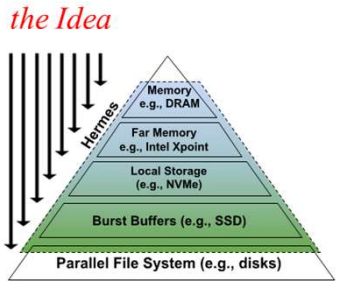
N. Zhang, B. Toonen, X-H. Sun, B. Allcock, "Performance Modeling and Evaluation of a Production Disaggregated Memory System," International Symposium on Memory Systems (MEMSYS'20), Sept. 2020

Dataflow under the von Neumann Machine

- Use **LPM** to utilize the memory system performance
  - Identify the number of layers and carry matching at each layer
- Use **C-AMAT** (concurrency) and others to perform the matching
- Goal: ***Dataflow*** from the source to compute unite with minimum stall



It is feasible



The *Dataflow*$_V$ Complexity

Possible Ways to Match

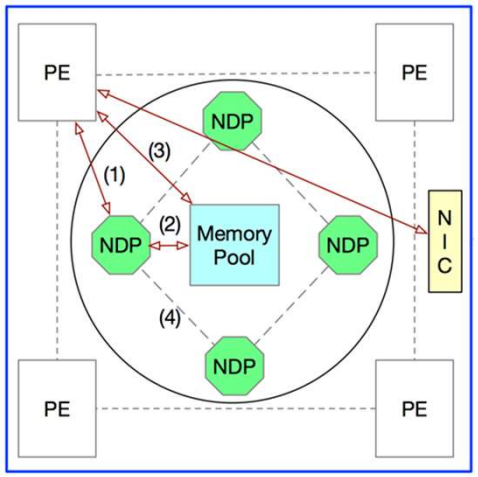
- Reduce request
  - Improve locality, **processing-in-memory (PIM) & in place computing,** algorithm improvement**,** data movement reducer, etc.
- Improve supply
  - Improve data access concurrency , buffer, new technology, etc.
- Mask the difference
  - **Overlapping** computing with data access delay (pure miss), prefetch, etc.
- A more balanced design, etc.

Hardware technology, compiler technology, application algorithm designs, system scheduling, system support, co-design, integration, etc.

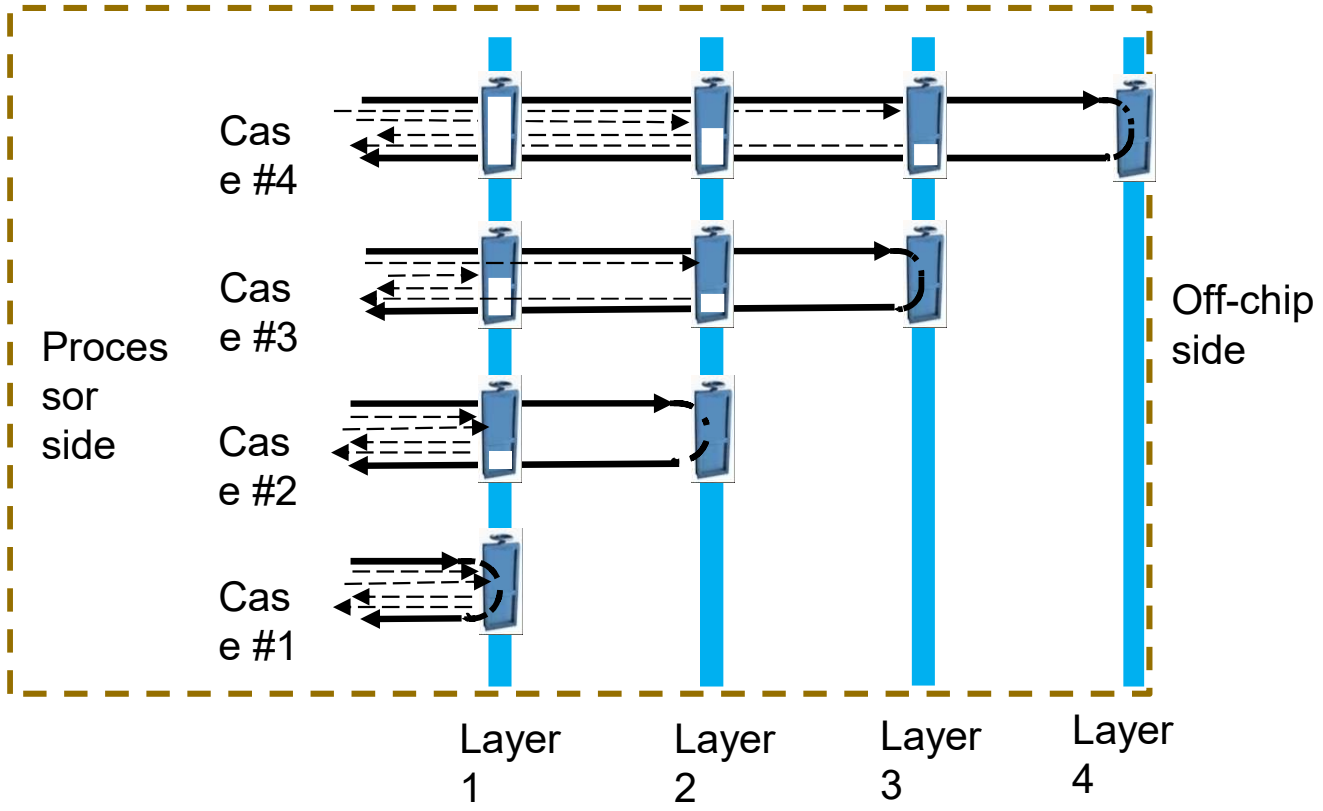
Action: Towards a Unified Memory-centric Computing System with Cross-layer Support

No delay data access: An integration of memory, storage, architecture (PIM, NDP. CXL, etc), OS, compiler, runtime systems

- **Is your new device/ technology helpful?**
- **How should they be used together?**



New advances in computer architecture

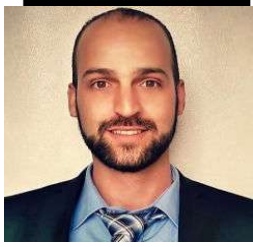


Dataflow under the von Neumann machine

H. Geng, X. Lu, Y. Che, Z. Tian, D. Cheng, X.-H. Sun, M. Niemier, X. Hu, "COSMOS: RL-Enhanced Locality-Aware Counter Cache Optimization for Secure Memory", in the proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO-58), Conference, Oct, 2025 (accepted to appear).

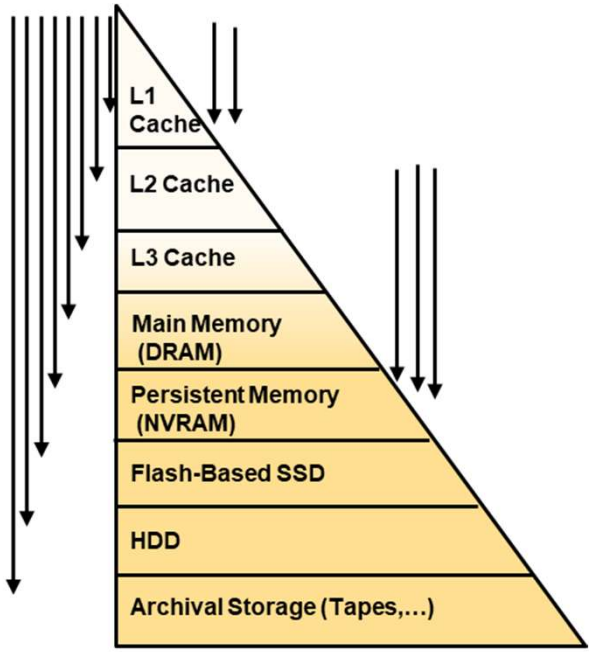
Put in Action: Hermes: A Smart Multi-tiered I/O Buffering System

OCI-1835764
CSSI: Frameworks

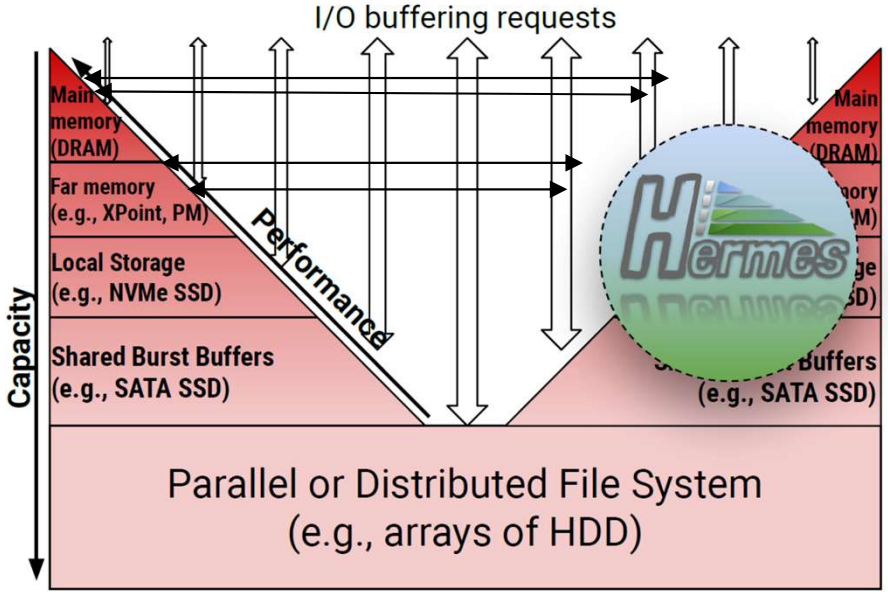


Anthony Kougkas

- Storage as a layer of the memory hierarchy
- Selective cache, data concurrency, matching
- Independent management of **each tier**
- Optimized Access Latency, Throughput, Capacity, Network topology, etc.



DMSH with Concurrence

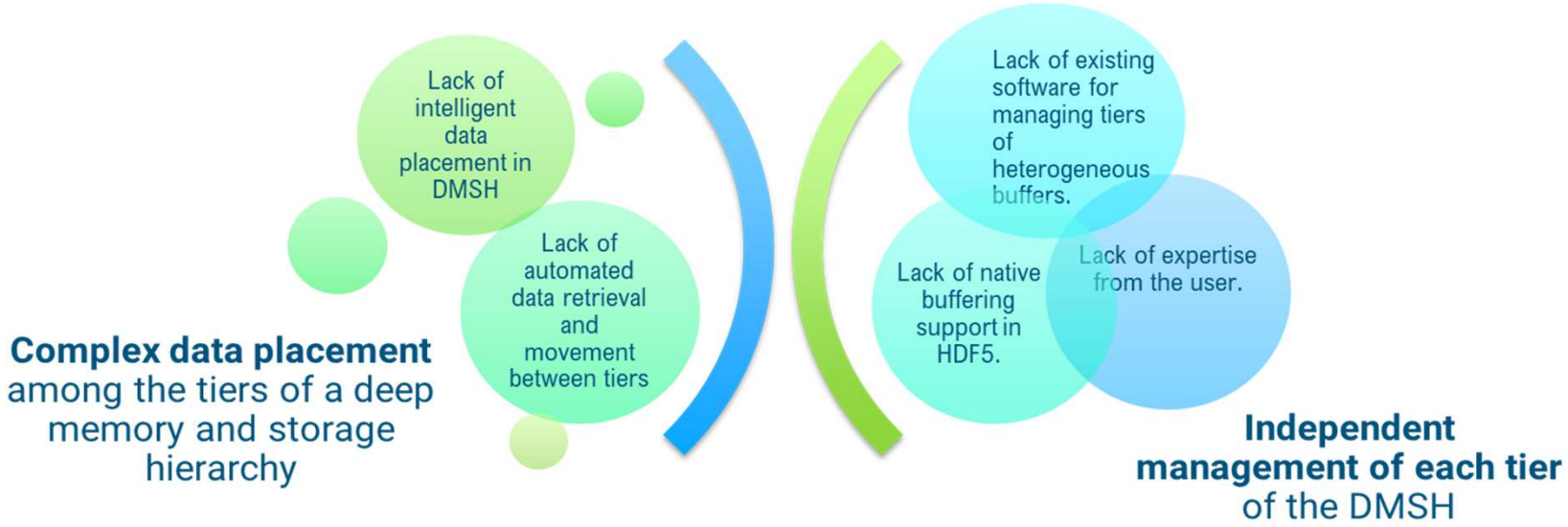


A. Kougkas, H. Devarajan, and X.-H. Sun, "I/O Acceleration via Multi-Tiered Data Buffering and Prefetching," Journal of Computer Science and Technology, vol. 35, no. 1, pp. 92-120, Jan. 2020

Our First Data Management System

The Development Challenges

- Application-aware multi-tier optimization
- Complexity in software development
- Complexity in decision making
- An example of memory/storage integration
- An implementation of the layered *matching and data flow* concept



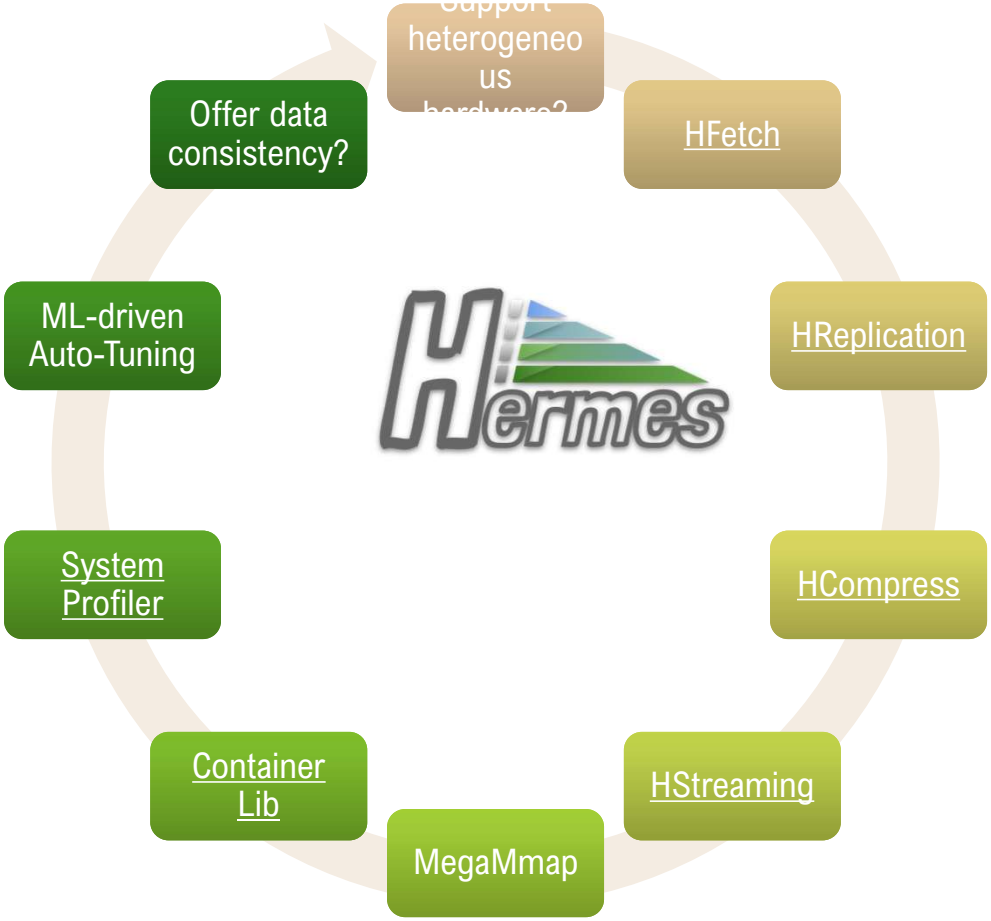
A. Kougkas, H. Devarajan, and X.-H. Sun, "Bridging Storage Semantics using Data Labels and Asynchronous I/O," ACM Transactions on Storage, Vol 16, No 4, 2020

Technical Preparation

Hermes introduce several independent technologies

Explore our code at https://github.com/grc-iit.

github.com/grc-iit/hfetch



github.com/grc-iit/hreplica

github.com/grc-iit/hcompress

github.com/grc-iit/hflow

ML-driven Auto-Tuning and Auto-Configuration: N. Rajesh, K. Bateman, S. Byna, J. L. Bez, A. Kougkas, X.-H. Sun. “TunIO: An AI-powered Framework for Optimizing HPC I/O”, IEEE IPDPS 2024.

Other Hermes Tools and Services

- **Application orchestrator[1]**:
  - schedules application's access to Hermes in multi-tenant scenarios, mitigating interference by shared buffers.
- **System telemetry[2]**:
  - captures in real-time the status of tiers (e.g., load, remaining capacity, etc.,) to enable efficient data placement by Hermes.
- **Application I/O profiling[3]**:
  - classifies the I/O phases of applications to enable optimal Hermes configuration tuning based on incoming access patterns.
- **Hermes Container Library[4]:**
  - implements STL-like data structures (maps, list, etc) that can be distributed across nodes and within available storage tiers.
- **Resource Management[5]**:
  - provision and allocate the best storage hardware resources available based on application demands and QoS guarantees
- **The Memory Management System**

1.A. Kougkas, H. Devarajan, X-H. Sun, and J. Lofstead. "Harmonia: An Interference-Aware Dynamic I/O Scheduler", In IEEE International Conference on Cluster Computing 2018 (Cluster'18)

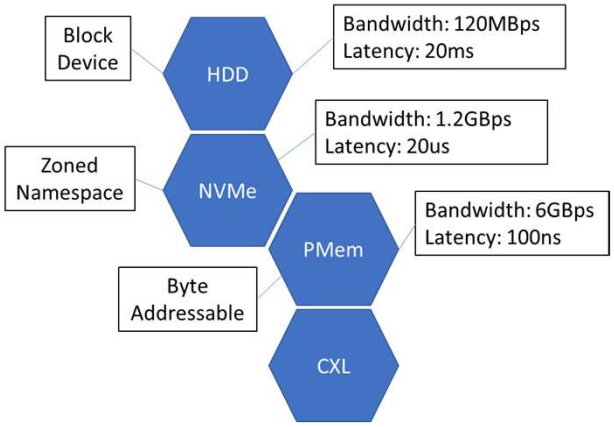
2.N. Rajesh, H. Devarajan, J. Cernuda Garcia, K. Bateman, L. Logan, J. Ye, A. Kougkas, and X-H. Sun. 2020. "Apollo: An ML-assisted Real-Time Storage Resource Observer". In 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC'21)

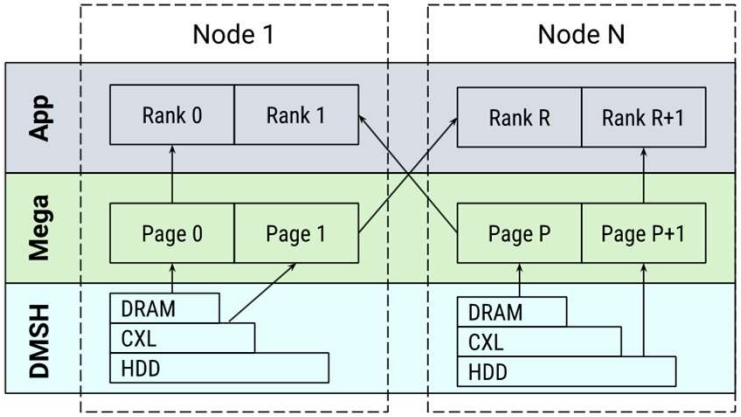
3.Hariharan Devarajan, Anthony Kougkas, P. Challa, Xian-He Sun, "Vidya: Performing Code-Block I/O Characterization for Data Access Optimization" In IEEE International Conference on High Performance Computing, Data, and Analytics 2018 (HiPC'18)

4.H. Devarajan, A. Kougkas, K. Bateman, and X-H Sun. "HCL: Distributing parallel data structures in extreme scales." In 2020 IEEE International Conference on Cluster Computing (Cluster'20).

5.K. Bateman, N. Rajesh, J. Cernuda, L. Logan, J. Ye, S. Herbein, A. Kougkas, and X.-H. Sun. "LuxIO: Intelligent Resource Provisioning and Auto-Configuration for Storage Services" In IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC'22)

Blurring the Line of Memory & Storage





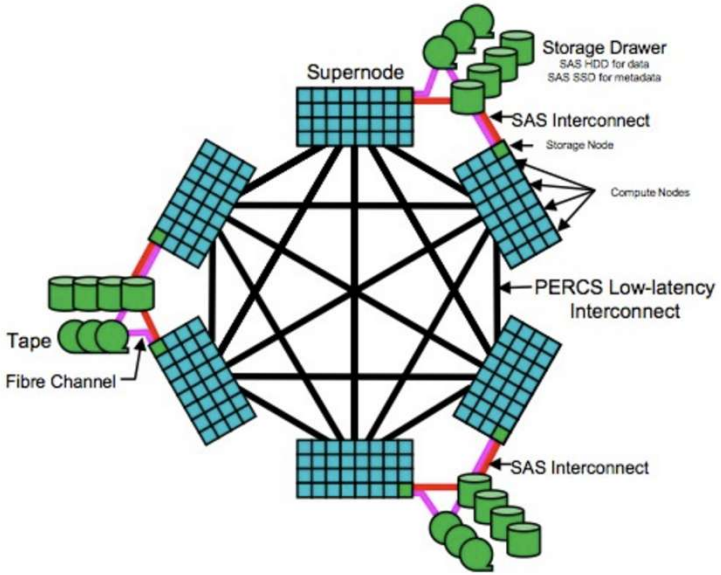
- Storage has emerged with similar performance to DRAM
- E.g., Compute Express Link (CXL) is emerging and allows storage to be accessed as memory
- **We can combine DRAM and storage into a tiered and portable distributed shared memory (DSM) abstraction**

Luke Logan

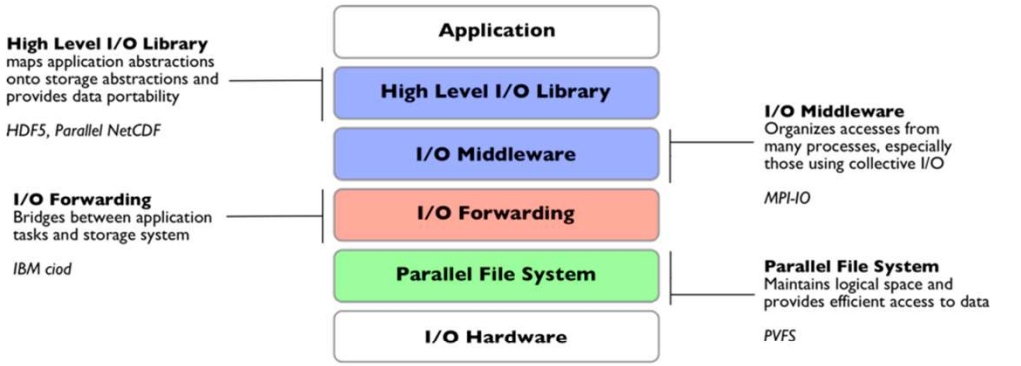
L. Logan, X.-H. Sun, A. Kougkas, “MegaMmap: Blurring the Boundary Between Memory and Storage for Data-Intensive Workloads”, in the Proceedings of IEEE/ACM SC'24, Atlanta, USA, Nov. 17-22, 2024.

The Complexity of HPC I/O Systems

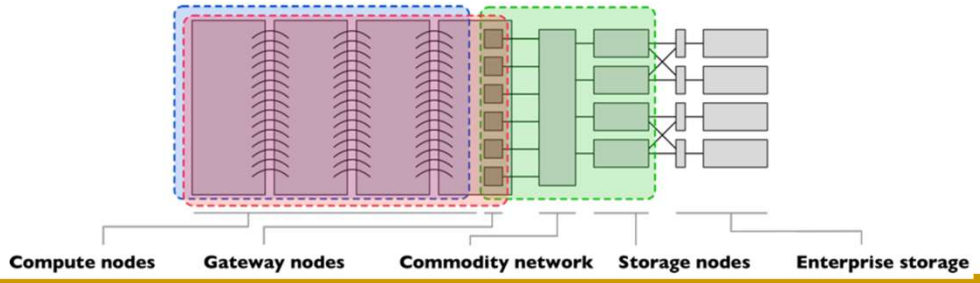
IBM Blue Gene/P system at Argonne National Laboratory.

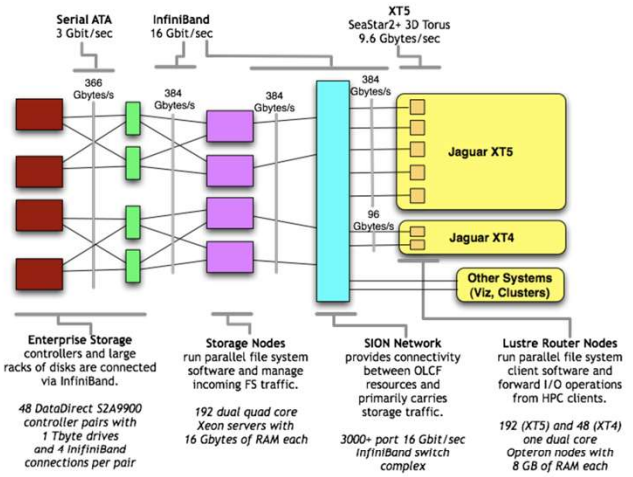


Bluewaters Storage System



The I/O Software Stack of Intrepid





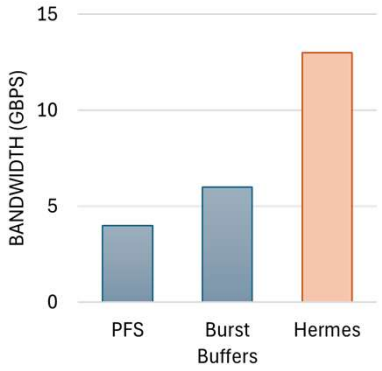
Jaguar Storage System

Courtesy from Argonne Leadership Computing Facility

Hermes System Results

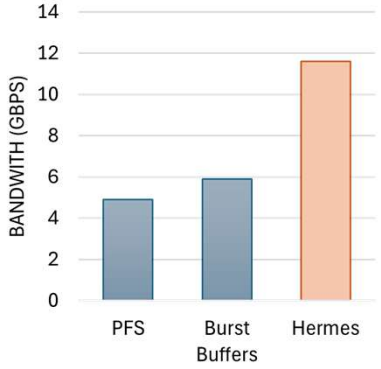
Utilizing data access concurrence and memory hierarchy system with Multi-tiered I/O buffering

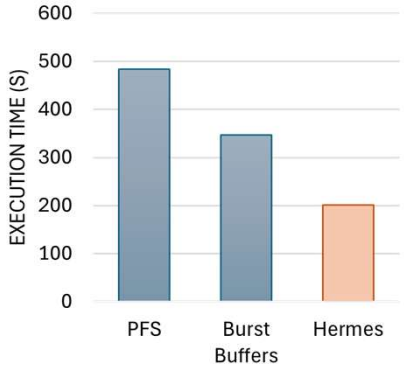
HPC Simulations and Big Data Analytics



- Vector Particle-In-Cell (VPIC)
  - Write intensive simulation
  - **3x faster** than baseline (PFS)
  - 2.2x improvement compared to single-tier buffering solutions.

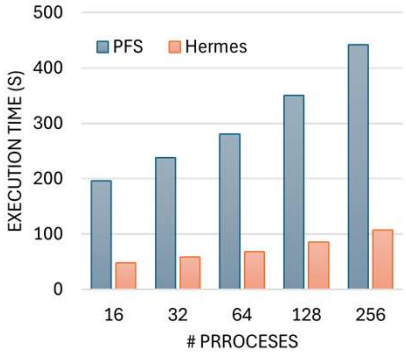
- Particle clustering (BD-CATS)
  - Read intensive analysis
  - 2.4x faster than baseline (PFS)
  - 2x improvement compared to single-tier buffering solutions.





- Gray-Scott reaction and diffusion models
  - Write intensive phase
  - **2.5x faster** than baseline (PFS)
  - 1.7x improvement compared to single-tier buffering solutions.

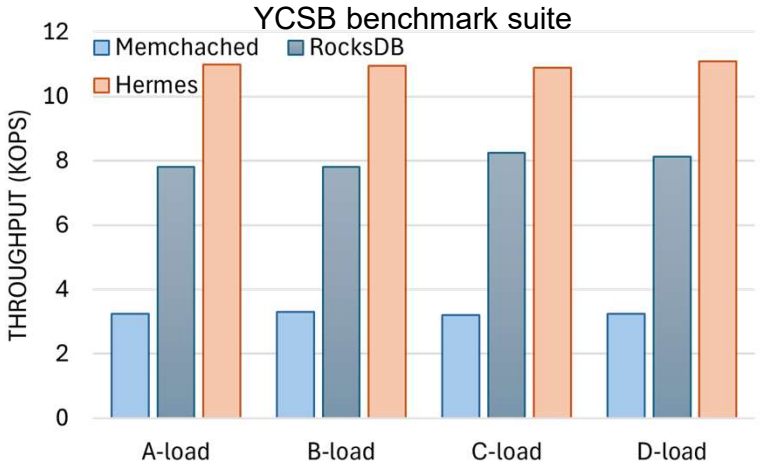
- Gray-Scott Analysis
  - Read intensive phase
  - 4x faster than baseline (PFS)



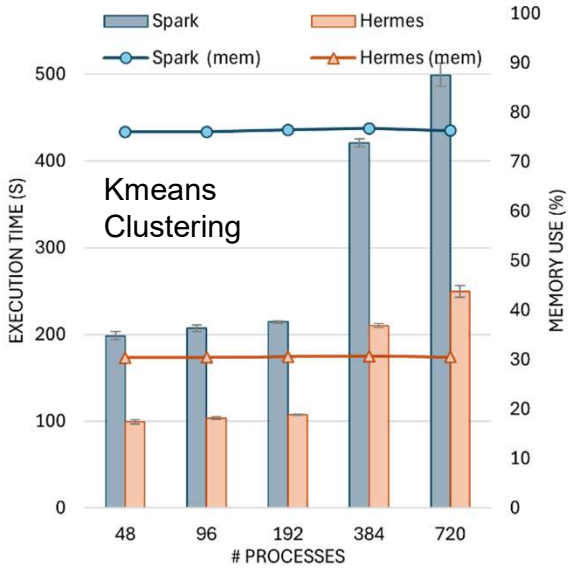
Hermes System Results

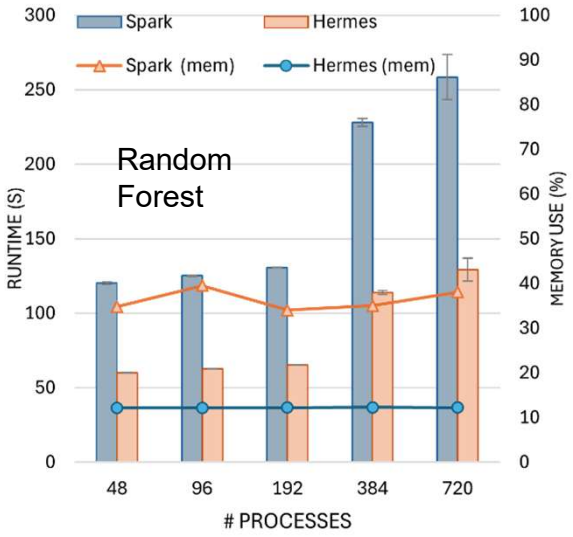
Utilizing data access concurrence and memory hierarchy system with Multi-tiered I/O buffering

Cloud and ML Workloads



- Performance superiority, using the same resources, for a variety of cloud workloads:
  A. Webserver
  B. Email Server
  C. DB queries
  D. KVS get/put





- Hermes outperforms SparkML in distributed ML algorithms by more than **3x highlighting** the performance benefits of data tiering.
- Hermes reduces DRAM usage by over **2x, enhancing data** processing capabilities without extra infrastructure overhead.

L. Logan, X.-H. Sun, A. Kougkas, "MegaMmap: Blurring the Boundary Between Memory and Storage for Data-Intensive Workloads", IEEE/ACM SC24, Nov. 2024

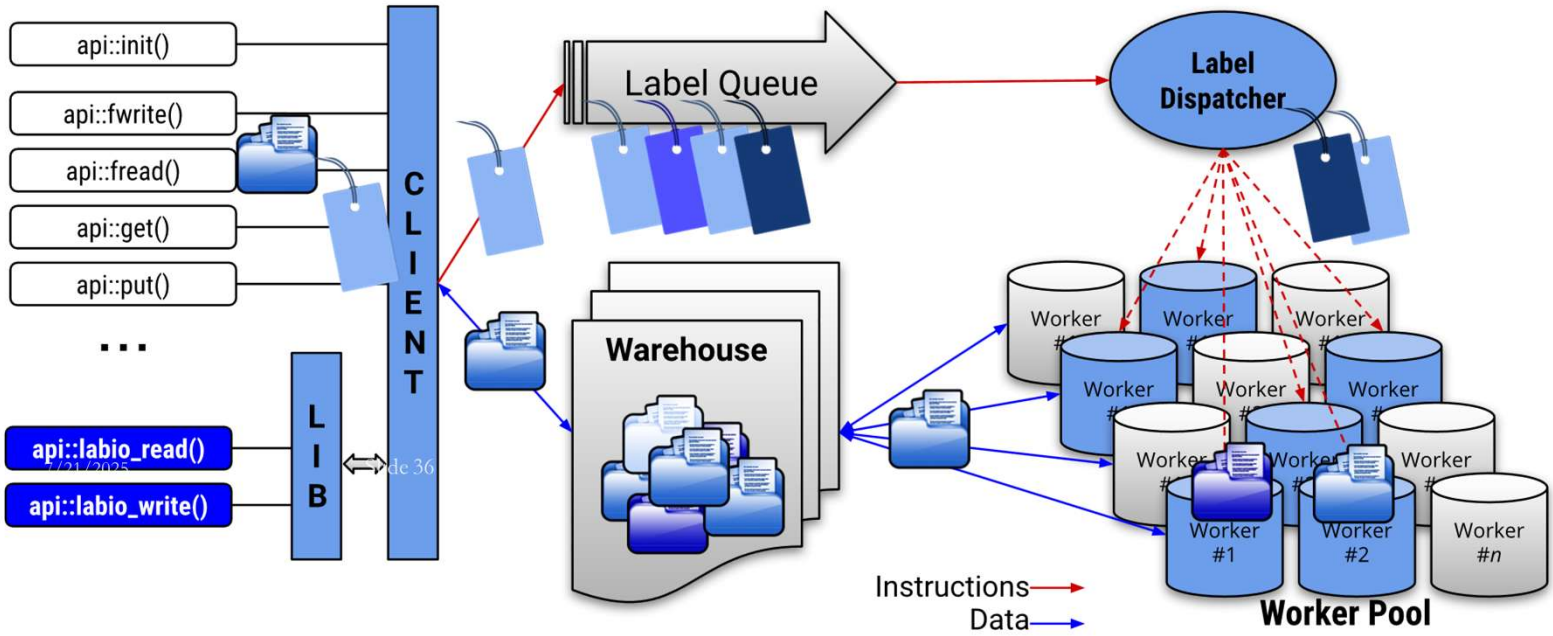
After Hermes

- Application and Collaboration
  - Extended to Active (log) data: the ChronoLog project
  - Extended to Meta data: the Coeus project
  - Extended to workflow environment, the Cloud environment
  - Extended to AI applications
- Fundamental: Data centric system integration
  - *Unified Memory-centric Computing System (UMC2)*
    - *PIM, DPU, OS and compiler, memory/storage*
  - LABIO: a data-centric I/O system
    - LabStor and DTIO
- The next environment for AI: IOWarp
  - Bending the I/O Fabric for Advancing **AI-Infused** Scientific **Workflows**
    - StoreHub

New application
New technology
New environments

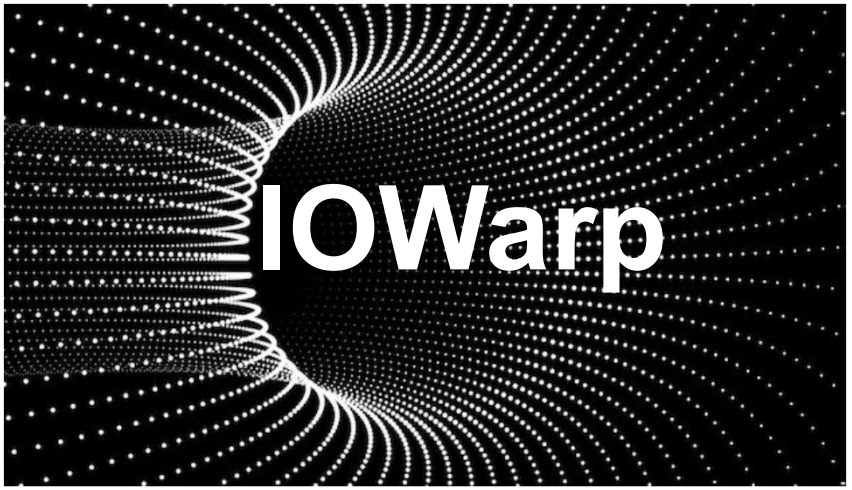
LABIOS: Data Operation with Label

- Data requests are transformed into (data) ***Label*** units
  - A label is a tuple of an operation and a pointer to the data
- A dispatcher distributes labels to the workers
- Workers execute labels independently (i.e., fully decoupled)



A. Kougkas, H. Devarajan, J. Lofstead, X.-H. Sun; "*LABIOS: A Distributed Label-Based I/O System", i*n Proceedings of ACM HPDC '19 (Best Paper Award)

Fundamental

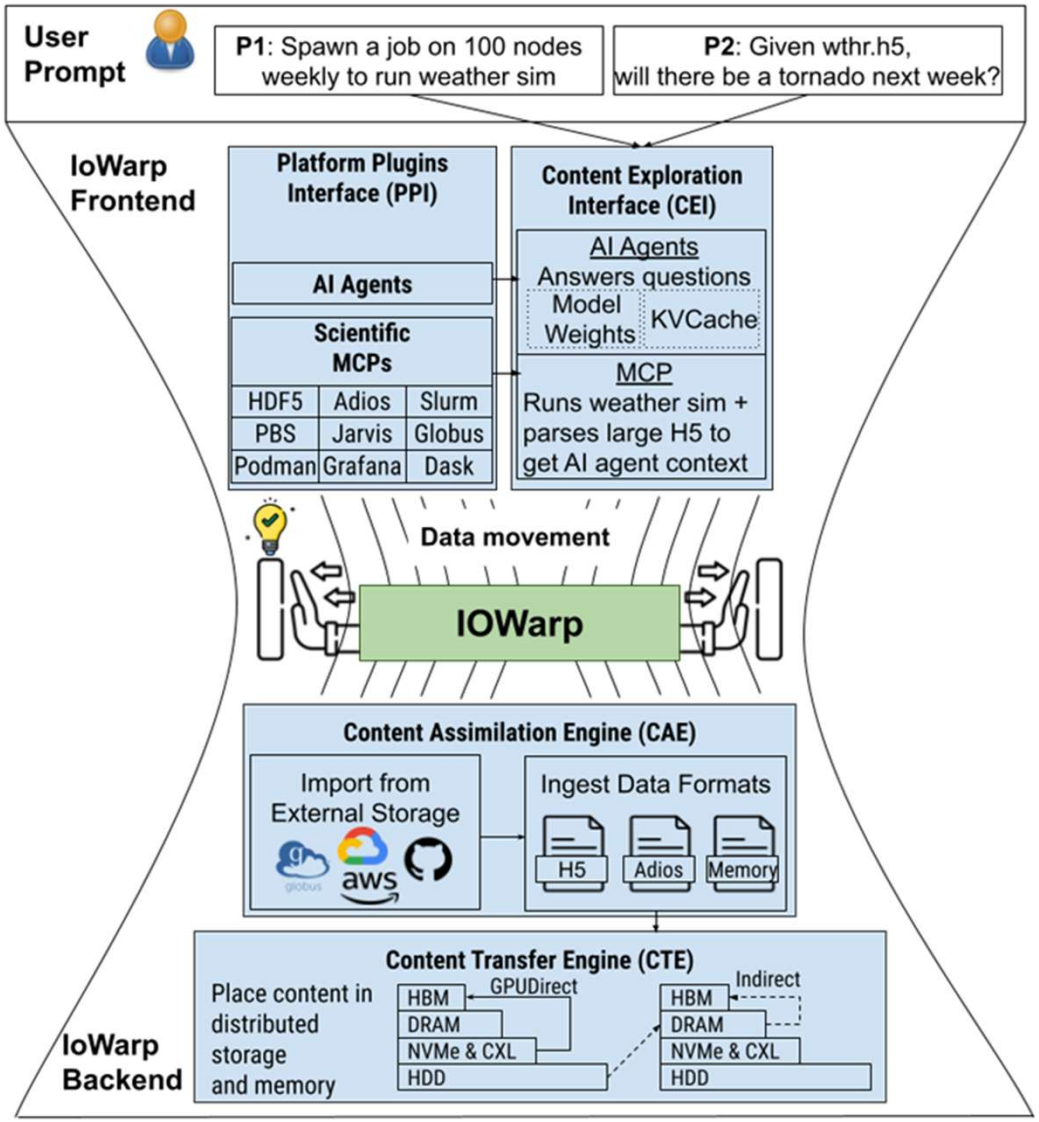


IOWarp: The Continuation

IOWarp is a **data management** platform designed to address the unique challenges in big data and AI applications

IOWarp: the Four Components

- Content Exploration
  - Comprehensive querying and indexing system and WarpGPT
- Platform Plugins
  - Connecting external tools and services
- Content Assimilation Engine
  - Convert diverse data formats into Content
- Content Transfer Engine
  - Manage efficient data transfer



How Can You Get Involved?

The IOWarp organization is the primary contact point for all IOWarp content https://github.com/iowarp

Inside, new users can:

- Get started quickly with our Installation Guides https://github.com/iowarp/iowarp-install
- Explore the design and more complex features in the Tutorial

Our code bases welcome collaborators:

- Our Runtime, modular execution engine for custom storage and data analytic pipelines.
- Jarvis, a software deployment environment that abstract hardware and shell complexities.
- A collection of Scientific MCPs and an Exploration Engine enabling a natural language interface to scientific data.
- The Assimilation Engine, a framework for transforming diverse format-specific data into an abstracted view called Content.
- The Transfer Engine, a multi-tiered, dynamic, and distributed I/O buffering system that manages all data movement and placement.

How Can You Get Involved?

StoreHub

A Community Testbed

Path Forward: A Community-Driven Initiative

- **Community-Centric Approach**: Identified needs through surveys, workshops (HDF User Group, PDSW, SSDBM), and direct conversations.
  https://forms.gle/c6fmQkiSrCYWQMij9
- **Build vital infrastructure** using community insights.
- **StoreHub** is the nation's premier *testbed for storage technologies*.
  https://grc.iit.edu/research/projects/storehub/

- **Pioneering Research**: Enabling breakthroughs in data management and storage research.
- **Broader Impact**: Enhancing education, workforce development, and fostering collaboration.

Take Home Messages

- **Modeling is the foundation** :
  - From memory-bound to C-AMAT and LPM
  - Understanding and Solution
- **Solution**: Dataflow under the von Neumann Machine
  - Reduce data access time under current systems
  - It is a new system paradigm based on performance modeling
- **Deliverable**: Scalable data management systems
  - The Hermes system for *Dataflow*$_V$
  - Performance engineering & performance-guided system design
- **Keywords**:
  - Data centric rethinking, Data system optimization, HPC4AI
- **Challenges**:
  - Understanding and Support of data concurrency and system integration (the ecosystem)
  - Hardware/software co-design
  - **Infrastructure** and **testbed**

Applications & Infrastructure

Conclusion (scalable data management)

- A new computer paradigm is introduced for AI-DL & big data
  - **Dataflow under the von Neumann Machine**
- Performance modeling and understanding is essential for the success of $Dataflow_V$
- Current Success: the Hermes System
  - Concurrent data access and system integration
- Ongoing Project: IOWrap
  - Geared up for AI applications
  - StoreHub: A community and community testbed

Key members of current research staff & students at Illinois Tech

Dr. Kun Feng
Research Software Engineer

Eneko Gonzalez
Research Software Engineer

Xiaoyang Lu
Research Professor

Anthony Kougkas,
Research Professor

Keith Bateman
PhD Candidate

Vadim Biryukov
PhD Student

Jaime Cernuda
PhD Candidate

Luke Logan
PhD Candidate

Isa Muradli
PhD Student

Shazzadul Islam
PhD Student

Neeraj Rajesh
PhD Candidate

Meng Tang
PhD Candidate

Hua Xu
PhD Student

Jie Ye
PhD Candidate

Izzet Yildirim
PhD Candidate

Thank you
Any questions?

AI & Data: Challenge & Opportunity in Computer System Research

Xian-He Sun

The Gnosis Research Center at the Illinois Institute of Technology

sun@iit.edu

www.cs.iit.edu/~scs

We would like to thank our sponsors the

National Science Foundation