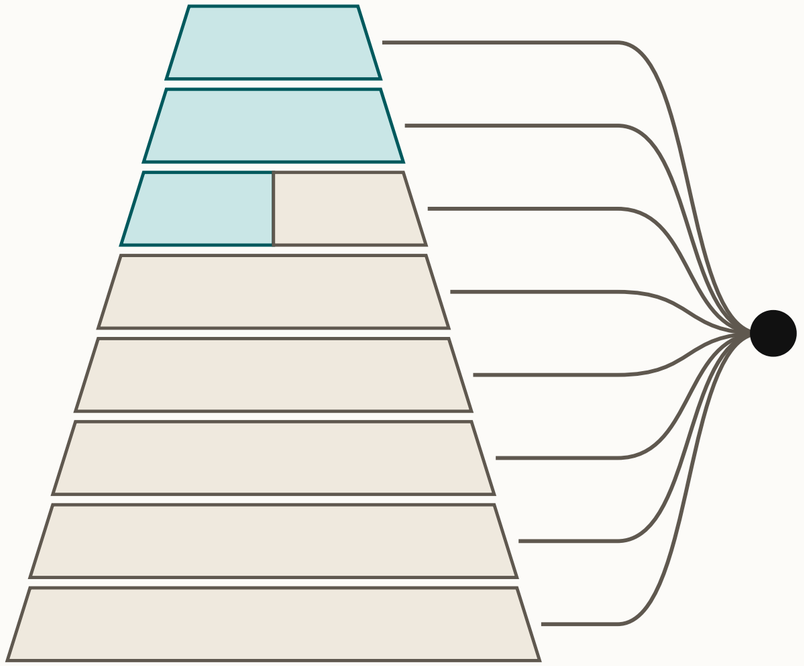
GRC GROUP MEETING, NOVEMBER 2026

One Destination, *N Tiers*



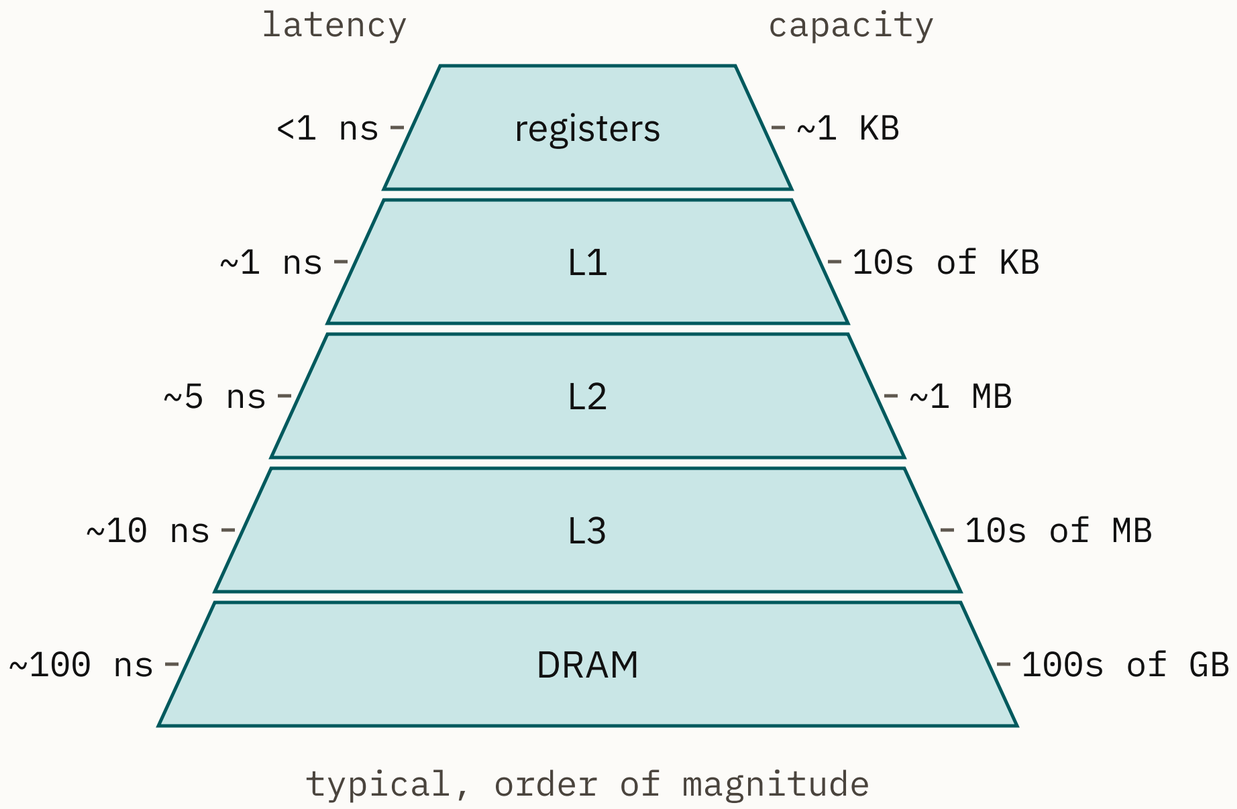
What happens after the data leaves memory

Anthony Kougkas

Associate Research Professor, Gnosis Research Center, Illinois Tech

The hierarchy you already trust

None of us has ever chosen which cache level a variable lives in. That invisibility is forty years of architecture work, some of it done in this lab.



One machine, one owner, managed by hardware and the OS.

Width is capacity and the fast tiers sit on top. Every pyramid in this deck uses the same grammar. **The I/O path below DRAM has no such invisibility.**

Below DRAM, a second pyramid

From IBM's first disk drive in 1956 until flash reached the data center, below main memory there was disk and tape. Then four families of hardware arrived.

Flash era

SSD, NVMe, burst buffers, node-local NVMe

enterprise SSD late 2000s; NVMe 1.0 2011; DataWarp 2015 to 2016; Summit NVMe 2018

Byte-addressable persistence

NVDIMM-N, PCM, memristor/ReRAM, STT-MRAM, 3D XPoint/Optane

memristor 2008; 3D XPoint 2015; Optane SSD 2017; Optane PMem 2019

Disaggregation and fabrics

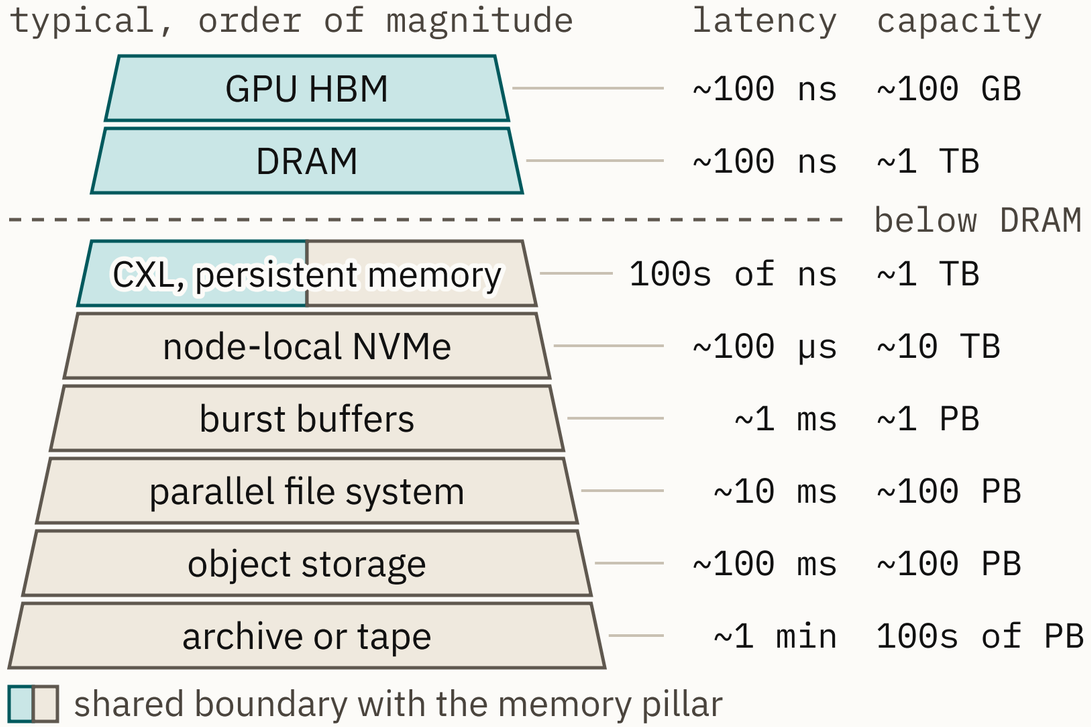
CXL, RDMA, NVMe-oF, DAOS, object stores

CXL 1.0 2019; Azure preview 2025; Meta in production 2026

Accelerator-attached

HBM, GPUDirect Storage, computational storage, DPUs

GPUDirect Storage announced 2019, generally available 2021



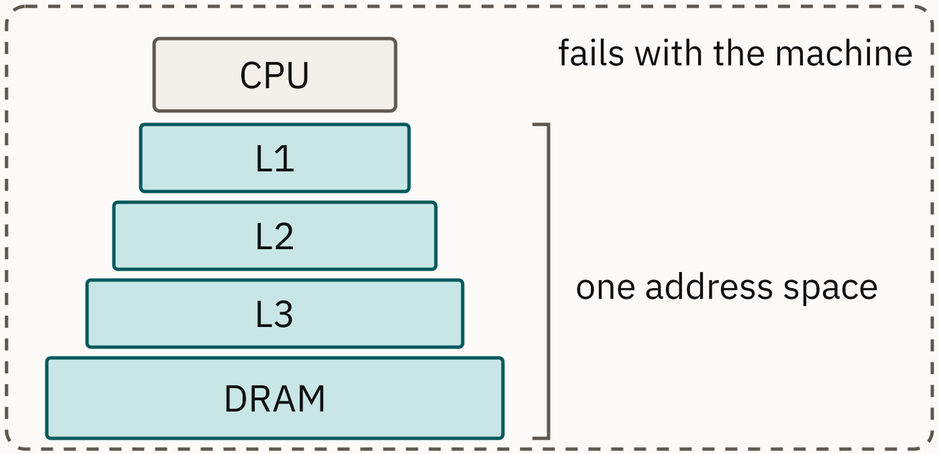
Devices arrive on vendor roadmaps. Software written against a device dies with it. Software written against the hierarchy survives.

Software support arrived one device at a time. **Intel wound Optane down in 2022; software written against the hierarchy kept running.**

An I/O tier is not just another cache level

Inside a memory system one architect designs every level. Beyond it there is no single architect.

CACHE LEVEL — one machine, one owner

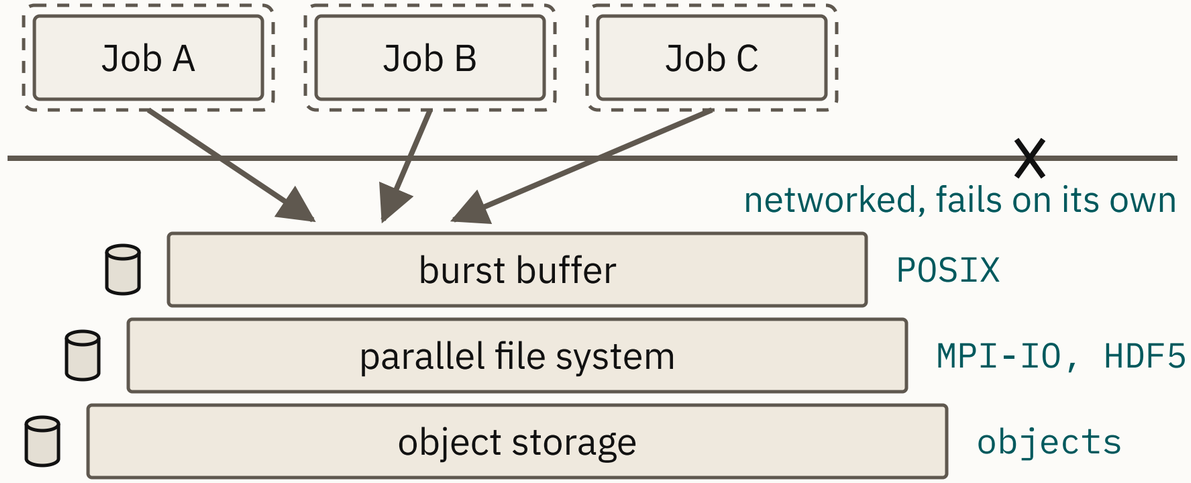


volatile

managed by hardware and the OS

nanoseconds

I/O TIER — many jobs, many owners, many domains



persistent, with durability semantics

managed one tier at a time, never as one hierarchy

microseconds to seconds

Shared principle, from Dr. Sun's C-AMAT and LPM: match the rate of requests to the rate of supply at every layer.

Same principle, different problem, different machinery.

Single tiers did have managers: DataWarp for the burst buffer, Data Elevator between burst buffer and file system, HSM for tape. **Nothing managed the hierarchy as one.** The I/O line starts here: burst buffers on Cori and Trinity, and jobs interfering on a shared tier.

THE THESIS

Every tier supplies the application at once. *The application sees one destination.*

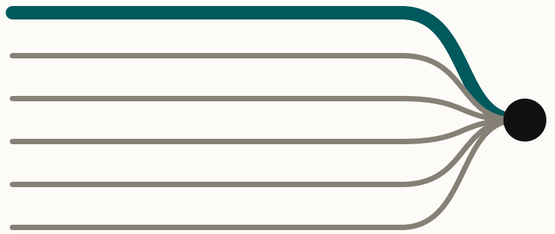
Six ideas. Each has a name, a system, and a person in this room or in a chair they used to sit in.



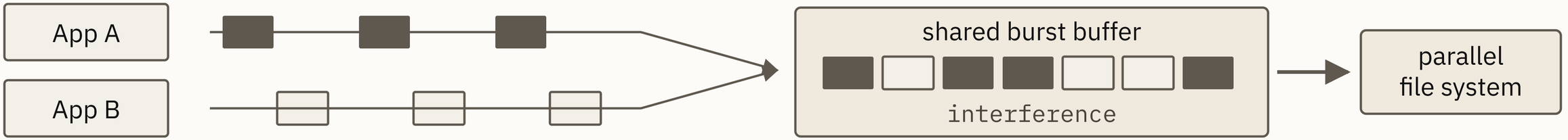
Not five mount points, and not a burst-buffer directive in a job script. They converge on context: what the client needs for its next decision.

2015 TO 2016: CRAY DATAWARP ON CORI AND TRINITY

A new tier arrived. *Who manages it?*



Each paper tamed one tier or one interface. The machines already had five tiers.



BBIO eScience'16

Coordinating burst buffers to prevent cross-application I/O interference.

Harmonia Cluster'18

An interference-aware I/O scheduler for shared burst buffers.

IRIS ICS'18

Files and objects behind one interface. The hierarchy also spans data models. Kun Feng carried it into NIOBE (BigData'19).

The problem appeared before any system did.

BBIO up to 2x (eScience'16). **Harmonia** up to 3x over state-of-the-art buffering management (Cluster'18). **IRIS** bridges file and object storage (ICS'18).

Anthony; **Hari Devarajan** (Harmonia, IRIS); **Kun Feng** (NIOBE); with Argonne (Matthieu Dorier, Rob Latham, Rob Ross) and Jay Lofstead (Sandia); advised by Dr. Sun

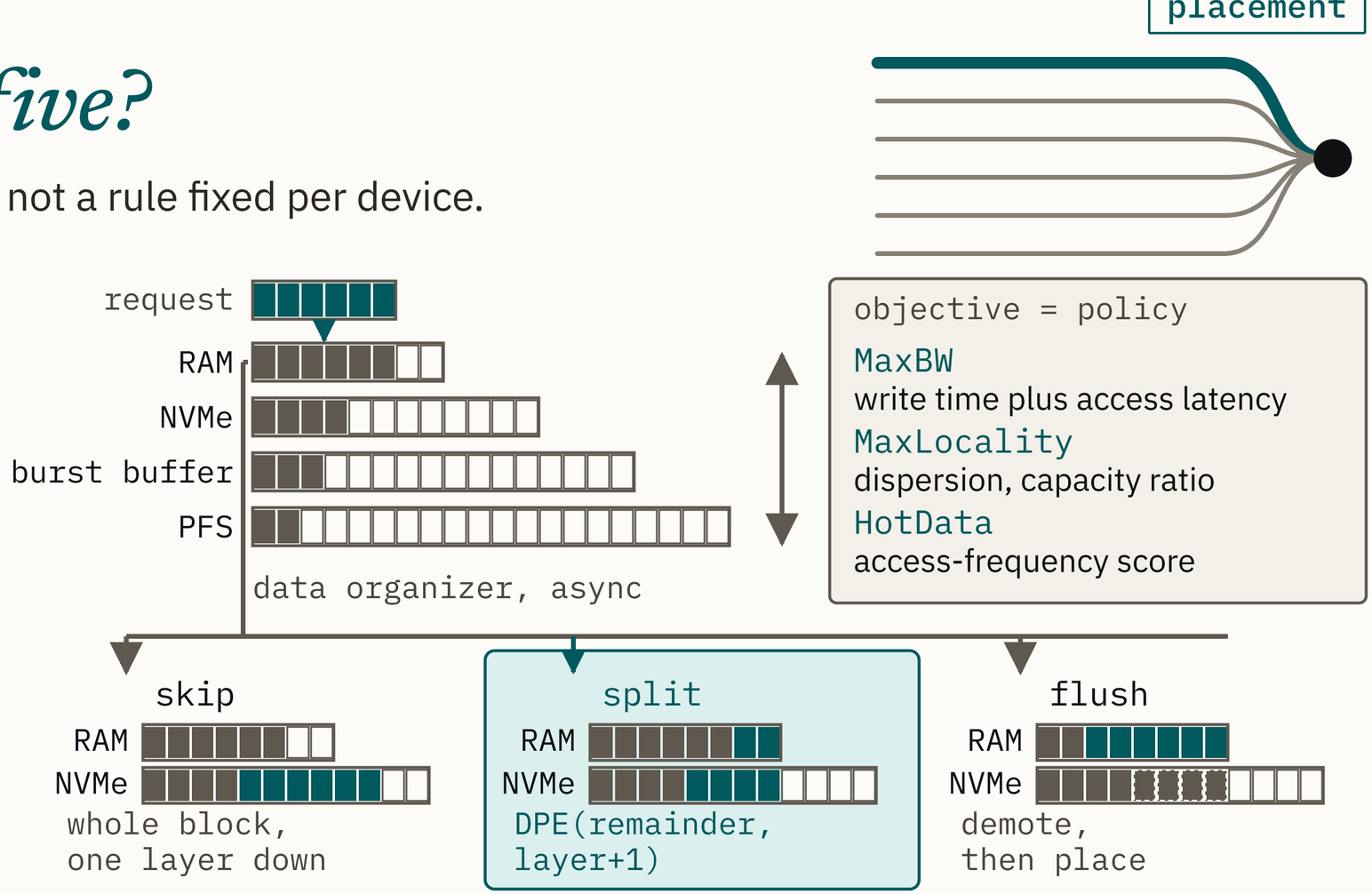
What if there are *five?*

Placement is solved per request. It is not a rule fixed per device.

IDEA DPE(request, layer): if it fits, place. Otherwise the policy picks skip, split or flush and recurses on what remains.

MECHANISM Dynamic programming with memoization. A data organizer promotes and demotes asynchronously, down the tiers and across nodes.

PRICE Single-writer, multiple-reader consistency. Durability follows the buffering mode: persistent (sync or async), non-persistent scratch, or bypass.

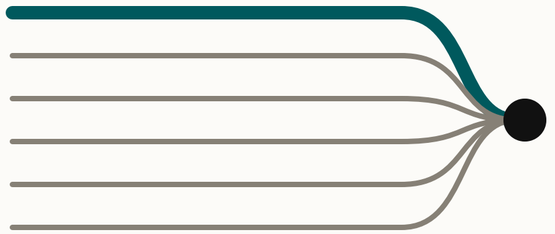


Hermes: 8x over no buffering; more than 2x over Data Elevator and DataWarp (HPDC'18). POSIX, MPI-IO, HDF5 adapters with no application changes. Released under The HDF Group (final v1.2.1, 2024); ran on Theta and Summit.

Anthony and **Hari**; runtime by **Luke Logan**; with The HDF Group (Gerd Heber); advised by Dr. Sun

What happens to every other optimization *once tiers exist?*

placement, as a platform



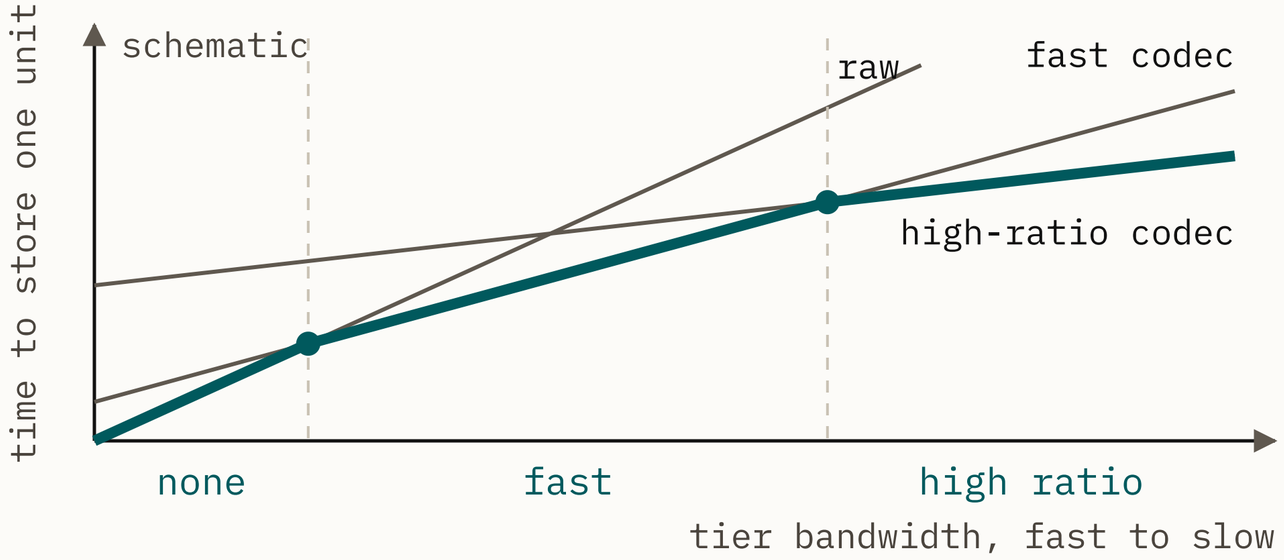
Once tiers exist, every classic optimization gains a new decision variable: which tier.

HFetch, IPDPS'20 | HCompress, IPDPS'20 | HReplica, BigData'20 | Eviction, SC'19 poster | HCL, Cluster'20 | HFlow, Cluster'21

PREFETCHING HFetch moves from client-pull to server-push. It sees every application's accesses, so hot data rises once, not once per client.

COMPRESSION HCompress picks the codec per tier: compress when compress time plus compressed size over tier bandwidth beats writing raw. Fast tiers want fast codecs; slow tiers want ratio.

REPLICATION HReplica chooses scheme, codec and tier jointly.

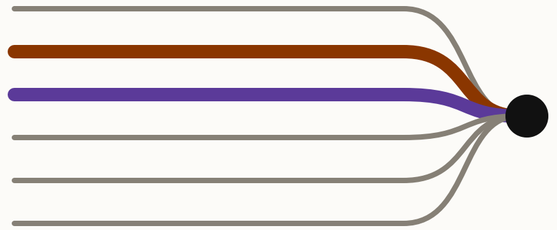


HFetch 10 to 35 percent over existing prefetchers, over 50 percent over none (IPDPS'20). **HCompress** up to 7x (IPDPS'20). **HReplica** 5.2x (BigData'20). Every one of these algorithms inferred what the application wanted from its I/O calls.

Hari (HFetch, HCompress, HReplica, HCL); **Jaime** (eviction, HFlow); **Luke** (runtime)

How does placement learn *what the application wants?*

intent, then task

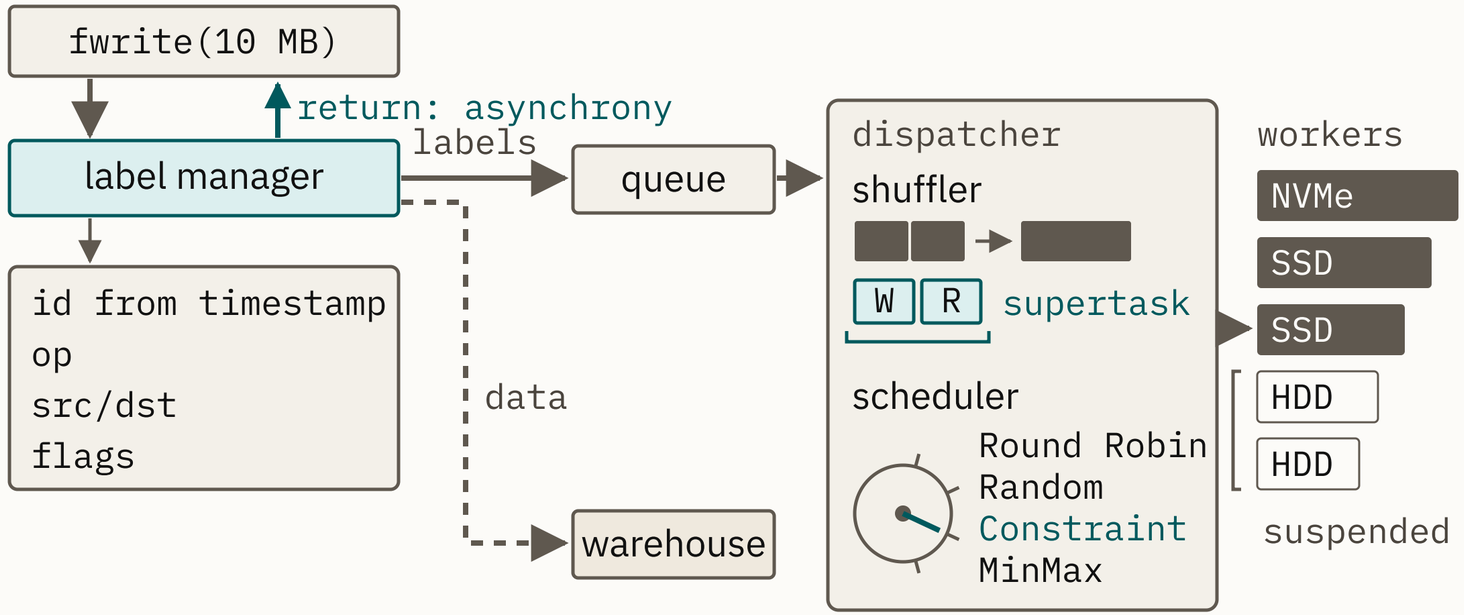


POSIX throws away what the application wants; the label puts it back. Then storage becomes a scheduling problem.

IDEA A label is (operation, data pointer, flags). Requests split 1-to-N or merge N-to-1. Labels go to a queue, data to a warehouse, metadata to an inventory.

MECHANISM The shuffler merges adjacent labels and binds read-after-write into an ordered supertask. The scheduler runs Round Robin, Random, Constraint (a weighted worker score) or MinMax (a multidimensional knapsack, NP-hard, approximate DP). Workers can be suspended or woken.

PRICE Waking a suspended worker takes about 3 s; MinMax is the slowest policy to dispatch.

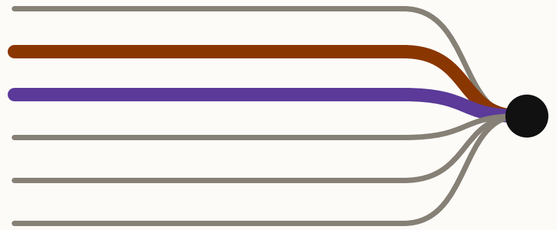


LABIOS up to 17x through asynchrony (HPDC'19); Round Robin and Random dispatch 55K to 125K labels/s. Karsten Schwan Best Paper; US Patent 11,630,834.

Anthony and **Hari**; with Jay Lofstead (Sandia) and Kathryn Mohror (LLNL); advised by Dr. Sun. DTIO by **Keith Bateman**, with Bogdan Nicolae and Franck Cappello (Argonne)

LABIOS, HPDC'19, KARSTEN SCHWAN BEST PAPER AWARD; US PATENT 11,630,834 (2023)

intent, then task



From the label *to the DataTask*

I used to call labels an optimization. I now see them as intent preserved across a system boundary.

DTIO, SSDBM'25

A DataTask adds formats, ordering and dependencies, so translation runs online and producer and consumer overlap. Now in CLIO Core.

→

LABIOS 2.x

The label as an IR that POSIX, HDF5 and MPI-IO adapters compile into. A direction.



HPDC 2019: Jay Lofstead, Anthony Kougkas, Xian-He Sun

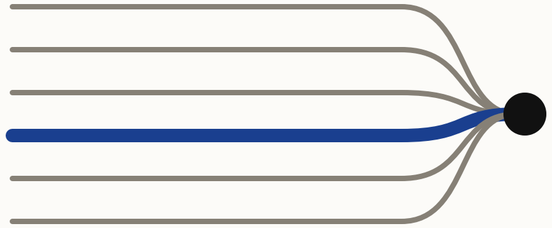
DTIO 49.6 percent versus offline translation on PtychoNN (SSDBM'25).

Anthony and **Hari**; with Jay Lofstead (Sandia) and Kathryn Mohror (LLNL); advised by Dr. Sun. DTIO by **Keith Bateman**, with Bogdan Nicolae and Franck Cappello (Argonne)

CHRONOLOG, MSST'20

order

What if the data carries *its own placement key?*

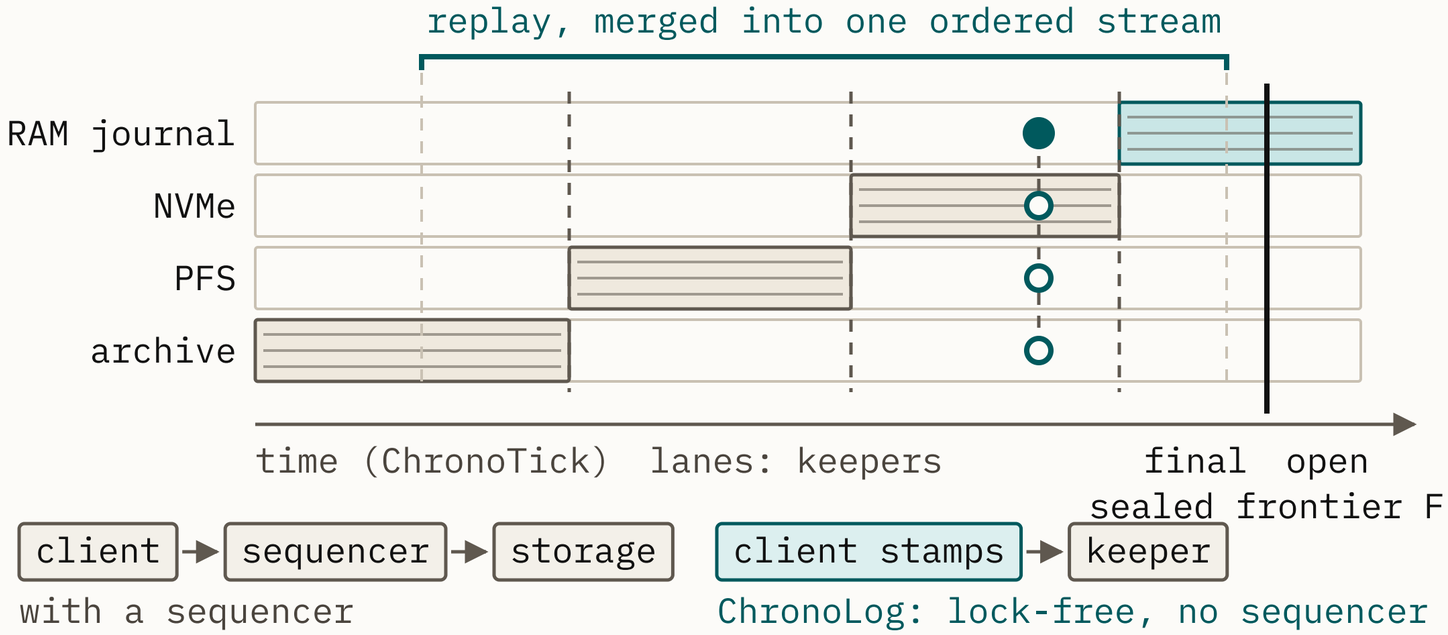


Physical time is the log sequence number. A writer knows where its entry goes without asking anyone.

Design (MSST'20) A ChronoTick is an offset from the chronicle's base time, synced with ChronoVisor. Lock-free, no sequencer on append.

Shipping (v3.x, 4.0 in progress) Demotion down the tiers; replay merged into one stream. 4.0: a hybrid logical clock orders under skew; a read completes once every keeper's sealed frontier passes its end; eviction never passes the persisted watermark (E <= W).

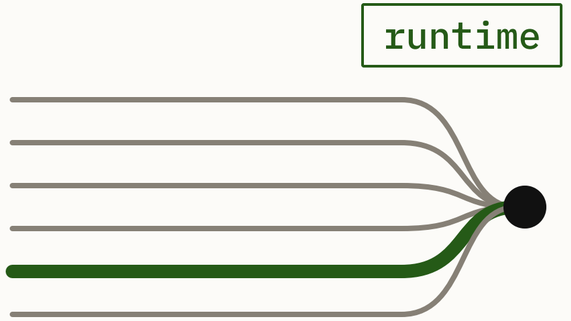
PRICE Bounded clock uncertainty; colliding ticks need a declared collision semantic. Promotion and prefetch are open.



MSST'20 prototype: over 1M tail operations per second; 16 GB/s record and 18 GB/s playback on 8 NVMe nodes. v3.0 (May 2026) added SQL, pub/sub and an MCP server.

Anthony and **Hari** with **Keith**, **Jaime** and **Neeraj** (MSST'20, first services); **Kun Feng** and **Eneko Gonzalez** (most of today's code); with Inna Brodkin and Kyle Chard (UChicago); with Dr. Sun

Why does every system *rebuild the same runtime?*



The kernel I/O stack is fixed. Workloads and devices are not. Make the stack a composition.

Hermes runtime
2020

→

LabStor
SC'22: modular user-space I/O stacks

→

Chimaera
tasks, queues, shared memory

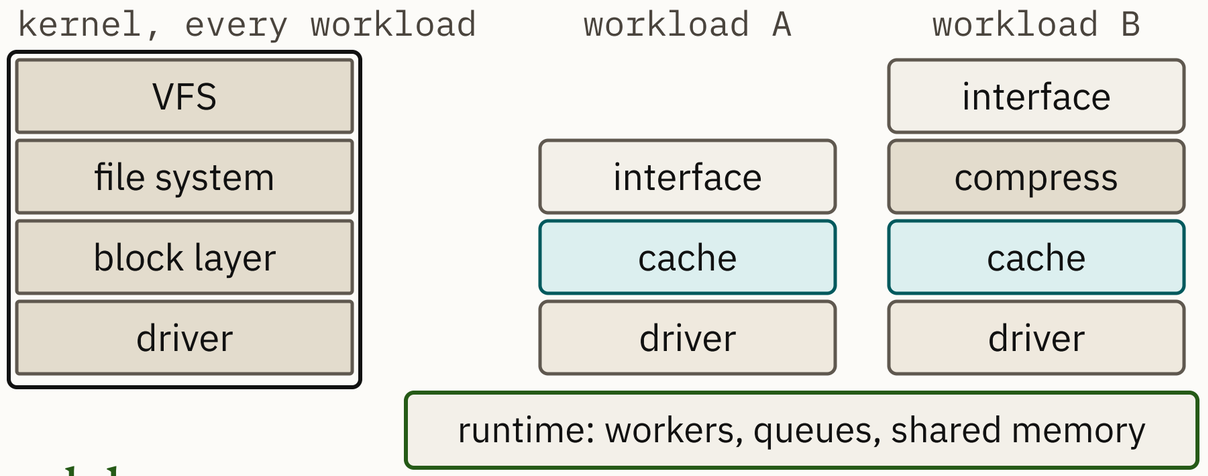
→

CLIO Runtime
2026, with Context Transport Primitives

`LABSTOR` Single-purpose I/O modules in user space, composed per workload and device.

`CHIMAERA` Every module is a pool of tasks on shared-memory queues; one module calls another by submitting a task. CLIO Core's interposition chain follows from this.

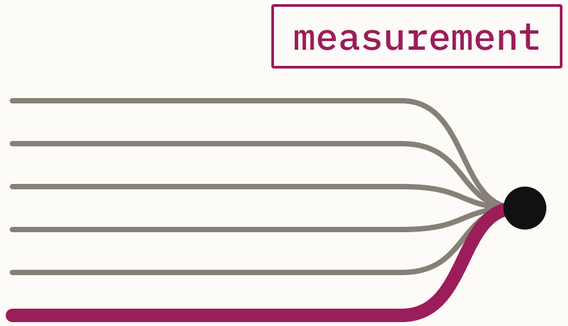
`MEGAMMAP` The same tiers behind a memory interface: distributed shared memory spills DRAM to NVMe and PFS, with hints for coherence.



Build the runtime once. Every system after it is a module.

LabStor up to 60 percent from tailored I/O stacks (SC'22). **MegaMmap** runs with up to 2.6x less DRAM (SC'24).

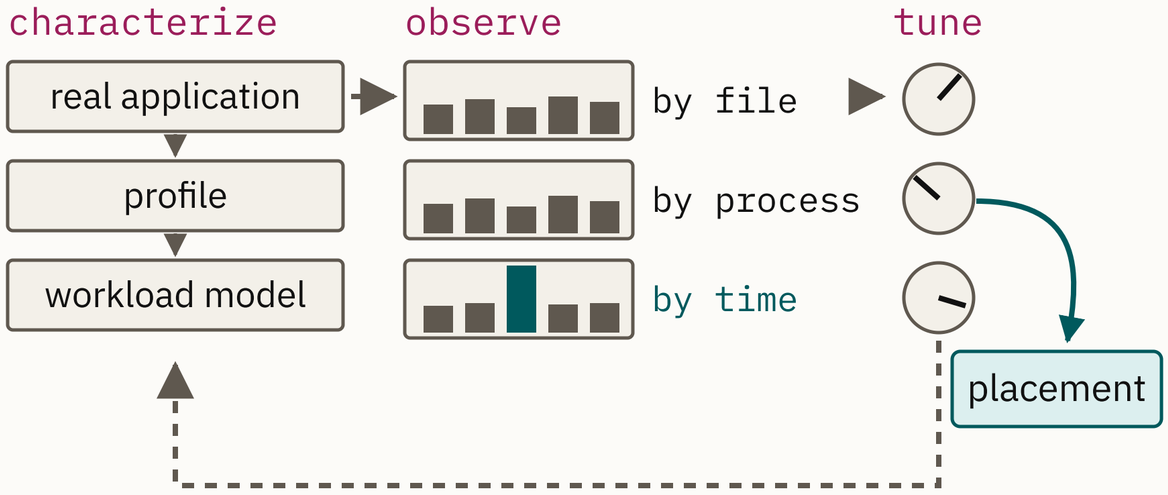
How do you place *what you cannot see?*



A benchmark is a model of a workload. DLIO emulates the I/O behavior of scientific deep learning, built from profiles on Theta. WisIO reads one trace from many views (file, process, time) and classifies bottlenecks from metrics, not fixed rules.

DLIO, built by Hari with Huihuo Zheng and Venkat Vishwanath (Argonne), Anthony and Dr. Sun, **became the workload engine of MLCommons MLPerf Storage.**

characterize	Vidya, HiPC'18, Hari	DLIO, CCGrid'21, Hari
	DaYu, Cluster'24, Meng	
observe	Apollo, HPDC'21, Neeraj	
	DFTracer, SC'24, Hari, LLNL	
	WisIO, ICS'25, Izzet, LLNL	
tune	LuxIO, HiPC'22, Keith	
	TunIO, IPDPS'24, Neeraj	



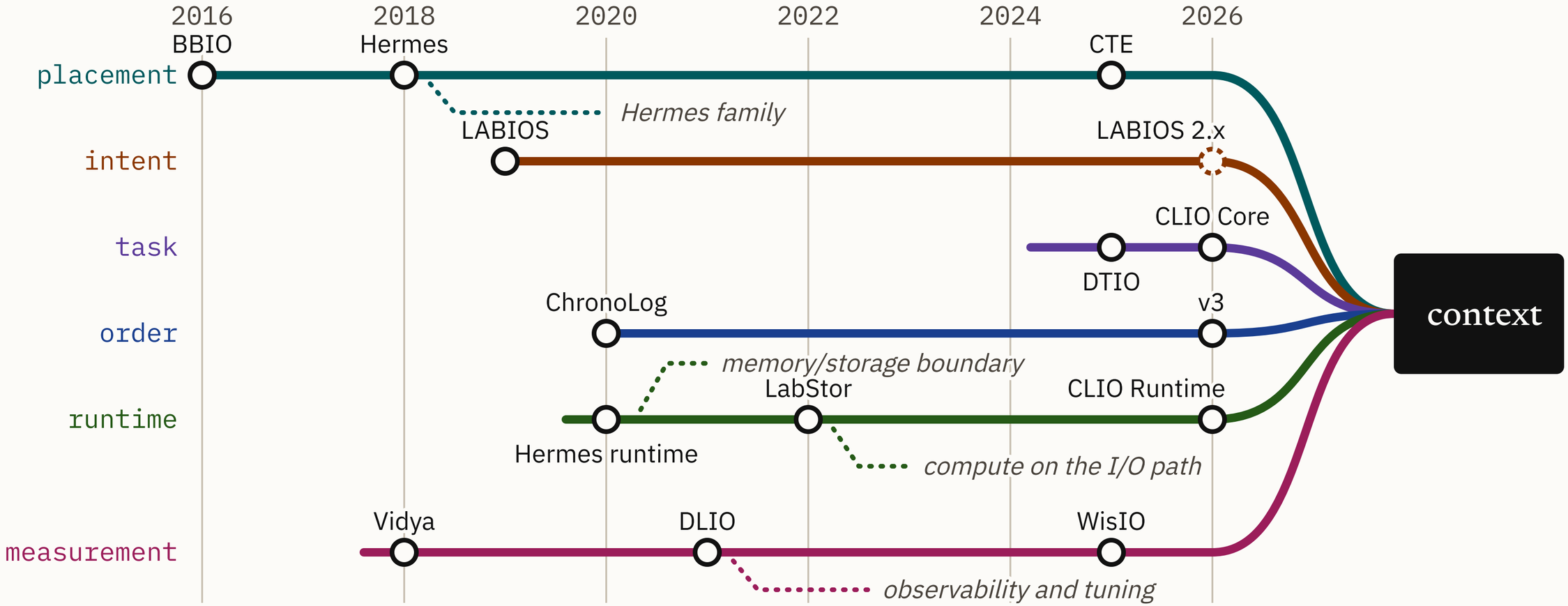
DLIO-guided optimizations cut training time by up to 6.7x (CCGrid'21 Best Paper). **WisIO** up to 805x bottleneck coverage versus a single view, and 11x faster analysis (ICS'25).

Hari, **Neeraj**, **Keith**, **Meng**, **Izzet**; with Argonne and LLNL; with Dr. Sun

The map

Six questions, six answers. Every I/O thesis in this room extends one of these lines.

Dots: systems. Dashed rings: directions. Dotted spurs: sister clusters.



Buffer every tier, no code changes **(Hermes)**

Shared AI storage workload **(DLIO, MLPerf Storage)**

Ordered, tiered log **(ChronoLog v3)**

I/O as schedulable units **(labels, DataTasks)**

Composable user-space I/O stacks **(LabStor to CLIO Runtime)**

Bottlenecks in terabyte traces **(DFTracer, WisIO)**

Installable stack **(CLIO Core and Kit on PyPI)**

HPDC'19 and CCGrid'21 best papers; US Patent 11,630,834 on the label representation.

Five dissertations, *one line of ideas*

Each dissertation handed its ideas to the next, and all five continue in IOWarp.

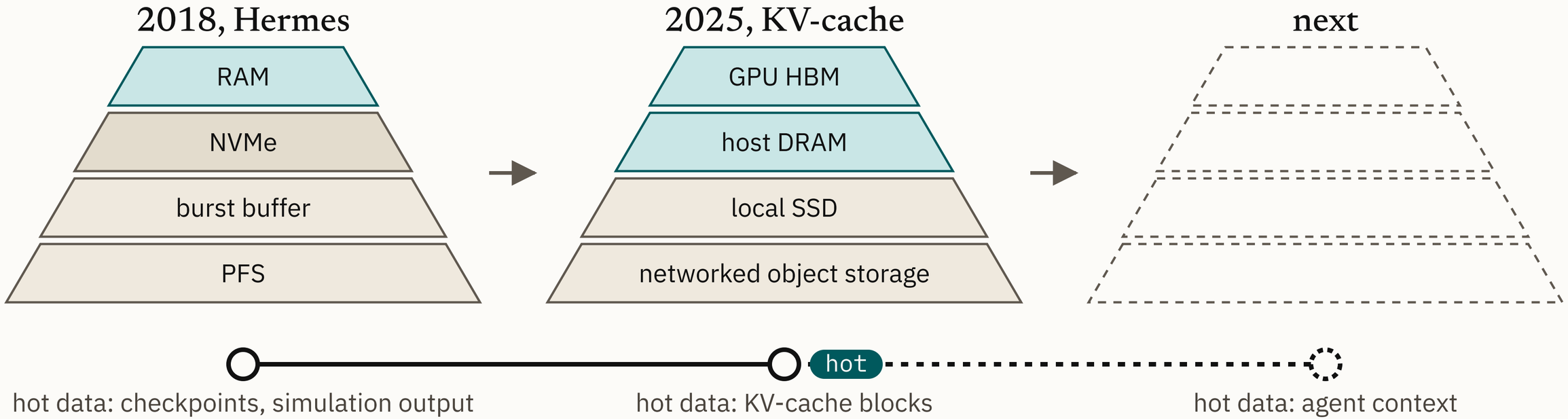
DISSERTATION	WHAT IT CONTRIBUTED	WHERE IT LIVES TODAY
Anthony Kougkas, 2019 *Accelerating I/O Using Data Labels*	Contention, multi-tier placement, labels: Harmonia, Hermes, LABIOS	CTE placement; intent and tasks in CLIO Core; the IOWarp program
Kun Feng, 2020 *A Multi-Level Data Integration Approach*	HPC and Big Data convergence, files and objects behind one interface: IRIS, NIOBE	File and object bridging in CAE ingest (POSIX, S3, Globus); ChronoLog, whose current code Kun wrote most of
Hariharan Devarajan, 2021 *Towards a Self-programmable Storage Solution*	Tier-aware algorithms and measurement: HFetch, HCompress, HReplica, HCL, DLIO	The CTE compressor, replication and cache modules; MLPerf Storage
Luke Logan, 2025 *RECOMBINANT*	pMEMCPY, LabStor, MegaMmap, the Hermes runtime	CLIO Runtime, Context Transport Primitives, CTE
Jaime Cernuda, 2026 *FlowForge*	HFlow, Hades, HStream, Jarvis, ChronoLog	Agent and Relay pipelines, Jarvis, derived-variable pipelines in CTE

Kun heads infrastructure at GRC. Hari is a computer scientist at LLNL. Luke and Jaime became Assistant Research Professors at GRC in January 2026.

Advised by Xian-He Sun (2019, 2020, 2021). Co-advised by Anthony Kougkas and Xian-He Sun (2025, 2026).

The same pyramid, *different hot data*

The hot data changed. The problem did not. In 2018 it was a checkpoint; today it is a KV-cache; next it is the context an agent needs for one decision.



Industry arrived at the same pyramid independently. ***That is the best evidence the problem is real.***

KV-cache tiering: **NVIDIA Dynamo** (GTC, March 2025), **LMCache**, **Mooncake** (FAST'25). CXL memory expansion: CXL 1.0 2019; Azure preview 2025; Meta in production 2026.

The client *changed*

Assembling the slice of context one decision needs, on time, from every tier, is an I/O problem.

`Program`

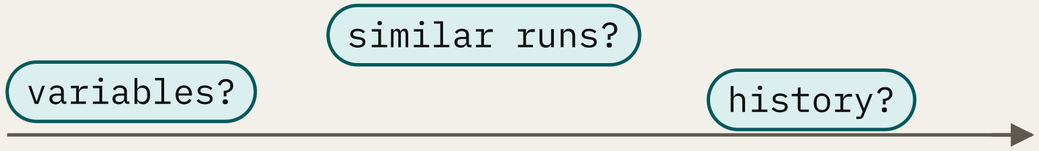
Its unit of work is the **byte**. Its access is periodic and knowable.

`offset, size`



`Agent`

Its unit of work is the **decision**. Its requests are semantic and produced by reasoning.



> **Context:** the slice of data, metadata, provenance, system state, tool capability and history that one decision needs.

`placement`	CTE
`intent, task`	CLIO Core
`order`	ChronoLog history
`runtime`	CLIO Runtime
`measurement`	characterizing agents as an I/O workload

IOWarp (2024 to 2029): Illinois Tech, The HDF Group, University of Utah. Each primitive from Part II has a home in the CLIO stack.

CLIO Core: *Hermes, grown up*

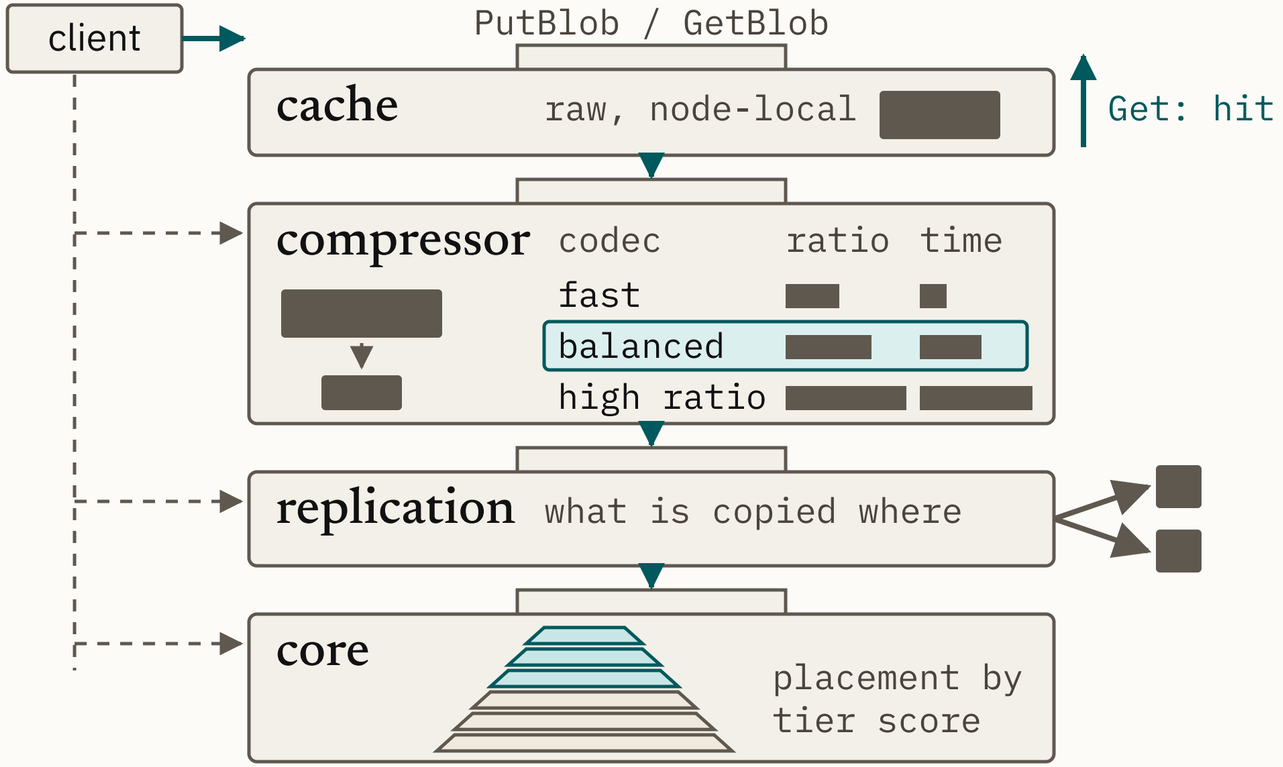
Hermes's policies became interposers on one task interface.

IDEA The chain is cache, then compressor, then replication, then core. Each module handles the core's own PutBlob and GetBlob and forwards the rest, so a client pointed at any link works unchanged.

CACHE A node-local raw copy for the shared-memory read path; writes back down the chain.

COMPRESSOR Per blob and candidate codec, learned models (Q-table and linear regression; dense NN optional) predict ratio, compress and decompress time, and PSNR if lossy. A time or ratio objective picks against the target tier's bandwidth.

REPLICATION Decides what is copied where; a returning owner pulls what its successors changed. Surviving a tier failure is new in CLIO 2.4.

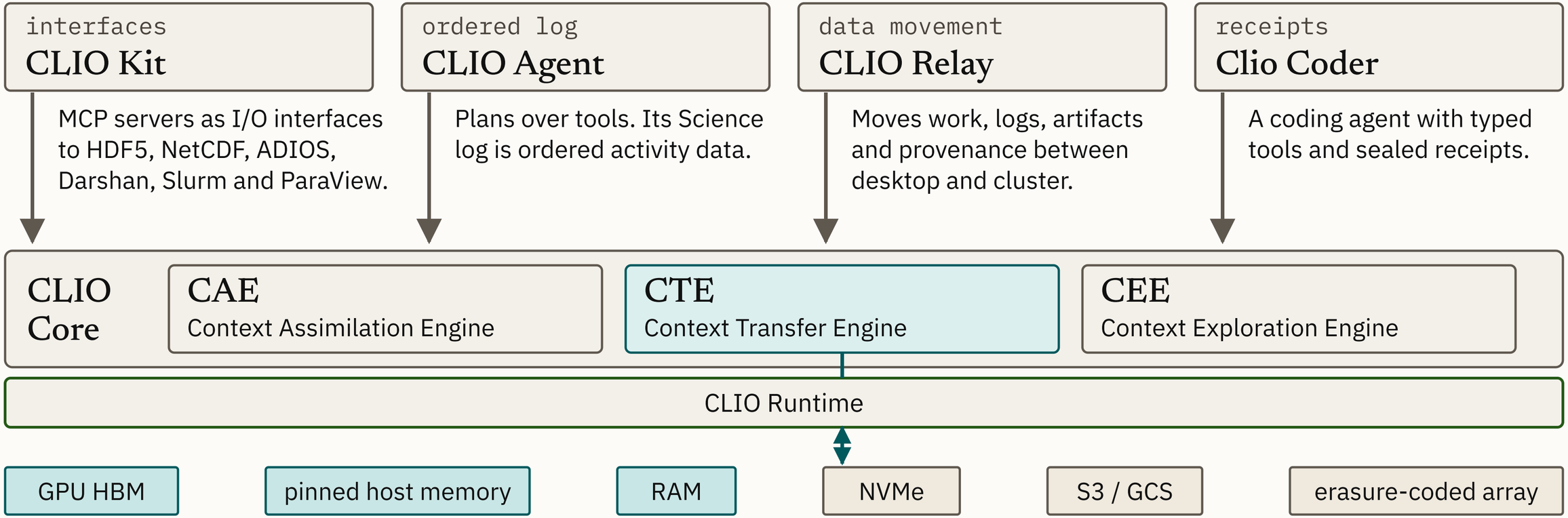


Releases 2.3 and 2.4 (Oct to Nov 2026): a full tier evicts; placement follows the tier score; a tier that goes offline loses no data, through failover reads and replication; the runtime runs on Aurora's Intel GPUs.

Luke leads Core engineering, with **Jaime Cernuda**, **Atharva Naik**, and **H. Joe Lee** and **Matthew Larson** (The HDF Group)

The CLIO stack, *seen as I/O*

Read every piece as an I/O system: interfaces, ordered logs, data movement, and receipts.



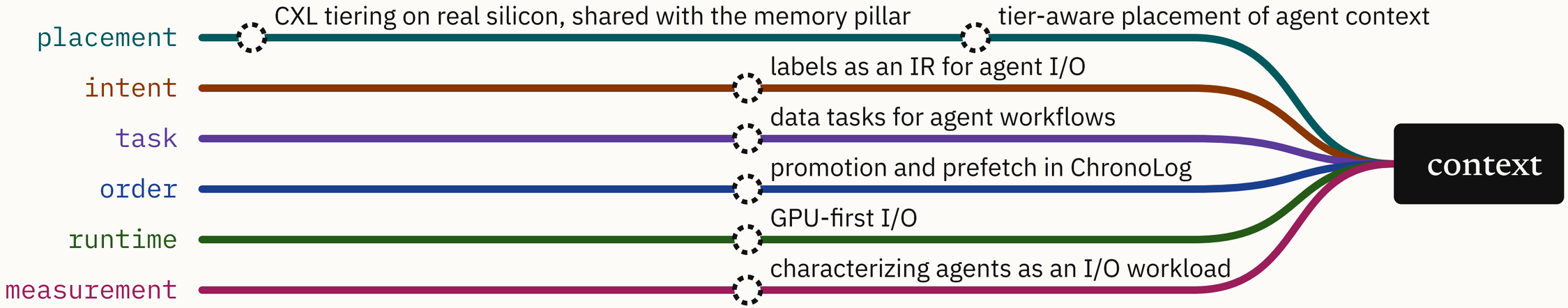
DOE Genesis Mission projects **SPOTTER-AI** (Argonne-led) and **PORTUS** (PNNL-led) build on CLIO. The full stack launches at SC26, November 15 to 20.

PyPI downloads through Nov 9, 2026: **Core 65,651**, **Kit 41,233**, **Agent 17,194**. Downloads, not users.

Jaime leads Agent and Relay; **Isa Muradli**, **Shazzadul Islam**, **Soham Sonar** and **Aum Sathwara** build Kit; **Anthony** builds Coder

Where your thesis sits

The open problems on each line carry no names. **The problems are yours to define.**



● KV-cache as a tier (placement line)

Jie Ye: Viper ICPP'24, UnboxKV IPDPS'25, PKAS HPDC'26

Zarif Chowdhury: pre-RoPE vs post-RoPE KV survivability

● GPU as a storage tier (runtime line)

Izzet Yildirim: GDSight, PDSW'26

Isa Muradli: GPUDirect, NIXL

Luke Logan, mentoring high-school researcher J. Fernandes: GPUH5, PDSW'26

Rajni Pawar: GnnPress, GNNs beyond HBM

● Agents as an I/O workload (measurement line)

Izzet: AgentStage eScience'26, StageOnce, ToolFuse, *Correct Is Not I/O Efficient*

Rajni: Acropolis eScience'26

Shazzadul: agentic search

Keith: Agent Error Simulator eScience'26

● Workflows and in situ (task line)

Meng Tang: dataflow for I/O-aware scheduling, IPDPS'26

Hua Xu: derived-quantity operators eScience'26; Trigger, Render, Reason at SC26

Neeraj Rajesh: from I/O observability to explainability

What this *taught me*

None of them is about I/O.

1 **The system existed before the project did.**

Hermes was a paper and a prototype before it was a funded project.

2 **One line of work compounds.**

One runtime, Hermes to LabStor to Chimaera, now under everything we ship.

3 **Software makes an idea negotiate.**

With compilers, failures, other people's workloads, and the cost of being understandable.

4 **The abstraction outlives the code.**

Hermes's final release was 2024. Its placement model runs inside CLIO today.

None of these systems started because someone asked for them. Each started because someone in this lab saw a thing that should exist and began building it well.

Pursuing excellence does not need justification.

GRC I/O AT A GLANCE

Every tier supplies the application at once. *The application sees one destination.*

placement
Hermes, then CTE

intent
LABIOS

task
DTIO

order
ChronoLog

runtime
Hermes runtime, then LabStor, then Chimaera, then CLIO Runtime

measurement
DLIO, WisIO

memory systems architecture | **parallel and distributed I/O** | resource management

agentic science and AI

Every system on these slides carries the names of the people who built it. The memory pillar matches demand to supply inside the machine, through architecture. The I/O pillar matches it across shared, persistent, distributed tiers, through systems software. IOWarp grows from the I/O pillar.

Acknowledgments: NSF, DOE, Argonne, LLNL; partners The HDF Group, Sandia, UChicago, University of Utah, PNNL.



One destination, N tiers.

APPENDIX: FOR LIFTING INTO OTHER DECKS

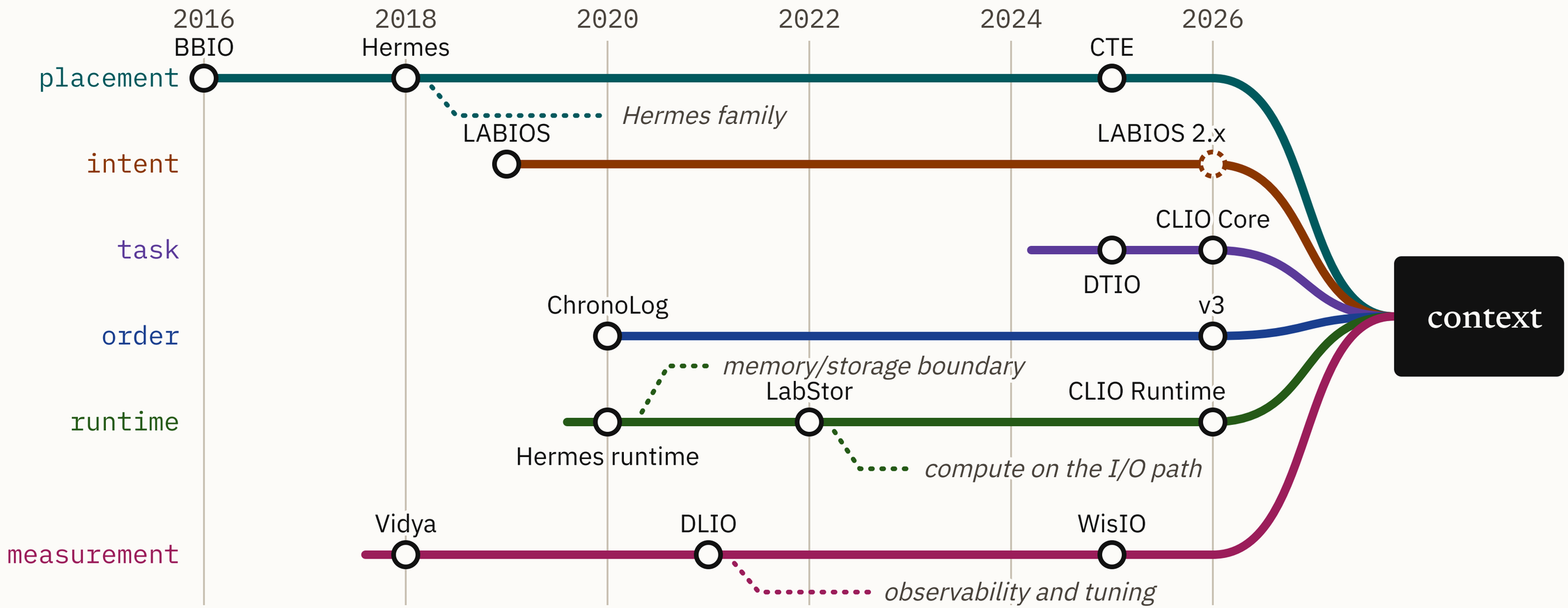
Beyond the memory bus: *the GRC I/O line*

1	GRC's work on the deep memory and storage hierarchy began in 2016 with burst-buffer interference on leadership machines, building on the lab's earlier parallel file system research, and grew into a family of systems: Hermes, LABIOS, ChronoLog, DTIO, and now IOWarp.
2	Our memory and I/O work share one principle, matching demand to supply at every layer, and pursue it in two places: inside the memory system through architecture, and across shared, persistent, distributed tiers through systems software.
3	Hermes (HPDC'18; Kougkas, Devarajan, Sun) is the ancestor of the transfer engine in IOWarp.
4	DLIO (CCGrid'21 Best Paper; Devarajan, Zheng, Kougkas, Sun, Vishwanath) became the workload engine of MLPerf Storage.

Same principle, different problem, different machinery.

The GRC I/O lineage, 2016 to 2026

One line per primitive; time runs left to right. Dots are systems, dashed rings are directions, dotted spurs are sister clusters. Every line ends at context.



Buffer every tier, no code changes **(Hermes)**

Shared AI storage workload **(DLIO, MLPerf Storage)**

Ordered, tiered log **(ChronoLog v3)**

I/O as schedulable units **(labels, DataTasks)**

Composable user-space I/O stacks **(LabStor to CLIO Runtime)**

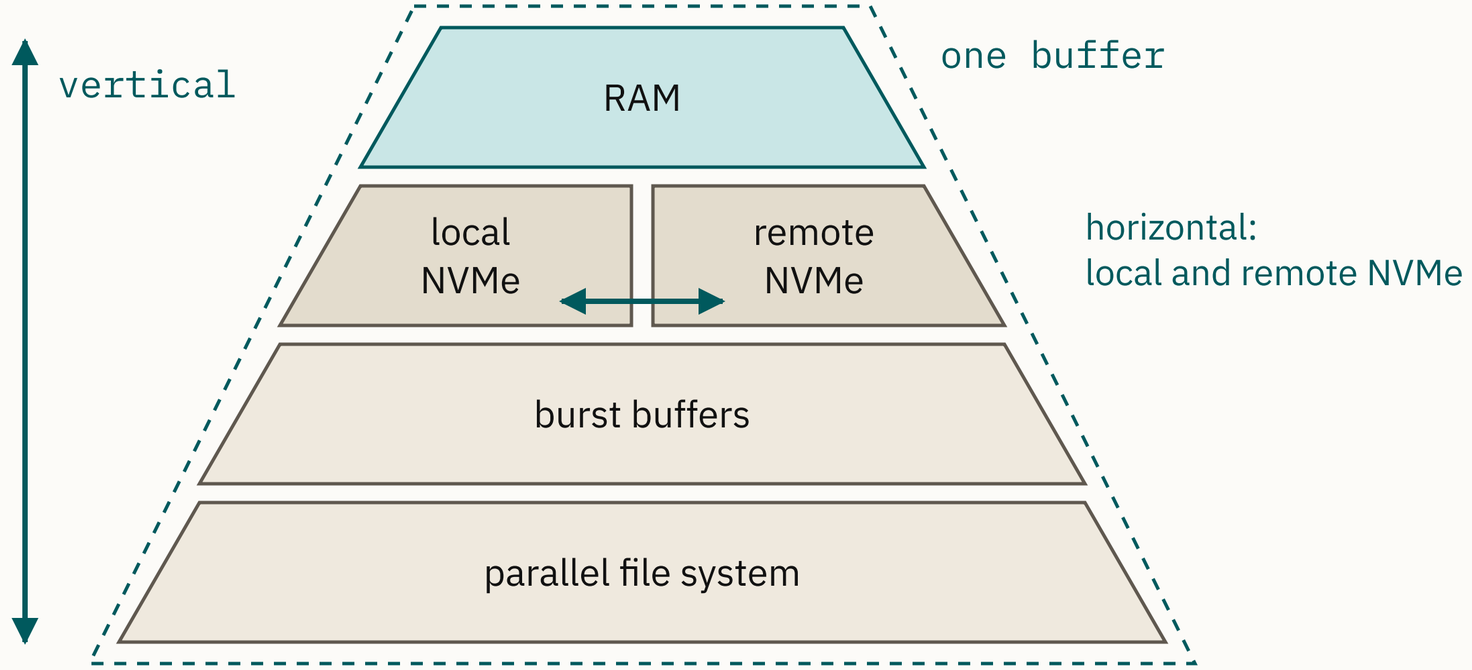
Bottlenecks in terabyte traces **(DFTracer, WisIO)**

Installable stack **(CLIO Core and Kit on PyPI)**

A2

Hermes on one slide

A multi-tiered, heterogeneous-aware buffering system (HPDC'18) that treats RAM, local and remote NVMe, burst buffers and the parallel file system as one buffer.



Placement: for every request, place, split, skip or flush at each layer. POSIX, MPI-IO and HDF5 adapters.

Platform: HFetch, HCompress, HReplica, multi-tier eviction, HCL, HFlow.

Release: versioned middleware under The HDF Group, final v1.2.1 in 2024; ran on Theta and Summit.

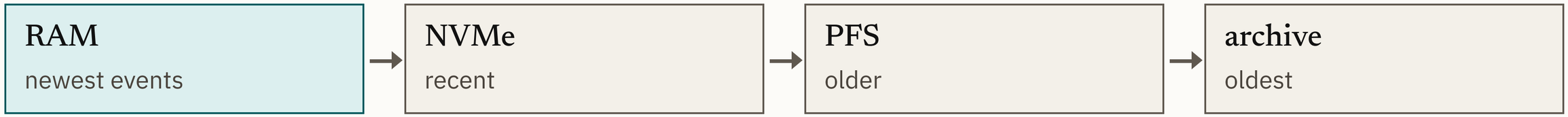
Today: its placement model continues in the CLIO Context Transfer Engine.

8x over no buffering and more than 2x over Data Elevator and Cray DataWarp (HPDC'18).

Anthony Kougkas and **Hariharan Devarajan**; runtime by **Luke Logan**; platform work by Hari and **Jaime Cernuda**; with The HDF Group (Gerd Heber); advised by Dr. Sun

ChronoLog on one slide

A distributed, shared, tiered log store that orders events by physical time, so a log's age order matches the hierarchy's speed order.



`Design (MSST'20)`

Physical time instead of a sequencer on the append path. Nodes for throughput, tiers for capacity, time for range queries.

`Shipping today (v3.x, 4.0 in progress)`

Demotion down the tier chain; replay merged across tiers into one ordered stream; SQL, pub/sub and an MCP server since v3.0. Promotion and prefetch are open.

MSST'20 prototype: over 1M tail operations per second; 16 GB/s record and 18 GB/s playback on 8 NVMe nodes.

Anthony and **Hari** with **Keith**, **Jaime** and **Neeraj** (MSST'20); **Kun Feng** and **Eneko Gonzalez** (today's code); with Inna Brodkin and Kyle Chard (UChicago); with Dr. Sun

IOWarp and CLIO on one slide

The I/O line applied to agents: assemble the context one decision needs, on time, from every tier.

CLIO Core

Context Assimilation, Transfer (CTE) and Exploration engines on the CLIO Runtime. CTE tiers span GPU HBM to erasure-coded arrays; modules for cache, compression and replication.

CLIO Kit

MCP servers for HDF5, NetCDF, ADIOS, Darshan, Slurm, ParaView.

Agent and Relay

Planning over tools; moving work, logs and provenance between desktop and cluster.

Clio Coder

A coding agent with typed tools and sealed receipts.

IOWarp (2024 to 2029): Illinois Tech, The HDF Group, University of Utah. DOE Genesis Mission projects SPOTTER-AI (Argonne-led) and PORTUS (PNNL-led) build on CLIO.

Core: **Luke Logan**; Agent and Relay: **Jaime Cernuda**; Kit: **Isa Muradli**, **Shazzadul Islam**, **Soham Sonar**, **Aum Sathwara**; Coder: **Anthony Kougkas**

APPENDIX: CONTRIBUTIONS

What the community *can do now*

1	Buffer across every tier without changing application code: Hermes, released under The HDF Group through v1.2.1; the placement model continues in CLIO's CTE.
2	Measure storage for AI training with a shared workload: DLIO, the workload engine of MLPerf Storage.
3	Keep a totally ordered, tiered log: ChronoLog v3, open source, with SQL, pub/sub and an MCP server.
4	Express I/O as schedulable units: the label (LABIOS, published; US Patent 11,630,834) and the DataTask (DTIO, prototype tagged ssdbm25, mechanisms in CLIO Core).
5	Compose user-space I/O stacks: LabStor, Chimaera, CLIO Runtime.
6	Find bottlenecks in multi-terabyte traces: DFTracer and WisIO, with LLNL.
7	Install the current stack: CLIO Core and Kit on PyPI (counts are downloads).

HPDC'19 Best Paper (LABIOS); CCGrid'21 Best Paper (DLIO).