STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

Fabric Attached Memory

Hardware and Software Architecture

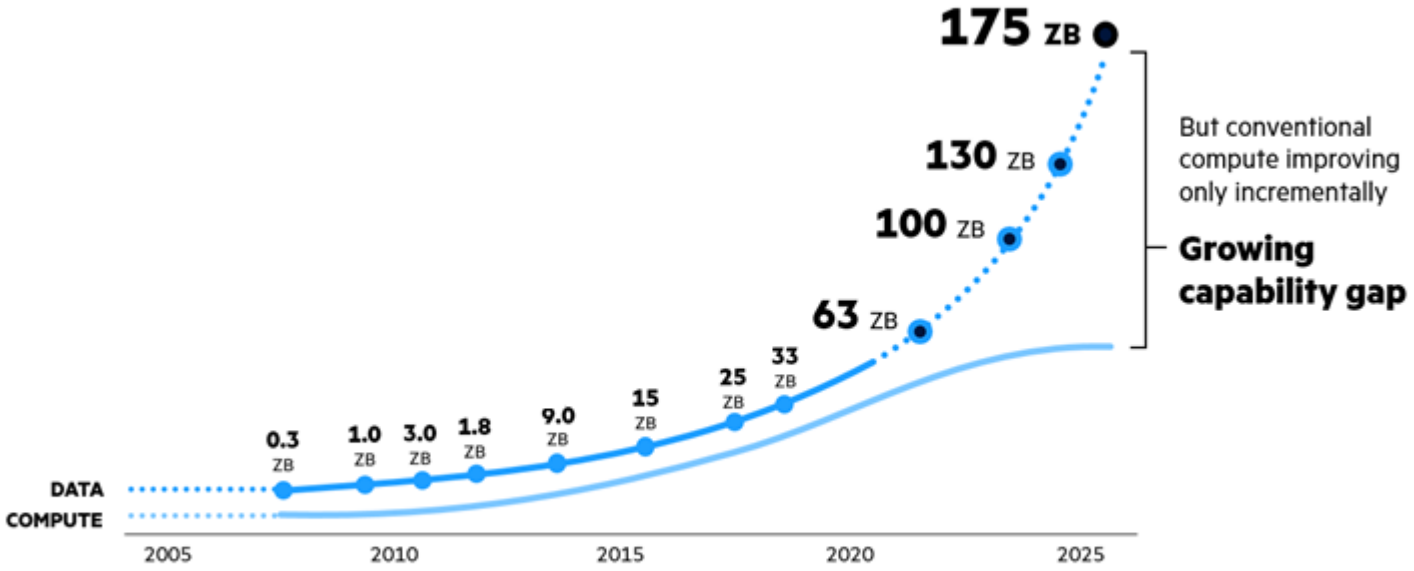
Clarete Crasta, Dave Emberson, Sharad Singhal

Agenda

- Motivation
- Using fabric attached memory in HPC
  - Architecture
  - Software stack
- Results and use cases
  - Microbenchmarks
  - Arkouda-based graph processing
- Summary & future work

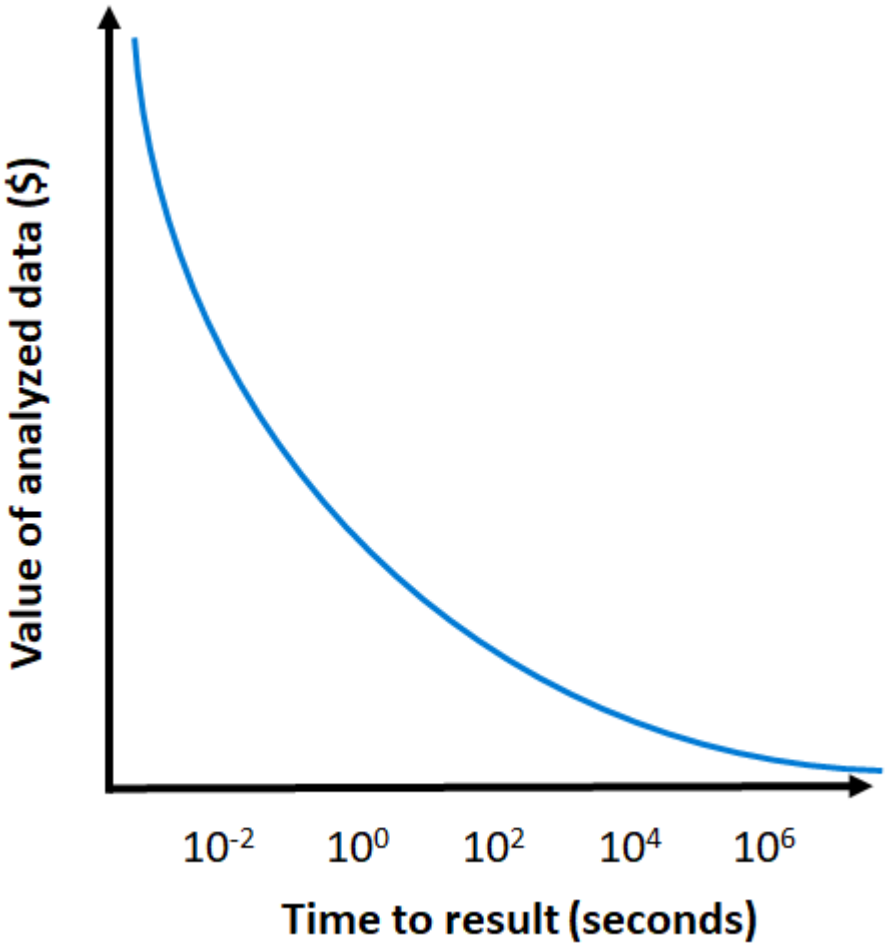
Need quick answers on larger data sizes

EXPONENTIALLY INCREASING DATA **X** EXPLODING DATA SOURCES **X** SHRINKING TIME TO ACTION **=** Massive advances in computing power **NEEDED EVERYWHERE**



Data nearly doubles every two years (2013-25)

Source: IDC Data Age 2025 study, sponsored by Seagate, Nov 2018



Characteristics of emerging applications

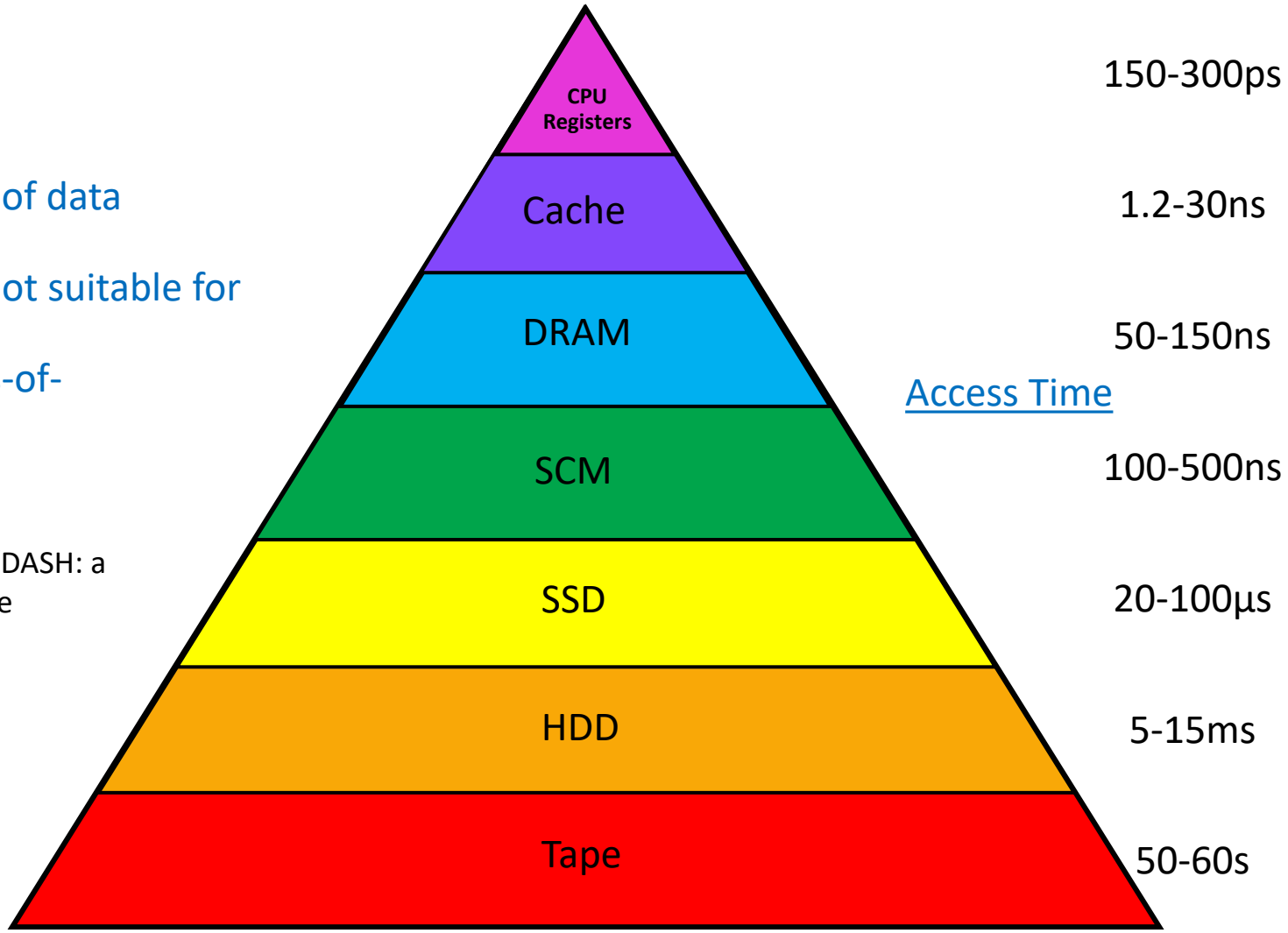
- AI and machine learning
  - Applications in simulation, modeling, large language models
- HPC workflows/pipelines such as those in genomics
  - Applications to transform data in a workflow with large intermediate data sets
- Large scale graphs
  - Applications in
    - *Security*: website reputation, malware detection
    - *Social networks*: Community detection, link prediction
    - *Advertising*: brand reputation, click-through prediction
    - *Internet of things*: traffic management, risk detection

- Applications have enormous memory footprints
  - Datasets can be 10s-100s of terabytes to multi-petabytes in size
  - Analytics performance is currently limited by the total amount of DRAM in the HPC cluster
- Random data access patterns
  - Processor caches are inefficient due to low hit rates
  - Distributed applications often require expert programmers
- Data movement introduces high latencies
  - Demand paging to SSDs is very slow
  - Moving data consumes time and energy
  - Difficult to optimize system resource locality or network performance

Merging memory and storage: the new hierarchy

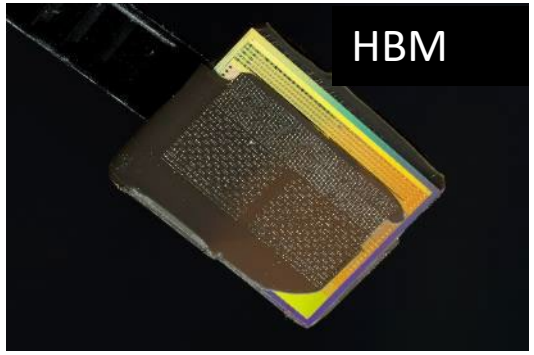
"We are entering the HPC era of data intensive applications.
Existing supercomputers are not suitable for th[ese] kind[s] of workloads. There is a 5-orders-of-magnitude gap in the current storage hierarchy."

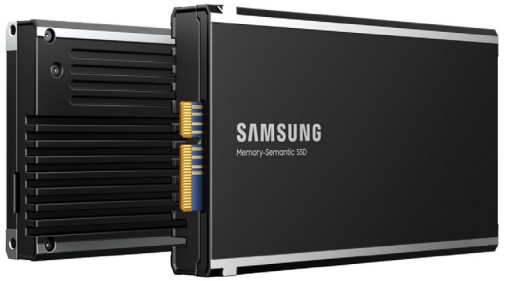
Jiahua He et al, Proceedings SC2010, "DASH: a Recipe for a Flash-based Data Intensive Supercomputer"



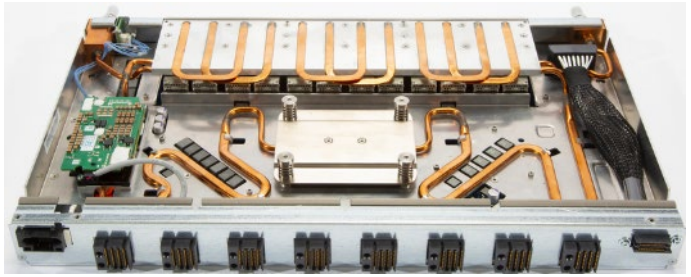
Technology advances

Advances to Memory Technologies



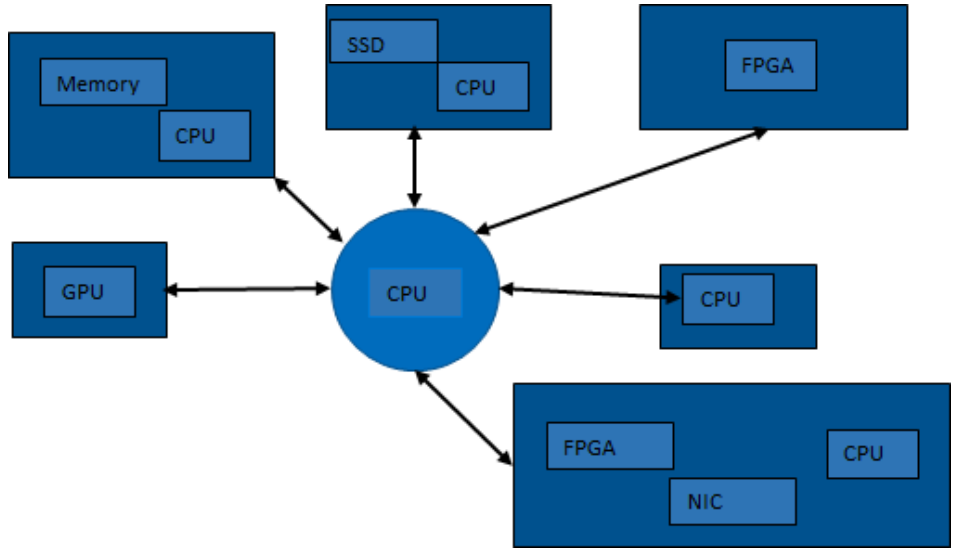


Advances to interconnects – Slingshot, CXL



CXL Compute Express Link

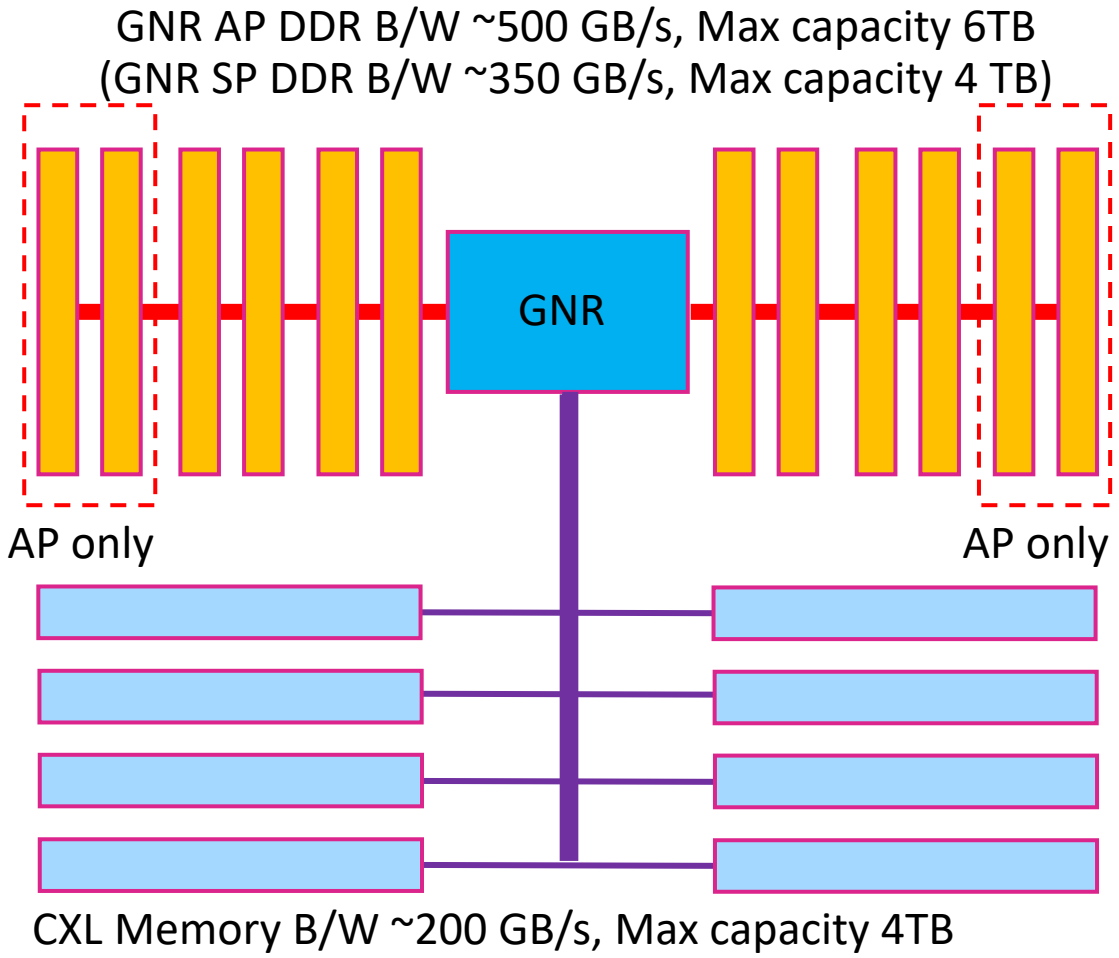
Heterogeneous Compute



SDC 23

Emerging system architectures

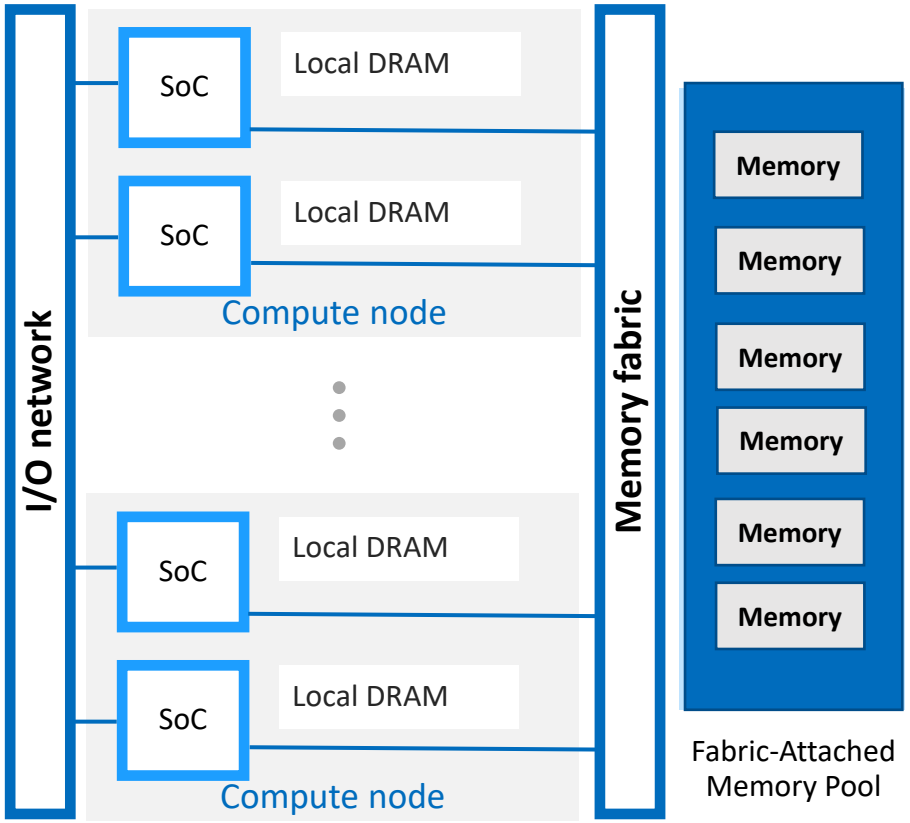
- E1.S modules
  - x8 CXL @ 32 GT/s = 32 GB/s max
  - Capacity up to 256GB per module
- E3.S modules
  - x8 CXL at 32 GB/s
  - x16 CXL at 64 GB/s max (~50 GB/s in practice)
  - Capacity up to 512GB per module
- Running 64 CXL lanes at 32 Gb/s = 256 GB/s max (~200 GB/s in practice)
  - CXL has higher latency than on-chip DDR controllers
  - Switches to extend memory capacity beyond the limits of directly connected modules, but incurs additional latency



→ GNR AP B/W increase 40%, capacity 67%
→ GNR SP B/W increase 57%, capacity 100%

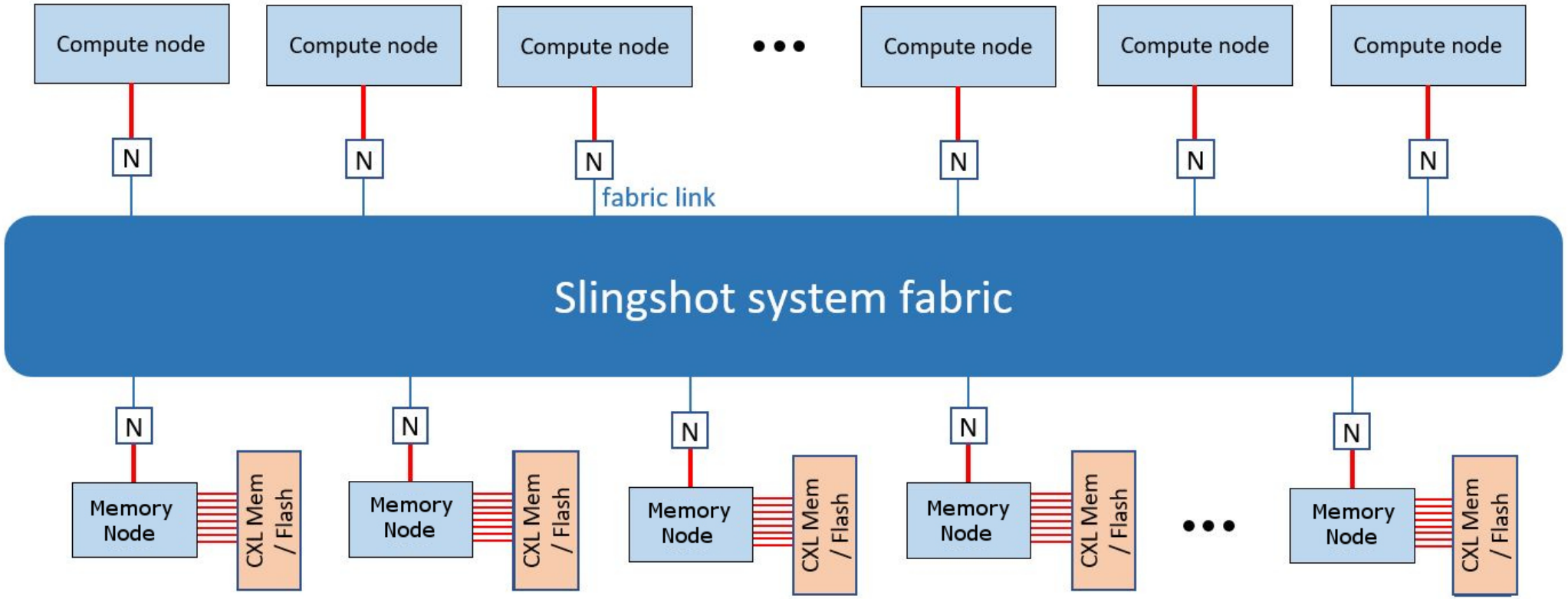
Fabric attached memory

- Benefits
  - Independent scaling of compute and memory
    - Improved utilization, reduced overprovisioning
    - Decoupling of failure domains
  - Reduced depth of I/O stack for large workloads
    - Direct, unmediated access
    - Accessible by all compute resources
- Challenges
  - Coherence domains are limited
    - Individual nodes on RDMA-based interconnects
    - Small clusters on CXL-based interconnects
  - Fabric latency is large compared to local memory
    - Software has to be (usually) refactored for performance

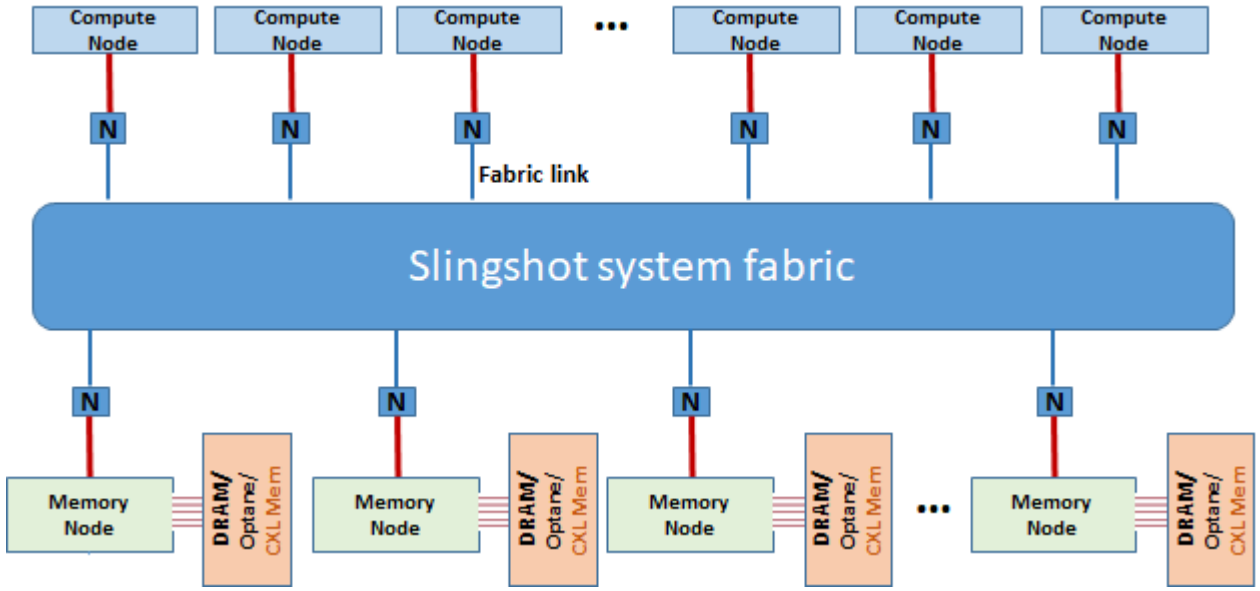


GoldenTicket data-driven HPC architecture

FAM on Slingshot is implemented with ***Memory Nodes***



Prototype implementation



Future: Memory servers with pool of CXL memory interconnected over Slingshot

Front view

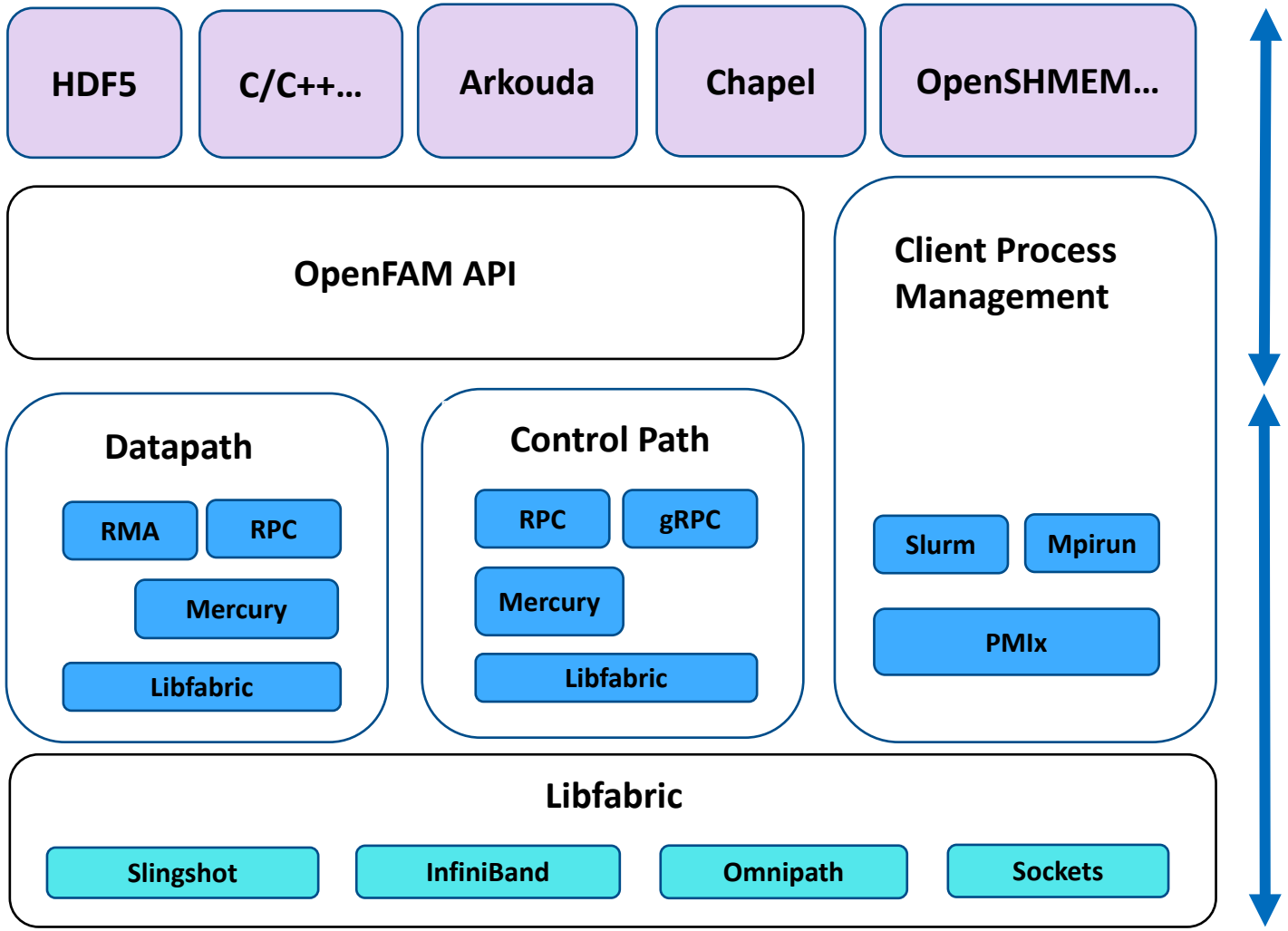
32 Compute Nodes
- DL385 Dual AMD "Milan" CPUs
- 1 TB DRAM each

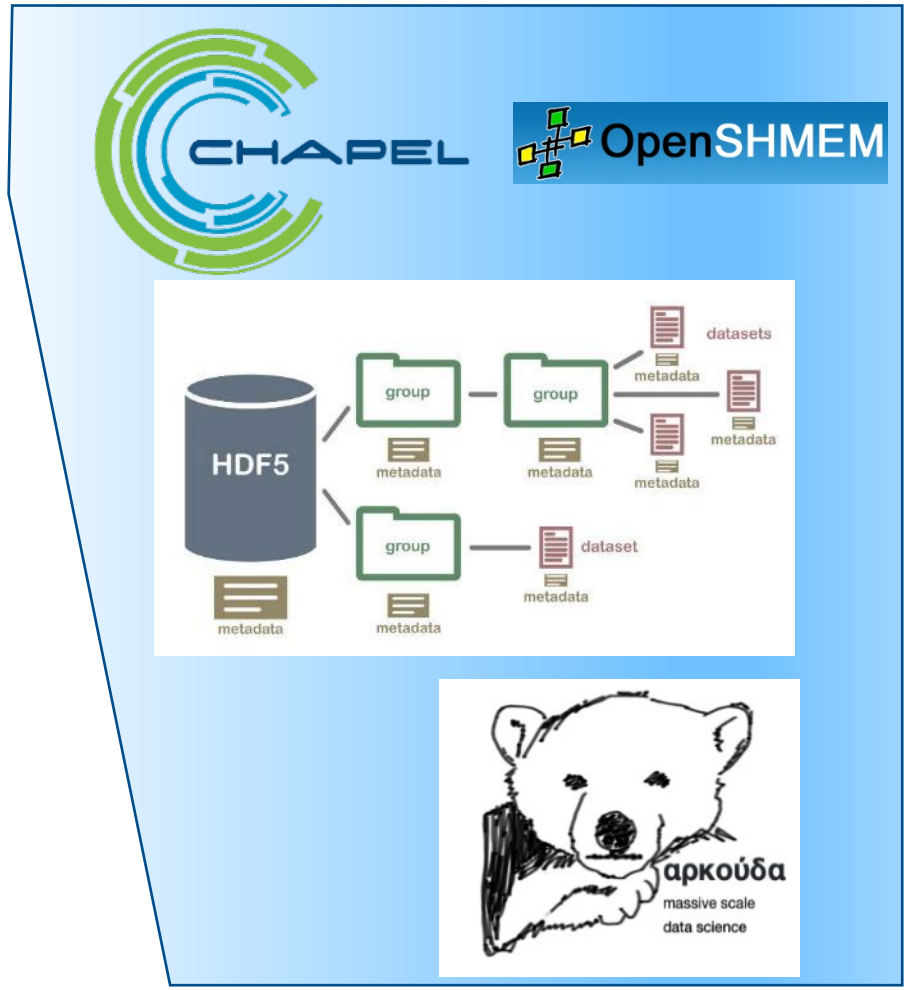
FAM Partition A (10 Nodes)
- DL385 Dual AMD "Milan" CPUs
- 4TB DRAM each

FAM Partition B (10 Nodes)
- DL380 Dual Intel "Ice Lake"
- 1 TB DRAM each
- 8TB "Barlow Pass" SCM each

400 Gbps Slingshot per node
50 TB total DRAM-based FAM
80 TB Optane-based FAM

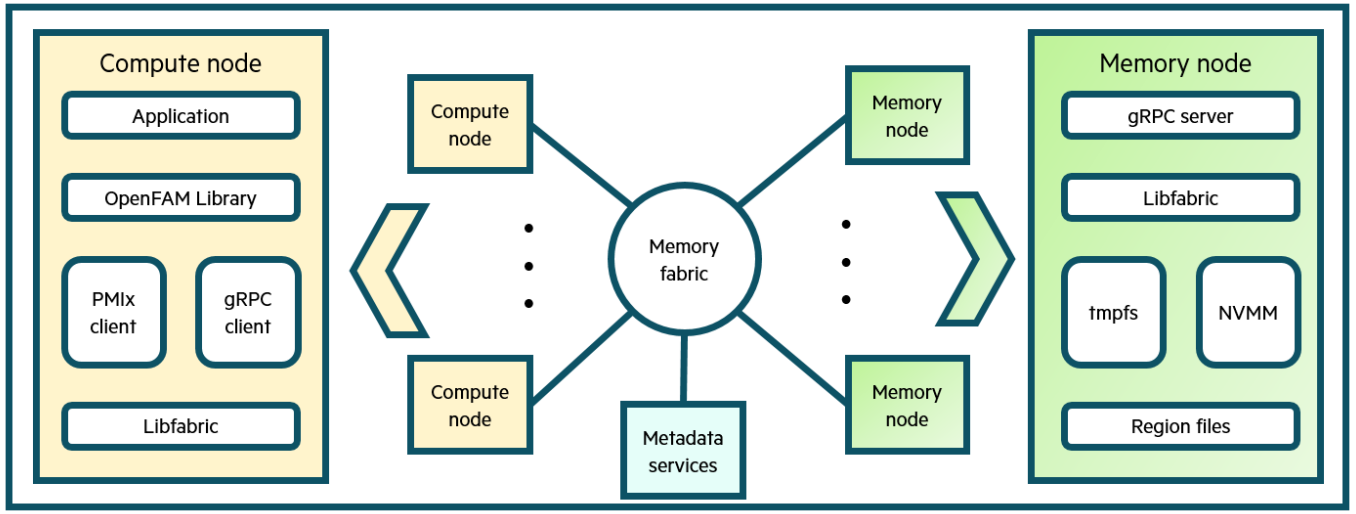
Software stack - components





OpenFAM

- API and reference implementation to program FAM
  - Targeted at scale-up, scale-out and emerging FAM architectures.
  - API is generic, supports both fabric attached persistent memory and volatile memory
  - Supports direct access and RDMA based on the environment.
  - Open source available on github
  - C++ implementation, with C++ and C APIs

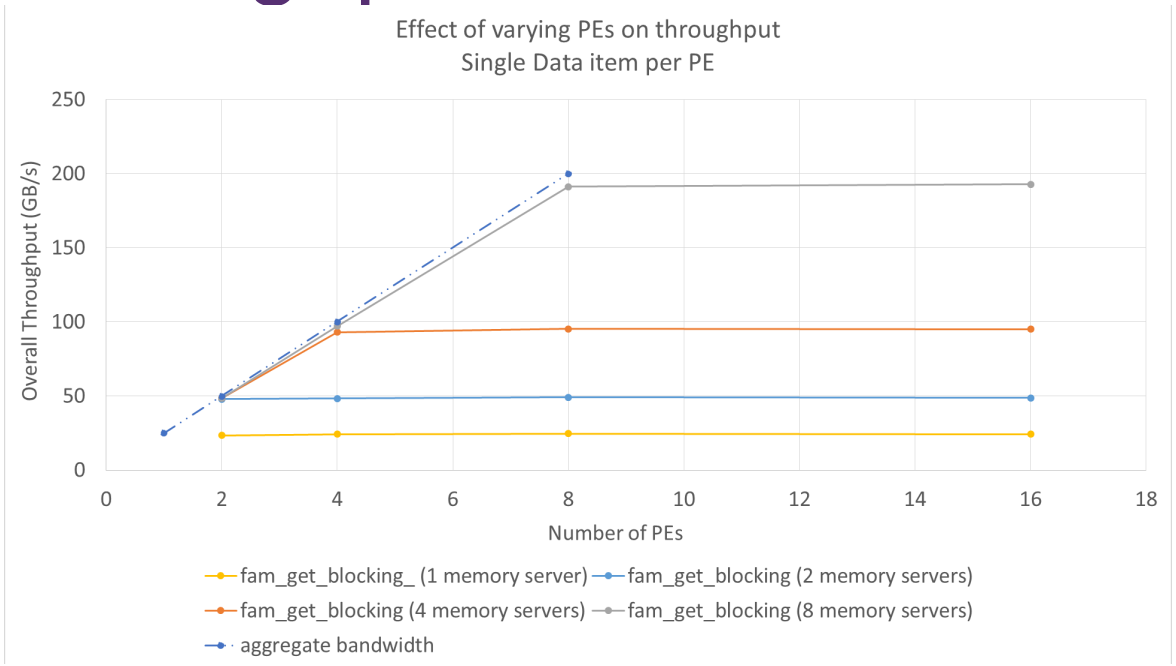


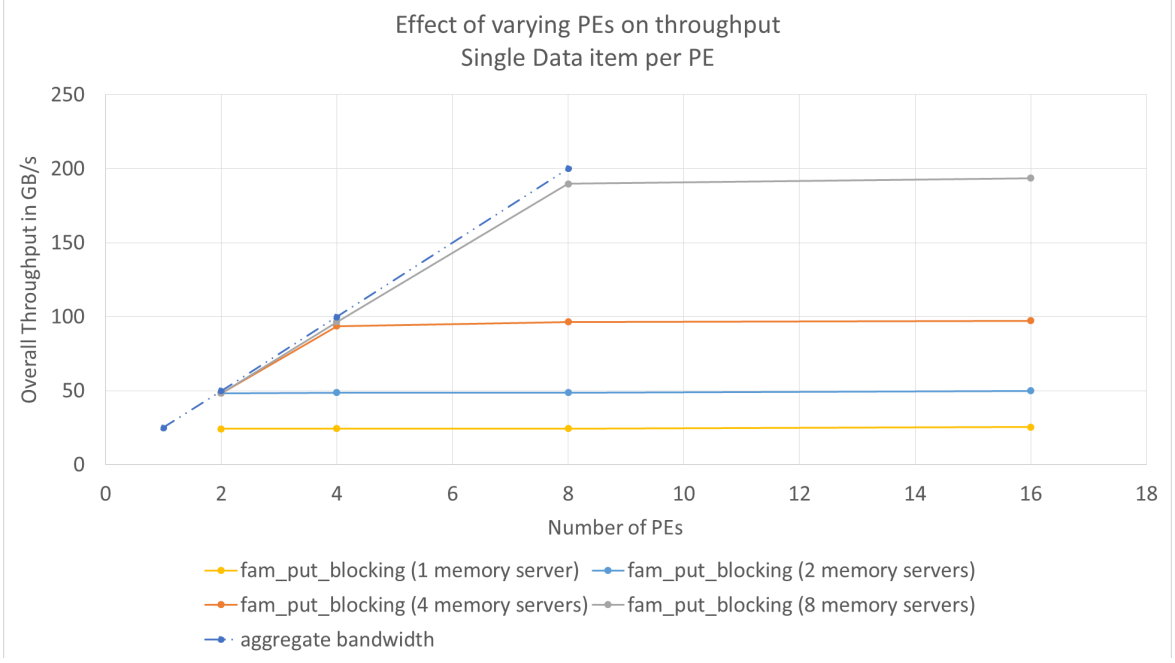
https://github.com/OpenFAM/OpenFAM

Functions supported

- Data path operations
  - Get, put, gather, scatter
    - Blocking, non-blocking
  - Fetching and non-fetching atomics
- Map/unmap
  - Map FAM to process address space
    - Available on scale-up environments (CXL memory in-plan)
- Memory ordering and collectives
  - Quiet, fence, barrier
  - Multi-threading and I/O contexts
- Memory management
  - Region creation, resizing, destruction
  - Data item allocation, deallocation
- Miscellaneous
  - FAM to FAM copy
  - Permission management
  - Data item/region lookup by name
  - FAM backup and restore

Throughput tests for blocking calls



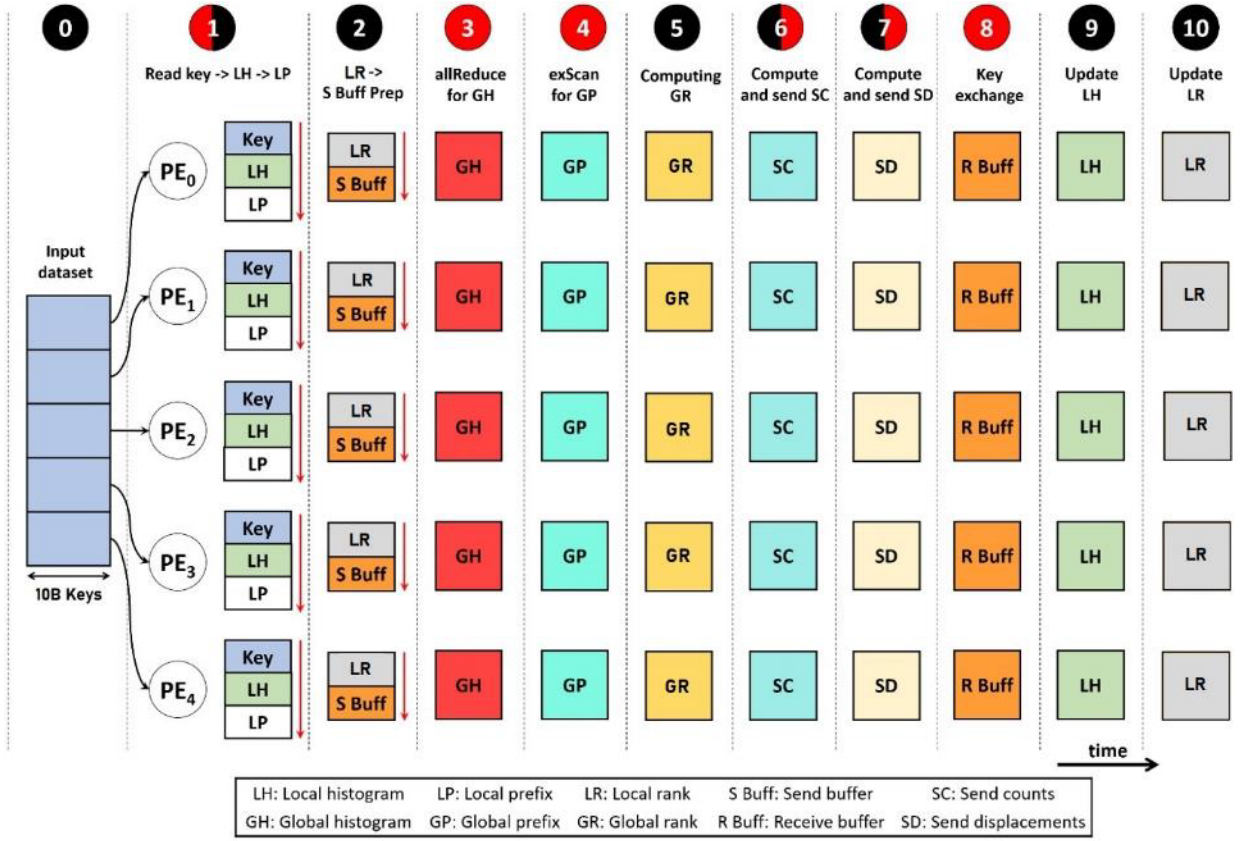


- Interleave size : 65,536
- I/O operations: 100
- Message size: 64 MiB
- Data items/PE: 1
- Number of memory servers: {1, 2, 4, 8}
- Number of PE's : {2, 4, 8, 16}

- Throughput scales linearly with PEs and memory servers
- For 8 memory servers (200 GB/s aggregate bandwidth):
  - fam_get_blocking: 192.8 GB/s
  - fam_put_blocking: 193.6 GB/s

LSD Radix Sort application

Series of stable sorts, iterating through a sequence of select functions, to create sorted result of 10-byte keys.

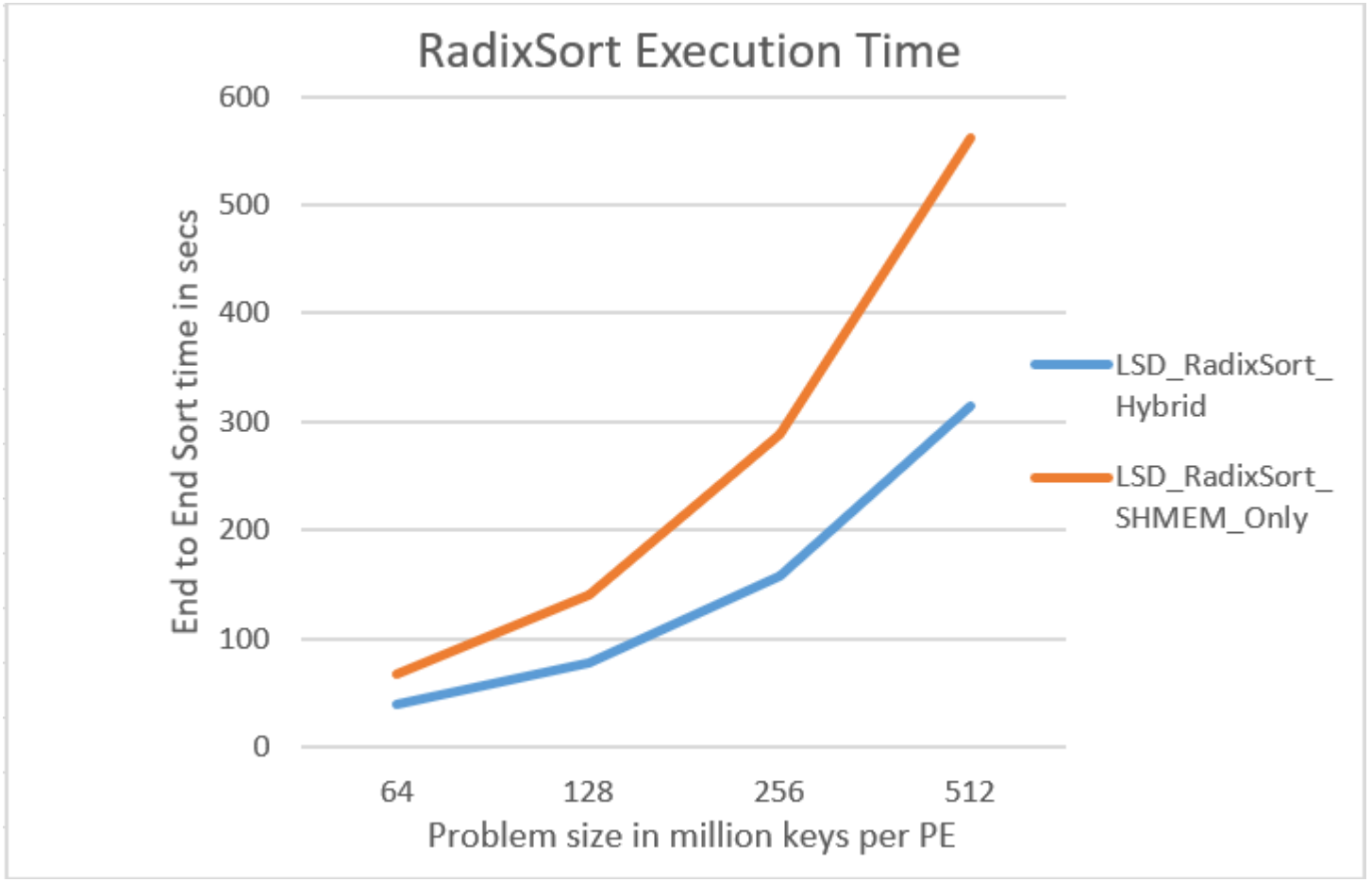


Phases of a Parallel LSD Radix Sort

- Changes to Radix Sort
  - Step 0: Data read from FAM instead of SSD
  - Step 3: Global histogram updates in FAM
  - Step 6: Count (all-to-all) updates use FAM

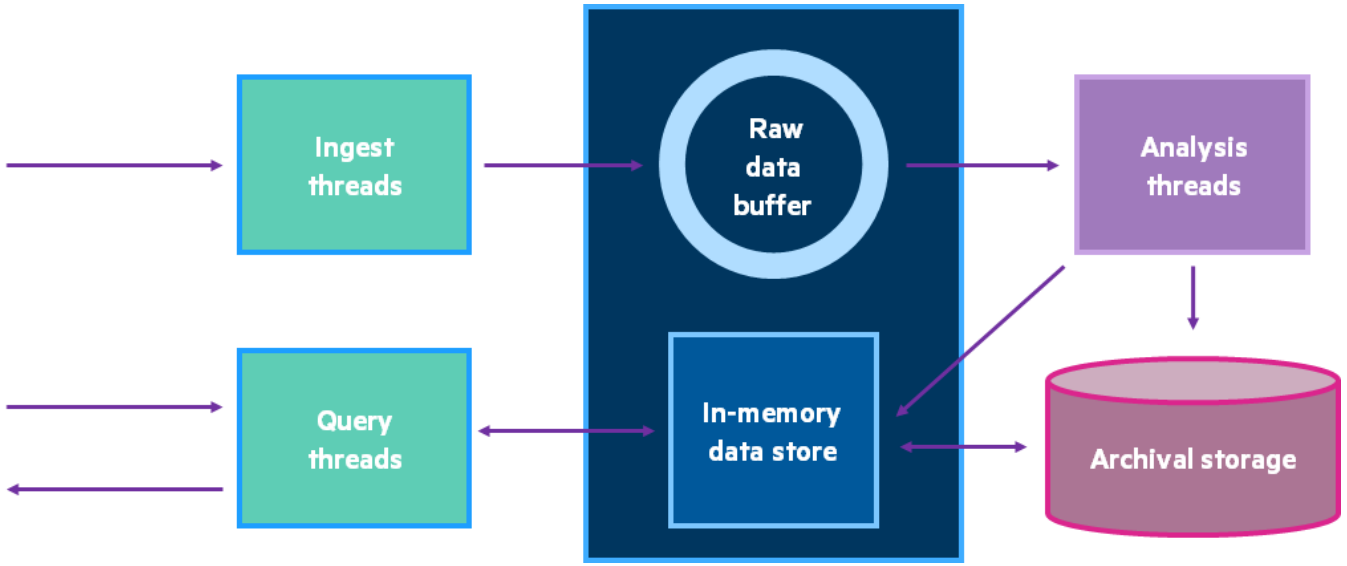
Benefit of FAM in SHMEM-based LSD radix sort

- **Purpose:**
  - Comparison of LSD radix sort SHMEM only version and Hybrid mode results.
  - Graphs show overall application execution time
- **Configuration:**
  - Problem size from 64 million to 512 million elements per PE – Total 32 PEs
    - 2,048 Million – 16,384 Million elements
  - 16 OMP threads per PE
  - For the FAM access (hybrid mode), 4 OpenFAM memory servers.
  - OpenFAM v3.1.0, Cray-OpenSHMEMx/11 5.6.
- **Inference**
  - From initial experiments, Hybrid mode performs better than the SHMEM only version as the problem size increases.
  - With the hybrid mode the end-to-end application time is ~45 to ~55 percent better than the SHMEM only version.



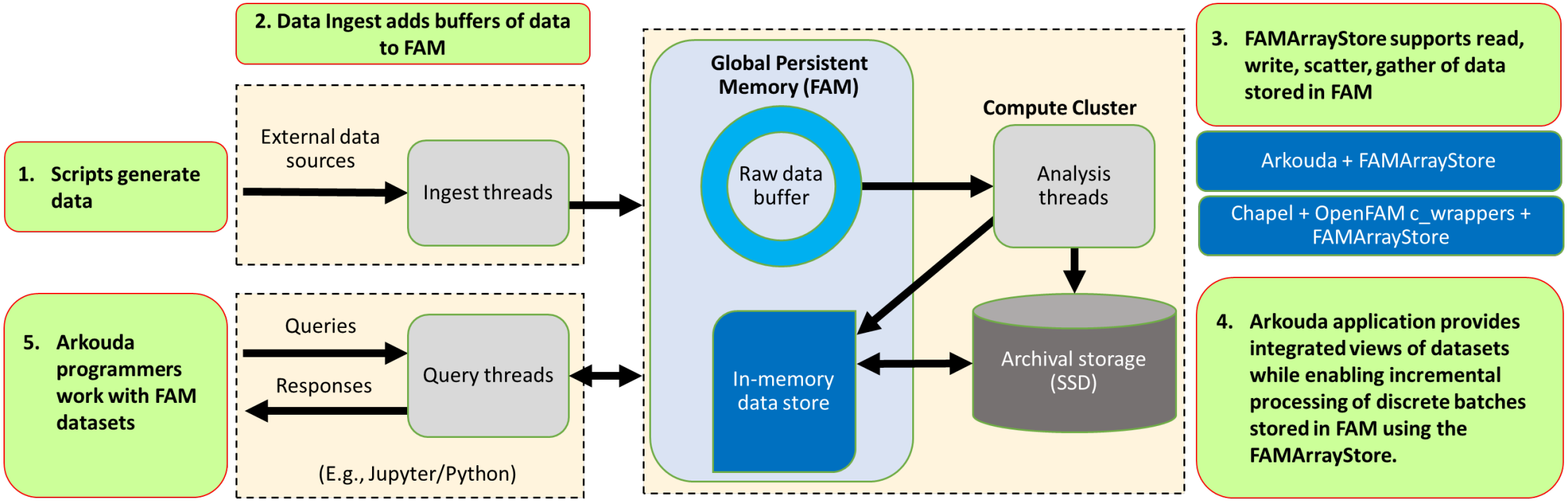
Generalized data-flow for HPC workflows

- Ingest threads
  - Periodically ingest data from external sources (e.g., scientific instruments)
- Analysis threads
  - Background processing of incoming data
  - Long running analyses
  - Interactive queries
- In-memory data store
  - High-speed access to data from queries or analysis threads
- Archival storage
  - Backup of data no longer in use



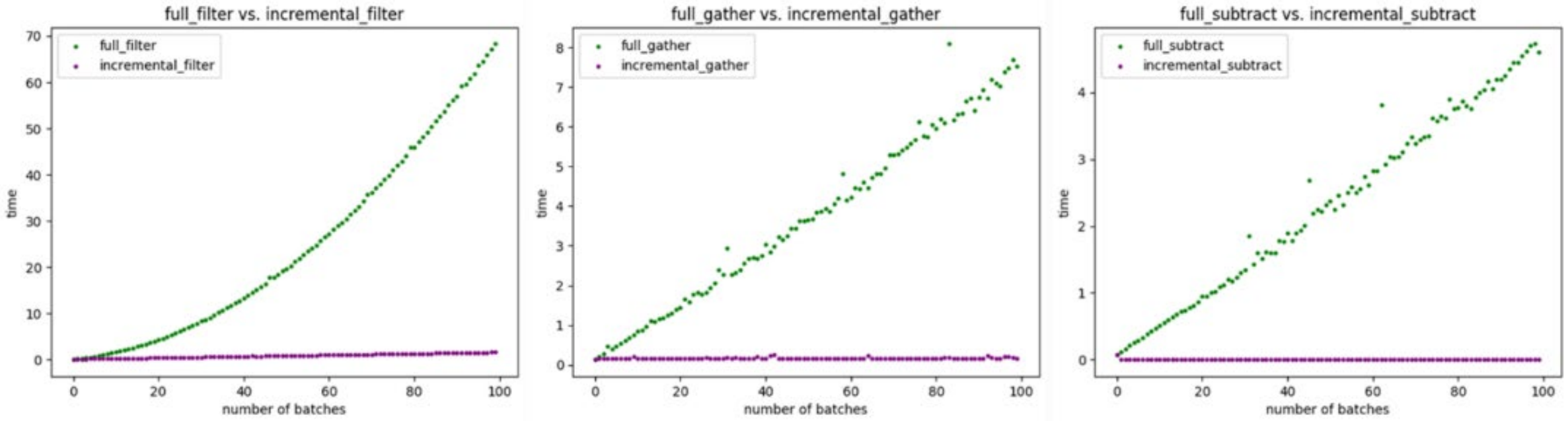
High performance computing at scale for interactive workloads

High performance interactive data analytics



Example application workflow using FAM-enhanced Arkouda.

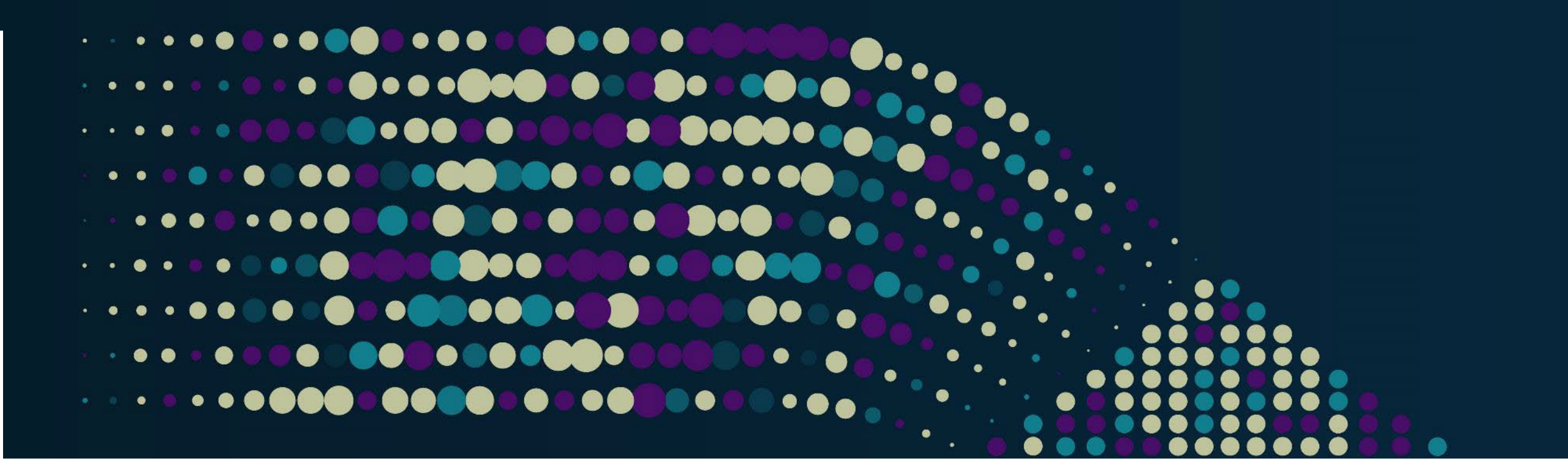
Benefit of FAM-based intermediate data store



Incremental processing leverages previously computed results stored in FAM to speed time to new results.

Summary and future work

- A fabric attached memory architecture for HPC applications
  - Defined the hardware components and configurations
  - Built the software stack to access and use FAM
  - Performance benchmarks for FAM access latencies and throughput
  - Use cases that can benefit from FAM
- Future work
  - CXL memory, future memory technologies on memory nodes
  - More ecosystem enablement – OpenSHMEM, Chapel, Arkouda, data formats



Please take a moment to rate this session.

Your feedback is important to us.

