SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Global Distributed Client-side Cache

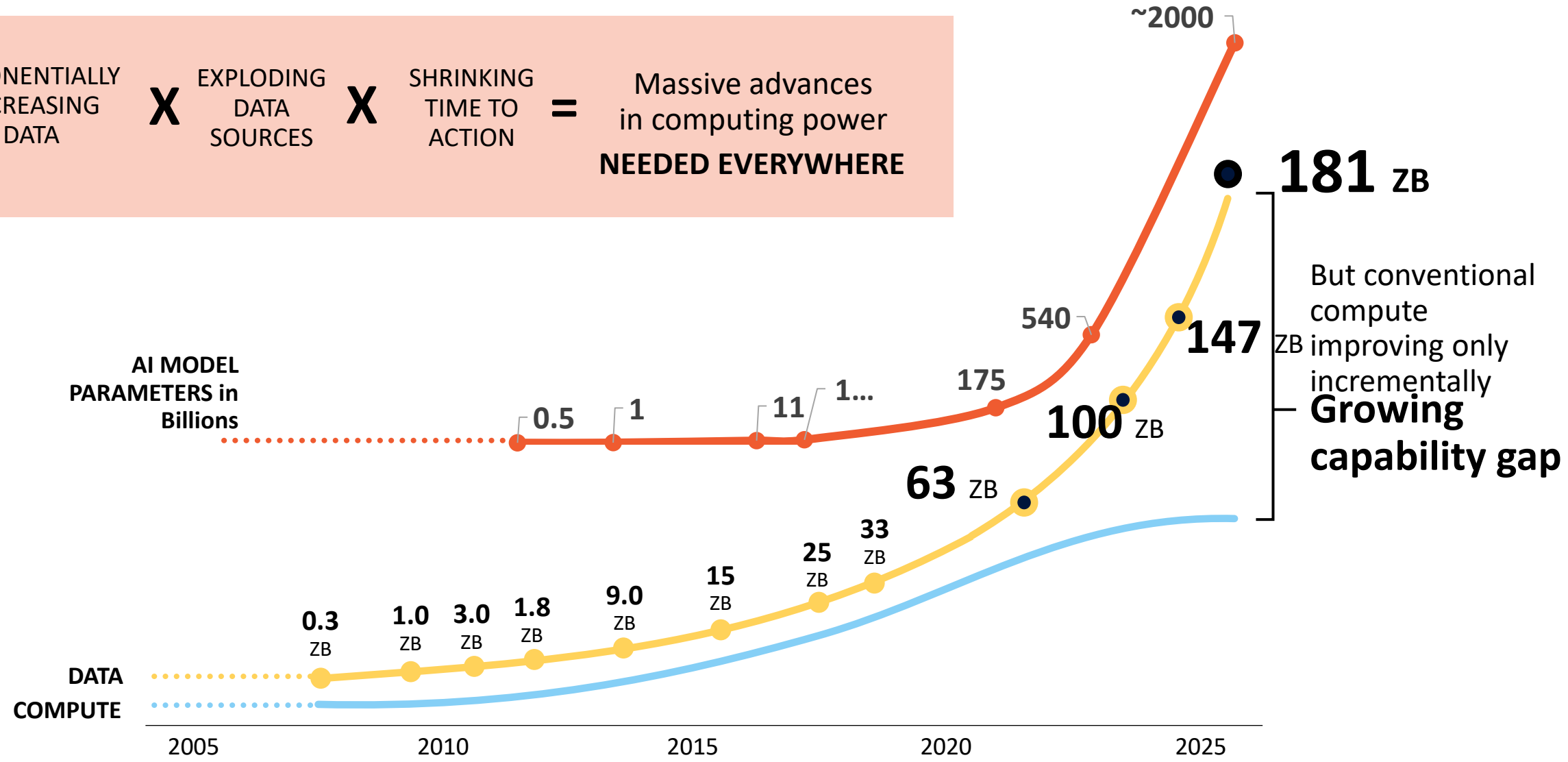
Clarete R. Crasta*, David Emberson*, Harumi Kuno°
*HPC Business Group, ° Hewlett Packard Enterprise Labs

Team:
Abhishek Dwaraki °, John L. Byrne °, Sekwon Lee°, Ramya Ahobala Rao°, Shreyas Vinayaka Basri K S°, Amitha C°, Chinmay Ghosh°, Sriram Ravishankar°, Porno Shome°

www.sniadeveloper.org

The new normal

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



Source: Hewlett Packard Enterprise

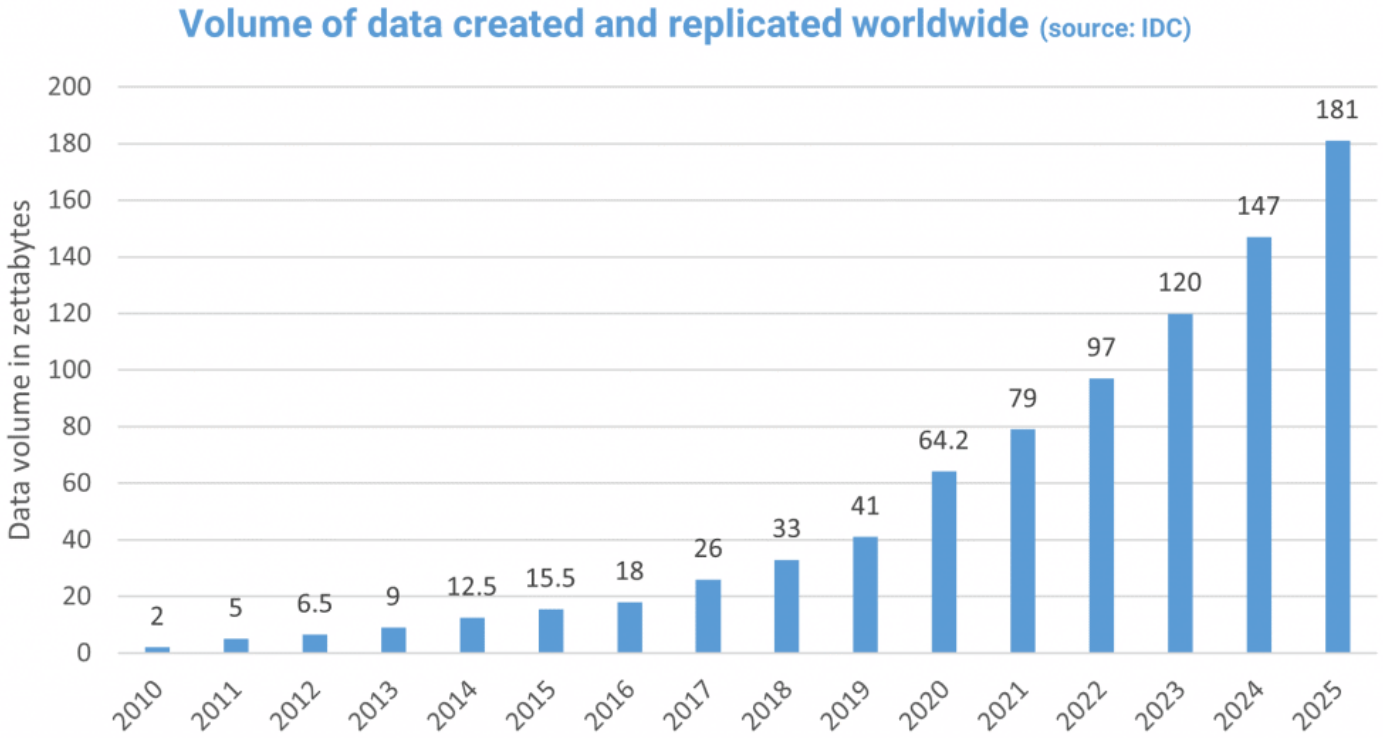
 ©2025 SNIA Developer Conference. All Rights Reserved.

Motivation

Challenges:

- HPC and AI workloads demand efficient handling of petabyte-scale data.
- Centralized storage impacts performance.
- Server-side caches constrained by DRAM and network bandwidth; scalability is limited as server resources cannot grow dynamically with workload demands.
- Most node-local client-side caches lack data sharing across nodes.

Need: An effective caching solution compatible with modern memory/storage technologies.

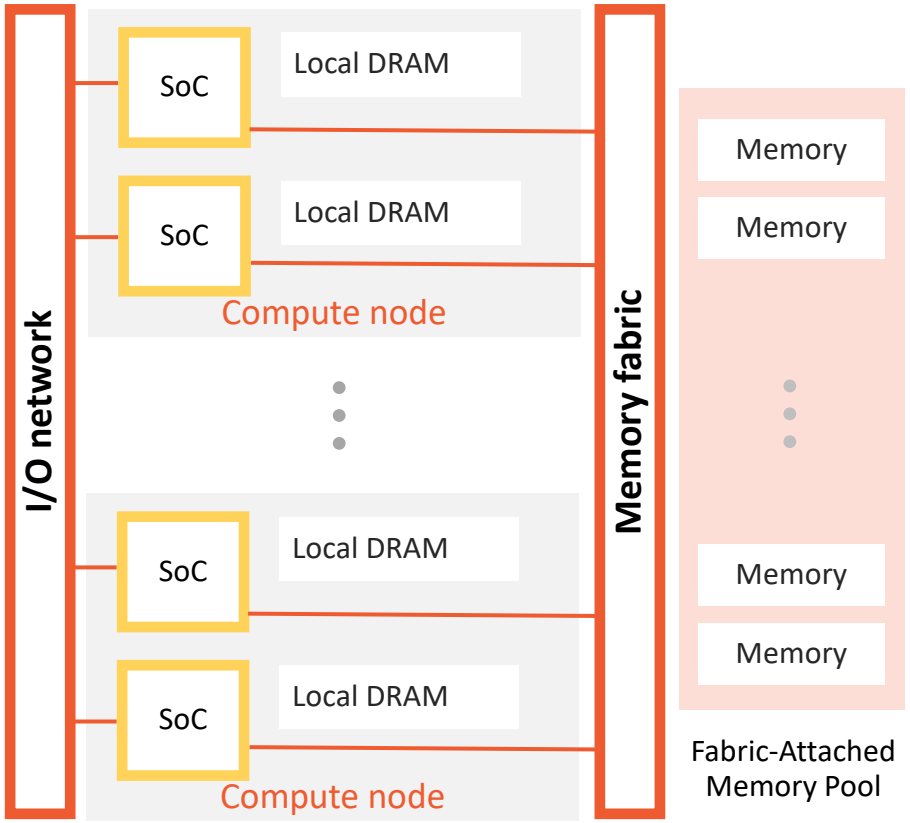


 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

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
    - VM instead of disks/SSDs
    - Support for in-memory formats
- Challenges
  - Coherence domains are limited to individual nodes
    - Transparent caching or paging limits scalability
  - Fabric latency is large compared to local memory
    - Software has to be (usually) refactored for performance
  - Few easy-to-use APIs available for developers
    - A number of efforts[1,2] underway
      - Generally use object store or file system abstractions



1. Intel DAOS: https://github.com/daos-stack/daos
2. M. Grodowitz, P. Shamis, and S. Poole, "OpenSHMEM I/O Extensions for Fine-Grained Access to Persistent Memory Storage," in *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, vol. 1315, J. Nichols, B. Verastegui, A. 'Barney' Maccabe, O. Hernandez, S. Parete-Koon, and T. Ahearn, Eds. Cham: Springer International Publishing, 2020, pp. 318–333; https://github.com/openucx/shmem-opensnapi

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

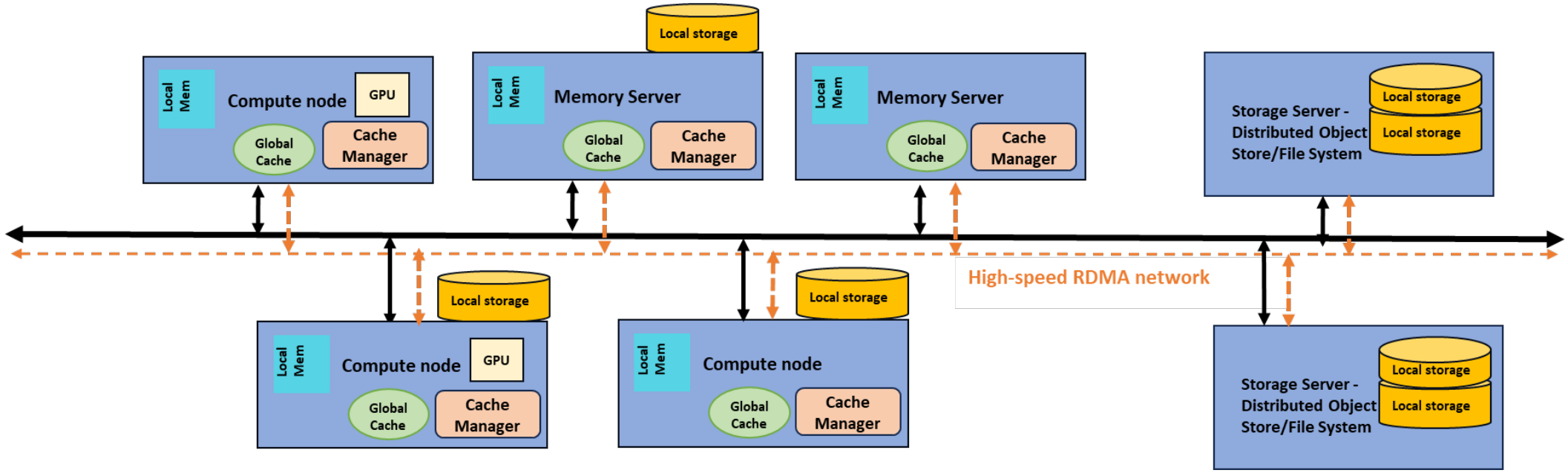
Outline

- Global client-side cache architecture
- Components of the Global Client-side cache
- Evaluation of the benefits of the cache with PageRank
- Future work
- Summary

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

Global client-side cache architecture



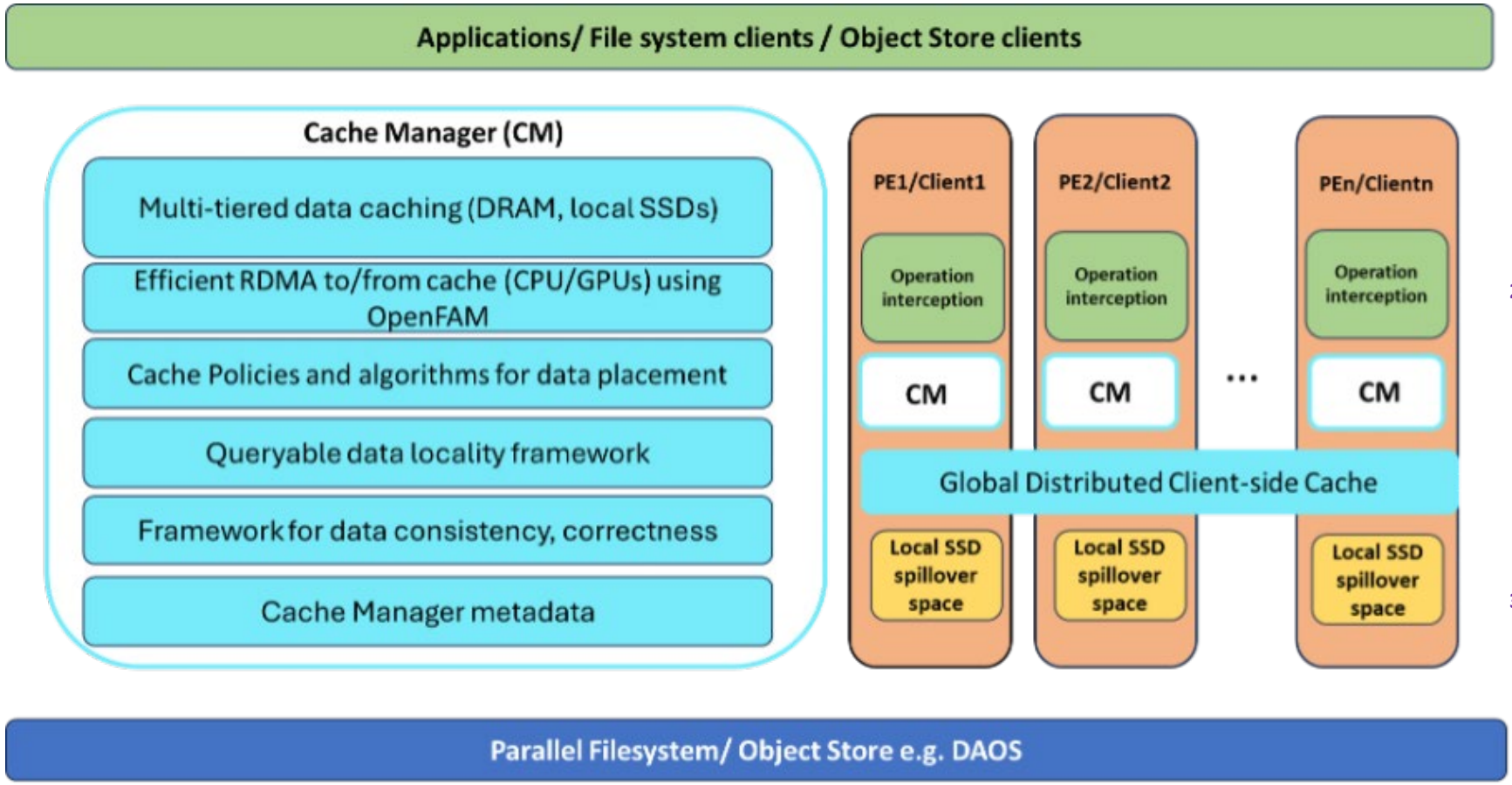
- **Shared Cache:**
  - Tiered Caching: DRAM, local NVMe SSDs, and GPU memory.
  - A distributed cache manager consolidates per-node resources and access latencies for each tier.
  - Cache is shared across applications running in a cluster.

- **Efficient Data Access:**
  - Local SSDs as spillover storage, balancing cost and performance.
  - Data cached locally to nodes with higher probability of access, enables scheduling tasks closer to data.
  - Data moved dynamically located across tiers (DRAM, SSDs, HDDs).
  - RDMA-based high-speed data movement.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

Components of the Global Client-side Cache

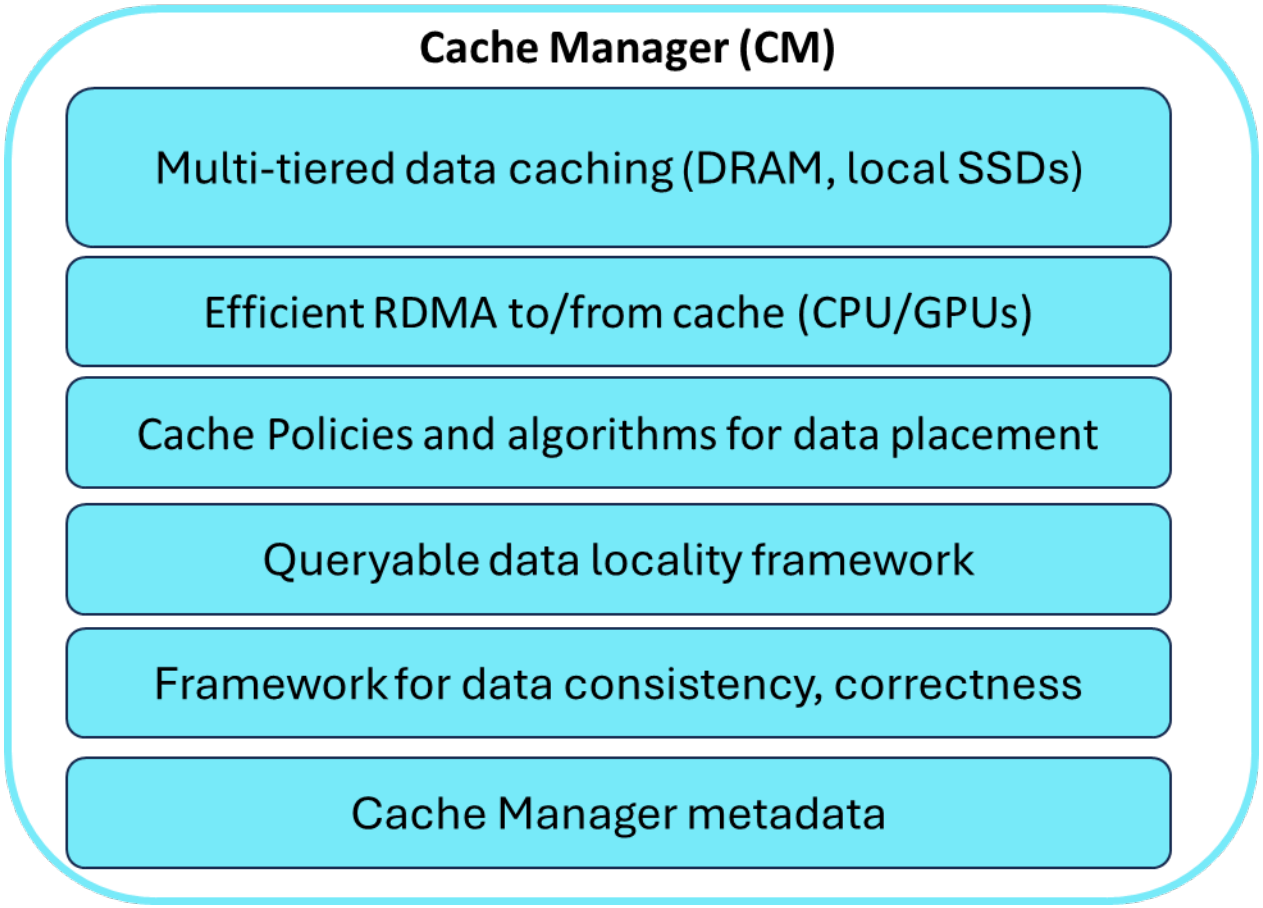


1. **Cache Manager**:
   - Coordinates cache capacity and usage across nodes.
   - Ensures data consistency and correctness.
2. **RDMA-Based Framework**:
   - Enables high-speed, low-latency data movement.
   - Uses OpenFAM for efficient memory management.
3. **API and Configuration**:
   - Comprehensive APIs for object management, data access, and metadata handling.
   - Configurable eviction policies and invalidation mechanisms.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

Cache Manager



Functionality:

- Tracks dirty bits and maintains global metadata.
- Supports configurable invalidation and eviction policies
- Monitors cache usage and moves data across nodes and memory/storage tiers.

Mechanisms for Data Access:

- **CPU and GPU Memory**:
  - Uses RDMA for remote memory access.
  - Leverages OpenFAM for efficient inter-node data transfers.
- **SSDs/Flash Storage**:
  - Accesses data through operating system or file system interfaces.
- **Distributed File Systems/Object Stores**:
  - Integrates with APIs to access data stored in distributed systems.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

API and Configuration Overview

Cache Manager APIs

- A robust set of APIs to manage distributed caching systems, enabling efficient data access, caching, and management across multiple nodes and memory/storage tiers.
- APIs are accessible to frameworks like runtime systems or storage clients (e.g., DAOS clients).
- User-visible APIs allow configuration of caching policies and parameters.

Key API Categories and Examples

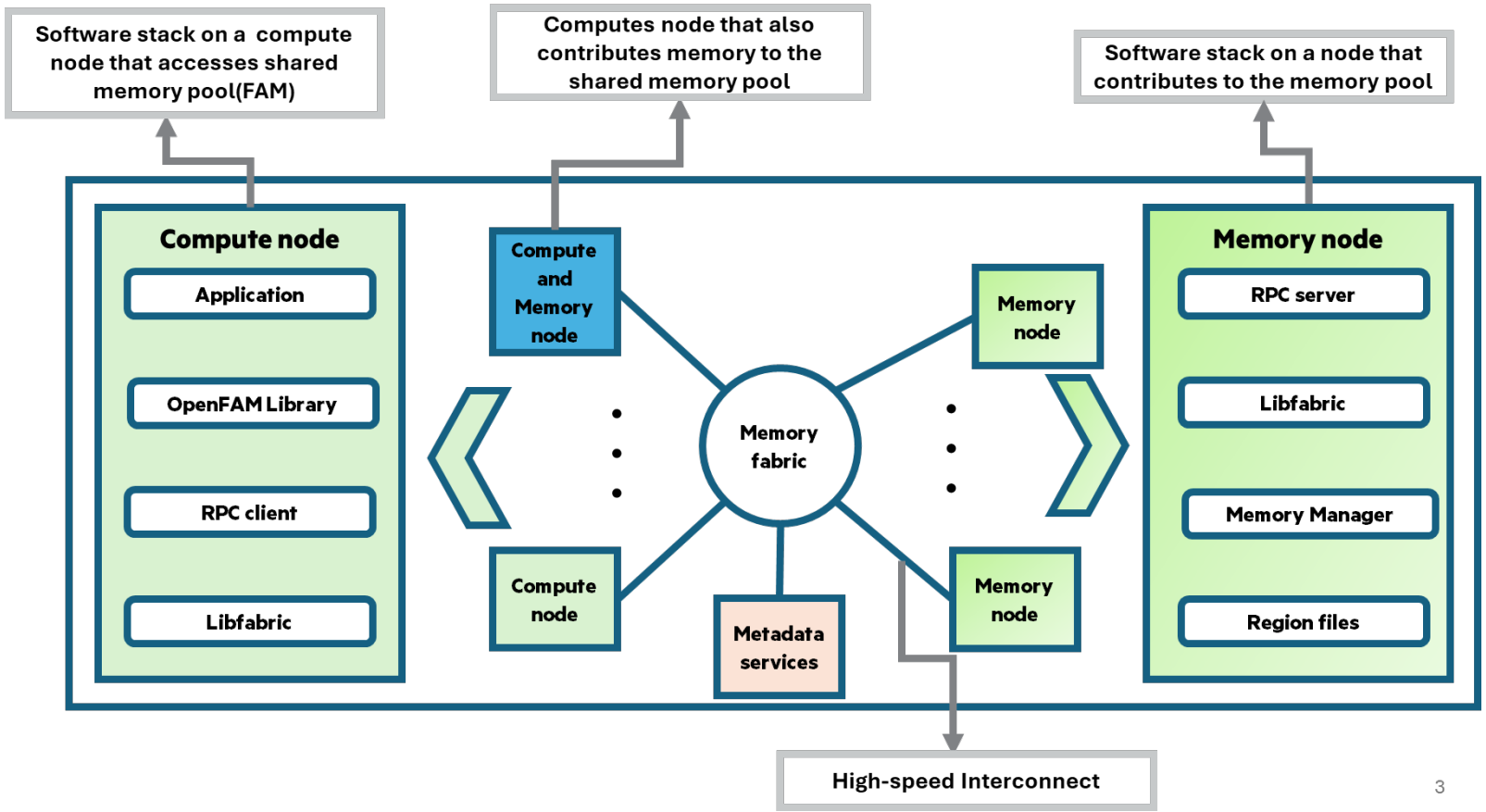
- **Object Management**
  cm_open_object, cm_close_object, cm_delete_object
- **Data Access**
  cm_read, cm_write, cm_preserve
- **Metadata and Locality**
  cm_get_cache_config, cm_get_locality, cm_set_cache_config
- **Directory Management**
  cm_mkdir, cm_rmdir

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

RDMA-Based Framework and Protocol for Global Cache

- **OpenFAM Integration**:
  - OpenFAM is an API and reference implementation for accessing a shared pool of memory over fabric.
  - Leverage OpenFAM for RDMA between nodes in the Global cache
- **OpenFAM Memory Servers**:
  - Manage memory contributions to the global cache on individual nodes.
  - Operate on each node to handle memory mapping, allocation, and release of cache segments.
  - Enables efficient data sharing across heterogeneous environments with CPUs and GPUs.



[OpenFAM: A library for programming Fabric-Attached Memory]

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

OpenFAM: API for programming fabric-attached memory

- Inspired by
  - OpenSHMEM[1]
  - HPE's work on The Machine program
- The API[2] largely parallels the OpenSHMEM 1.3 API
  - Get/put, scatter/gather, atomics, memory ordering (fence/quiet)
- Differences between OpenFAM and OpenSHMEM environments:
  - Focus is on fabric-attached memory (FAM) instead of remote processing element (PE) memory
    - Node memory is treated as private, while FAM is shared
    - Any PE can access FAM-resident data independently of other PEs
  - FAM-resident data can outlive the application
    - Named data items, directory operations (e.g., lookup)
    - Permissions associated with allocations
  - FAM is not associated with any PE
    - Allocation/deallocation can be done from any PE
    - Any PE can access any allocation

1. OpenSHMEM : http://www.openshmem.org/site/Specification
2. Keeton K., Singhal S., Raymond M. (2019) *The OpenFAM API: A Programming Model for Disaggregated Persistent Memory*. In: Pophale S., Imam N., Aderholdt F., Gorentla Venkata M. (eds) OpenSHMEM and Related Technologies. OpenSHMEM in the Era of Extreme Heterogeneity. OpenSHMEM 2018. Lecture Notes in Computer Science, vol 11283. Springer, Cham

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

OpenFAM API summary[1]

API

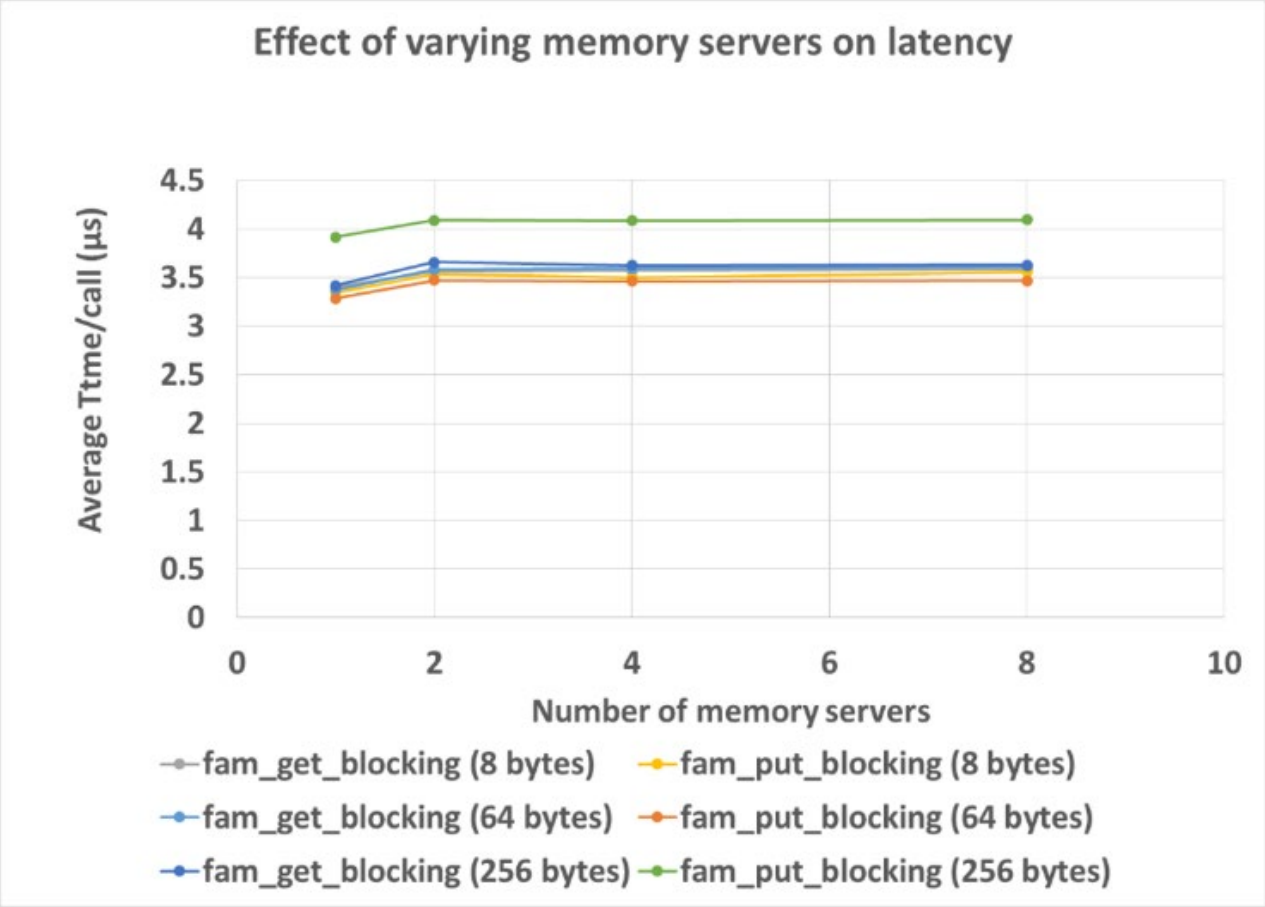
- Data path operations
  - Get, put, gather, scatter
    - Blocking, non-blocking, atomic variants
- FAM atomics
  - Fetching and non-fetching atomics
- Memory ordering and collectives
  - Quiet, fence, barrier
  - Thread safety and contexts
- Memory management
  - Region creation, resizing, destruction
  - Data item allocation, deallocation

1. The complete API is documented at https://OpenFAM.github.io

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

OpenFAM performance numbers



Cluster configuration

- 52-node Slingshot-based HPC cluster.
- Compute nodes: 2-socket AMD EPYC 7763, 1 TiB DRAM.
- Memory nodes: 4 TiB DRAM.

- **Results**:
  - Latency < 5 microseconds for short messages.

 ©2025 SNIA Developer Conference. All Rights Reserved.

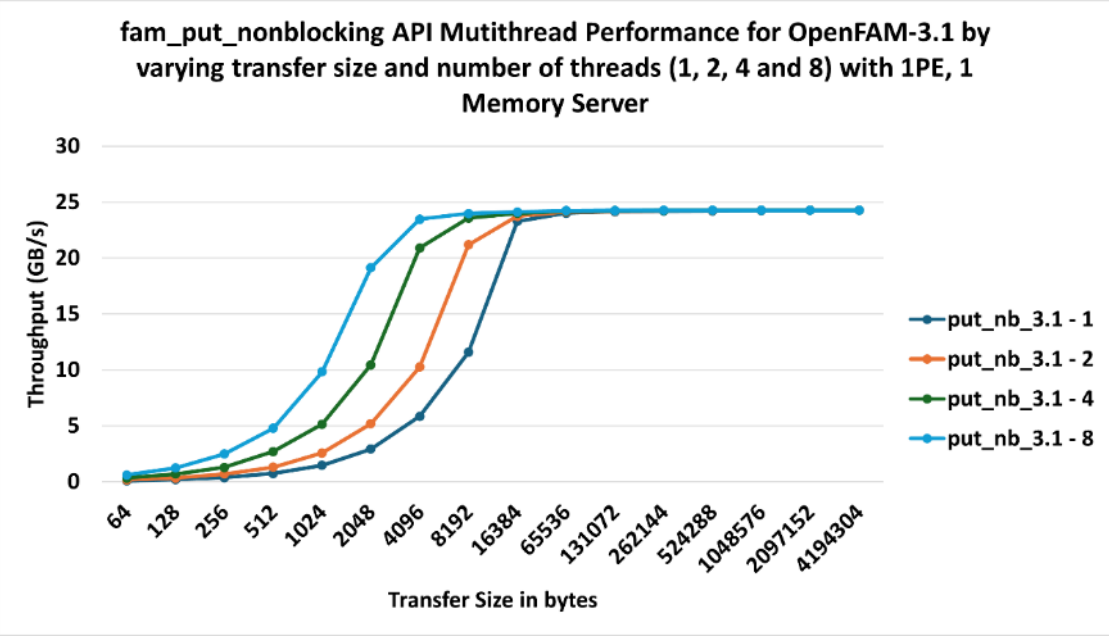
HPE Proprietary

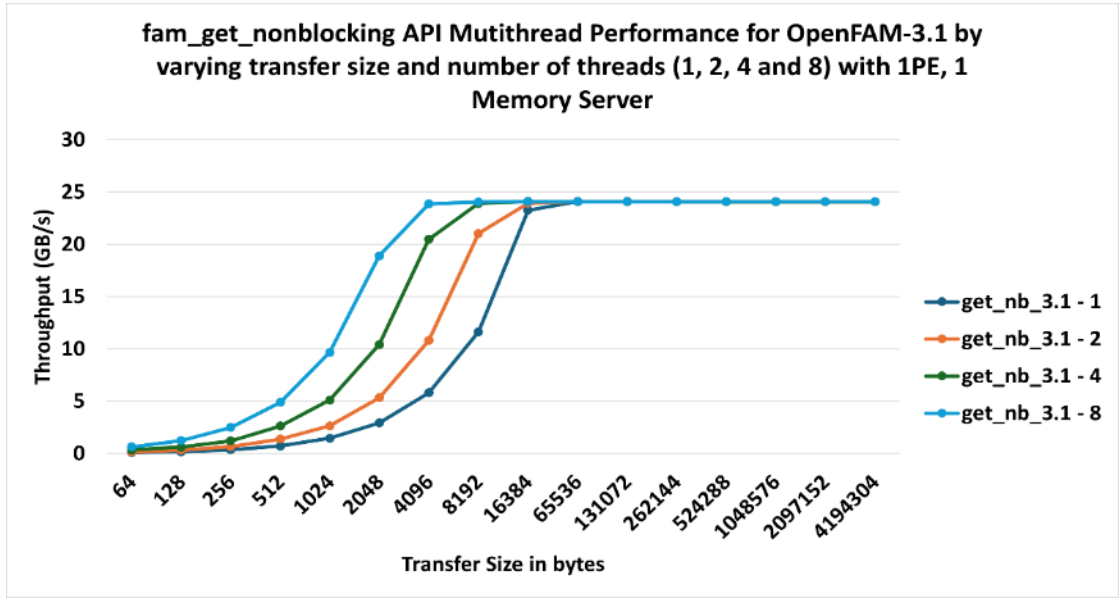
OpenFAM performance numbers

Cluster configuration

- 52-node Slingshot-based HPC cluster.
- Compute nodes: 2-socket AMD EPYC 7763, 1 TiB DRAM.
- Memory nodes: 4 TiB DRAM.

- **Results**:
  - Blocking and non-blocking RDMA operations achieve near link bandwidth.



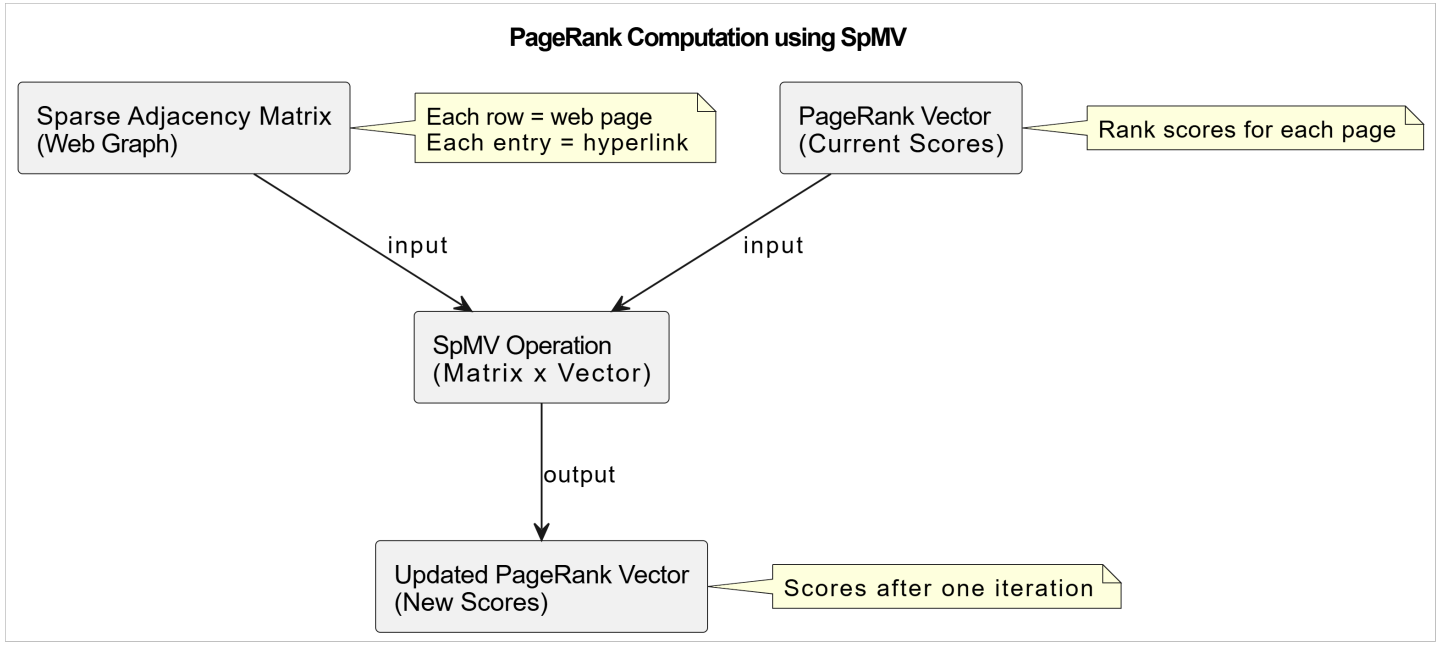


 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

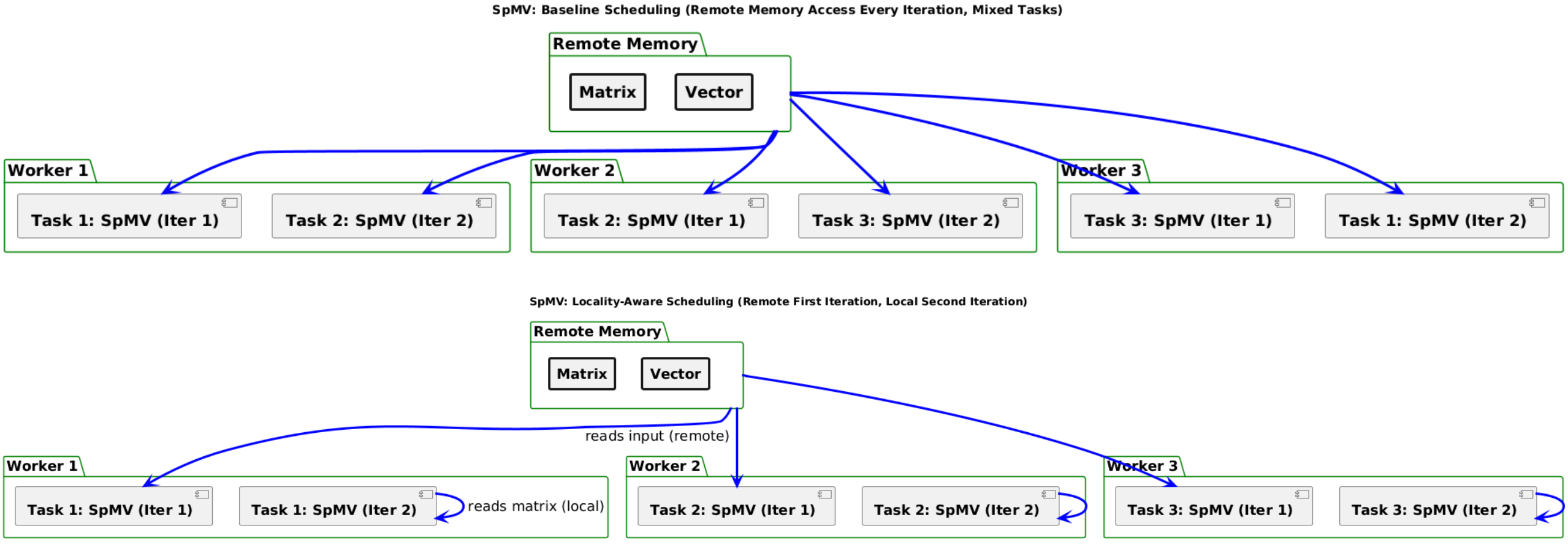
PageRank Algorithm & Cache Optimization

- **PageRank** is a web page ranking algorithm developed by Google founders Larry Page and Sergey Brin.
- Models the web as a directed graph:
  - **Nodes** = web pages
  - **Edges** = hyperlinks
- A page's importance is determined by the quantity and quality of incoming links.
- **Computation:**
  - Uses **Sparse Matrix-Vector Multiplication (SpMV)**
  - Web graph → sparse adjacency matrix
  - PageRank vector updated iteratively:
    - Repeats until scores converge



 ©2025 SNIA Developer Conference. All Rights Reserved.

PageRank locality-aware scheduling and Client-side caching



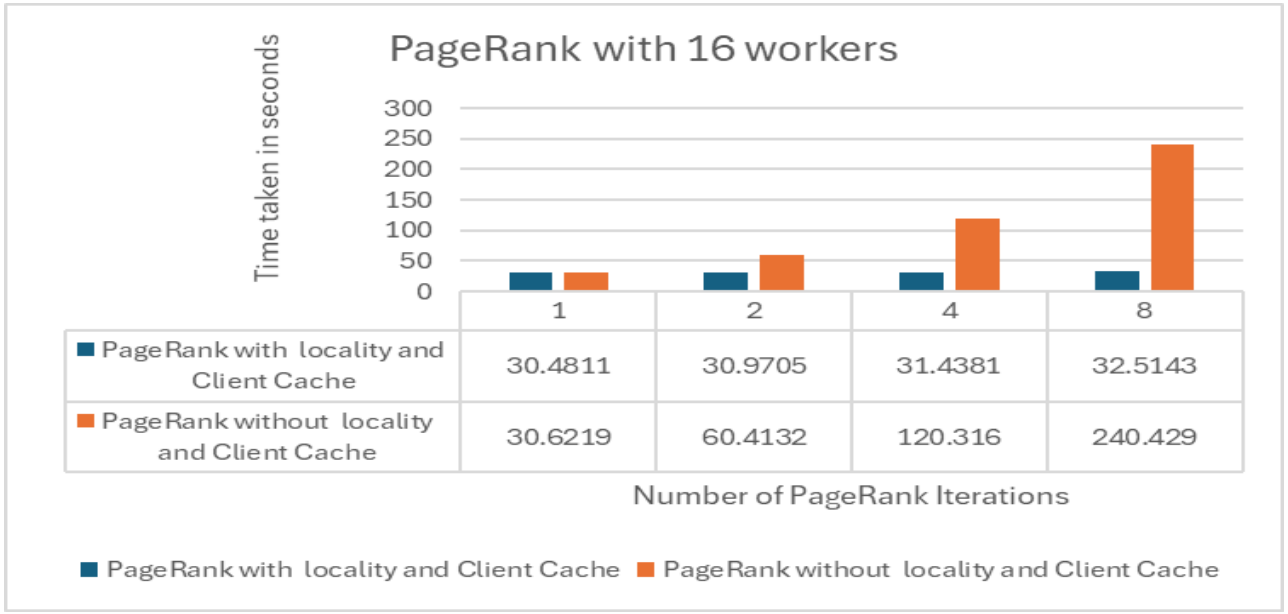
 ©2025 SNIA Developer Conference. All Rights Reserved.

Demonstration of the benefits of locality-aware scheduling and client-side caching

Experiment Setup:

- Baseline: data fetched from remote memory every iteration.
- Optimized: Locality-aware scheduling with client-side caching.
- Matrix size: 16M x 16M.
- Hardware: 16 workers, 4 memory servers, Slingshot interconnect.
- Two-socket AMD EPYC 7763 64-Core Processor

Note: This comparison focuses solely on local vs. remote DRAM access and does not include any evaluation of DAOS operations.



- **Results**:
  - Preliminary experiments show optimized version ~7x faster than baseline after 8 iterations.
- **Insights**:
  - Locality-aware scheduling significantly improves performance.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

Future Vision

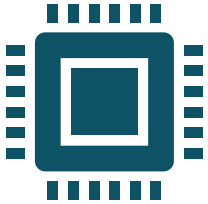
- **Planned Features**:
  - Multiple configurable caching polices.
  - Configurable invalidation mechanisms.
  - Enhanced GPU integration.
  - Runtime that takes advantage of the cache.

- **Long-Term Vision**:
  - Dynamic capacity adjustments for varying workloads.
  - Expand support to other HPC file systems.
  - Exploration of extension to server-side cache.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

Summary



Proposed Solution:

Global, multi-tiered client-side cache for HPC filesystems and object stores.

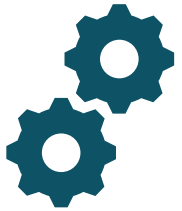
Supports heterogeneous compute environments with CPUs, GPUs, and accelerators.

Key Benefits:

Reduces latency and improves efficiency.

Scales independently of storage servers.

Adapts to diverse workloads and environments.



Future Potential:

Extend compatibility to additional file systems.

Optimize for emerging HPC and AI workloads.

 ©2025 SNIA Developer Conference. All Rights Reserved.

Acknowledgements

- **Special Thanks**:
  - Lance Evans, Johann Lombardi for insights into DAOS and its capabilities.
  - Pete Haddad, Eric Wu, and Binoy Arnold for their continuous support with hardware and software resources.
  - The HPE DAOS Engineering team, Kevan Rehm and Sherin George for discussions and inputs on DAOS integration and evaluation.
  - Chris Rickett and Rangan Srinivasan for their efforts with the IDS use case.
  - Sergey Serebryakov, Taeklim Kim, Paolo Faraboschi, and the AI Labs for their assistance with GPU integration and other explorations.
  - Michelle Strout for insightful discussions around Arkouda and the Cache.
  - Mike Woodacre, Larry Kaplan for their valuable inputs and support.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary

References

[1] Lustre: A scalable, high-performance file system cluster, 2003. Accessed: 2025- 01-24

[2] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Distributed afs, 2025. Accessed: 2025-04-08.

[3] Syed Ismail Faizan Barmawer, Gautham Bhat Kumbla, Mashood Abdulla Kodavanji, Clarete Riana Crasta, Sharad Singhal, and Ramya Ahobala Rao. Client allocation of memory across memory servers, 2024. US Patent App. 17/815,366.

[4] Benjamin Berg, Daniel S. Berger, Sara McAllister, Isaac Grosof, Sathya Gunasekar, Jimmy Lu, Michael Uhlar, Jim Carrig, Nathan Beckmann, Mor Harchol-Balter, and Gregory R. Ganger. The cachelib caching engine: Design and experiences at scale. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2020.

[5] M. Blaze and R. Alonso. Dynamic hierarchical caching in large-scale distributed file systems. In [1992] Proceedings of the 12th International Conference on Distributed Computing Systems, pages 521–528, 1992.

[6] Compute Express Link Consortium. Compute express link (cxl), 2025. Accessed: 2025-04-08.

[7] Michael D. Dahlin, Randolph Y. Wang, Thomas E. Anderson, and David A. Patterson. Cooperative caching: using remote client memory to improve file system performance. In Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, OSDI '94, pages 19–es, USA, 1994. USENIX Association. event-place: Monterey, California.

[8] daos stack. Daos storage stack, 2020. Accessed: 2020-08-27.

[9] Semiconductor Engineering. High-bandwidth memory, 2025. Accessed: 2025-04-08.

[10] Dave Henseler et al. Architecture and design of cray datawarp. In Proceedings of the Cray User Group (CUG) Conference 2016, 2016. Accessed: 2025-04-08.

[11] NVM Express. What is nvme technology?, 2025. Accessed: 2025-04-08.

[12] MPI Forum. Mpi forum documentation, 2025. Accessed: 2025-04-08.

[13] K. Harms. Daos is your future distributed asynchronous object store. Presented at the Argonne Leadership Computing Forum Hands-on HPC Workshop, October 2023. Accessed: 2024-09-30.

[14] M. Hennecke. Understanding daos storage performance scalability. In Proceedings of the HPC Asia 2023 Workshops, pages 1–14. ACM, 2023.

[15] IBM. Storage-class memory: The next storage system technology. IBM Journals & Magazine | IEEE Xplore, 2010. Accessed: 2025-04-08.

[16] Petros Koutoupis. The lustre distributed filesystem. Linux J., 2011(210), October 2011. Article 3.

[17] Nimrod Megiddo and Dharmendra S. Modha. Arc: A self-tuning, low overhead replacement cache. In Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST), 2003.

[18] Jeffrey C. Mogul. Recovery in spritely nfs. Comput. Syst., 7(2), Spring 1994.

[19] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications. In Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation (OSDI), pages 561–577. USENIX Association, 2018.

[20] NVIDIA. Rdma architecture overview, 2025. Accessed: 2025-04-08.

[21] OpenSHMEM. Openshmem specification, 2025. Accessed: 2025-04-08.

[22] Yingjin Qian, Xi Li, Shuichi Ihara, Andreas Dilger, Carlos Thomaz, Shilong Wang, Wen Cheng, Chunyan Li, Lingfang Zeng, Fang Wang, Dan Feng, Tim Süß, and André Brinkmann. Lpcc: Hierarchical persistent client caching for lustre. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '19), 2019.

[23] Redis. Client-side caching, 2025. Accessed: 2025-04-08.

[24] S. Singhal et al. Openfam: A library for programming disaggregated memory. In OpenSHMEM and Related Technologies: OpenSHMEM in the Era of Extreme Heterogeneity, volume 13694 of Lecture Notes in Computer Science, pages 18–32. Springer, 2022.

[25] S. Singhal et al. Openfam: Programming disaggregated memory. Concurrency and Computation: Practice and Experience, 35(5):e7291, 2023.

[26] Sharad Singhal, Clarete R. Crasta, et al. Openfam: A library for programming disaggregated memory. In Proceedings of the Cray User Group (CUG) 2022, 2022. Accessed: 2025-04-08.

[27] Foteini Strati, Xianzhe Ma, and Ana Klimovic. Orion: Interference-aware, finegrained gpu sharing for ml applications. In Proceedings of the Nineteenth European Conference on Computer Systems, pages 1075–1092, 2024.

[28] H. Tang et al. Toward scalable and asynchronous object-centric data management for hpc. In 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pages 113–122, 2018.

[29] Teng Wang, Kathryn Mohror, Adam Moody, Kento Sato, and Weikuan Yu. An Ephemeral Burst-Buffer File System for Scientific Applications. In SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 807–818, November 2016. ISSN: 2167-4337.

[30] Weka. Weka architecture white paper, 2023. Accessed: 2025-04-08.

[31] Wikipedia. Pagerank, 2025. Accessed: 2025-04-08.

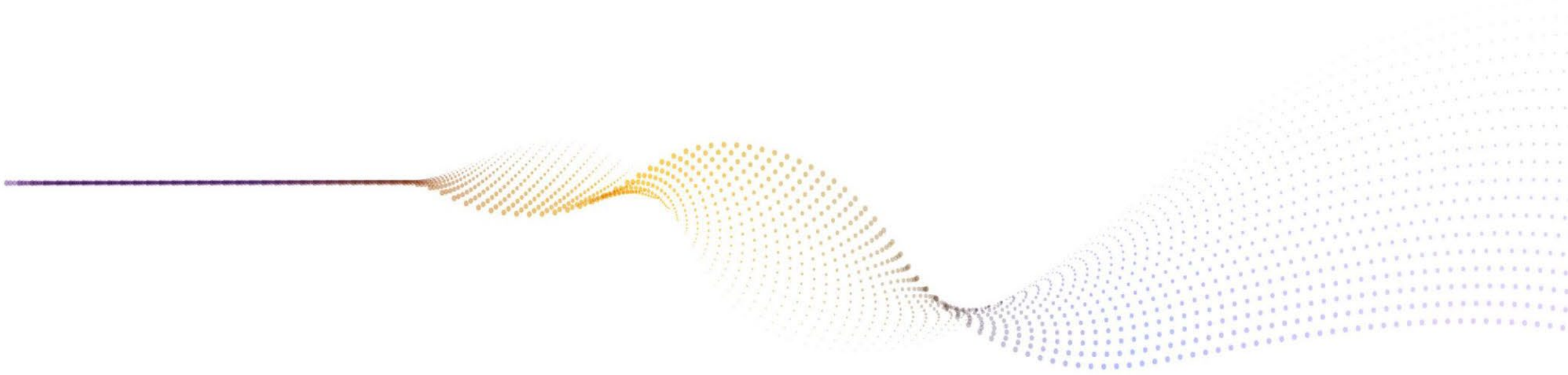
Global Distributed Client-side Cache for DAOS CUG '25, May 4–8, 2025, NYC, NY

[32] Gala Yadgar, Michael Factor, Kai Li, and Assaf Schuster. Management of multilevel, multiclient cache hierarchies with application hints. ACM Trans. Comput. Syst., 29(2):Article 5, 51 pages, May 2011.

[33] Bo Zhang, Philip E. Davis, Nicolas Morales, Zhao Zhang, Keita Teranishi, and Manish Parashar. Optimizing data movement for gpu-based in-situ workflow using gpudirect rdma. In Euro-Par 2023: Parallel Processing: 29th International Conference on Parallel and Distributed Computing, pages 323–338. Springer-Verlag, Berlin, Heidelberg, 2023.

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

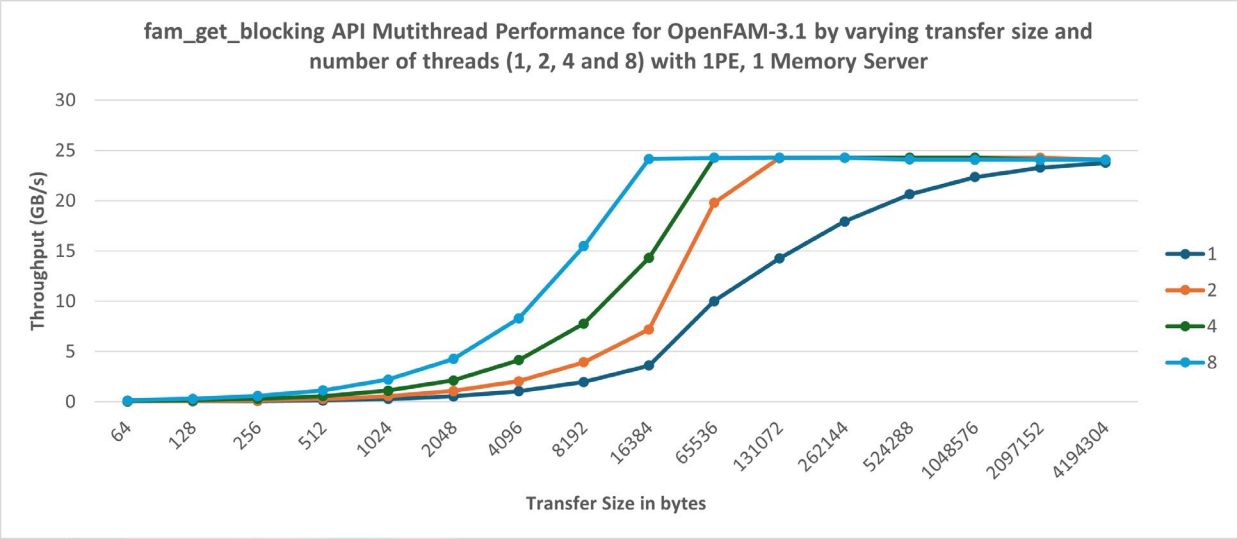
 ©2025 SNIA Developer Conference. All Rights Reserved.

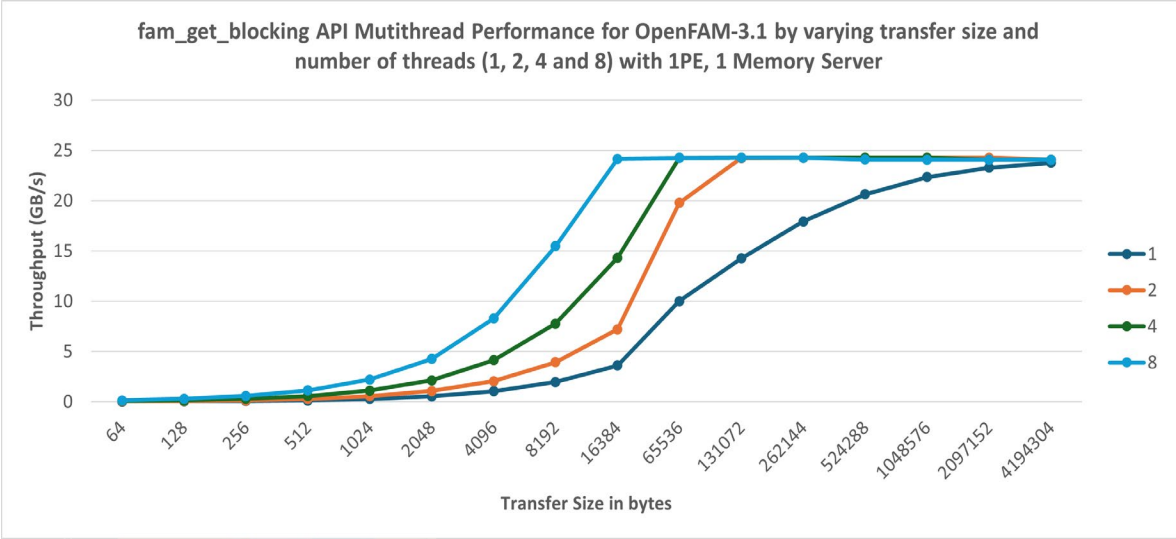
Backup

 ©2025 SNIA Developer Conference. All Rights Reserved.

© 2025 Hewlett Packard Enterprise Development LP

FAM blocking operations





 ©2025 SNIA Developer Conference. All Rights Reserved.

 ©2025 SNIA Developer Conference. All Rights Reserved.

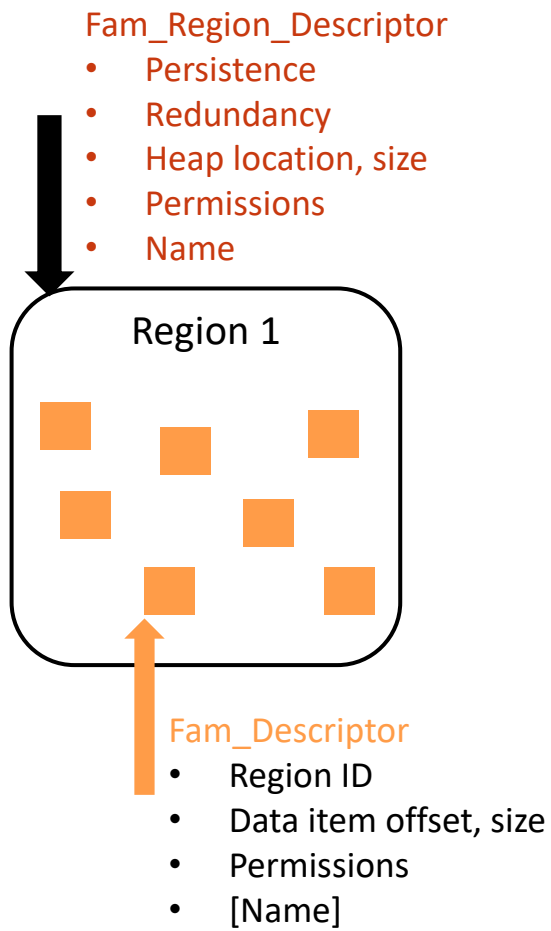
 ©2025 SNIA Developer Conference. All Rights Reserved.

 ©2025 SNIA Developer Conference. All Rights Reserved.

 ©2025 SNIA Developer Conference. All Rights Reserved.

FAM addressing

- FAM is addressed using descriptors
  - Opaque read-only data structures that uniquely identify a location in FAM
  - Descriptors are accessible across OS instances and applications
    - Use base + offset addressing
  - While it is possible to map FAM to a "flat address", descriptors carry additional information



- General API pattern:

```
void fam_get_blocking(void *local, Fam_Descriptor
*descriptor, uint64_t offset, uint64_t nbytes);
```

Local DRAM buffer

FAM data item descriptor

Byte offset within the data item

Number of bytes to transfer

 ©2025 SNIA Developer Conference. All Rights Reserved.

HPE Proprietary