Hewlett Packard
Enterprise

Global Distributed Client-side Cache

Cray User's Group (CUG25) May 2025

Clarete R. Crasta*, John L. Byrne[o], Abhishek Dwaraki[o], David Emberson*, Harumi Kuno[o], Sekwon Lee[o], Ramya Ahobala Rao[o], Shreyas Vinayaka Basri K S[o], Amitha C[o], Chinmay Ghosh[o], Rishi Kesh Kumar Rajak[o], Sriram Ravishankar[o], Porno Shome[o], Lance Evans*

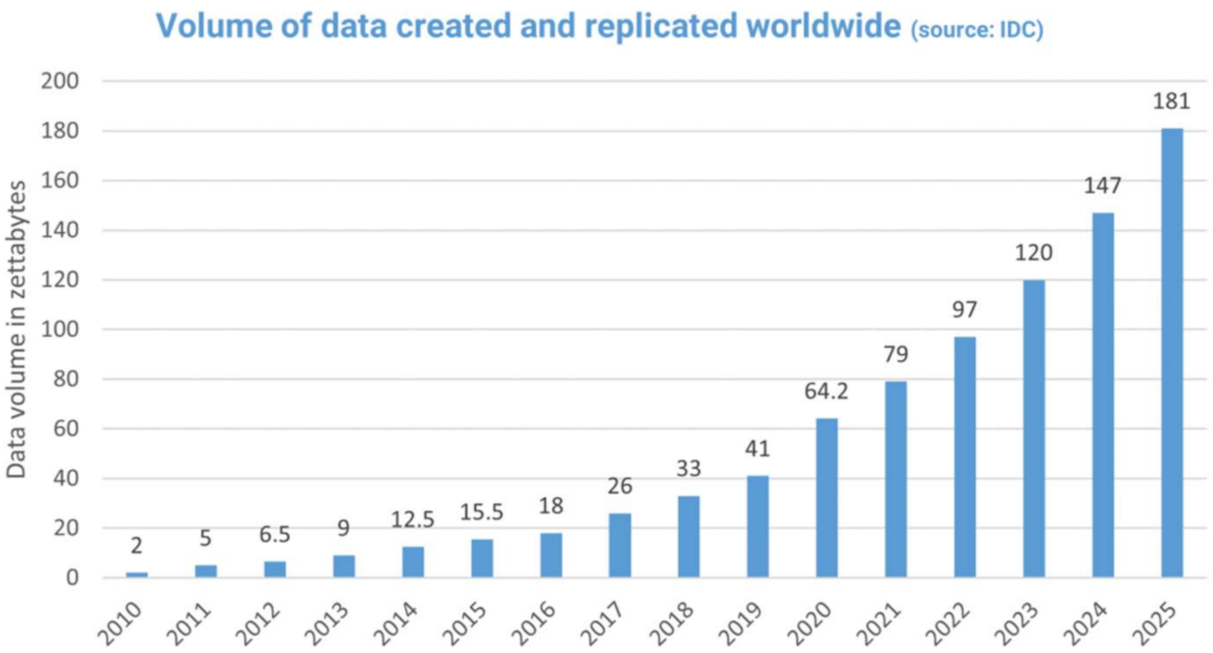
**HPC Business Group, [o] Hewlett Packard Labs*

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.



Outline

- Motivation
- Global client-side cache architecture
- Components of the Global Client-side cache
- Evaluation of the benefits of the cache with PageRank
- Current status and future work
- Summary

Vision and Goal

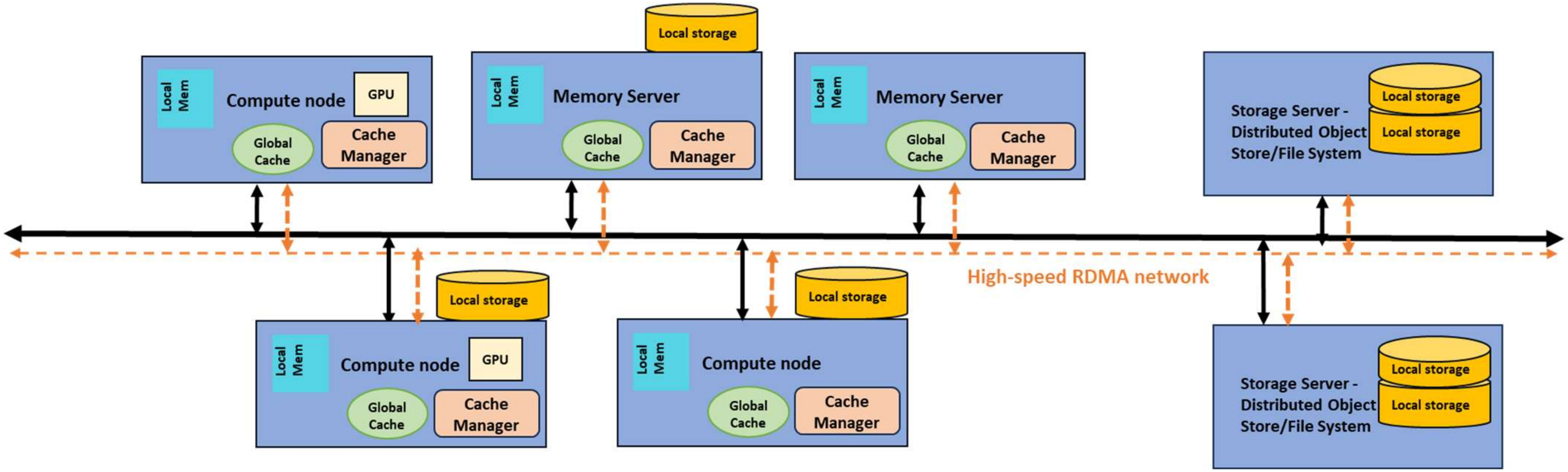
Opportunity with Global Client-Side Caching:

- Leverages higher aggregate client-side resources, such as DRAM and network bandwidth.
- Scales independently of the number of server nodes, offering a more flexible and efficient caching solution.

Our Goal:

- Integrate global client-side caching with Distributed Asynchronous Object Storage (DAOS), a high-performance exascale storage stack recently acquired by HPE and then with other HPC filesystems such as Lustre.

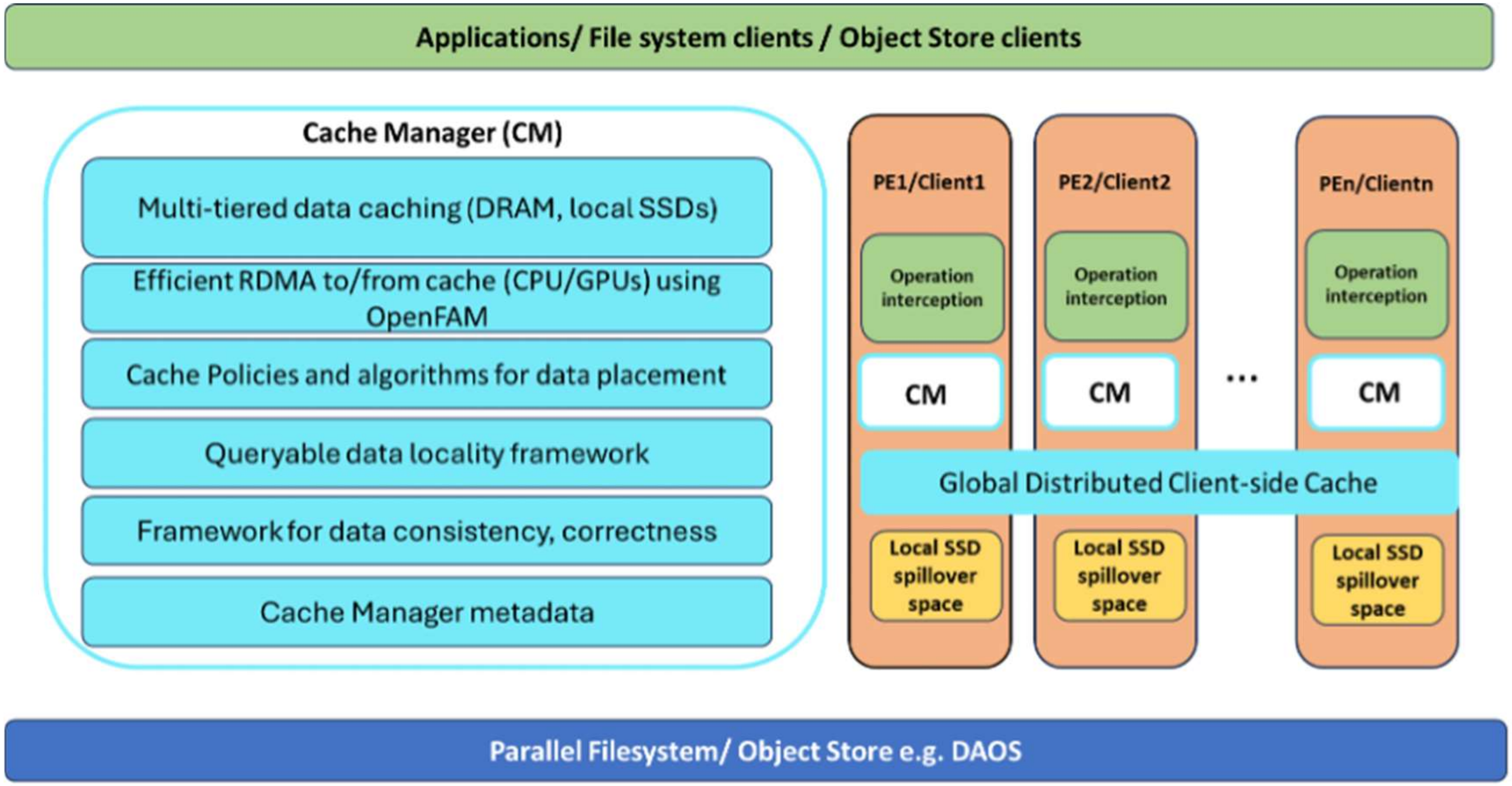
Global client-side cache architecture



- **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.

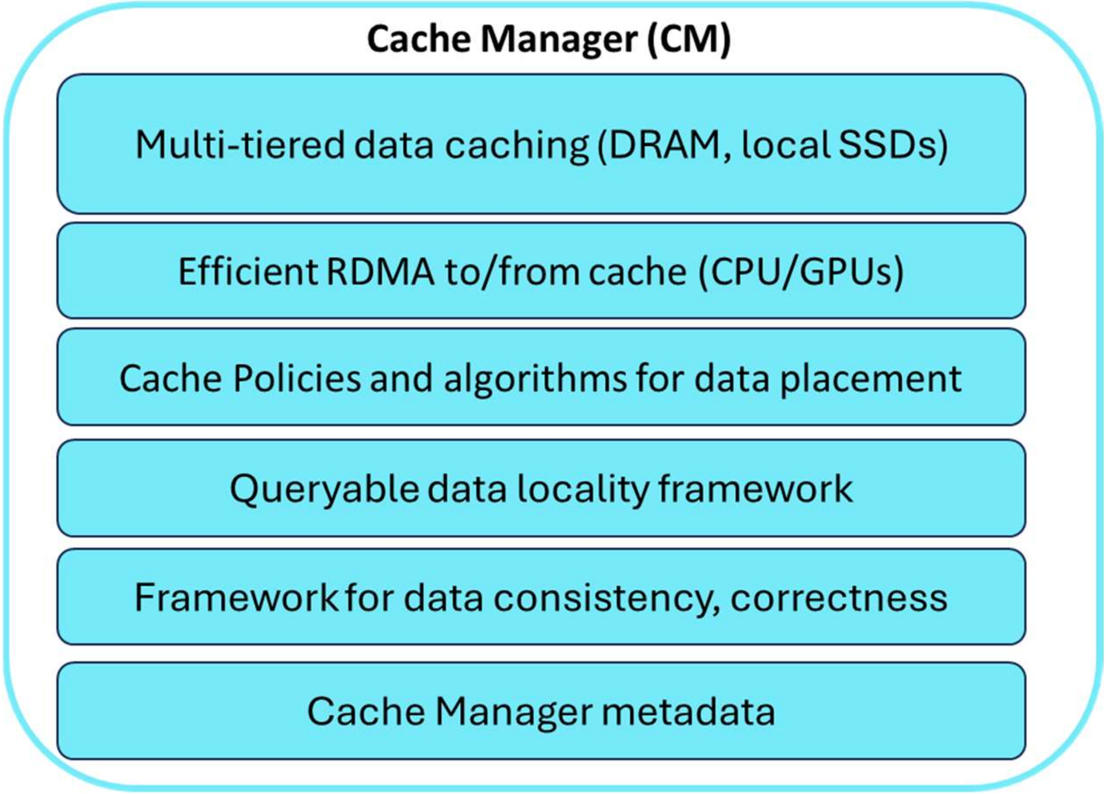
- **Efficient Data Access:**
  - Local SSDs as spillover storage, balancing cost and performance.
  - Data cached locally to nodes with higher probability of access, enables scheduling tasks closer to data.
  - Data moved dynamically located across tiers (DRAM, SSDs, HDDs).
  - RDMA-based high-speed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

Components of the Global Client-side Cache


Applications/ File system clients / Object Store clients
Cache Manager (CM)
Multi-tiered data caching (DRAM, local SSDs)
Efficient RDMA to/from cache (CPU/GPUs) using OpenFAM
Cache Policies and algorithms for data placement
Queryable data locality framework
Framework for data consistency, correctness
Cache Manager metadata
PE1/Client1
PE2/Client2
PEn/Clientn
Operation interception
Operation interception
Operation interception
CM
CM
...
CM
Global Distributed Client-side Cache
Local SSD spillover space
Local SSD spillover space
Local SSD spillover space
Parallel Filesystem/ Object Store e.g. DAOS

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.

Cache Manager


Cache Manager (CM)
Multi-tiered data caching (DRAM, local SSDs)
Efficient RDMA to/from cache (CPU/GPUs)
Cache Policies and algorithms for data placement
Queryable data locality framework
Framework for data consistency, correctness
Cache Manager metadata

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.

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**

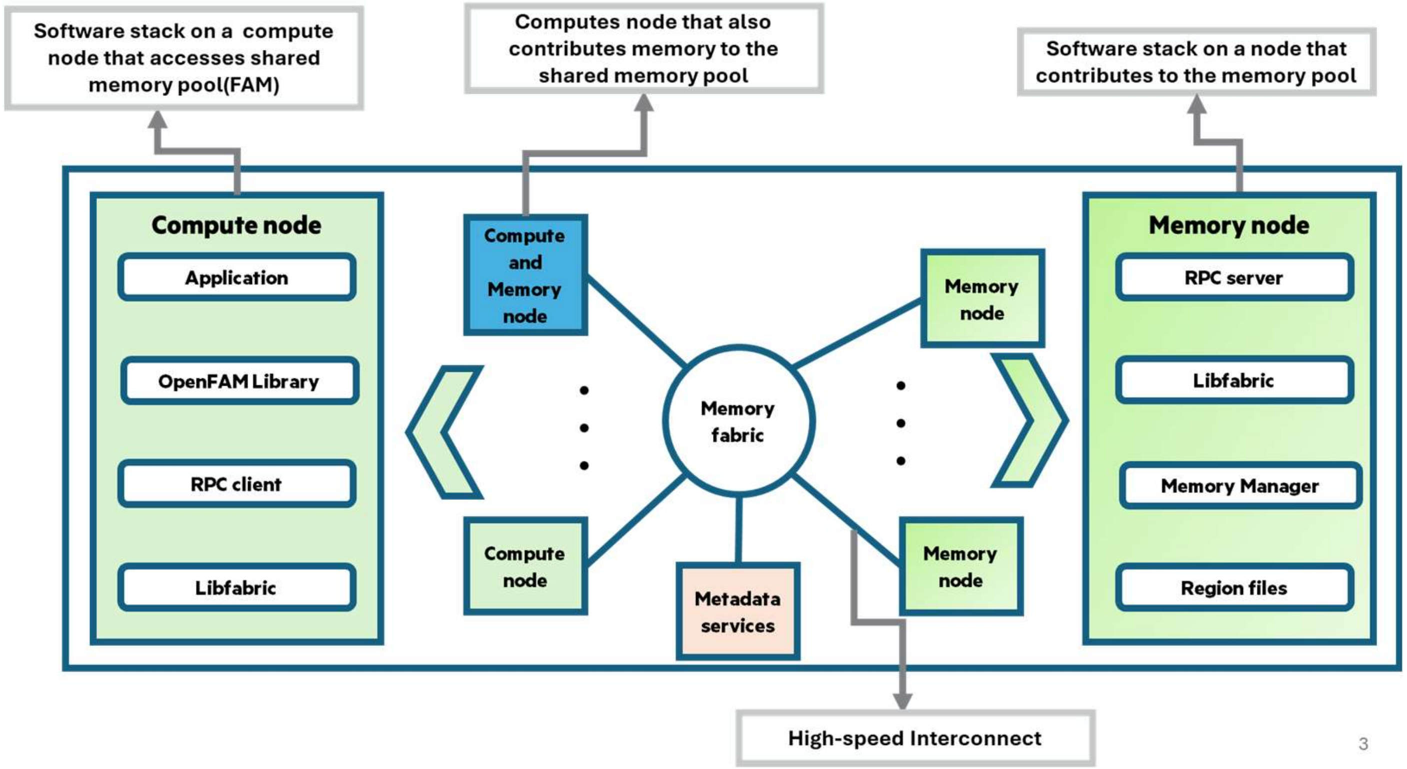
  cm_read, cm_write, cm_preserve
- **Metadata and Locality**

  cm_get_cache_config, cm_get_locality, cm_set_cache_config
- **Directory Management**

  cm_mkdir, cm_rmdir

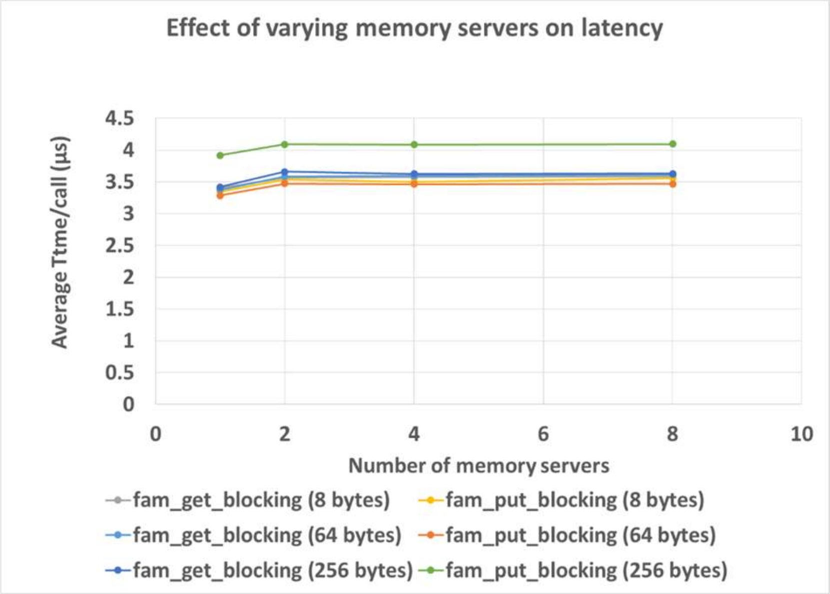
RDMA-Based Framework and Protocol for Global Cache

- **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

OpenFAM performance numbers

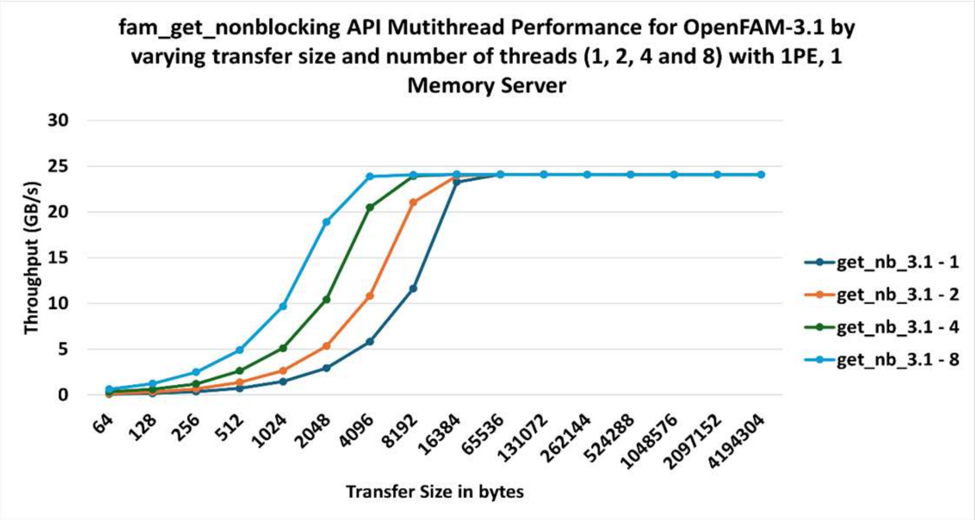


Effect of varying memory servers on latency

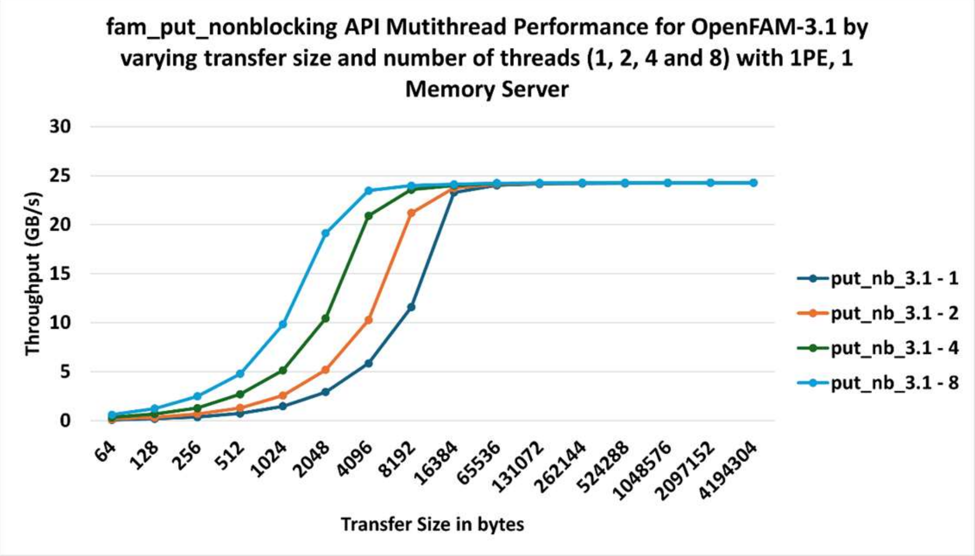
Cluster configuration

- 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.
  - Latency < 5 microseconds for short messages.



fam_get_nonblocking API Mutithread Performance for OpenFAM-3.1 by varying transfer size and number of threads (1, 2, 4 and 8) with 1PE, 1 Memory Server



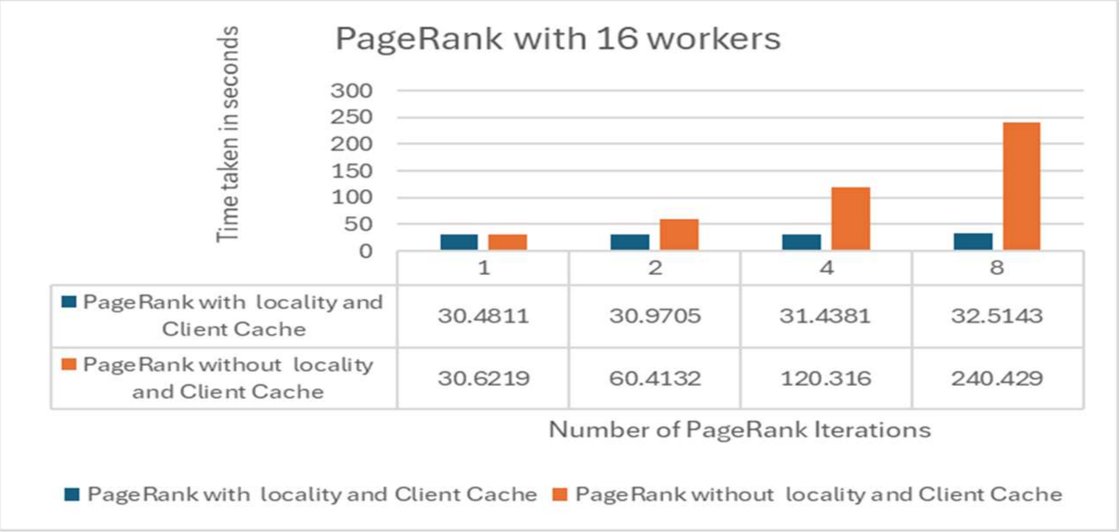
fam_put_nonblocking API Mutithread Performance for OpenFAM-3.1 by varying transfer size and number of threads (1, 2, 4 and 8) with 1PE, 1 Memory Server

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

Experiment Setup:

- Baseline: No client cache; 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.



	1	2	4	8
🟦 PageRank with locality and Client Cache	30.4811	30.9705	31.4381	32.5143
🟧 PageRank without locality and Client Cache	30.6219	60.4132	120.316	240.429

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

Current Status of Global Client-Side Cache Development

Cache Manager:

- Initial Functional model of the cache manager is complete.
- Detailed design specifications and implementation of complete Cache Manager functionality in progress.

Runtime System:

- Building a runtime system that
  - Leverages data locality information from the cache manager.
  - Enables optimal workload scheduling across the cluster.

Collaboration:

- With HPE DAOS team to integrate client-side global cache.
- Evaluating performance using standard DAOS benchmarks.

Use Cases

- Integration of the Global Client-side Cache with IDS (Intelligent Data Store) and other frameworks
- **https://doi.org/10.1002/cpe.7982**

Future Work

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

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

Summary

Proposed Solution:

Global, multi-tiered client-side cache for DAOS.

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.

Acknowledgements

- **Special Thanks**:
  - Pete Haddad, Eric Wu, and Binoy Arnold for their continuous support with hardware and software resources.
  - The HPE DAOS Engineering team, Johann Lombardi, Kevan Rehm and Sherin George for discussions and inputs on DAOS integration and evaluation.
  - HPE Simulation team for their simulation experiments.
  - 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, Cullen Bash, Andrew Wheeler, and Bob Wisniewski for their valuable inputs and support.

Thank you

clarete.riana@hpe.com

References

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

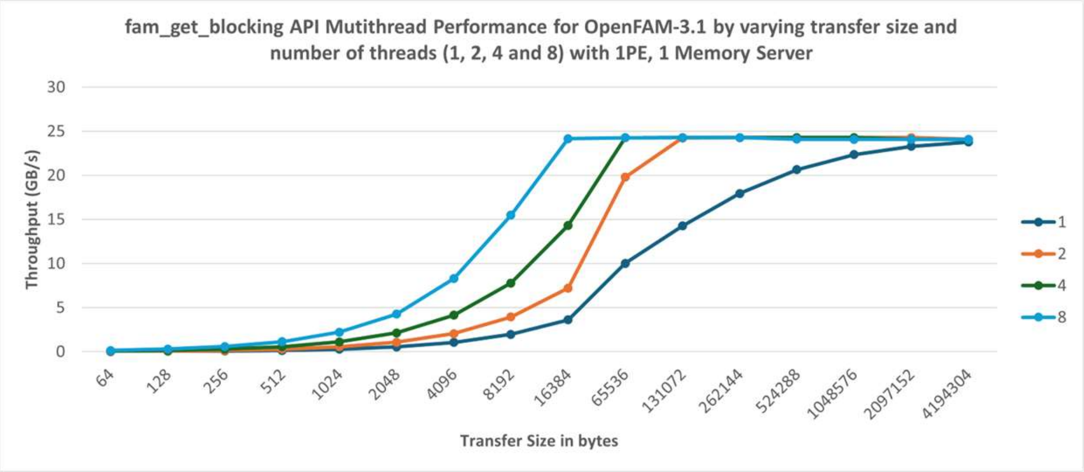
[2] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Distributed afs, 2025. Accessed: 2025-04-08.
[3] Syed Ismail Faizan Barmawer, Gautham Bhat Kumbla, Mashood Abdulla Kodavanji, Clarete Riana Crasta, Sharad Singhal, and Ramya Ahobala Rao. Client allocation of memory across memory servers, 2024. 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.

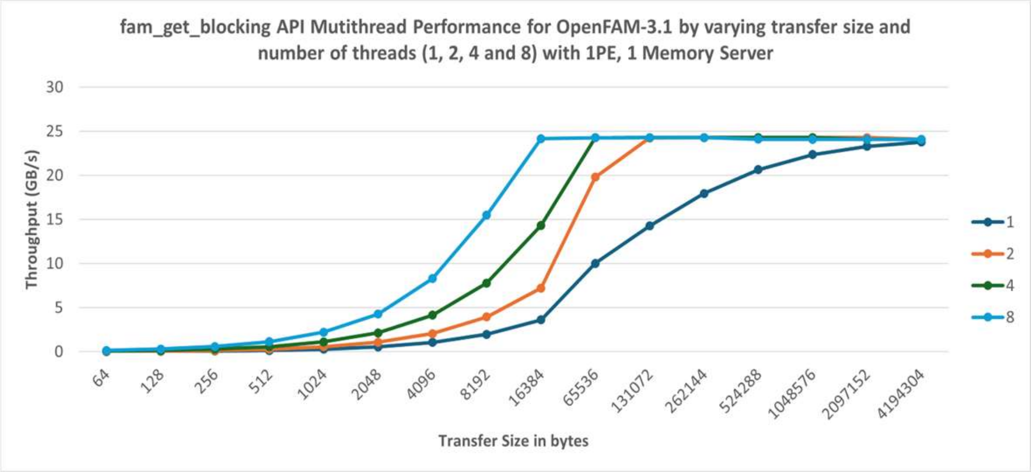
[20] NVIDIA. Rdma architecture overview, 2025. Accessed: 2025-04-08.
[21] OpenSHMEM. Openshmem specification, 2025. Accessed: 2025-04-08.
[22] Yingjin Qian, Xi Li, Shuichi Ihara, Andreas Dilger, Carlos Thomaz, Shilong Wang, Wen Cheng, Chunyan Li, Lingfang Zeng, Fang Wang, Dan Feng, Tim Süß, and André 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.

Backup

FAM blocking operations



fam_get_blocking API Mutithread Performance for OpenFAM-3.1 by varying transfer size and number of threads (1, 2, 4 and 8) with 1PE, 1 Memory Server



fam_get_blocking API Mutithread Performance for OpenFAM-3.1 by varying transfer size and number of threads (1, 2, 4 and 8) with 1PE, 1 Memory Server