XIO: Toward eXplainable I/O for HPC Systems

Sarah Neuwirth*
Johannes Gutenberg University Mainz
Mainz, Germany
neuwirth@uni-mainz.de

Hariharan Devarajan*
Lawrence Livermore National Laboratory
Livermore, CA, USA
hariharandev1@llnl.gov

Chen Wang
Lawrence Livermore National Laboratory
Livermore, CA, USA
wang116@llnl.gov

Jay Lofstead
Sandia National Laboratories
Center for Computing Research
Albuquerque, NM, USA
gflofst@sandia.gov

Abstract

Modern HPC workloads exchange vast data to drive scientific discoveries, relying on diverse storage infrastructures. Current monitoring tools only capture the I/O interfaces, which is insufficient to explain suboptimal I/O performance, as the bottleneck can be within one or more layers of the parallel I/O stack.

To address this, we analyze HPC ranking lists to highlight storage diversity and build the Master Architectural Plan (MAP) that enables identification of current gaps in data paths in monitoring tools. Finally, we present DataCrumbs, a lightweight tool that fills these gaps by examining the user and kernel I/O stack to uncover causal relationships to achieve eXplainable I/O (XIO).

CCS Concepts

• **General and reference** → **Performance**; **Measurement**; **Evaluation**; • **Software and its engineering** → **Software infrastructure**; • **Information systems** → **Storage management**.

Keywords

Explainable I/O, DataCrumbs, HPC Storage Performance, I/O Monitoring and Profiling, Kernel-level Profiling, HPC System Monitoring

ACM Reference Format:
Sarah Neuwirth, Hariharan Devarajan, Chen Wang, and Jay Lofstead. 2025. XIO: Toward eXplainable I/O for HPC Systems. In *The International Conference on Scalable Scientific Data Management 2025 (SSDBM 2025), June 23–25, 2025, Columbus, OH, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3733723.3733741

1 Introduction and Motivation

Modern high-performance computing (HPC) workloads are increasingly data-intensive, requiring efficient storage architectures and interconnects to support large-scale simulations, AI-driven analytics, and scientific data processing. Workflows such as the Multiscale Machine-Learned Modeling Infrastructure (MuMMI) [25] rely on heterogeneous computing elements–leveraging GPUs for large-scale simulations and CPUs for post-processing analysis–while using shared memory, node-local burst buffers, and parallel file system for data exchange. Optimizing data movement in such workflows is critical to enhancing simulation fidelity, resolution, and execution efficiency. Despite advances in HPC storage solutions, a fundamental challenge remains: understanding and optimizing component contributions from within the I/O stack to I/O performance at scale.

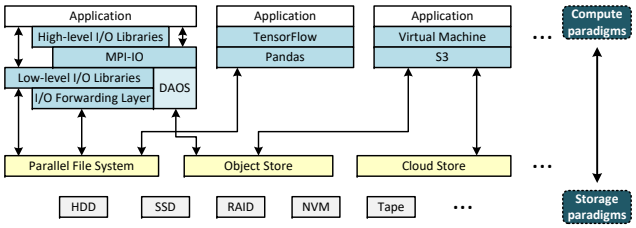


Figure 1: Complexity of hardware / software storage stacks.

Background and Related Work: Current HPC monitoring tools provide fragmented insights that often do not capture the full data flow across compute and storage layers. Application-level profilers (e.g., Darshan [12], Recorder [37] and Score-P [19]) track high-level I/O calls but overlook kernel-level interactions that influence performance. System-level tools (e.g., LDMS [9], DCDB [26] and TACC-Stats [17]) aggregate hardware metrics but lack fine-grained visibility into workload-specific I/O behavior. Although these tools are essential for optimizing scientific workloads in HPC systems [28, 34], they generally focus on the *what* of workload performance, identifying potential bottlenecks, but failing to explain *why*. This critical gap in I/O explainability makes it difficult to diagnose cache inefficiencies, storage bottlenecks, and interconnect-related slowdowns.

The complexity in providing explanations results from the multi-layered nature of modern HPC storage stacks, as shown in Figure 1. At higher levels, applications interact with the storage via specialized I/O libraries (e.g. HDF5 [16]) or frameworks (e.g. TensorFlow [8]) that translate requests into lower-level system calls. Middleware (e.g. MPI-IO) and optimization layers (e.g. I/O forwarding) introduce additional abstractions before the data reaches the underlying storage subsystems, which include parallel file systems, object storage and tiered storage hierarchies that include NVMe SSDs, HDDs and tape archives. The complex dependencies between these layers contribute to unpredictable performance behavior.

*Authors contributed equally to this work.

This work is licensed under a Creative Commons Attribution International 4.0 License.

SSDBM 2025, Columbus, OH, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1462-7/25/06
https://doi.org/10.1145/3733723.3733741

Several frameworks have been proposed for holistic I/O performance analysis. TOKIO [22], for example, relies on a combination of front-end tools (Darshan, Recorder) and back-end tools (LMT), while MAWA-HPC (Modular and Automated Workload Analysis for HPC Systems) [39] aims to develop a generic workflow and tooling suite that can be transparently applied to applications and workloads. Due to its modular design, the suite is able to support various community tools, which increases its compatibility with different applications and HPC systems. Our work complements these approaches by providing a better understanding of the variability of I/O performance and its causes.

Contributions: We describe a systematic approach to understanding I/O behavior in HPC systems. First, we analyze HPC ranking lists to identify trends in storage, interconnects, and compute elements. Second, we introduce the Master Architectural Plan (MAP), a generalized framework for modeling HPC architectures that offers a structured view of storage tiers, device types, interconnects, and compute elements for performance analysis (Figure 2). Third, we assess existing monitoring tools, highlighting gaps in storage visibility, compute-storage interaction tracking, and multi-layer performance modeling. Finally, we propose DataCrumbs, a lightweight profiling tool that enables fine-grained, cross-layer I/O tracing, bridging application-level profiling with kernel-space monitoring.

2 Methodology

This study aims to improve understanding and transparency of I/O performance in HPC systems. We employ a four-step methodology.

1) Survey of HPC Ranking Lists: To understand the architectural evolution of HPC systems, we analyze popular ranking lists: TOP500 [3] (compute-focused), IO500 [2] (I/O performance), and the Comprehensive Data Center List (CDCL) [4]. These lists should provide insights into storage architectures, interconnects, and computing trends. The Graph500 list [1] was considered but excluded due to its focus on graph analytics rather than storage infrastructure. From our analysis, we identify three key research questions: (1) *What are the key computing, storage, and networking features in modern HPC ecosystems?*, (2) *What trends and emerging technologies are shaping future HPC architectures?*, and (3) *What are the main limitations in current HPC infrastructure data collection?*

2) Developing the Master Architectural Plan: Understanding data movement across HPC components is crucial for explainable I/O. To systematically represent storage and compute interactions, we develop the ***Master Architectural Plan*** (MAP), a comprehensive framework capturing computing elements, storage tiers, interconnects, and device types. MAP integrates findings from the surveyed lists and aligns system components across different HPC generations. It serves as a blueprint for identifying data trails and providing an overview of data paths in modern HPC systems. Furthermore, MAP serves as a baseline for evaluating existing monitoring tools and identifying missing coverage areas.

3) Assessment of I/O Monitoring Tools: A major limitation in understanding HPC I/O behavior is the fragmentation of monitoring tools. Most I/O tools focus on recording ***what*** happened without explaining ***why*** it happened, resulting in opaque performance data that complicates debugging and optimization. We evaluate tools across three categories: (1) *application-level monitoring*, (2) *system-level monitoring*, and (3) *end-to-end monitoring*. We claim that a critical shortfall in current tools is their inability to correlate I/O operations with their underlying causes (i.e., the why), making it challenging to model I/O performance in modern HPC systems. These systems exhibit significant variability due to jitter and complex interactions between the intricate software and hardware stack of I/O subsystems. Based on our analysis, we propose ***DataCrumbs***, a lightweight tool that provides a comprehensive view of the I/O stack and thus has the potential to close the gap.

Table 1: Features of different lists and their coverage.

		TOP500	IO500	CDCL
Compute Nodes	**Processor**	✓	✓	✓
	Network	✓	✓	✓
	Storage Device	✗	[✓]	[✓]
	Storage Controller	✗	[✓]	[✓]
	Storage Software	✗	[✓]	[✓]
Storage Nodes	**Processor**	✗	✗	[✓]
	Network	✗	✗	[✓]
	Storage Device	✗	[✓]	[✓]
	Storage Controller	✗	[✓]	[✓]
	Storage Software	✗	✓	[✓]
Topology		✗	✗	[✓]

4) Case Study: Toward Explainable I/O: To illustrate the limitations of existing tools and highlight the benefits of explainable I/O, we conduct a case study using the IOR benchmark [23] with the ***DataCrumbs*** tool. Unlike other monitoring tools, DataCrumbs employs the extended Berkeley Packet Filter (eBPF) technology to capture multi-stack profiling, including user-space, middleware, and kernel-level interactions. We demonstrate the capabilities of DataCrumbs and examine the impact of buffered vs. unbuffered I/O operations on kernel performance, identify key bottlenecks related to OS page caching and buffer management, and demonstrate how fine-grained kernel instrumentation improves I/O performance analysis. The results show how deeper monitoring capabilities can drive optimizations in scientific computing workflows.

3 Preliminary Results and Findings

Next, we apply the methodology outlined in Section 2 resulting in an abstract architectural diagram, position current monitoring tools, and introduce our tool for capturing these data paths.

3.1 Insights from HPC Ranking Lists

To evaluate the evolving landscape of HPC systems, we analyzed three major ranking lists: TOP500, IO500, and CDCL. Table 1 provides an overview of the information captured by these lists, highlighting whether specific features are included (✓), omitted (✗), or optional ([✓]). Although these lists provide valuable insights into HPC architectures, they vary in scope, coverage, and completeness, leading to critical gaps in understanding system performance and storage infrastructure. The lists reviewed reflect an ongoing evolution in HPC system design driven by advances in hardware and software. Three main trends can be identified:

1) Shift from CPU-Centric to Heterogeneous Architectures: HPC systems are transitioning from traditional multi-core CPU architectures to heterogeneous computing models integrating GPUs, FPGAs, and domain-specific accelerators [27, 30]. The TOP500 list highlights this shift, showing that 182 out of the 500 top systems

employ GPU acceleration, with 32 different GPU models in use. The rise of DPUs (Data Processing Units) and in-network computing further diversifies system architectures, optimizing workloads that require high-throughput I/O and complex data processing.

2) Convergence of Compute and Storage Interconnects: Historically, compute and storage interconnects have been distinct and separately optimized, but recent trends point to shared high-speed interconnects [35]. The emergence of technologies such as NVMe-over-Fabrics (NVMeOF), RDMA [32], and SmartNICs [18] is blurring the boundary between computation and storage, enabling more efficient data movement across the machine.

3) Rise of Storage and Networking Optimizations: The IO500 and CDCL lists reveal a growing emphasis on optimized storage architectures, including node-local burst buffers, in-memory caching, and object storage solutions. While Lustre, Spectrum Scale, BeeGFS, and DAOS remain dominant, newer solutions like UnifyFS [11], Hermes [20], Univistor [38], and GekkoFS [36] are gaining traction due to their performance benefits in domain-specific workloads. Additionally, smart interconnects and in-network computation technologies, such as CXL switches and GPU Direct Storage, are becoming more prevalent, reducing I/O bottlenecks. Our investigation included 441 unique HPC systems at 252 sites in 35 countries, derived from over 1,500 entries. These systems feature 26 processor types, 44 interconnects, and 44 storage configurations. While these ranking lists offer valuable insights, they *fail to provide a comprehensive picture of HPC storage infrastructures* due to several challenges.

First, the TOP500 list focuses primarily on compute resources, often omitting critical details about storage and interconnect architectures. IO500 provides detailed but incomplete storage insights—some entries specify storage controllers, interconnects, and device types, while others only report aggregate performance metrics. Second, IO500 entries frequently use non-standardized file system labels (e.g., vague terms like *Parallel File System* or *SSD*), making it difficult to interpret system configurations. The lack of a common storage architectures classification scheme leads to ambiguities, particularly in differentiating between persistent storage systems and volatile storage. Finally, storage connectivity details are largely absent, even though high-speed storage interconnects (e.g., PCIe, NVMeOF, InfiniBand) play a crucial role in I/O performance. Only 19 of 500 TOP500 systems provide network details, limiting the ability to analyze end-to-end data movement.

3.2 Refining the Master Architectural Plan

The ***Master Architectural Plan (MAP)*** is a comprehensive framework intended to map the evolution of HPC storage architectures for past, current and potential future designs. The main goal of MAP is to provide a standardized yet adaptable concept for researchers to tune software and monitoring tools for different HPC configurations. By abstracting commonalities in system design, MAP facilitates a structured approach to evaluating storage tiers, computing elements, interconnects, and data management strategies.

3.2.1 ***HPC Ecosystem Features:*** HPC architectures integrate a variety of hardware components, each of which influences overall system efficiency. MAP formalizes this complexity by classifying it along four key dimensions: (1) *computing elements*, (2) *storage tiers*, (3) *storage devices*, and (4) *interconnects*.

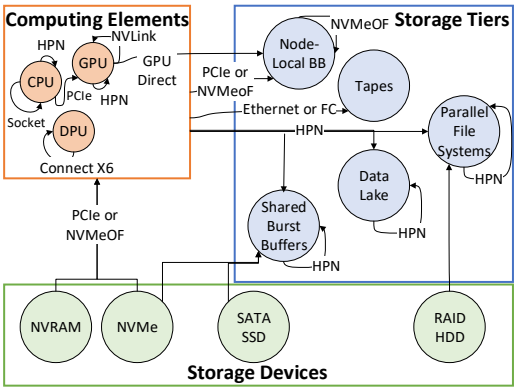


Figure 2: Overview of the graph visualization with MAP.

Traditional HPC systems were mainly based on multicore CPUs, while modern systems include GPUs, many-core architectures, and specialized accelerators such as Data Processing Units (DPUs) [7, 13]. These components are connected via high-bandwidth PCIe, NVLink [6] and InfiniBand (IB) networks to optimize throughput. In recent years, HPC systems have evolved from global parallel file systems (e.g., Lustre, Spectrum Scale) to multi-tier storage hierarchies that include node-local burst buffers, shared NVMe-based storage, and archival tiers (HDDs, tapes). The IO500 dataset reveals that DAOS (10%), Lustre (9%), Spectrum Scale (3%) and BeeGFS (3%) have a large share of HPC installation (out of 32 unique file systems). This indicates an emerging trend towards high-speed and configurable storage solutions. MAP categorizes storage devices based on their interface type (e.g., SATA SSD, NVMe, NVRAM) and performance characteristics. Although many ranking lists include aggregate storage capacity, only the IO500 list and to some extent the CDCL provide detailed device-specific insights, such as internal interconnect topologies and metadata separation. Finally, modern HPC workloads require fast and scalable inter-node and storage interconnects. While the TOP500 list focuses on compute node interconnects, the IO500 provides limited but important insights into storage fabric configurations. High-performance networks (HPN) such as RDMA over Converged Ethernet (RoCE), InfiniBand and NVMe-over-Fabrics are increasingly being used to accelerate data exchange across hierarchical storage layers.

MAP serves as *graph-based representation* of these components and their connections, as shown in Figure 2, enabling researchers to *model data flows within HPC systems.*

3.2.2 ***Emerging Trends in HPC Architectures:*** The evolution of HPC architectures shows notable trends in storage and compute integration. Traditionally, compute nodes and storage systems were physically separated to avoid storage-centric compute node failures, but new architectures are co-locating storage within compute nodes to minimize off-node storage access. This is evident in Frontier (ORNL) and El Capitan (LLNL), where local NVMe and high-speed RAM are tightly coupled with processing elements to minimize data movement overhead. In addition, the rise of software-defined storage (SDS), AI-driven caching strategies, and hierarchical burst buffers is transforming HPC workflows. The increasing use of shared burst buffers and NVMe-based tiering improves access latency and enables dynamic workload-driven storage orchestration.

HPC interconnects have evolved from dedicated fiber/Ethernet links to software-defined network fabrics that dynamically optimize data paths based on real-time I/O workloads. This shift has driven the adoption of RDMA-enabled fabrics, NVMe-over-Fabrics [21], and GPU Direct Storage [31]. Finally, with exascale computing, systems are now designed for hybrid workloads that include traditional simulations, AI-driven processing, and in-situ data analytics. As a result, cross-layer data movement optimization is becoming an important research focus. These trends demonstrate a *paradigm shift* in HPC system design, where storage and compute infrastructures are becoming increasingly *integrated and performance-aware*.

3.2.3 ***Challenges in HPC System Analysis and Validation:*** While MAP provides a structured model, its validation is challenging due to data inconsistencies across HPC ranking lists. IO500, while the most comprehensive for storage analysis, suffers from inconsistent data collection practices, making it difficult to gain detailed insights on storage controllers, interconnects, and tiering mechanisms. While compute interconnects (e.g., Slingshot, NVLink, and PCIe) are well documented, storage-specific interconnect details (e.g., between NVMe SSDs and controllers, between nodes and parallel file systems) are often omitted in existing datasets. Only four IO500 systems overlap with the TOP500, limiting the ability to validate performance metrics across storage and compute tiers. In addition, HPC production systems often omit specific hardware configurations, making it difficult to track the behavior of real workloads across different storage architectures.

To address these gaps, we manually examined major HPC systems and validated MAP on Tier-0/1 exascale platforms (Frontier, El Capitan), Tier-2 HPC systems (Polaris), and tier-3 testbed environments (Wombat at ORNL). These case studies confirmed that MAP accurately models interconnect diversity, storage tiering, and emerging trends in workload-driven I/O performance optimization.

3.3 I/O Monitoring Infrastructure Assessment

Understanding I/O performance in HPC systems requires a comprehensive monitoring infrastructure capable of capturing data across different hardware and software layers. Although there are various monitoring tools, they mainly focus on isolated components, leaving critical gaps in holistic system analysis. This section reviews the landscape of popular HPC monitoring tools.

1) Gaps in Existing HPC Monitoring Tools: Current HPC monitoring tools can be categorized into three broad groups. First, *application-level monitoring tools* such as Darshan [12], TAU [33], and Recorder [37] capture high-level I/O calls at the application layer, providing insights into workload behavior. However, they lack visibility into the underlying storage system interactions, kernel buffering mechanisms, and network data flows, limiting their ability to explain performance bottlenecks. Second, *system-level monitoring tools* like LDMS [9], DCDB [26], and Pika [15] focus on global system metrics, including hardware health, I/O throughput, and resource utilization. While useful for trend analysis, these tools provide coarse-grained data and often lack per-application breakdowns, making it difficult to correlate I/O performance to specific workloads. Third, *end-to-end monitoring frameworks* such as Ganglia [24], Nagios [10], and Apollo [29] attempt to bridge the gap between application and system monitoring by aggregating data across different layers. However, they still lack fine-grained insights into storage interconnects, kernel stack interactions, and network data flows, preventing full I/O performance explainability.

A main limitation of these tools is their *inability to integrate storage, data processing, and network monitoring into a unified view*, as reflected by MAP. Existing solutions work in silos, resulting in fragmented insights that complicate troubleshooting and optimization.

2) Challenges in HPC Monitoring Tools: Several challenges hinder effective HPC monitoring. First, many HPC sites do not fully disclose their monitoring strategies, thus making it difficult to assess gaps in their tool infrastructure. While sites like NERSC reference Darshan, monitoring practices for systems like Frontier at ORNL remain largely undocumented [5]. Second, different monitoring tools apply inconsistent data collection mechanisms, making cross-system comparisons difficult. The lack of a standardized approach to tracking I/O timing, compute-storage interactions or network traffic related to data movement. Third, while many monitoring tools capture what is happening at the application or system level, they are unable to explain why it is happening. The reason for this is the lack of instrumentation at the kernel and interconnect layers, where critical performance bottlenecks often occur. Finally, many tools are developed as R&D projects with a limited budget, which causes them to stall or not be further pursued. As a result, the long-term evolution of tools and adaptation to modern HPC architectures remains a challenge.

3) Toward a Comprehensive Monitoring Framework: Our findings in applying MAP reveal a substantial shortcoming in current monitoring infrastructures. Most tools are designed for specific use cases rather than providing a holistic view of system behavior. To address this, the following actions are required. First, a unified monitoring approach needs to close the gap between workload execution, storage interactions, and network data flow analysis. Second, advanced tools should address the gap between application-level profiling, kernel-space instrumentation, and system-wide performance metrics. Solutions such as eBPF-based tracing offer promising ways to accomplish this. Finally, a consistent methodology for collecting and correlating I/O performance data across different HPC sites is critical for meaningful cross-platform comparisons.

By addressing these challenges, *a more transparent, efficient, and explainable I/O monitoring ecosystem* can be designed that supports both system administrators and end users optimize performance at scale, a goal that our proposed DataCrumbs tool directly tackles.

3.4 Toward Explainable I/O with DataCrumbs

Achieving ***eXplainable I/O (XIO)*** in HPC systems requires deep visibility into the entire I/O stack, from user applications to kernel-level interactions. Traditional monitoring tools focus on high-level I/O calls, but fail to identify low-level system behavior that directly affects performance. We introduce ***DataCrumbs*** [14], a profiling tool that provides multi-layered I/O tracing by leveraging the extended Berkeley Packet Filter (eBPF) technology. Our case study demonstrates how DataCrumbs enables fine-grained I/O performance analysis, uncovering causal relationships between application behavior, system calls, and kernel-level storage interactions.

3.4.1 ***DataCrumbs: Low-Overhead Multi-Layer Profiling.*** HPC applications can run millions of I/O operations, interacting

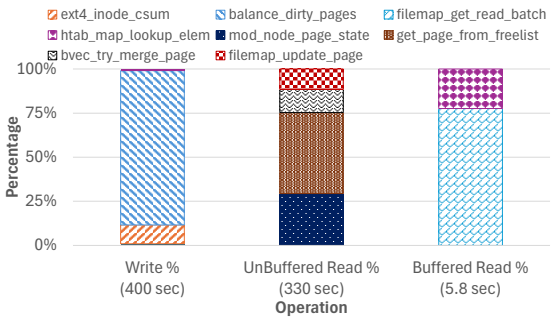


Figure 3: IOR performance breakdown with DataCrumbs.

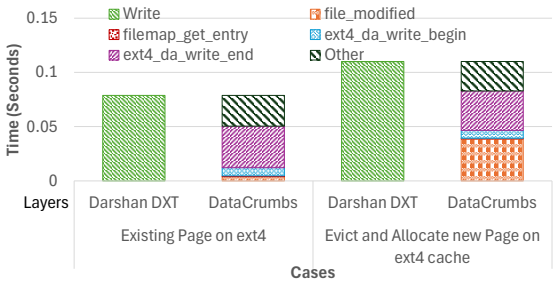


Figure 4: IOR performance breakdown with DataCrumbs and Darshan DXT for ext4 cache hit versus miss.

with multiple software layers, including user-space libraries (e.g., HDF5, MPI-IO, POSIX), middleware and file system layers (e.g., VFS, EXT4, OS cache), and Kernel stack components (e.g., buffer management, block layer, storage drivers). Existing tools like Darshan and Recorder capture user-space I/O calls but overlook kernel-level behavior that affects data movement, caching, and storage access latency. *This lack of visibility makes it difficult to explain I/O slowdowns, performance inconsistencies, and inefficiencies in HPC workflows.*

DataCrumbs enhances I/O explainability by capturing system-wide profiling data across multiple software layers with minimal overhead. This is achieved by (1) attaching lightweight eBPF probes to user applications, middleware, and kernel components for transparent monitoring, (2) sampling aggregated kernel data at configurable intervals to reveal I/O patterns, buffer states, and system interactions, and (3) correlating I/O function calls with kernel stack events to identify bottlenecks such as page cache inefficiencies, metadata overhead, and system call latencies.

This multi-layered approach enables the reconstruction of a detailed execution timeline that illustrates *how I/O operations propagate through the system and software stack.*

3.4.2 ***Case Study: Profiling the IOR Benchmark.*** To demonstrate DataCrumbs' explainability capabilities, we conducted two experiments using the IOR benchmark, a widely used HPC I/O benchmarking tool that measures performance under various I/O patterns, including sequential and random access. The experiments were run on a Chameleon Cloud instance with Intel Skylake bare-metal, 128 GB of memory, and 12 processes on a single node. DataCrumbs was run as a background service on the same node.

First, we ran IOR with a transfer size of 4 MB and a block size of 8 GB using a sequential file-per-process access pattern. The results, shown in Figure 3 for a single process, include three scenarios: (1) write with POSIX I/O, (2) buffered read with POSIX I/O, and (3) unbuffered I/O with POSIX I/O. We ensured that all caches were cleared for the unbuffered read. The runtimes were vastly different: write took 400 seconds, unbuffered read 330 seconds, and buffered read 5.8 seconds. The x-axis represents different operations, while the y-axis shows the percentage of time spent on each kernel call. As more than 12K calls were traced, only those accounting for more than 5% of the total time are shown. Substantial performance variations were observed for the different I/O scenarios. Write operations using POSIX I/O consumed 82% of execution time due to OS page cache management, highlighting the impact of dirty page flushing. Unbuffered read operations had a high overhead due to frequent page cache misses, with dominant kernel calls such as `mod_node_page_state` and `get_page_from_freelist`. In contrast, buffered reads relied on various kernel calls, such as `filemap_get_read_batch` and `htab_map_lookup_elem`, reflecting the differences in the handling of buffered and unbuffered reads.

Our second experiment compares DataCrumbs with the widely used Darshan DXT tool in two buffered write scenarios: (1) writing to an existing page in the ext4 page cache and (2) writes that trigger a swap and reallocation from the ext4 page cache for new data. We ran IOR with a transfer size of 1 MB and a block size of 8 GB using a sequential file-per-process access pattern. In this test, the total I/O is more than 50%, so the OS cache is evicted for the reported operations. Figure 4 shows a performance breakdown for a single process as observed by Darshan DXT and DataCrumbs. On the left, the write is stored in the ext4 page cache without being immediately written to disk. On the right, the write triggers the flushing of dirty inodes in ext4, followed by reallocation for new writes. This second scenario occurs due to exausting of existing inode pages in ext4. The results indicate that write costs within the ext4 layer remain consistent across both cases. However, in the second scenario, most of the additional time is spent on eviction and reallocation within the buffering layer before the write operation can continue.

The *breakdown of kernel function contributions* in Figures 3 and 4 highlights how *different I/O operations follow distinct execution paths*, leading to *highly variable performance outcomes*, a variability that becomes even more pronounced in HPC environments. With only 1.3% of overhead, DataCrumbs provides valuable insights into the root causes of I/O bottlenecks and variability, enabling more accurate I/O performance modeling, incorporating both user- and kernel-level factors, optimized workload execution, allowing scientists to fine-tune file access patterns to reduce cache inefficiencies, and better-informed HPC system design, as architects gain a clearer understanding of how I/O traffic interacts with storage subsystems. DataCrumbs uses kernel-space buffering, decouples event capture and storing, and minimizes probing functions to keep the overhead minimal with respect to other tools. Unlike traditional profiling tools, DataCrumbs exposes granular interactions within the I/O stack, fostering a *new level of explainability in HPC storage analysis.*

4 Conclusion

Modern HPC storage infrastructures are complex, involving multiple computing elements, diverse storage devices, intricate software abstractions, and high-performance network links. Researchers face

three main challenges with current performance lists and monitoring solutions: they fail to capture interconnects between computing elements and storage tiers, offer limited visibility into the I/O software stack at the kernel and network levels, and often include optional fields that result in sparse storage details. To address these issues, we introduce DataCrumbs, a tool designed to enhance visibility into both user and kernel I/O stacks, facilitating better understanding of I/O performance on HPC compute nodes. This study underscores the need for a holistic view of modern HPC software and hardware to achieve explainable I/O for large-scale HPC systems.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 24-SI-005 (LLNL-CONF-872903).

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] 2024. Graph500. https://graph500.org
[2] 2024. IO500 List. https://io500.org/releases
[3] 2024. TOP500 List. https://top500.org/lists/top500
[4] 2025. Comprehensive Data Center List (CDCL). https://www.vi4io.org/hpsl/
[5] 2025. Frontier User Guide — OLCF User Documentation. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#system-updates
[6] 2025. NVIDIA Mellanox ConnectX-6 SmartNIC Adapter. https://www.nvidia.com/en-sg/networking/ethernet/connectx-6/
[7] 2025. NVIDIA NVLink: High-Speed GPU Interconnect. https://www.nvidia.com/en-us/design-visualization/nvlink-bridges/
[8] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. 265–283.
[9] Anthony Agelastos, Benjamin Allan, Jim Brandt, Paul Cassella, Jeremy Enos, Joshi Fullop, Ann Gentile, Steve Monk, Nichamon Naksinehaboon, Jeff Ogden, et al. 2014. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*.
[10] Wolfgang Barth. 2008. *Nagios: System and Network Monitoring*. No Starch Press.
[11] Michael J Brim, Adam T Moody, Seung-Hwan Lim, Ross Miller, Swen Boehm, Cameron Stanavige, Kathryn M Mohror, and Sarp Oral. 2023. UnifyFS: A user-level shared file system for unified access to distributed local storage. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*.
[12] Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. 2009. 24/7 Characterization of Petascale I/O Workloads. In *IEEE International Conference on Cluster Computing and Workshops*. 1–10.
[13] Kevin Deierling. 2020. What Is a DPU? https://blogs.nvidia.com/blog/whats-a-dpu-data-processing-unit/
[14] Hariharan Devarajan and Meng Tang (Candice). 2025. hariharan-devarajan/datacrumbs: Release v0.0.2. https://doi.org/10.5281/zenodo.14983482
[15] Robert Dietrich, Frank Winkler, Andreas Knüpfer, and Wolfgang Nagel. 2020. PIKA: Center-Wide and Job-Aware Cluster Monitoring. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. 424–432. https://doi.org/10.1109/CLUSTER49012.2020.00061
[16] Pourmal Elena and Knox Larry. 2017. HDF5 Data Compression Demystified #2: Performance Tuning.
[17] Todd Evans, William L Barth, James C Browne, Robert L DeLeon, Thomas R Furlani, Steven M Gallo, Matthew D Jones, and Abani K Patra. 2014. Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats. In *2014 First International Workshop on HPC User Support Tools*. 13–21.
[18] Elie F Kfoury, Samia Choueiri, Ali Mazloum, Ali AlSabeh, Jose Gomez, and Jorge Crichigno. 2024. A Comprehensive Survey on SmartNICs: Architectures, Development Models, Applications, and Research Directions. *IEEE Access* (2024).
[19] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen Malony, et al. 2012. Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir. In *Tools for High Performance Computing 2011*. Springer, 79–91.
[20] Anthony Kougkas, Hariharan Devarajan, and Xian-He Sun. 2018. Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*. 219–230.
[21] Rohit Kumar. 2017. Mellanox BlueField NVMeoF SoC Solution. https://www.servethehome.com/mellanox-bluefield-nvmeof-soc-solution/
[22] Glenn K. Lockwood, Nicholas J. Wright, Shane Snyder, Philip Carns, George Brown, and Kevin Harms. 2018. TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis. *2018 Cray User Group Conference (CUG)* (1 2018).
[23] Huong Luu, Marianne Winslett, William Gropp, Robert Ross, Philip Carns, Kevin Harms, Mr Prabhat, Suren Byna, and Yushu Yao. 2015. A Multiplatform Study of I/O Behavior on Petascale Supercomputers. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*. Association for Computing Machinery, New York, NY, USA, 33–44. https://doi.org/10.1145/2749246.2749269
[24] Matt Massie, Bernard Li, Brad Nicholes, and Vladimir Vuksan. 2012. *Monitoring with Ganglia*. " O'Reilly Media, Inc.".
[25] Joseph Y. Moon, Francesco Di Natale, Helgi I. Ingolfsson, Harsh Bhatia, and Joseph R. Chavez. 2021. *Multiscale Machine-Learned Modeling Infrastructure RAS*. Technical Report MuMMI RAS. Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States). https://doi.org/10.11578/dc.20211209.3
[26] Alessio Netti, Micha Müller, Axel Auweter, Carla Guillen, Michael Ott, Daniele Tafani, and Martin Schulz. 2019. From Facility to Application Sensor Data: Modular, Continuous and Holistic Monitoring with DCDB. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–27.
[27] Sarah Neuwirth. 2023. Modular Supercomputing and its Role in Europe's Exascale Computing Strategy. *PoS* LATTICE2022 (2023), 245.
[28] Sarah Neuwirth and Arnab K Paul. 2021. Parallel I/O Evaluation Techniques and Emerging HPC Workloads: A Perspective. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. https://doi.org/10.1109/Cluster48925.2021.00100
[29] Neeraj Rajesh, Hariharan Devarajan, Jaime Cernuda Garcia, Keith Bateman, Luke Logan, Jie Ye, Anthony Kougkas, and Xian-He Sun. 2021. Apollo:: An ML-assisted Real-Time Storage Resource Observer. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*. 147–159.
[30] Daniel Reed, Dennis Gannon, and Jack Dongarra. 2022. Reinventing High Performance Computing: Challenges and Opportunities. https://doi.org/10.48550/arXiv.2203.02544
[31] Gilad Shainer, Ali Ayoub, Pak Lui, Tong Liu, Michael Kagan, Christian R. Trott, Greg Scantlen, and Paul S. Crozier. 2011. The development of Mellanox/NVIDIA GPUDirect over InfiniBand—a new model for GPU to GPU communications. *Computer Science - Research and Development* 26, 3 (June 2011), 267–273. https://doi.org/10.1007/s00450-011-0157-1
[32] Pavel Shamis, Manjunath Gorentla Venkata, M Graham Lopez, Matthew B Baker, Oscar Hernandez, Yossi Itigin, Mike Dubman, Gilad Shainer, Richard L Graham, Liran Liss, et al. 2015. UCX: an open source framework for HPC network APIs and beyond. In *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*.
[33] Sameer S. Shende and Allen D. Malony. 2006. The Tau Parallel Performance System.
[34] Konstantin S. Stefanov, Sucheta Pawar, Ashish Ranjan, Sanjay Wandhekar, and Vladimir V. Voevodin. 2021. A Review of Supercomputer Performance Monitoring Systems. *Supercomputing Frontiers and Innovations* 8, 3 (2021), 62–81. https://doi.org/10.14529/jsfi210304
[35] A Tekin, A Tuncer Durak, C Piechurski, D Kaliszan, F Aylin Sungur, F Robertsén, and P Gschwandtner. 2021. *State-of-the-Art and Trends for Computing and Interconnect Network Solutions for HPC and AI*. Technical Report. PRACE.
[36] Marc-André Vef, Nafiseh Moti, Tim Süß, Markus Tacke, Tommaso Tocci, Ramon Nou, Alberto Miranda, Toni Cortes, and André Brinkmann. 2020. GekkoFS – A temporary burst buffer file system for HPC applications. *Journal of Computer Science and Technology* 35 (2020), 72–91.
[37] Chen Wang, Jinghan Sun, Marc Snir, Kathryn Mohror, and Elsa Gonsiorowski. 2020. Recorder 2.0: Efficient parallel I/O tracing and analysis. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*.
[38] Teng Wang, Suren Byna, Bin Dong, and Houjun Tang. 2018. UniviStor: Integrated hierarchical and distributed storage for HPC. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. 134–144.
[39] Zhaobin Zhu, Sarah Neuwirth, and Thomas Lippert. 2022. A Comprehensive I/O Knowledge Cycle for Modular and Automated HPC Workload Analysis. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. https://doi.org/10.1109/CLUSTER51413.2022.00076