Optimizing I/O for an Exascale Implicit Kinetic Plasma Simulation using the Rabbit Storage System

Ian Lumsden*, Hariharan Devarajan†, Izzet Yildirim‡, Stefano Markidis§, Andong Hu§, Ivy Peng§, Luca Pennati§, Dewi Yokelson¶, Stephanie Brink†, Olga Pearce†, Tom Scogland†, Bronis R. de Supinski†, Gian Luca Delzanno‖, Anthony Kougkas‡, Xian-He Sun‡, Michela Taufer*

* Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA
† Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA
‡ Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA
§ School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden
¶ Advanced Micro Devices, Inc.
‖ T-5 Applied Mathematics and Plasma Physics, Los Alamos National Laboratory, Los Alamos, NM, USA

***Abstract*—Exascale computing revolutionizes scientific research by facilitating large-scale simulations and data analysis. As applications generate massive, heterogeneous data streams, the I/O subsystem increasingly dominates runtime, limiting scalability and time-to-solution. Emerging storage accelerators such as the Rabbit system offer a promising path forward. Rabbit provides dynamically configurable hybrid storage, combining high-bandwidth, node-local NVMe (e.g., XFS) with shared, distributed file systems (e.g., Lustre), to match diverse I/O patterns more efficiently. In this work, we evaluate Rabbit's capability to mitigate I/O bottlenecks using iPIC3D, a representative implicit Particle-in-Cell (PIC) code for exascale scientific workloads. We perform a detailed characterization of the key phases of I/O (restart, field, and moment) and identify the dominant access patterns in these phases. We then benchmark these patterns using IOR in Rabbit hybrid storage configurations, identifying the ideal performance achievable for each access pattern under node-local and distributed storage modes. We use phase-aware mapping of these patterns to Rabbit storage systems, achieving a 4.85 times improvement in I/O throughput, reducing the I/O share of runtime from 38% to 11% and delivering an end-to-end speedup of 1.45 times. These results outline the importance of hardware-software co-design and highlight Rabbit as a scalable, data-intensive storage solution.**

***Index Terms*—High performance computing, I/O optimization, Rabbit nodes, I/O accelerators, iPIC3D, IOR.**

I. Introduction

Exascale computing revolutionizes scientific research by facilitating large-scale simulations and data analysis. However, the I/O subsystem poses a significant bottleneck, often using substantial application runtime. As computational speed surpasses storage capabilities, the challenge of efficiently handling large amounts of data increases. Sustained exascale performance demands hardware advancements alongside strategic co-design of storage systems, middleware, and applications.

Emerging technologies, such as the Rabbit storage system [1], provide a promising path forward. Rabbit is a rack-level, dynamically configurable I/O accelerator that combines node-local NVMe drives with software-defined orchestration to deliver both high-bandwidth, low-latency local storage (e.g., XFS) and shared, distributed storage (e.g., Lustre). Rabbit's flexible architecture is now a key feature of leading supercomputers such as El Capitan [2], illustrating that storage accelerators are no longer experimental but are becoming an integral part of next-generation HPC systems.

Optimizing applications to take advantage of Rabbit, however, is a non-trivial task. Scientific codes must align their diverse and heterogeneous I/O patterns with the capabilities of Rabbit's hybrid storage modes. Such an alignment requires a hardware-software co-design approach in which application-level I/O phases are characterized, mapped, and tuned for the available storage backends.

In this work, we focus on iPIC3D [3]–[6], a state-of-the-art implicit Particle-in-Cell (PIC) code for space and plasma physics, which exhibits all the I/O challenges typical of exascale workloads. iPIC3D's runtime is dominated by multiple I/O phases—large file-per-process writes, small collective outputs, and frequent structured data dumps—each stressing different aspects of the storage stack. Addressing these challenges requires phase-aware optimization strategies that leverage Rabbit's ability to use multiple file systems, dynamically tailored to each pattern. This paper makes the following contributions:

- We perform a detailed analysis of iPIC3D's I/O patterns, identifying the dominant phases (restart, field, and moment data) and quantifying their performance characteristics on production HPC systems.
- We benchmark these I/O patterns using IOR on Rabbit's XFS and Lustre configurations, demonstrating how phase-aware mapping improves performance.
- We integrate the optimized configurations into iPIC3D, achieving a 4.85× improvement in I/O throughput and reducing the I/O share of runtime from 38% to 11%, resulting in a 1.45× overall speedup.

The paper highlights the broader implications of hybrid workload-driven storage strategies for exascale applications.

II. I/O Technology and Application Benchmark

The ability to achieve sustained performance at exascale requires not just faster compute hardware but also innovative

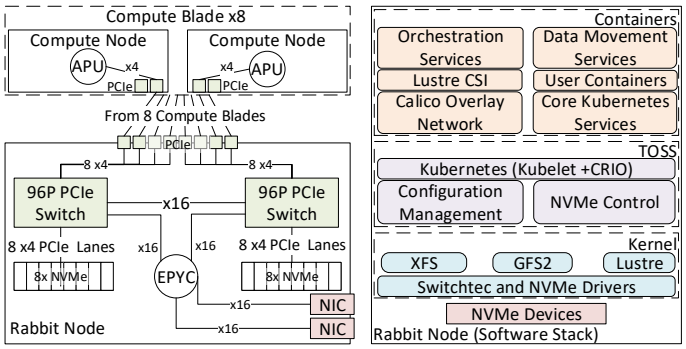


Fig. 1. Rabbit storage system architecture: hardware topology connecting compute blades to Rabbit nodes via PCIe (Left) and software stack for containerized file system management and orchestration (Right).

I/O subsystems capable of matching data movement demands. The Rabbit storage system addresses these requirements through dynamic, workload-aware storage provisioning. At the same time, applications like iPIC3D, with their heterogeneous I/O phases, provide an ideal use case for quantifying the benefits of such I/O technologies.

A. Rabbit Storage System

The Rabbit storage system is a rack-level, software-defined I/O accelerator engineered to bridge the gap between compute and storage performance on exascale systems [1]. Rabbit integrates high-bandwidth NVMe devices with flexible orchestration layers to dynamically provision storage configurations tailored to application needs. Figure 1 provides an overview of its design, with the hardware layout [7] on the left and the software architecture [8] on the right. Each Rabbit node connects to multiple compute blades via high-speed PCIe links, providing direct access to 16 NVMe drives with an aggregate peak bandwidth of 56 GB/s. Rabbit's dual AMD EPYC processors and 256 GB of DRAM enable high-throughput, low-latency data paths between compute and storage tiers. Rabbit's software layer leverages Kubernetes-based orchestration to deploy and manage multiple file systems (e.g., XFS, GFS2, Lustre) with tunable parameters such as striping, record sizes, and RPC concurrency [9]. This approach allows hybrid I/O modes, combining node-local file systems for high-bandwidth writes with distributed configurations for collective I/O patterns. Rabbit nodes are integrated into Top500 supercomputers such as El Capitan, where they act as burst buffers and acceleration layers, significantly reducing I/O bottlenecks.

B. iPIC3D as a Benchmark Application

iPIC3D is a GPU-enabled implicit Particle-in-Cell (PIC) code designed for large-scale space and plasma physics simulations [3]–[6]. Its ability to simulate planetary-scale magnetospheric dynamics across multiple spatial and temporal scales makes it representative of the complexity and scale of exascale workloads. iPIC3D's diverse I/O patterns stress both large sequential throughput and metadata-intensive collective writes, which are key performance dimensions for Rabbit. As such, iPIC3D serves as an ideal testbed for evaluating Rabbit's hybrid storage capabilities and exploring phase-aware

TABLE I
CHARACTERIZATION OF THE I/O BEHAVIOR OF IPIC3D ON 32 NODES.

Job Time	776.8s	Number of Files	965
I/O Time	283.1s	Number of I/O Ops	197K
Number of Nodes	32	Total I/O Size	14.1 TB
Total Processes	128	Agg. Bandwidth	50.9 GB/s
I/O Threads/Process	1	Avg. Transfer Size	74.7 MB
Compute Threads/Process	4	Mount Point	/p/lustre5

	Restart Data	Field Data	Moment Data
Number of Files	128	1	1
Processes per File	1	128	128
Total I/O per File (MB)	115074	128	64
Transfer Size (MB)	498.2 ± 0.2	1.0 ± 0.5	0.5 ± 0.0
Percent I/O Time	93.564%	0.007%	0.006 %
I/O Bandwidth (MB/s)	2029.4	27.4	26.4

optimizations. In particular, iPIC3D exhibits three distinct heterogeneous I/O patterns across three separate phases of its application. First, the *Restart Data* phase involves writing a large-scale, file-per-process output exceeding 100 GB per process using ADIOS2 [10]. This phase is primarily characterized by sequential write operations for checkpointing. Second, during the *Field Data* phase, moderate-sized collective write operations are performed using MPI-IO [11] to record the physical energy fields (e.g., magnetic, electric). The data produced in this phase is written as a collection of VTK [12] files for visualization purposes. Third, the *Moment Data* phase is marked by frequent and small-scale collective MPI-IO write outputs, which systematically document particle moments along with relevant diagnostic information. Like field data, moment data is written as a collection of VTK files for visualization purposes. These I/O phases collectively account for more than 35% of the runtime of iPIC3D, making I/O a significant performance bottleneck.

III. CHARACTERIZING AND OPTIMIZING IPIC3D I/O WITH RABBIT

To systematically evaluate and optimize iPIC3D's I/O performance on the Rabbit system storage, we adopt a twofold methodology. First, we characterize iPIC3D's I/O behavior, identifying its dominant access patterns across the three phases. Second, we map these patterns to Rabbit's hybrid storage configurations using IOR-based benchmarking to assess how node-local (XFS) and distributed (Lustre) setups affect performance and scalability.

A. Characterization of iPIC3D's I/O

iPIC3D exhibits highly I/O-intensive characteristics, with a significant portion of the job time dedicated to data movement. We run the iPIC3D workload on 32 nodes and collect its I/O behavior using DFTracer [13], then analyze the traces with DFAnalyzer [14], [15] to identify key characteristics and pinpoint critical files requiring optimization. Statistics on these characteristics and critical files are shown in Table I. Out of the total runtime of 776.8 seconds, I/O accounts for 283.1 seconds, representing more than 36% of the execution time. The application generates 14.1 TB of data distributed across 965 files through 197K I/O operations. Running on 32 nodes with 128 MPI processes, each process utilizes a

single I/O thread and four compute threads. The aggregate I/O bandwidth achieved is 50.9 GB/s, with an average transfer size of 74.7 MB per operation, indicating that improving aggregate I/O bandwidth would significantly enhance the application's overall performance. All I/O operations target a Lustre file system mounted at /p/lustre5, which is typical for large-scale HPC workloads.

A closer examination of the I/O phases reveals that restart data accounts for the majority of the overall I/O footprint. This phase alone accounts for 93.56% of the total I/O time, writing 115 GB per file across 128 files, each handled by a single process. The restart phase achieves an impressive bandwidth of 2029.4 MB/s per file, with a large transfer size of approximately 498 MB, reflecting a highly aggregated, sequential write pattern. In contrast, field and moment data collectively contribute negligible I/O time (less than 0.02%), writing small files of 128 MB and 64 MB, respectively, with transfer sizes of 1 MB and 0.5 MB per operation. These phases exhibit bandwidths of 27.4 MB/s and 26.4 MB/s, characteristic of small, fragmented writes. Improving the performance for these three data types, namely, restart, field, and moment, would cover the vast majority of I/O cases in iPIC3D, as they encompass all significant data movement patterns from large, sequential checkpointing writes to smaller, parallel outputs. Addressing these patterns holistically would result in substantial end-to-end performance gains for the application.

If we analyze the workload using I/O time as the metric, the restart files dominate the profile, accounting for 93.564% of total I/O time. However, when examining I/O bandwidth, a different picture emerges. While restart files achieve relatively high per-process bandwidth due to large sequential writes, the field and moment data are severely constrained, reaching only 27.4 MB/s and 26.4 MB/s, respectively. These phases involve collective operations with small transfer sizes (1 MB and 0.5 MB), leading to fragmented and inefficient data movement. As a result, although their contribution to I/O time appears minimal, their poor bandwidth utilization represents an opportunity for significant optimization. Improving the handling of field and moment outputs, along with further improving efficiency for restart files, is essential for boosting overall I/O performance and reducing end-to-end runtime in iPIC3D.

B. Optimizing iPIC3D's I/O

To optimize iPIC3D's I/O performance, we adopt a methodology centered on leveraging the Rabbit Storage System as a high-performance acceleration layer. The first step involves analyzing the three primary I/O file patterns in iPIC3D (i.e., restart, field, and moment data) and representing each file pattern using the IOR benchmark [16], a widely accepted tool for emulating scientific application I/O behaviors with high fidelity. This approach allows us to systematically evaluate how different Rabbit configurations impact the performance of these distinct file types. We conduct a series of benchmark experiments across multiple configurations: (i) Rabbit XFS, which provides node-local access over PCIe with dedicated XFS namespaces created from 16 NVMe drives mounted exclusively for compute nodes, and (ii) Rabbit Lustre, which enables distributed access over the Slingshot network with two object storage targets (OSTs) across the same NVMe drives. The Lustre setup is carefully tuned for performance, considering layout parameters (record size, raidz level), resilience features (raidz, checksumming), and space optimizations (compression), as well as operational factors such as RPC concurrency, client core binding, and autotrim.

Once benchmarking results identify the best-performing configurations for each data type, we integrate these optimizations into the actual iPIC3D application, redirecting its I/O streams to Rabbit-enabled configurations. This design enables the selective acceleration of I/O phases by mapping restart data, field data, and moment data to either local XFS namespaces or a shared Lustre namespace, based on their respective performance profiles. A key advantage of the Rabbit Storage System is its ability to support both modes simultaneously on the same hardware, enabling hybrid acceleration strategies within a single iPIC3D run. Finally, we validate the effectiveness of these configurations through end-to-end experiments, demonstrating substantial reductions in I/O time and overall job runtime for iPIC3D on Rabbit-enabled HPC systems.

IV. Evaluating Phase-Aware I/O Strategies

To evaluate Rabbit hybrid storage's effect on various iPIC3D I/O patterns, we establish test cases and record performance metrics to address these questions: (i) How does Rabbit compare to the global Lustre file system in bandwidth and scalability for various I/O patterns? (ii) Which Rabbit setup (XFS vs. Lustre) aligns better with iPIC3D's specific I/O stages (restart, field, moment)? (iii) Does merging node-local and distributed Rabbit modes offer a superior phase-aware I/O strategy compared to a solo global setup? (iv) What performance improvements are realized by embedding Rabbit optimizations directly into iPIC3D?

A. Testbed

We executed our tests on the Tuolumne system at Lawrence Livermore National Laboratory [17], utilizing a maximum of 128 out of 1100 nodes. Each node features an AMD MI300A APU [18] with a 96-core 4th Gen AMD EPYC processor, four CDNA3 GPUs, and 512GB HBM3 RAM. The network operates on a 4$\times$200 Gb/s HPE Slingshot 11 interconnect. The system uses a Global Lustre file system for node access. Rabbit nodes include 16 2-TB NVMe drives, of which we configure 20TB, achieving up to 56GB/s in sequential bandwidth. Each Rabbit is equipped with 256GB RAM, a 64-core AMD EPYC 7713 processor, and two 200 Gb/s network interfaces.

Experiments utilize Tuolumne with the TOSS 4 OS. Key software includes HPE Cray CCE 18.0.1, Cray MPICH 8.1.25 for MPI, and CMake 3.29.2 for builds. ROCm 6.2.4 enables GPU acceleration on AMD MI300A APUs, while ADIOS2 2.10.1 handles iPIC3D restart data. Performance data is gathered using Caliper [19], DFTracer [13], and DFAnalyzer [14], [15].

1) IOR Benchmark Configuration: The setup pertains to restart, field, and moment files, detailing transfer size, block dimensions, access method (collective or file-per-process), and I/O interface (POSIX or MPI-IO). Multiple runs are conducted to ensure reliable performance metrics. This systematic technique helps align iPIC3D's I/O patterns with realistic benchmarks prior to workload implementation.

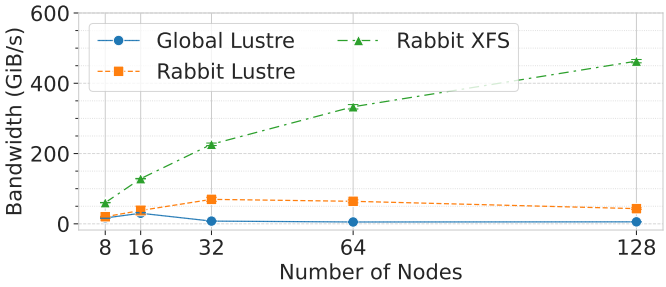


Fig. 2. Aggregate bandwidth for iPIC3D restart I/O is 43 GB/s for Global Lustre and 462.9 GB/s on Rabbit XFS at 128 nodes, delivering over 10× speedup through node-local acceleration.

2) iPIC3D: For iPIC3D, we use input decks representing a 3-dimensional magnetic reconnection problem designed to strong scale across all of El Capitan. We focus specifically on the input decks for 32, 64, and 128 nodes. Each run of iPIC3D performs 300 cycles of computation, and each run writes restart, field, and moment data every 50 cycles. Additionally, each run writes one final set of restart data after the simulation has completed. We configure iPIC3D to write all field and moment data to the same directory and to write all restart data to a different directory. By directing similar types of writes (i.e., file-per-process, collective I/O) to the same directories, we simplify the process of tuning iPIC3D's I/O using Rabbit.

B. IOR

1) Restart Data: To approximate the restart I/O pattern observed in iPIC3D, we configure the IOR benchmlrk to use file-per-process (FPP) mode with large, contiguous writes. Each process writes 512 MB transfer sizes with a total block size of 128 GB, ensuring that the files were large enough to emulate the restart data characteristics identified during characterization. We run the benchmark across scales from 8 to 128 nodes, using 4 MPI processes per node, and repeat 10 iterations per configuration to account for variability. This configuration closely reflects the application's checkpointing behavior, which is dominated by large sequential writes, even though it does not replicate it exactly.

The results are summarized in Figure 2, which plots aggregate bandwidth (GB/s) versus node count (8 to 128) for three configurations: Global Lustre (baseline), Rabbit Lustre (distributed mode over Slingshot), and Rabbit XFS (node-local NVMe access over PCIe). On Global Lustre, performance begins at 16.6 GB/s for 8 nodes, briefly improves to 30 GB/s at 16 nodes, but drops sharply beyond 32 nodes to around 5 GB/s, indicating significant scaling limitations due to network or metadata contention. Rabbit Lustre offers

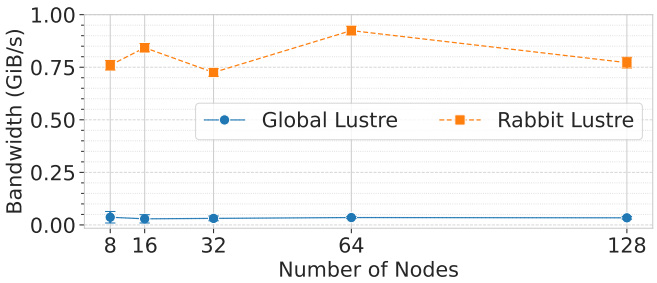


Fig. 3. Aggregate bandwidth for iPIC3D field data scales from 36.6 MB/s on Global Lustre and 924.3 MB/s on Rabbit Lustre at 64 nodes, delivering up to a 25× speedup and reducing I/O bottlenecks for small collective writes.

better scaling, reaching 20 GB/s at 8 nodes and peaking at 69 GB/s at 32 nodes, though performance plateaus at 64 nodes and declines slightly to 43 GB/s at 128 nodes. In contrast, Rabbit XFS demonstrates near-linear scalability, starting at 60 GB/s on 8 nodes and reaching 462.9 GB/s on 128 nodes, highlighting the advantage of node-local NVMe access over PCIe without network bottlenecks.

These results confirm that restart I/O in iPIC3D, dominated by large sequential writes, benefits most from node-local acceleration. Among the tested options, Rabbit XFS is the optimal configuration, providing more than an order-of-magnitude improvement over the global Lustre file system and enabling sustained high bandwidth at scale. Leveraging Rabbit with XFS for checkpointing phases can significantly reduce overall job time and improve system utilization.

2) Field Data: To approximate the field data I/O pattern in iPIC3D, we use the IOR benchmark configured for MPI-IO collective writes to a shared file, reflecting the small, coordinated output characteristic of this phase. Each process writes with a transfer size of 1 MB and a block size of 16 MB, using the MPI-IO interface with collective mode. We execute the benchmark at scales ranging from 8 to 128 nodes with 4 processes per node, and each test is repeated 10 times to capture performance variability. This benchmark provides a close approximation by modeling collective access and small aggregate output sizes.

Figure 3 plots aggregate bandwidth in MB/s against node count (8 to 128) for Global Lustre and Rabbit Lustre. On Global Lustre, performance is extremely limited, ranging from 36.6 MB/s at 8 nodes to 33.8 MB/s at 128 nodes, with minimal improvement across scales and significant variability at smaller scales. Rabbit Lustre shows a dramatic improvement, starting at 759.4 MB/s for 8 nodes, peaking at 924.3 MB/s at 64 nodes, and slightly reducing to 771.7 MB/s at 128 nodes. While the trend for Rabbit Lustre flattens at larger scales, it consistently provides an order-of-magnitude improvement compared to Global Lustre.

Field data writes using MPI-IO collective semantics are severely constrained on Global Lustre due to metadata and coordination bottlenecks, achieving less than 40 MB/s even at large scales. In contrast, Rabbit Lustre improves throughput to nearly 1 GB/s, substantially reducing I/O overhead for this phase. Therefore, for field data, Rabbit Lustre is the optimal configuration, providing strong improvements without

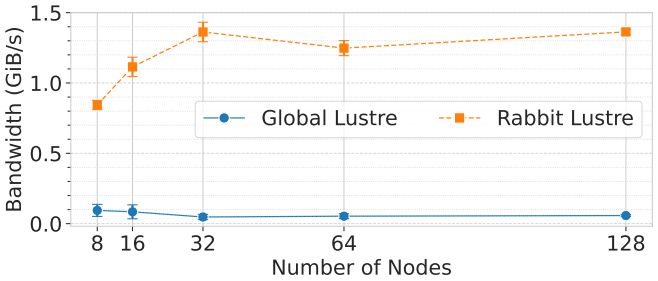


Fig. 4. Aggregate bandwidth for iPIC3D moment data increases from 94 MB/s on Global Lustre to 1.36 GB/s on Rabbit Lustre at 128 nodes, achieving over a 14× speedup and enabling efficient scaling for small collective writes.

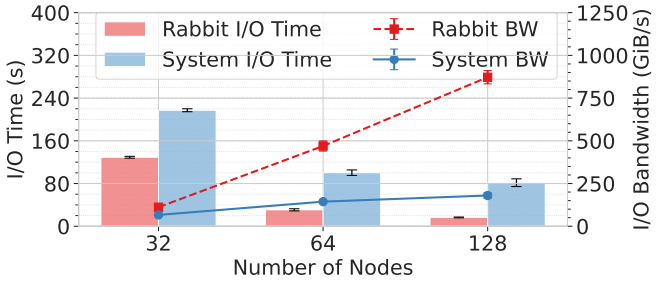


Fig. 5. I/O time and aggregate bandwidth for iPIC3D – across 32-128 nodes, iPIC3D's Rabbit hybrid setup boosts bandwidth from 66 GiB/s to 872 GiB/s, cuts I/O time, and achieves up to a 4.85× I/O speedup and a 1.45× overall application acceleration.

application-level changes and enabling efficient scaling for small collective output patterns.

3) Moment Data: To emulate the I/O pattern of moment data in iPIC3D, we configure the IOR benchmark using MPI-IO with collective writes and smaller transfer sizes to reflect the frequent, small, structured outputs observed during characterization. Each process writes using a transfer size of 512 KB and a block size of 4 MB, representing finer-grained I/O operations with aggregation across processes. We execute the runs on 8 to 128 nodes with 4 processes per node, and each experiment is repeated 10 times to account for variability. This configuration closely models the small data size and collective write patterns for the Moment Data.

Figure 4 plots aggregate bandwidth (MB/s) on the y-axis versus node count (8 to 128) on the x-axis, comparing Global Lustre and Rabbit Lustre configurations. For Global Lustre, bandwidth remains very low, starting at 94.4 MB/s at 8 nodes and gradually decreasing to 57.6 MB/s at 128 nodes, with notable performance variability at smaller scales. In contrast, Rabbit Lustre achieves significantly higher throughput, beginning at 843.6 MB/s for 8 nodes and scaling up to 1,363.4 MB/s at 128 nodes, with its best performance observed at 32 and 128 nodes (approximately 1.36 GB/s). While some fluctuations are present beyond 32 nodes, Rabbit Lustre consistently maintains over 1 GB/s, which is more than an order-of-magnitude improvement over Global Lustre.

Moment data, characterized by small, collective I/O operations, experiences severe performance limitations on Global Lustre due to coordination and metadata bottlenecks, never exceeding 100 MB/s at any scale. Rabbit Lustre dramatically improves performance to 1.3 GB/s, demonstrating its ability to handle small collective writes efficiently. Therefore, Rabbit Lustre is the recommended configuration for moment data, providing substantial scaling and throughput benefits without requiring application-level modifications.

C. iPIC3D

For iPIC3D, we select the best-performing configurations identified from the IOR-based evaluation. We redirect restart data to Rabbit XFS, and we direct moment and field data to Rabbit Lustre, creating a hybrid configuration that aligns the underlying file system with each phase's I/O pattern. This strategy exploits Rabbit XFS for large, sequential, and file-per-process writes and Rabbit Lustre for smaller, collective writes, ensuring phase-aware optimization of the workload's diverse access characteristics.

Figure 5 plots I/O time in seconds on the left y-axis and aggregate bandwidth (GiB/s) on the right y-axis against node scale (32, 64, and 128 nodes), comparing the baseline system configuration with the Rabbit-optimized configuration. Significant improvements were observed at all scales. At 32 nodes, bandwidth increased from 66.28 GiB/s to 111.43 GiB/s, providing a 1.68× speedup. At 64 nodes, Rabbit achieved 469.65 GiB/s, compared to 143.91 GiB/s on the system, a 3.26× improvement. At the largest scale, 128 nodes, Rabbit delivered 872.16 GiB/s, nearly 4.85× faster than the baseline's 179.97 GiB/s. This demonstrates Rabbit's ability to enable phase-aware, multi-backend storage strategies for HPC applications with heterogeneous I/O demands.

These results confirm that Rabbit can be tuned to deploy multiple file systems simultaneously, each tailored to the specific requirements of different I/O phases within the same workload. This hybrid approach minimizes contention and significantly accelerates I/O by 4.85× faster than baseline.

V. Conclusions

Rabbit's flexible architecture can significantly reduce I/O bottlenecks in HPC applications by dynamically matching storage configurations to application-specific access patterns. Our study provides three generalizable insights: (1) hybrid I/O strategies using multiple file systems in a single workflow deliver superior performance; (2) workload characterization is critical for precise mapping of access patterns to storage backends; and (3) rack-level storage accelerators like Rabbit, with Kubernetes-based orchestration for on-demand deployment of tunable file systems, enable dynamic adaptability for diverse workloads. Applying these strategies to iPIC3D, we achieved a 4.85× improvement in I/O performance, reducing the I/O share of runtime from 38% to 11% and resulting in a 1.45× overall application speedup, ultimately accelerating time-to-solution and improving system efficiency.

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-2009589). IL and MT acknowledge the support of NSF #2138811 and #2331152.

REFERENCES

[1] B. Behlendorf and O. Faaland, "Rabbit Storage for El Capitan: Fast I/O through Big, Pointy Teeth," Presented at the Lustre User Group (LUG), May 2023. [Online]. Available: https://www.opensfs.org/wp-content/uploads/Fast-IO-El-Capitan-Rabbits.revised.pdf

[2] "El Capitan | HPC @ LLNL." [Online]. Available: https://hpc.llnl.gov/hardware/compute-platforms/el-capitan

[3] S. Markidis *et al.*, "Exascale Implicit Kinetic Plasma Simulations on El˜Capitan for Solving the Micro-Macro Coupling in Magnetospheric Physics," *arXiv preprint arXiv:2507.20719*, 2025.

[4] S. Markidis, G. Lapenta, and Rizwan-uddin, "Multi-scale simulations of plasma with iPIC3D," *Mathematics and Computers in Simulation*, vol. 80, 2010.

[5] P. Kumar, S. Markidis, G. Lapenta, K. Meerbergen, and D. Roose, "High Performance Solvers for Implicit Particle in Cell Simulation," in *Proceedings of the International Conference of Computational Science (ICCS)*, 2013.

[6] C. P. Sishtla, S. W. D. Chien, V. Olshevsky, E. Laure, and S. Markidis, "Multi-GPU Acceleration of the iPIC3D Implicit Particle-in-Cell Code," in *Proceedings of the International Conference of Computational Science (ICCS)*, 2019.

[7] "Using El Capitan Systems: File Systems and Rabbits | HPC @ LLNL." [Online]. Available: https://hpc.llnl.gov/documentation/user-guides/using-el-capitan-systems/file-systems-rabbits

[8] "NearNodeFlash/NearNodeFlash.github.io," 2025. [Online]. Available: https://github.com/NearNodeFlash/NearNodeFlash.github.io

[9] A. M. Beltre, P. Saha, M. Govindaraju, A. Younge, and R. E. Grant, "Enabling HPC Workloads on Cloud Infrastructure Using Kubernetes Container Orchestration Mechanisms," in *IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*, 2019.

[10] W. F. Godoy *et al.*, "Adios 2: The adaptable input output system. a framework for high-performance data management," *SoftwareX*, vol. 12, 2020.

[11] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard Version 2.0*, Jul. 1997. [Online]. Available: https://www.mpi-forum.org/docs/mpi-2.0/mpi2-report.pdf

[12] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit (4th ed.)*. Kitware, 2006.

[13] H. Devarajan *et al.*, "DFTracer: An Analysis-Friendly Data Flow Tracer for AI-Driven Workflows," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2024.

[14] I. Yildirim, H. Devarajan, A. Kougkas, X.-H. Sun, and K. Mohror, "WisIO: Automated I/O Bottleneck Detection with Multi-Perspective Views for HPC Workflows," in *International Conference on Supercomputing (ICS)*, 2025.

[15] I. Yildirim and H. Devarajan, "Data flow analyzer (dfanalyzer)," https://github.com/LLNL/dfanalyzer, Jul. 2025.

[16] J. Kunkel, "HPC IO Benchmark v 4.0.0," 2025. [Online]. Available: https://github.com/hpc/ior/releases/tag/4.0.0

[17] "Tuolumne | HPC @ LLNL." [Online]. Available: https://hpc.llnl.gov/hardware/compute-platforms/tuolumne

[18] A. Smith *et al.*, "Realizing the AMD Exascale Heterogeneous Processor Vision : Industry Product," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[19] D. Boehme *et al.*, "Caliper: Performance Introspection for HPC Software Stacks," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.