HPE Lawrence Livermore National Laboratory

Chasing the Rabbit:

A Systematic Exploration of Software-Defined HPC Storage



Hariharan Devarajan

ESSA 26

LLNL:

Brian Behlendorf, Gian-Carlo Defazio, James Corbett, Elsa Gonsiorowski, Cameron Stanavige, Nikoli Dryden, Roger Pearce, Kathryn Mohror, and Bronis R. de Supinski

HPE:

Blake Devcich, Dean Roehrich, Matt Richerson, Tony Floeder, Brent Petit, Don Boyd, Trace Hamada, and Tim McCree

Prepared by LLNL under Contract DE-AC52-07NA27344.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Modern Workloads are Data-Intensive.

Runtime

24%

Montage

Astronomy workflow that uses seven executables with ~5300 tasks and exchange 25K files.

18%

Cosmoflow

AI training workload that reads 2TB dataset with 512MB HDF5 Samples to detect the structure of universe.

41%

1000 Genomes Project

A bioinformatics workflow that performs 72TB of reads across 1.3M files from ~4000 tasks.



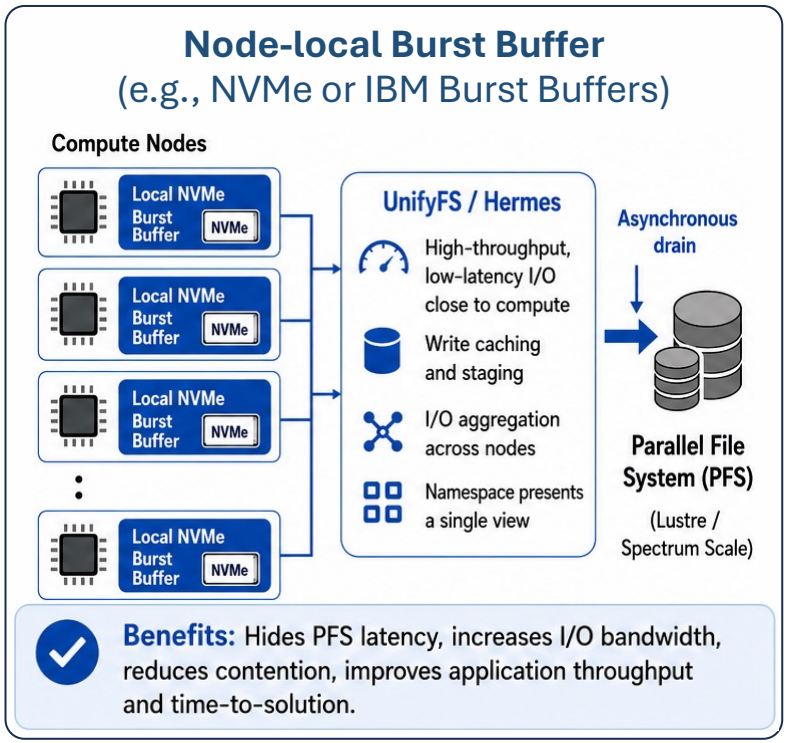
DDN IME **HPE DataWarp**

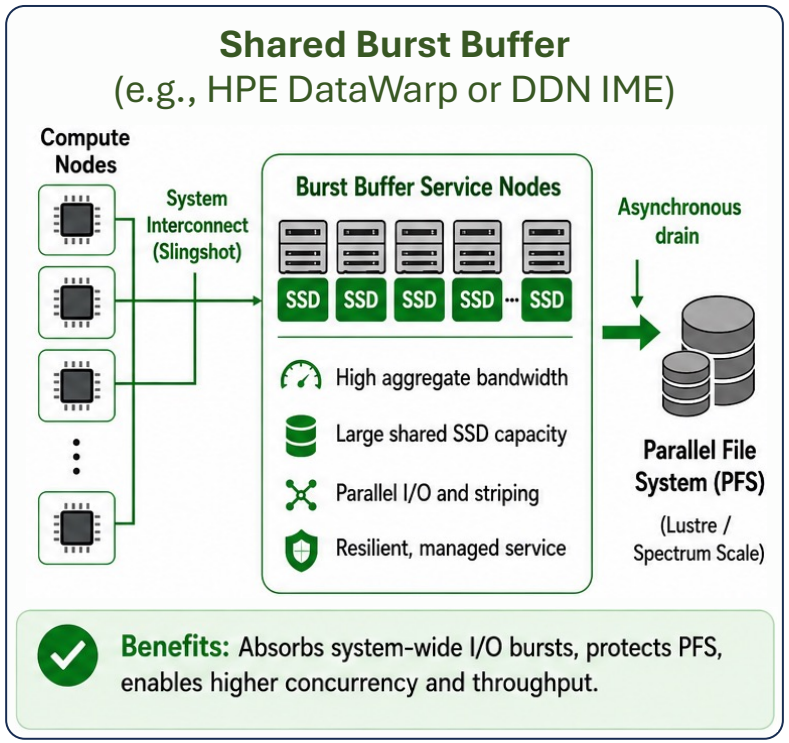
Storage accelerators reduce I/O bottlenecks

up to 3x

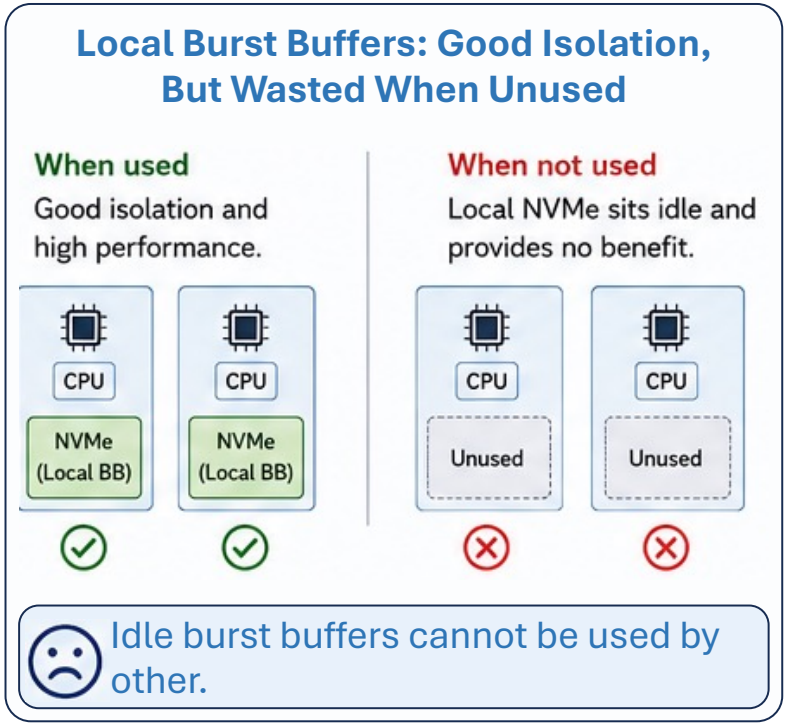


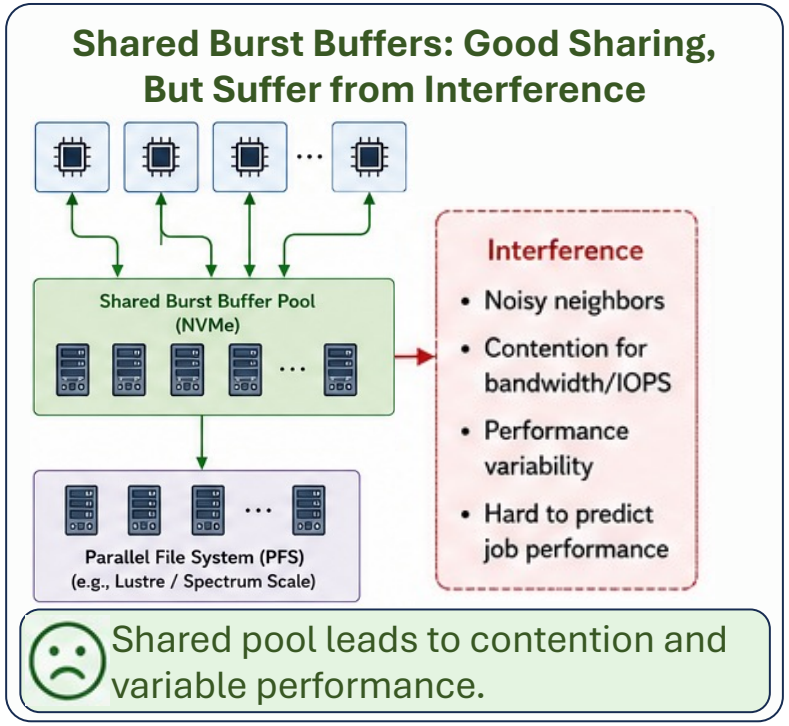
Burst Buffers absorb intensive I/O at high bandwidth close to compute, then drain to the PFS in the background.





Existing burst buffer designs use static allocation which limits utilization efficiency.





Existing burst buffer designs cannot adapt to diverse and dynamic workload needs.

Job Type / Need	Wants Local BB	Wants Shared BB	Example
Checkpoint-heavy simulation	✓	✗	High-rate checkpoints
Multi-stage workflow	✗	✓	Stage data between steps
Data-sharing ensemble	✗	✓	Share outputs across jobs
In-situ analytics + sharing	✓	✓	Local scratch + share results
Light I/O / compute-bound	✗	✗	Needs neither

Why static allocations fall short

- Over-provisioned for some, under-provisioned for others
- Cannot adapt as phases of a job change
- Leads to wasted capacity or poor performance
- Administrators must guess upfront—often wrong

Static Designs Lead to Wasted Resources, Interference, and Poor Utilization

1. Wasted Capacity
Idle local burst buffers cannot be used by others.

→

2. Interference
Shared pools cause contention and unpredictable performance.

→

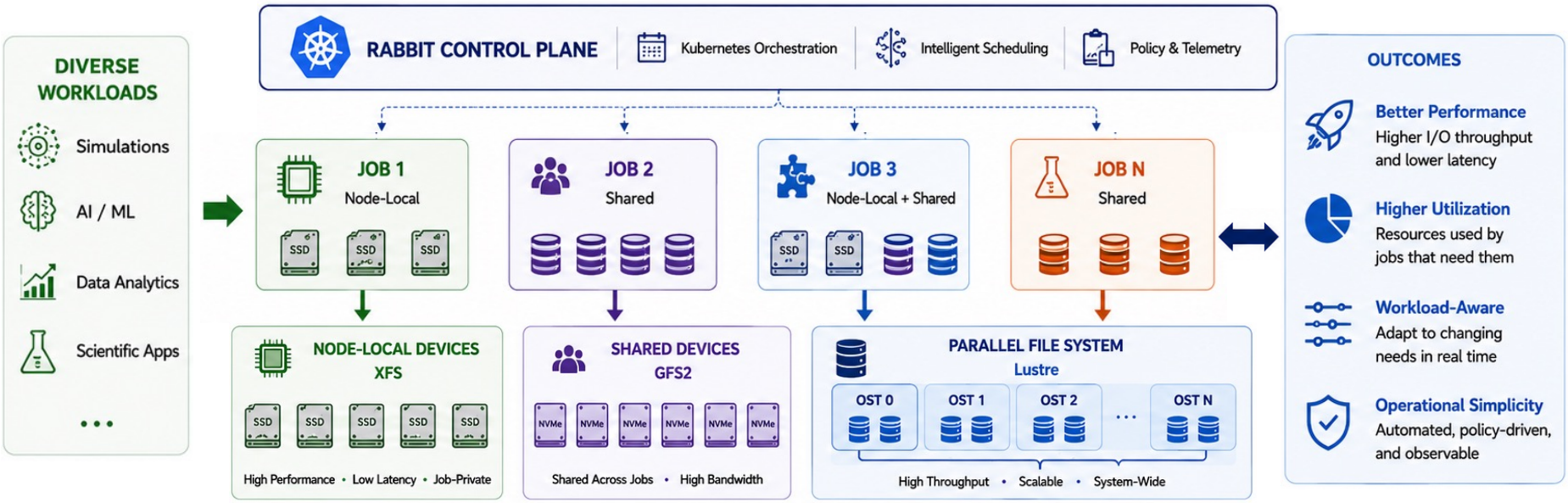
3. Lack of Adaptability
Jobs need different combinations (neither, one, other, or both). Static allocations cannot adapt.

→

Poor Utilization and Lower System Efficiency

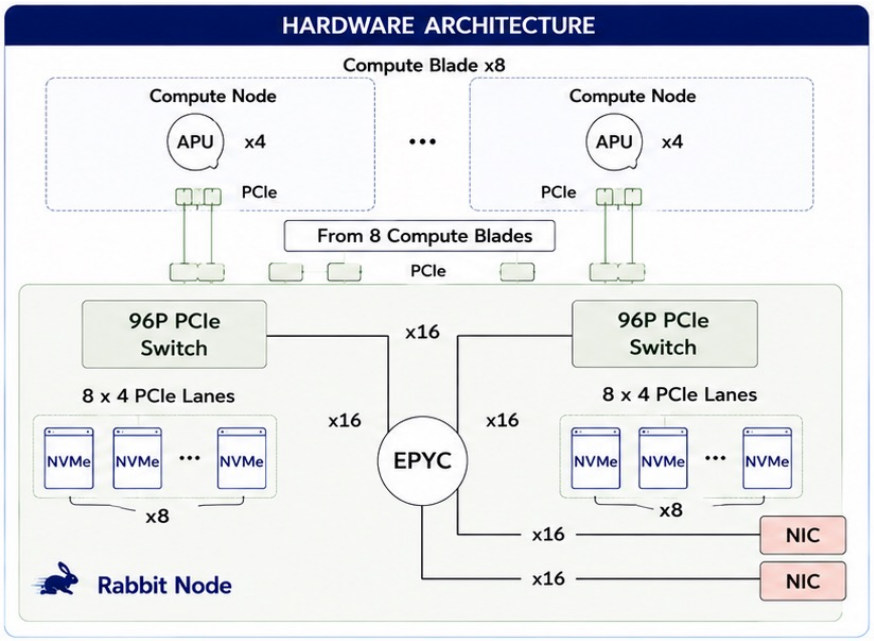
Introducing Rabbits

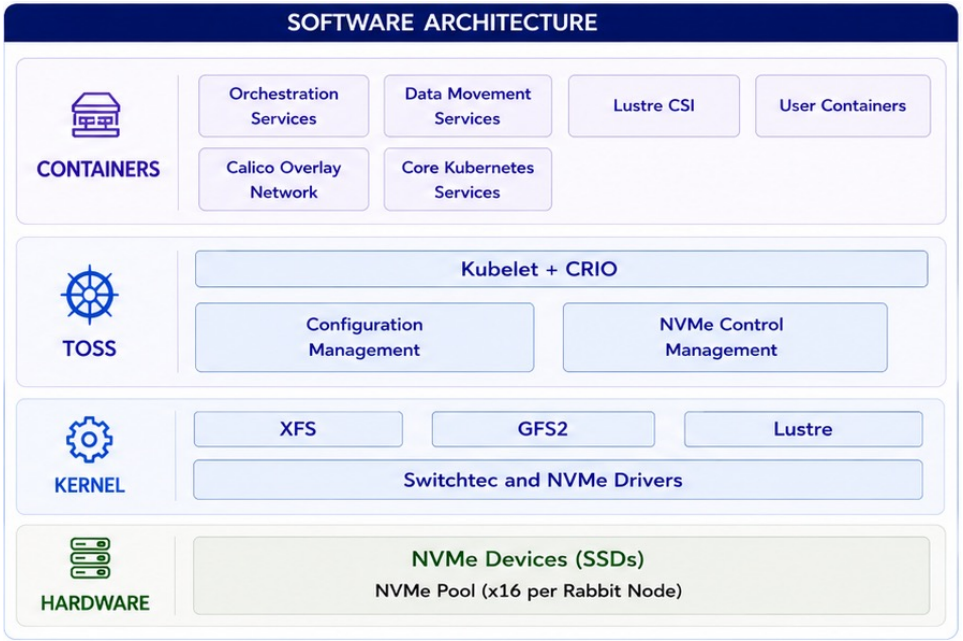
Dynamic. Flexible. Efficient.



Rabbits Architecture

High-Performance Hardware and Intelligent Software





Key Gaps in Research for Rabbits

Open questions that impact utilization



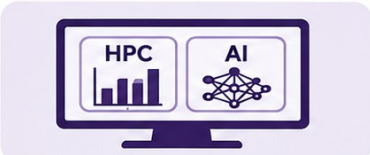
01 How will different file systems (XFS, GFS2, and Lustre) perform on Rabbit Hardware.



02 What are the optimal workload-centric Lustre tuning settings on Rabbits?



03 How do the rabbits scale across multiple nodes for shared block and file system?



04 What is the end-to-end Performance for realistic HPC and Al workloads?

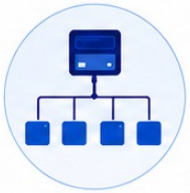


Key Research Question

" What are the **capabilities** and **performance characteristics** of Rabbits across software stacks, configurations, and workloads, and how we can **optimize** them?"

CORE CHALLENGES

1



Novel Hardware Architecture

Rabbits connect to a group of 16 compute nodes via both PCIe switches and network links.

→ Performance depends on communication path and file system used.

2



Complexity of the Software Layer

Same hardware can be accessed through XFS, GFS2, and Lustre—each with different data locality semantics, tuning options, and I/O behavior.

→ Right configuration is key to maximizing performance for different workloads.

3

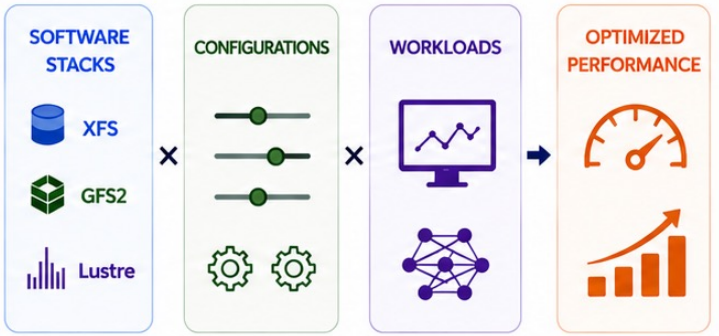


Dynamic Provisioning of Hardware

Resources can be allocated dynamically per job or persist across jobs, supporting flexible usage patterns.

→ Leads to varying performance due to heterogeneous and time-varying hardware contexts.

TOWARD OUR GOAL



A systematic and comprehensive evaluation to understand Rabbits' potential and optimize performance for data-intensive scientific workloads.

Experimental Setup

1 HARDWARE PLATFORM (Tuolumne @ LLNL)

System Overview

- Tuolumne system 1152 nodes
- Up to 256 nodes used in evaluation

1 Rabbit node per 16 compute nodes

Compute Node

4 × AMD MI300A APUs

4 × AMD MI300A APUs

Per MI300A APU

- 24 AMD Zen 4 x86 CPU cores
- 1 AMD CDNA 3 GPU
- 128 GB HBM3 Memory

Total per Compute Node

- 96 CPU cores (4 × 24)
- 4 CDNA 3 GPUs (4 × 1)
- 512 GB HBM3 Memory (4 × 128 GB)
- 4 × 200 Gb/s HPE Slingshot 11 NIC
- Global Lustre (baseline file system for all compute nodes)

Rabbit Node

- 16 × 2 TB NVMe Drives Configured 20 TB space
- 3.5 GB/s (peak per drive sequential R/W)
- Up to 56 GB/s aggregate sequential bandwidth
- 64-core AMD EPYC 7713 Processor
- 256 GB Memory
- 2 × 200 Gb/s Slingshot NICs

2 SOFTWARE CONFIGURATION (TOSS[4])

- OS: Operating System — TOSS[4]
- Compiler — AMD Compiler 6.2.4
- MPI — Cray MPICH 8.1.25
- Build System — CMake 3.29.2
- ROCm: ROCm Stack — ROCm 6.2.4 (for AMD MI300A GPUs)

3 WORKLOADS

IOR

- 4 MB read/write per process
- 16 MB total block size per process
- File-per-Process (FPP)
- POSIX interface
- Sequential access pattern

IO500

- Worldwide configuration
- Direct I/O
- 1 GB stack size
- 512B segmenter

DIO Benchmark

- Module: FEXT, Downloader
- Implemented in TensorFlow
- Training on TFX model datasets

Each test executed ten times to capture performance variability.

4 RABBIT CONFIGURATIONS

Local Access over PCIe (Node-Local)

XFS

- Namespace from each of the 16 NVMe drives
- Mounted exclusively on compute nodes

GFS2

- Namespace from each of the 16 NVMe drives
- Mounted concurrently on compute nodes

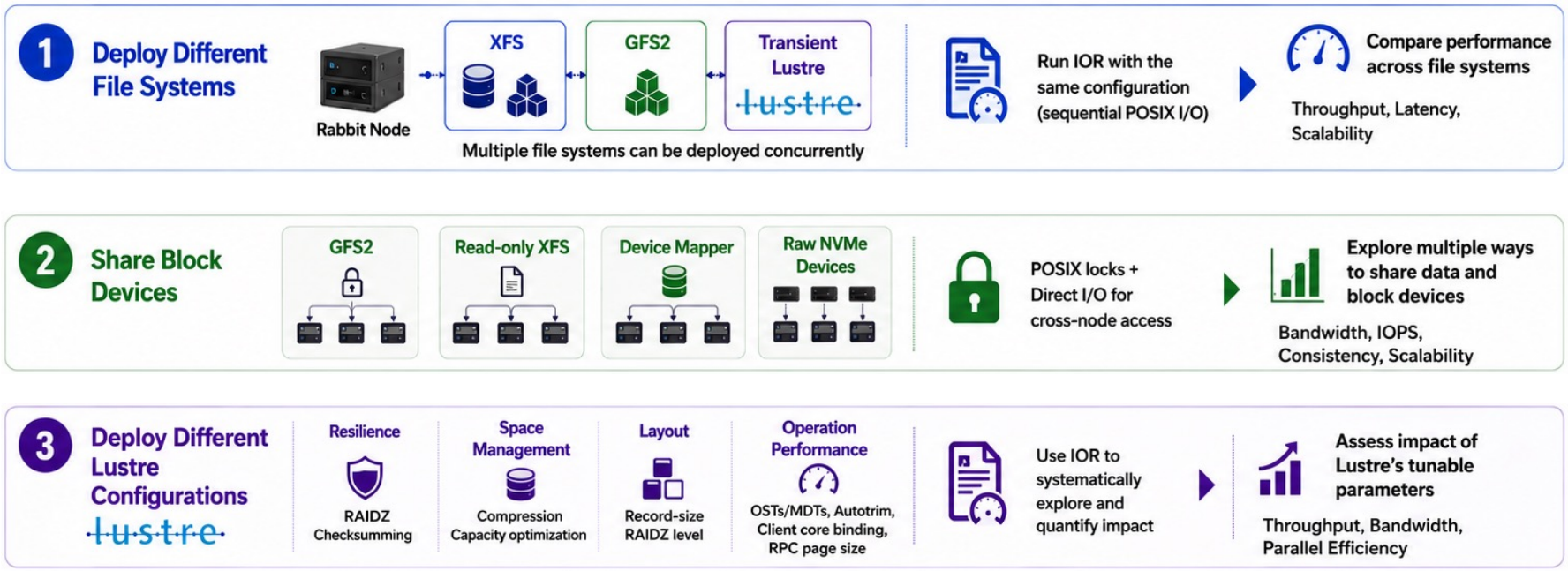
Distributed Access over Slingshot

lustre

- One OST across 16 NVMe drives
- Highly tunable (server & client)
- Deployed in two modes:
  - Transient — Scoped to lifetime of a single job
  - Persistent — Accessible across multiple jobs

Key Test Scenarios

Evaluating the capabilities and Performance Characteristics of Rabbit Storage System



Key Test Scenarios

Evaluating the capabilities and Performance Characteristics of Rabbit Storage System

4 Deploy Different Hardware Configurations

Transient Lustre
Deployed at job launch
Scales with compute nodes



Persistent Lustre
Pre-provisioned on fixed Rabbit nodes
Independent of job layout



Use IOR to compare transient vs. persistent deployments

Evaluate flexibility and performance at scale

Throughput, Scalability, Efficiency

5 Performance with IO500 and DLIO Benchmarks

IO500

- Mixed I/O workloads
- Data throughput (ior-easy/hard)
- Metadata (mdtest-easy/hard)
- Directory traversal (pfind)

DLIO

- BERT (compute-heavy)
- CosmoFlow (I/O-heavy)
- Implemented in TensorFlow

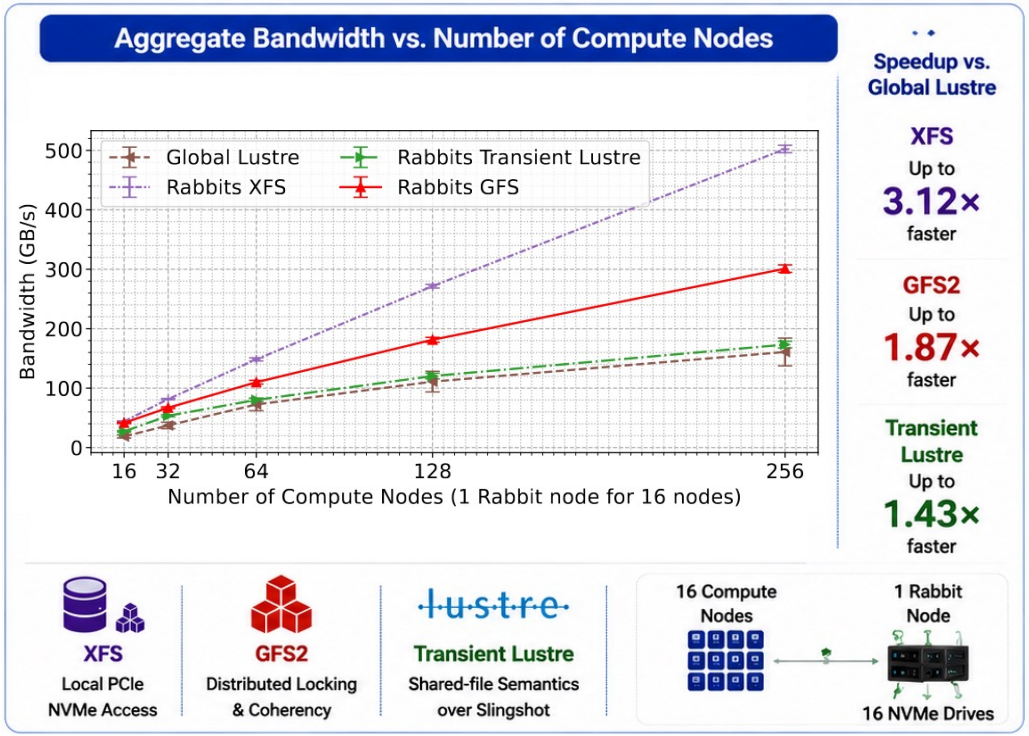
Evaluate across XFS, Transient Lustre, Persistent Lustre

Assess end-to-end performance for HPC & AI workloads

Throughput, IOPS, Variability, Time-to-Solution

Deploy Different File Systems

Scaling IOR (Sequential POSIX I/O) from 16 to 256 Nodes



Observations and Analysis

XFS is the fastest
65% to 3.12× higher bandwidth than Global Lustre due to local PCIe access, minimal hops, and file-per-process alignment.

GFS2 outperforms Global Lustre
by 1.5× to 1.87×, but is 5% to 66% slower than XFS due to distributed locking and coherency overheads.

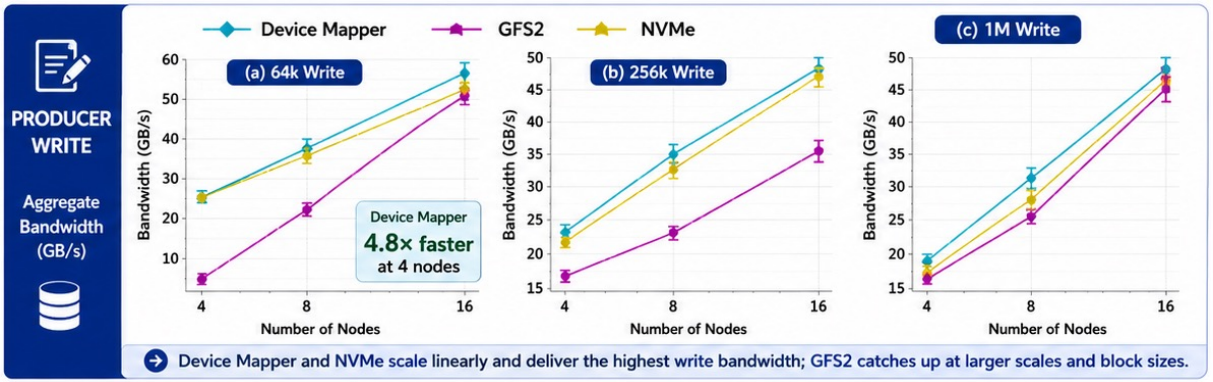
Transient Lustre provides 7% to 43% higher bandwidth than Global Lustre
but gains diminish with scale as metadata traffic grows with shared-file semantics.

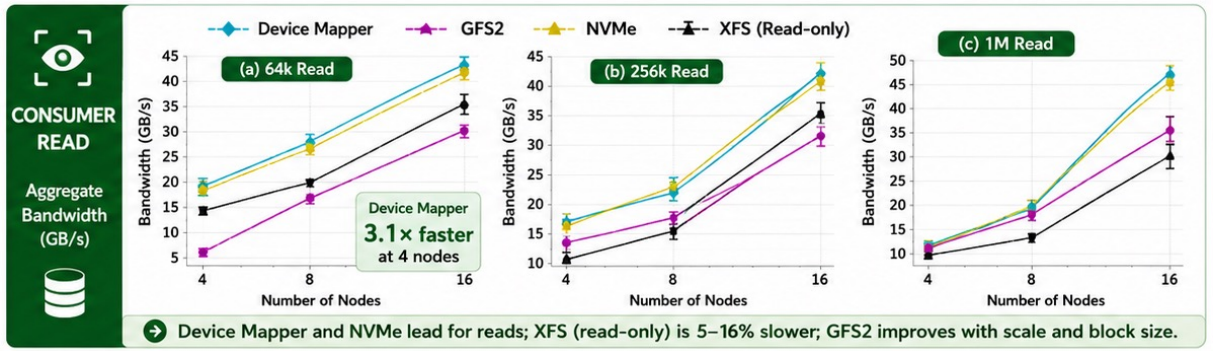
XFS > GFS2 > Lustre for sequential POSIX workloads
XFS avoids cross-node coordination, while GFS2 and Lustre pay for coherency.

At larger scales,
GFS2 benefits from more concurrency, and XFS benefits from cleaner client-side caching paths.

Shared Block Devices

Producer-Consumer I/O across 16 compute nodes.





Observations and Analysis

Producer Write Performance
Device Mapper ~ NVMe across all scales.
Near-linear scaling.
GFS2 is weaker at small scales but catches up at larger block sizes.

Consumer Read Performance
Device Mapper ~ NVMe at the top.
GFS2 lags at small scales.
XFS (read-only) is 5–16% slower than Device Mapper.

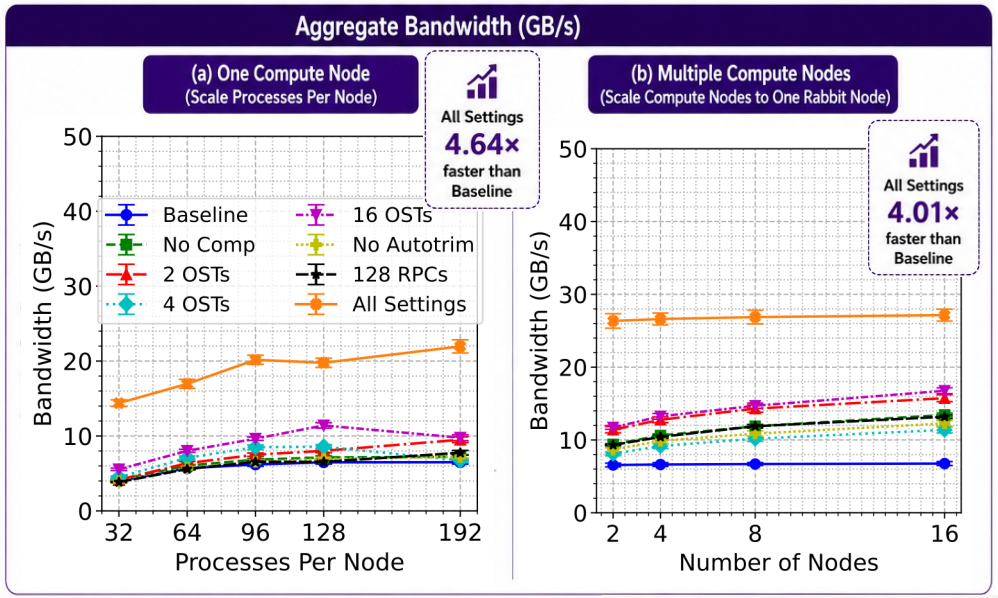
Comparative Analysis
- Device Mapper ≈ NVMe with minimal overhead (<5%).
- GFS2 converges at large scales and block sizes.
- XFS read-only has 5–19% penalties.

Scaling Behavior
Device Mapper & NVMe scale linearly.
GFS2 shows steeper curves at small block sizes due to coordination overhead.

Why It Matters
Raw block access eliminates metadata and synchronization overheads, maximizing I/O throughput.

Deploy Different Lustre Configurations

IOR (4MB I/O per process, 16GB total) with 192 processes per node





Observations and Analysis

Single Compute Node

- No Comp: 6–18% improvement
- More OSTs: 5–95% gains
- No Autotrim: 5–11% improvement
- 128 RPCs: helps at higher concurrency
- All Settings: 2.9×–4.64× over baseline

Multi-Node Scaling (2–16 Nodes)

- No Comp: 1.11×–1.98×
- More OSTs: 1.23×–2.47×
- No Autotrim: 11–81% improvement
- 128 RPCs: 19–95% improvement
- All Settings: 3.37×–4.01× over baseline
- Bandwidth saturates ~26 GB/s with just 2 nodes (NVMe/PCIe limit)

Why It Matters

Tuning reduces contention, increases parallelism, and removes overheads, maximizing sequential I/O bandwidth.

Workload Sensitivity

Small-block/high-concurrency workloads benefit most from RPC and OST tuning; large-block workloads are more bandwidth-limited.

Deploy Different Hardware Configurations

Comparing Transient vs. Persistent Lustre with IOR

Two Deployment Models

Transient Lustre

Job-specific Lustre file system instantiated and scaled with compute nodes

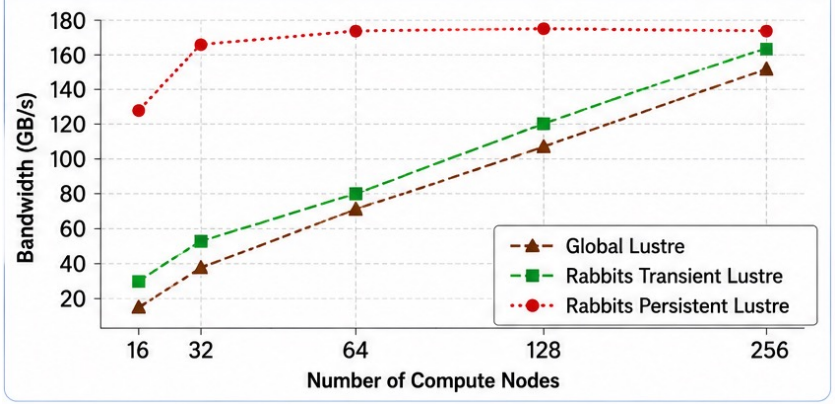
16:1 Compute-to-Rabbit Ratio

Persistent Lustre

Pre-provisioned Lustre file system spanning 16 Rabbit nodes (fixed)

Fixed 16 Rabbit Nodes

Aggregate Bandwidth vs. Number of Compute Nodes (IOR)



Observations and Analysis

Global Lustre: ≈160 GB/s with ±15% variability across scales due to external contention.

Transient Lustre: 1.07× to 1.21× faster (±5% variance); greatest gains at small scales where locality and NVMe dominate.

Persistent Lustre: 1.08× (large scales) to 47× (small scales) faster (±4% variance) with a fixed 16-Rabbit pool.
Scales well to 32 nodes, then saturates as Rabbit bandwidth becomes the limiting factor.

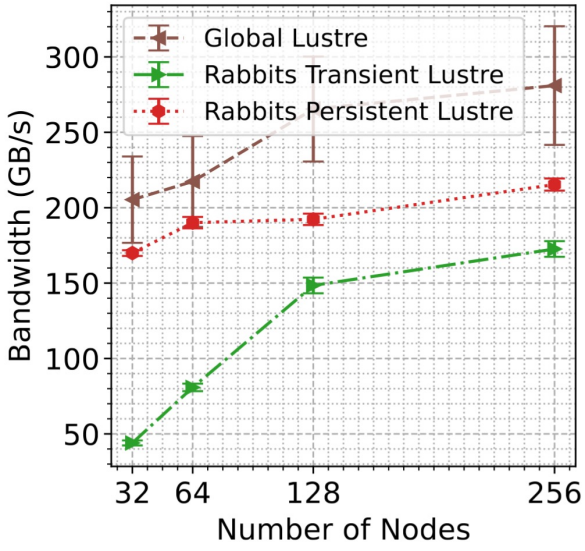
Transient and Persistent converge at the largest scale once they use the same Rabbit count.

Lower variability for Rabbit configurations reflects dedicated hardware and reduced system noise, enabling performance predictability.

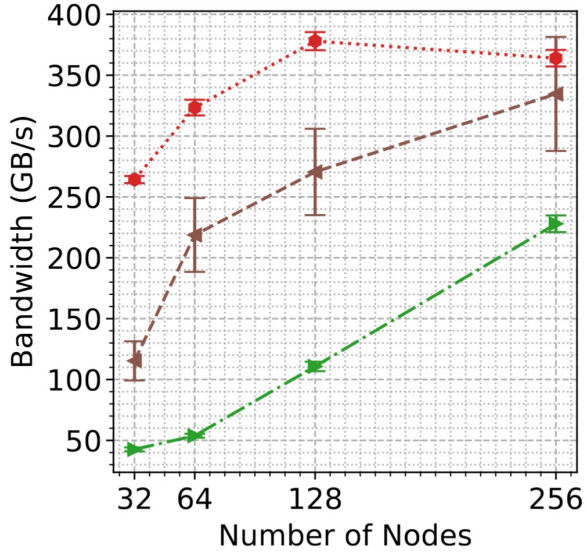
Hardware allocation, rather than software mode, is the primary driver of throughput at scale.

Performance with IO500: IOR Easy

Configurations: Global Lustre (Baseline) | Rabbit Transient Lustre (1 Rabbit node per 16 compute nodes) | Rabbit Persistent Lustre (16 Rabbit nodes)



(a) IOR Easy Write



(b) IOR Easy Read

Performance Highlights

IOR Easy Write (256 nodes)

- Global Lustre: **275 GB/s**
- Rabbits Transient Lustre: **170 GB/s**
- Rabbits Persistent Lustre: **215 GB/s**

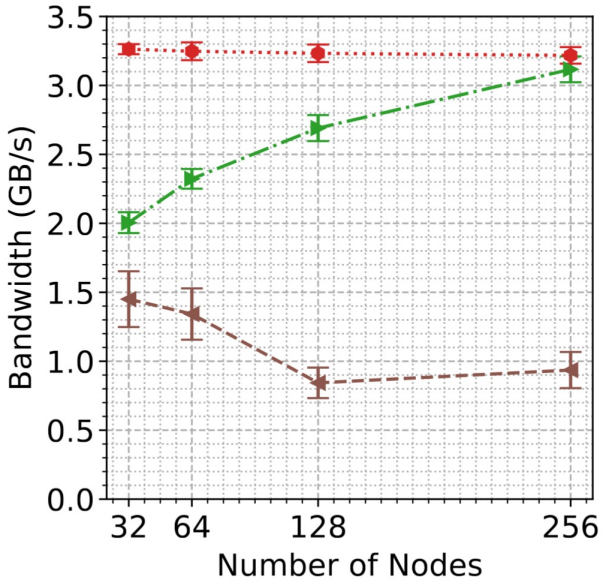
IOR Easy Read (256 nodes)

- Global Lustre: **330 GB/s**
- Rabbits Transient Lustre: **230 GB/s**
- Rabbits Persistent Lustre: **360 GB/s**

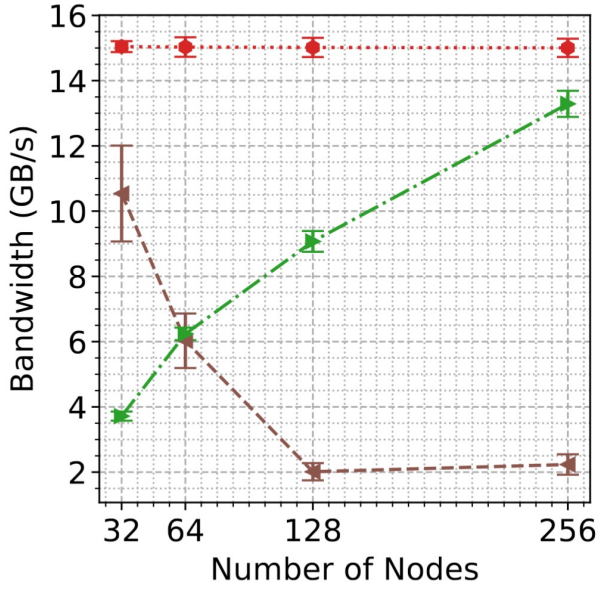
Rabbit Persistent Lustre delivers up to **1.6×** higher read bandwidth and up to **1.5×** higher write bandwidth at scale.

Performance with IO500: IOR Hard

Configurations: Global Lustre (Baseline) | Rabbit Transient Lustre (1 Rabbit node per 16 compute nodes) | Rabbit Persistent Lustre (16 Rabbit nodes)



(c) IOR Hard Write



(d) IOR Hard Read

Performance Highlights

IOR Hard Write (256 nodes)

- Global Lustre: **0.95 GB/s**
- Rabbits Transient Lustre: **3.15 GB/s**
- Rabbits Persistent Lustre: **3.25 GB/s**

IOR Hard Read (256 nodes)

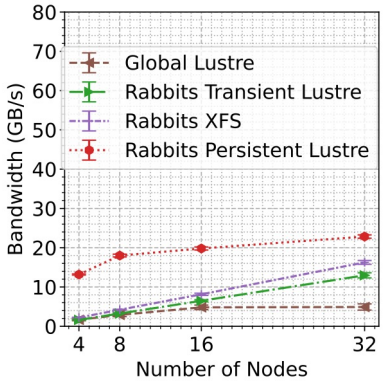
- Global Lustre: **2.10 GB/s**
- Rabbits Transient Lustre: **13.20 GB/s**
- Rabbits Persistent Lustre: **15.10 GB/s**

Rabbit Persistent Lustre delivers up to **7.2×** higher read bandwidth and up to **3.4×** higher write bandwidth at scale.

Performance with DLIO Benchmark

BERT

Compute-Heavy

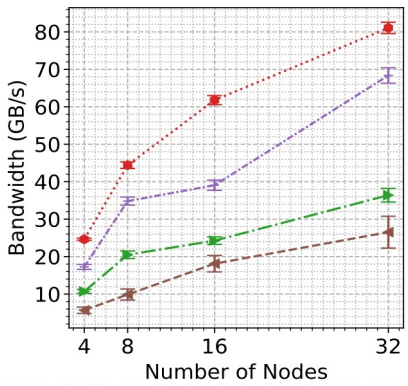


SPEEDUP @ 32 NODES

Global Lustre	1.0×
Rabbit XFS (16 GB/s vs 4.8 GB/s)	3.3×
Rabbit Transient Lustre (13 GB/s vs 4.8 GB/s)	2.7×
Rabbit Persistent Lustre (~23 GB/s vs 4.8 GB/s)	4.8×

CosmoFlow

I/O-Intensive



SPEEDUP @ 32 NODES

Global Lustre	1.0×
Rabbit XFS (68 GB/s vs 150 GB/s)	2.5×
Rabbit Transient Lustre (102 GB/s vs 150 GB/s)	3.4×
Rabbit Persistent Lustre (175.8 GB/s vs 150 GB/s)	5.8×

KEY TAKEAWAYS

BERT (compute-bound): benefits from moderate I/O improvements; sub-linear gains.

CosmoFlow (I/O-bound): scales with more Rabbit nodes; Persistent Lustre leads.

XFS scales linearly with local reads; Transient Lustre offers throughput–resource trade-offs.

Conclusion

First systematic study of Rabbit Storage System

1 SOFTWARE-DEFINED CONFIGURABILITY

Deploy different file systems and tune parameters for workload-specific optimization.

XFS | GFS2 | Lustre

UP TO **3.12×** higher bandwidth than global Lustre (IOR)

2 TUNED LUSTRE MAXIMIZES PERFORMANCE

Tuning configurations (OST counts, compression, RPC settings) delivers substantial gains.

OST Counts | Compression | RPC Settings | Other Tuning

UP TO **4.6×** improvement over baseline

26 GB/s per Rabbit node (combined optimized)

3 STRONG END-TO-END PERFORMANCE

Benchmark evaluation shows Rabbit delivers exceptional I/O and metadata performance across HPC and AI workloads.

IO500

Peak Sequential Read Bandwidth

413 GB/s (Persistent Lustre)

METADATA PERFORMANCE

3.3×–7× better than global Lustre

DLIO WORKLOADS (Persistent Lustre)

BERT **22.83 GB/s**

CosmoFlow **175.8 GB/s**

Rabbit is a VERSATILE ACCELERATOR for modern workloads,

bridging performance gaps in I/O-INTENSIVE SIMULATIONS | ANALYTICS | AI PIPELINES

THANK YOU!
QUESTIONS?



RABBIT
STORAGE SYSTEM

Hariharan Dev
hariharandev1@llnl.gov

KEY TAKEAWAYS

1 SOFTWARE-DEFINED FLEXIBILITY
Deploy different file systems (XFS, GFS2, Lustre) and tune parameters for workload-specific optimization.

Up to **3.12×** higher bandwidth than global Lustre (IOR tests)

2 TUNED LUSTRE, MAXIMUM PERFORMANCE
Tuning Lustre configurations (e.g., OST counts, compression, RPC settings) boosts performance significantly.

Up to **4.6×** improvement over baseline

Up to **26 GB/s** per Rabbit node

3 STRONG END-TO-END PERFORMANCE
Benchmark evaluation shows Rabbit excels across IO500 and DLIO workloads.

- IO500 peak sequential read: **360 GB/s** (Persistent Lustre)
- Metadata performance: **3.3× – 7×** better over global Lustre
- DLIO workloads: up to **22.83 GB/s** (BERT) and **175.8 GB/s** (CosmoFlow) with Persistent Lustre

