Analytics4X: General-Purpose Framework for Analysis and Optimization of HPC Data Movement

Ian Lumsden[1] (Student), Michela Taufer[1] (PhD Advisor)

Other Mentors: Stephanie Brink[2], Olga Pearce[2], Hariharan Devarajan[2], Jae-Seung Yeom[2], Tom Scogland[2]

[1]University of Tennessee, Knoxville [2]Lawrence Livermore National Laboratory

Dissertation Statement

To continue to accelerate scientific discovery in the exascale era and beyond, we need a general-purpose, adaptable analytic framework for optimizing data movement in both monolithic and modular workflow-based applications.

To design this tool, we first aim to understand and optimize I/O and data movement across diverse HPC applications:

- 3 applications (i.e., AMG2013, MuMMI, iPIC3D)
- 3 adapted tools (i.e., Caliper, Thicket, IOR benchmarking)
- 3 lessons learned (i.e., reveal data movement patterns, align I/O with workflow execution, use phase-aware monitoring)

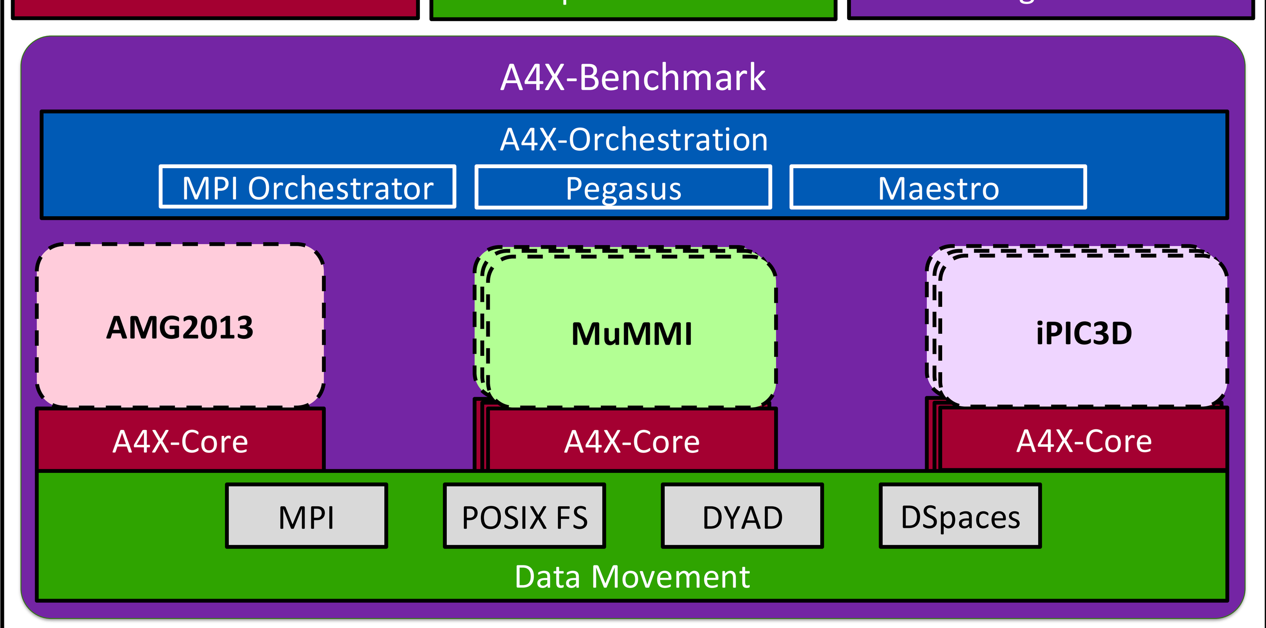
We integrate features derived from these lessons learned into a unified Analytics4X framework that supports diverse application types and I/O patterns

Community Challenges / Our Solutions

Community Challenges	Our Solutions
Profiling tools can capture compute performance well, but they lack the ability to filter and focus on data movement and I/O layers at scale	Develop and integrate a novel Call Path Query Language into LLNL's Thicket performance analysis library to enable complex filtering of performance profiles and focused analysis on data movement and I/O layers
Workflow applications introduce complex and irregular I/O patterns that are often not well-optimized, causing data movement inefficiencies and longer time-to-solution	Instrument workflow execution with middleware-level I/O tracking using Thicket to capture detailed I/O patterns and guide workflow-aware optimizations
In simulations, I/O phases vary in frequency, size, and access pattern; static storage configurations fail to provide optimal performance for all phases, causing bottlenecks	Deploy performance monitoring and IOR benchmarking to identify application I/O phases and dynamically match them to the most suitable storage configurations

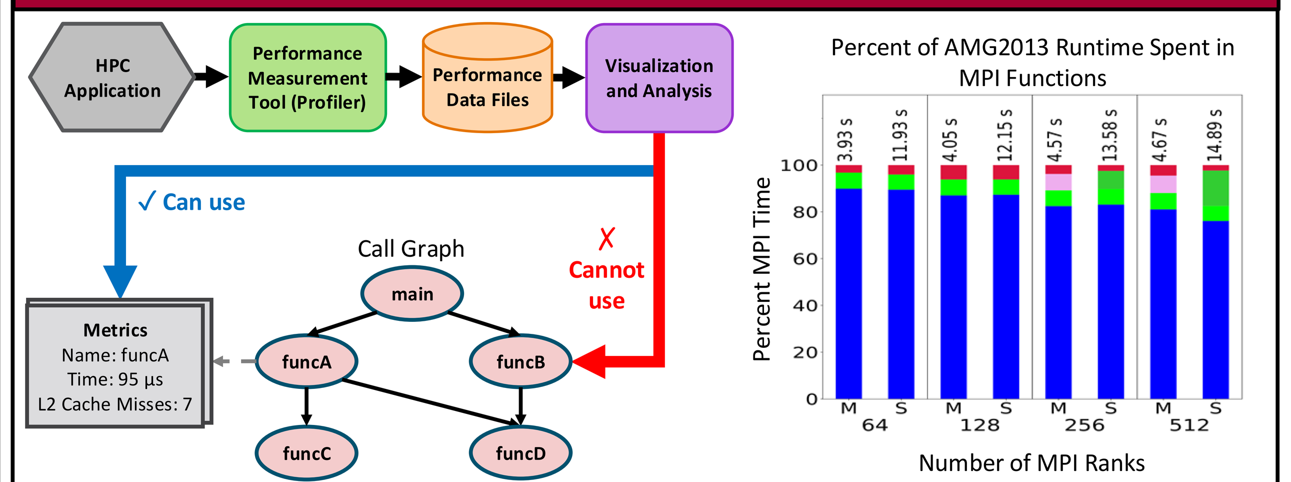
Analytics4X Framework

Feature:	**Feature:**	**Feature:**
Support for fine-grained I/O layer filtering within performance profiles	Built-in capability to integrate middleware-level performance data into workflow optimization	Automated phase detection and benchmark-driven mapping to optimal storage configurations

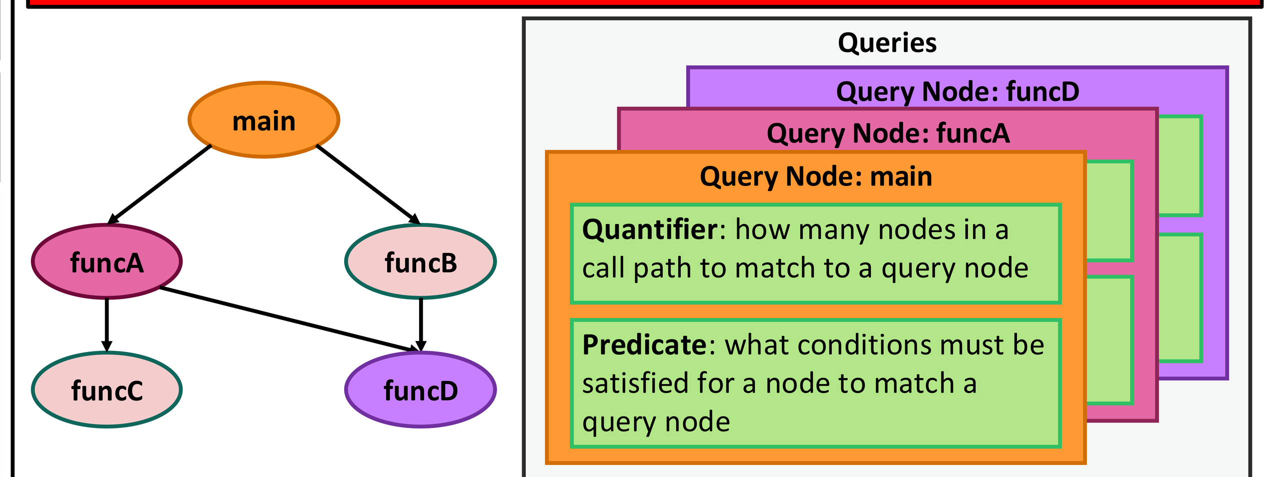


A4X-Core:	**A4X-Orchestration:**	**A4X-Benchmark:**
Common abstractions and built-in performance monitoring for data movement tools	Common abstractions for configuring different workflow management systems	Benchmark comprised of proxies representing common data movement motifs in applications

AMG2013: Proxy for Fluid Dynamics



Profiling tools can capture compute performance well, but they lack the ability to filter and focus on data movement and I/O layers at scale



Query Example: Find all subgraphs rooted at a MPI node with more than 5 L2 cache misses

Base Syntax

```
query = QueryMatcher().match(
    ".",
    lambda row: re.match(
        "MPI_.*",
        row["name"] )
    is not None
    and row["PAPI_L2_TCM"] > 5
).rel("*")
```

- \+ Support any query
- \- Require Python libs knowledge
- \- Work with Python only

Object-based Dialect

```
query = [
    (".", {
     "name": "MPI_.*",
     "PAPI_L2_TCM": "> 5"
    }),
    "*"
]
```

- \+ Use built-in Python objects
- \- Support limited queries
- \- Work with Python only

String-based Dialect

```
query = """
MATCH (".", p)->("*")
WHERE p."name" =~ "MPI_.*"
AND p."PAPI_L2_TCM" > 5
"""
```

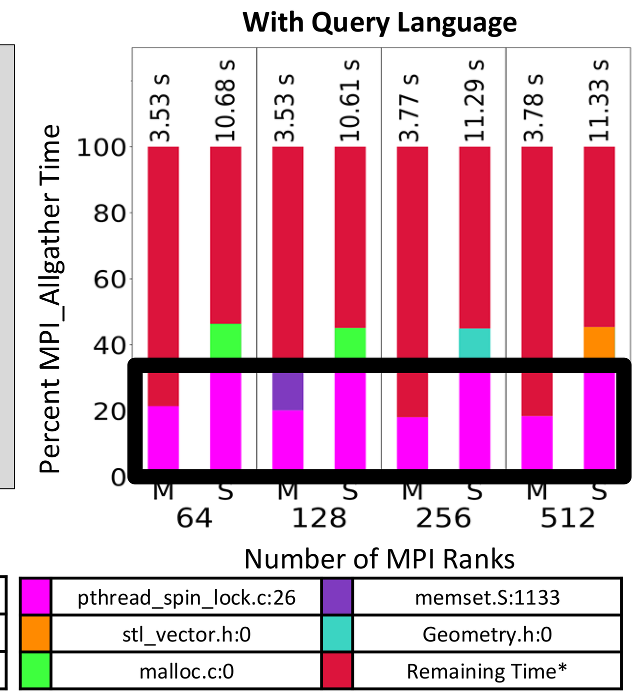
- \+ Work with any language
- \- Support limited queries

We develop a novel Call Path Query Language and integrate it into LLNL's Thicket performance library to enable complex filtering of profiles using the call graph

We apply the following query to the profiles of AMG2013 to identify the bottlenecks *within* MPI_Allgather:

```
query = [
  (".", {
    "name": "P?MPI_Allgather"
  }),
  "*"
]
```

M = MVAPICH S = Spectrum-MPI



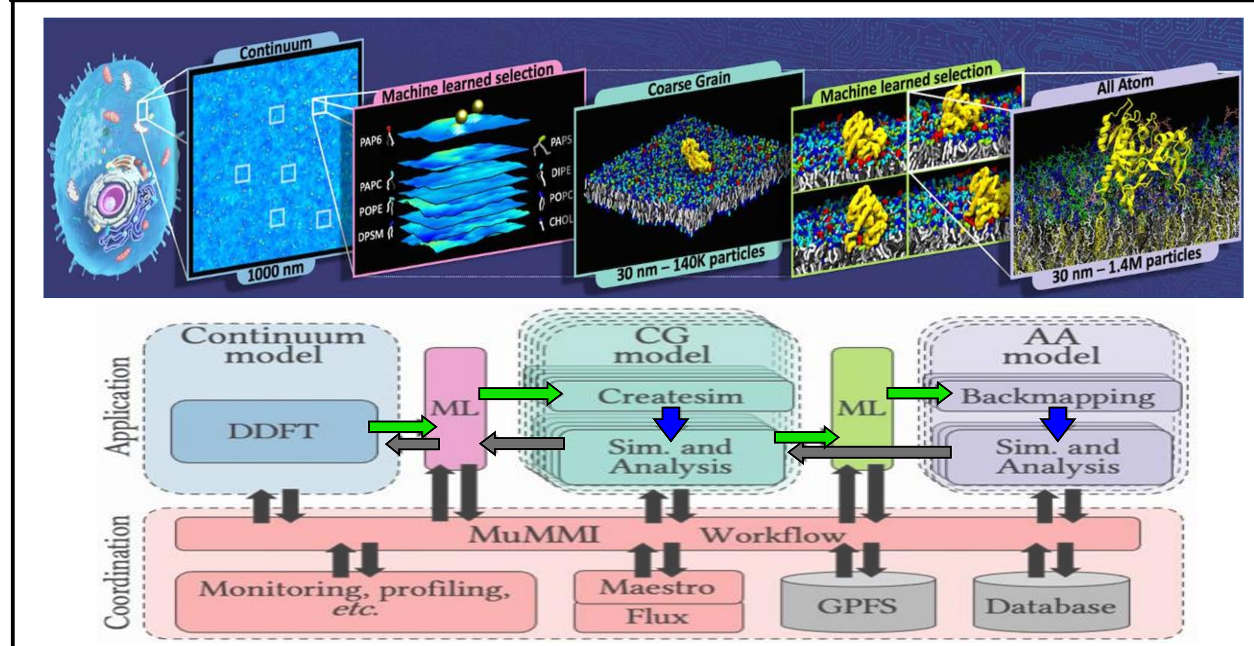
* "Remaining Time" categories = functions that take up less than 5% of the total time considered

By applying the query language to runs of AMG2013 with both MVAPICH and Spectrum-MPI, we can not only identify that MPI_Allgather is the largest bottleneck, but we can also identify that pthread_spin_lock is the largest single bottleneck *within* MPI_Allgather

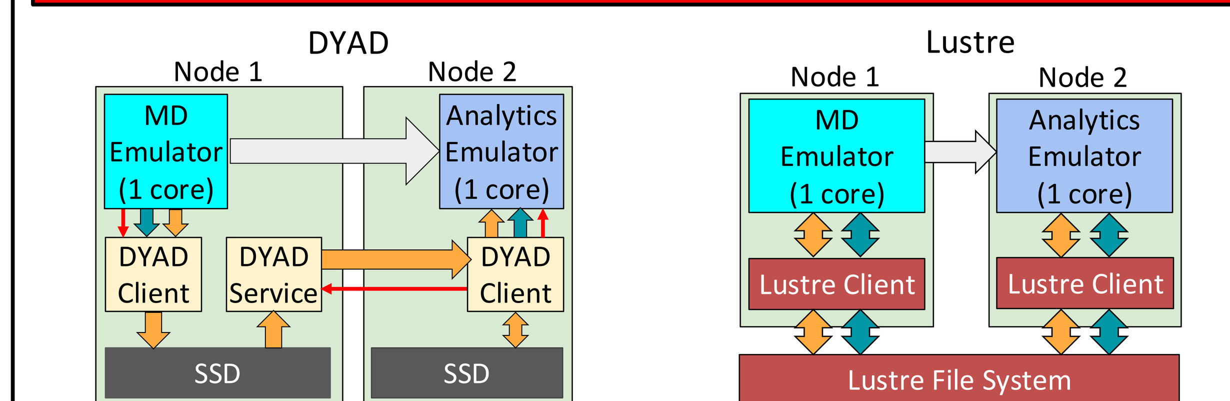
Lesson Learned:

Profile-level performance analysis can be extended to reveal detailed data movement patterns

MuMMI: Workflow for Molecular Dynamics



Workflow applications introduce complex and irregular I/O patterns that are often not well-optimized, causing data movement inefficiencies and longer time-to-solution

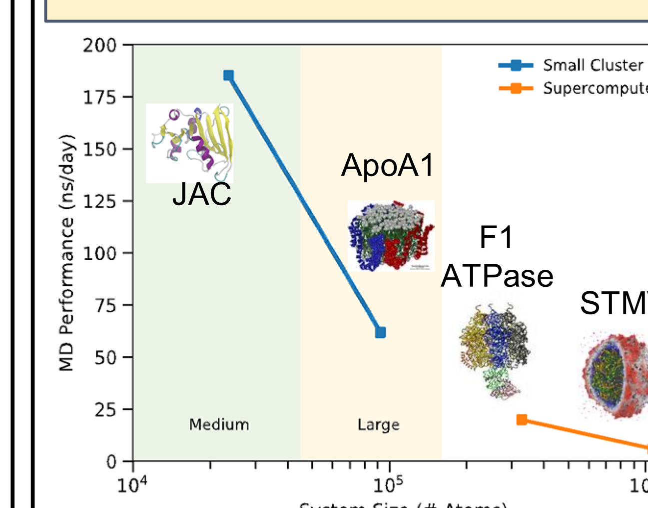


DYAD

- Pros:
  - Provides easy use of local storage
  - Provides built-in sync
- Cons:
  - Supports only write-once, read-many I/O

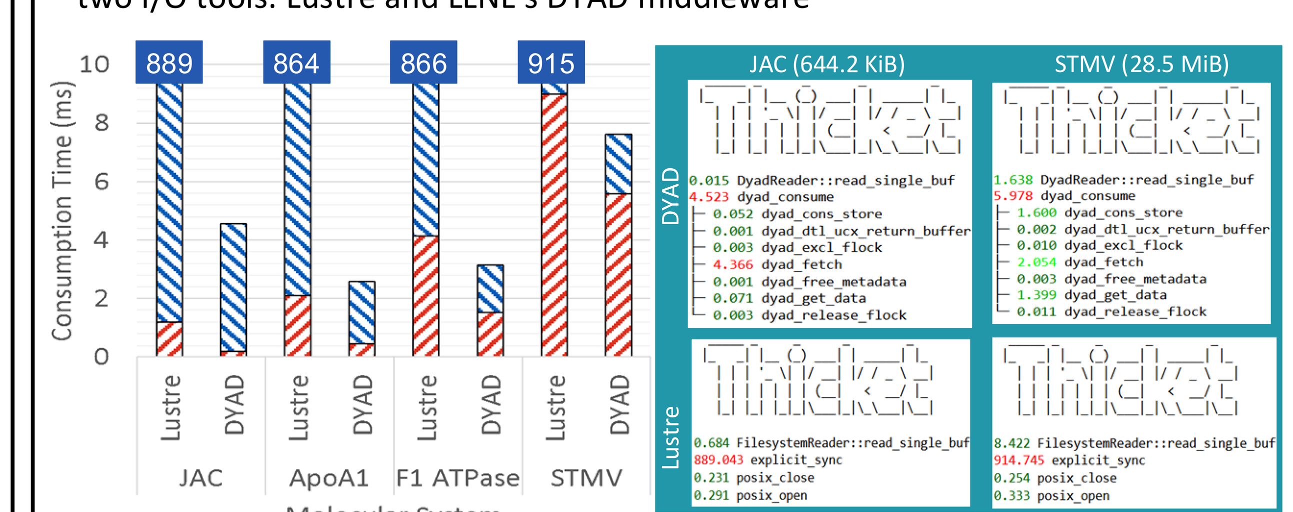
Lustre

- Pros:
  - Provides high throughput for large, bulk-synchronous I/O
- Cons:
  - Struggles with small or unsynchronized I/O
  - Does not provide built-in sync



JAC: • 23.5K atoms • 644.2 KiB Frames • 880 stride	ApoA1: • 92.2K atoms • 2.5 MiB Frames • 294 stride
F1 ATPase: • 327.5K atoms • 8.8 MiB Frames • 92 stride	STMV: • 1.1M atoms • 28.5 MiB Frames • 28 stride

We instrument a molecular dynamics workflow with an ensemble of one-to-one I/O patterns using LLNL's Caliper profiler and analyze the resulting performance data with Thicket and our Call Graph Query Language to study the behavior of two I/O tools: Lustre and LLNL's DYAD middleware

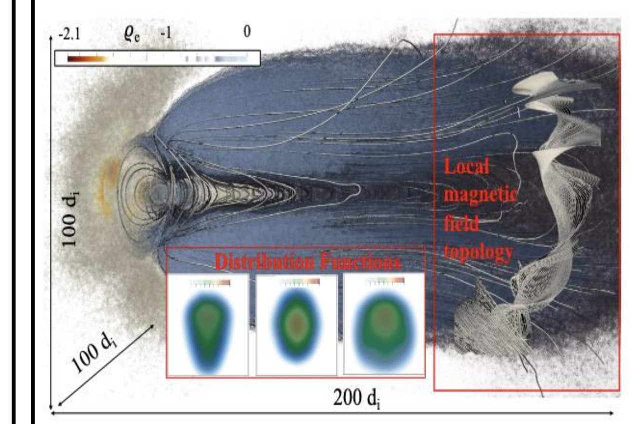


We find that leveraging local resources and efficient communication protocols enables better scalability as data sizes (represented by molecular model) increase

Lesson Learned:

Middleware can substantially improve data movement efficiency for workflows by aligning I/O with workflow execution patterns

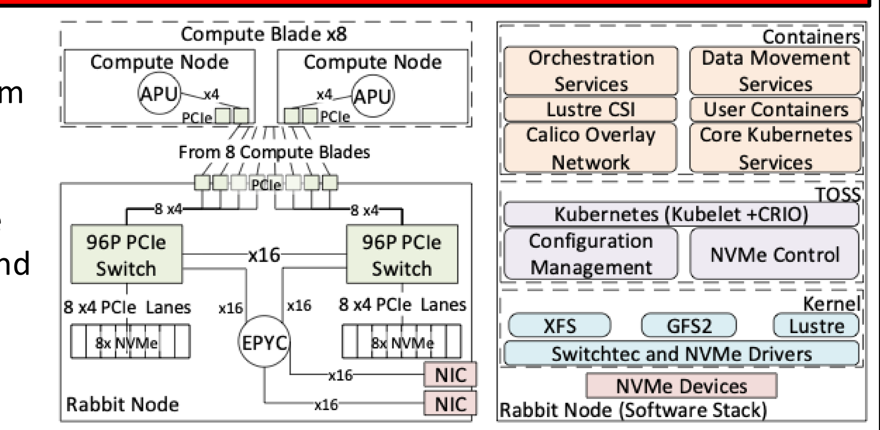
iPIC3D: Simulation of Planetary-Scale Plasma Physics



High-Level I/O Characteristics of iPIC3D	
Job Time	766.8s
I/O Time	283.1s
I/O Phases	Restart Data, Field Data, Moment Data

In simulations, I/O phases vary in frequency, size, and access pattern; static storage configurations fail to provide optimal performance for all phases, causing bottlenecks

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

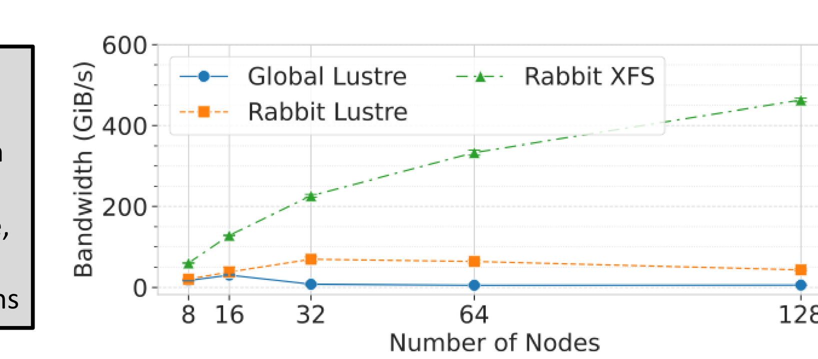


We deploy LLNL's Caliper and DFTracer performance monitoring tools, LLNL's DFAnalyzer performance analysis tool, and IOR benchmarking to identify iPIC3D's I/O phases and dynamically match them to the most suitable storage configurations on LLNL's Tuolumne supercomputer

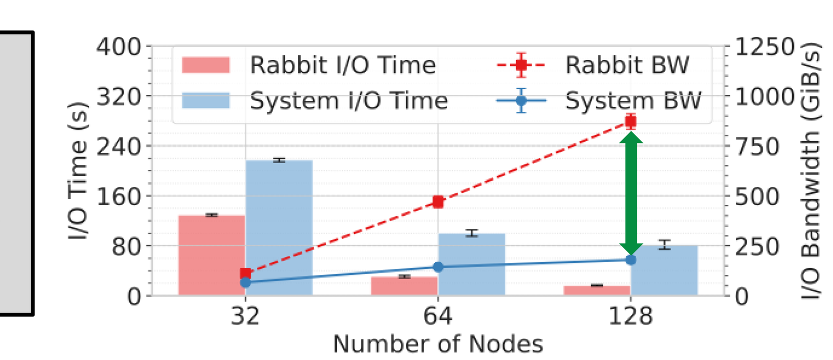
Step 1: Analyze iPIC3D's I/O to identify dominant phases and quantify their performance

	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

Step 2: Benchmark each phase with IOR on the Rabbit storage system and the system-wide, global Lustre to determine optimal storage configurations



Step 3: Integrate the optimized storage configurations back into iPIC3D



By mapping each I/O phase of iPIC3D to its optimal Rabbit configuration based on IOR benchmarking, we achieve up to a 4.85x I/O throughput improvement and up to a 1.45x overall application speedup

Lesson Learned:

Matching I/O phases to targeted storage systems can yield substantial performance gains, but requires phase-aware monitoring and tuning

eScience 2022 Paper:	IPDPS 2024 Workshop Paper:	Cluster 2025 Workshop Paper:	CV:

We acknowledge the feedback and support of the following colleagues: Andong Hu (KTH Royal Institute of Technology, USA), Anthony Kougkas (Illinois Institute of Technology, USA), Bronis R. de Supinski (Lawrence Livermore National Laboratory, USA), Connor Scully-Allison (University of Chicago, USA), David Boehme (Lawrence Livermore National Laboratory, USA), Dewi Yokelson (Advanced Micro Devices, Inc., USA), Gian Luca Delzanno (Los Alamos National Laboratory, USA), Ivy Peng (KTH Royal Institute of Technology, USA), Izzet Yildirim (Illinois Institute of Technology, USA), Jack Marquez (University of Tennessee, USA), Jakob Luettgau (Inria, France), Katherine E. Isaacs (University of Utah, USA), Luca Pennati (KTH Royal Institute of Technology, USA), Stefano Markidis (KTH Royal Institute of Technology, USA), Vanessa Lama (Oak Ridge National Laboratory, USA), Xian-He Sun (Illinois Institute of Technology, USA).

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-POST-2010147). This material is based upon work supported by the U.S. National Science Foundation under Award No. 1841758, 1900888, 2138811, and 2331152.

FIND OUT MORE AT: globalcomputing.group