Towards End-to-End I/O Analysis and Optimization of HPC Systems

Hammad Ather
Advisors: Hank Childs & Allen D. Malony

UNIVERSITY OF OREGON
Lawrence Livermore National Laboratory
BERKELEY LAB

ABSTRACT

I/O remains a major performance bottleneck in HPC systems, especially with the growing complexity of AI, ML, and data-intensive workloads. Existing optimization tools struggle due to a disconnect between collected trace data and actionable insights, and often rely on costly, non-scalable tuning methods. This work proposes two solutions: (1) an interactive I/O visualization framework to bridge trace analysis and optimization, and (2) a lightweight runtime workflow that enables on-the-fly I/O tuning without prior training or profiling. Together, these approaches reduce tuning overhead and enhance I/O performance for scientific and AI workloads.

INTRODUCTION

High-performance computing (HPC) enables large-scale simulations and data processing at unprecedented speed, driving scientific innovation. However, achieving peak performance is challenging due to bottlenecks across subsystems.

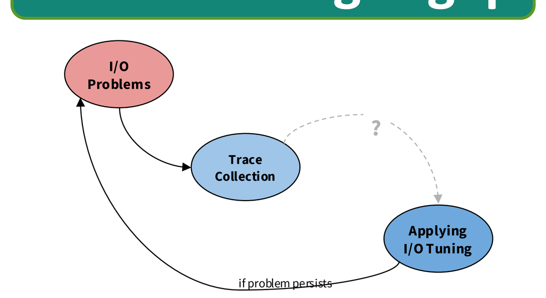
Central challenge: I/O subsystem is often a performance bottleneck in HPC systems

Why does I/O bottleneck occur?

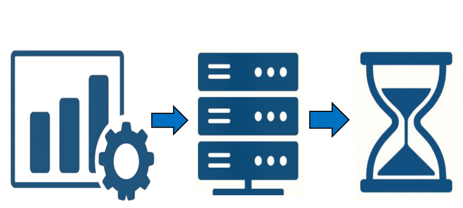
- Unbalanced workloads
- Resource contention
- Poor cross-layer tuning

Key limitations in state-of-the-art

Trace-to-insight gap



High Tuning Cost

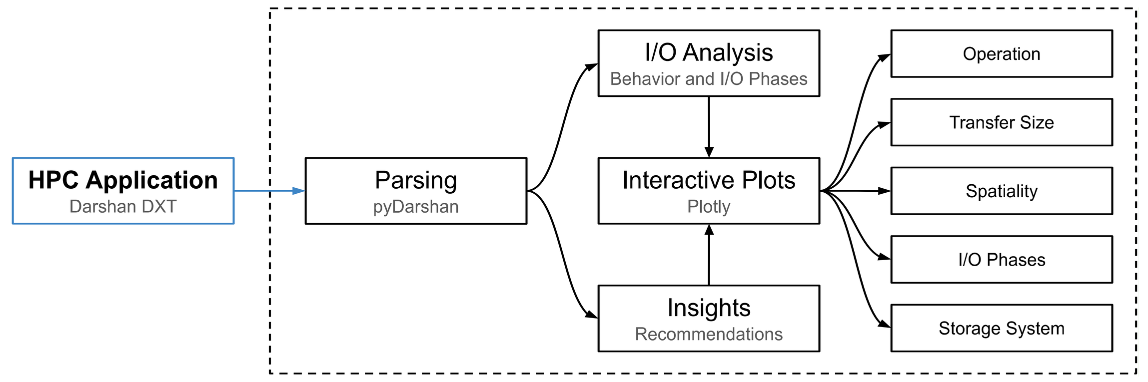


In this work, I focus on ***TWO*** thrusts to analyze and optimize I/O performance

INTERACTIVE I/O VISUALIZATION

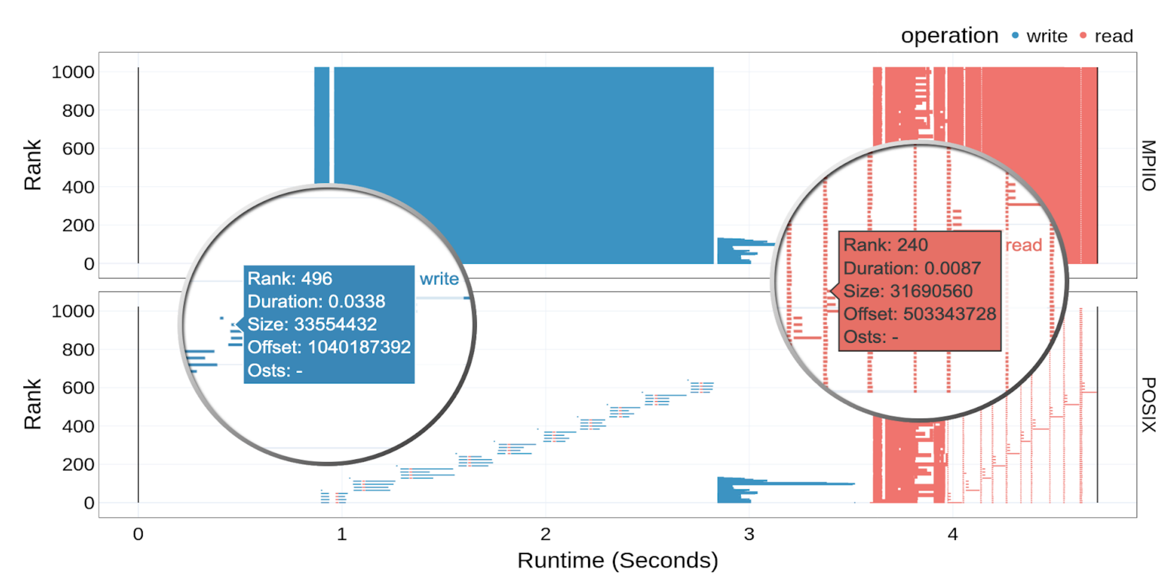
Propose a novel interactive, user-oriented visualization, and analysis framework called ***DXT-Explorer***

- Extract I/O behavior from log data
- Visualize large-scale I/O trace data
- Provide actionable recommendations

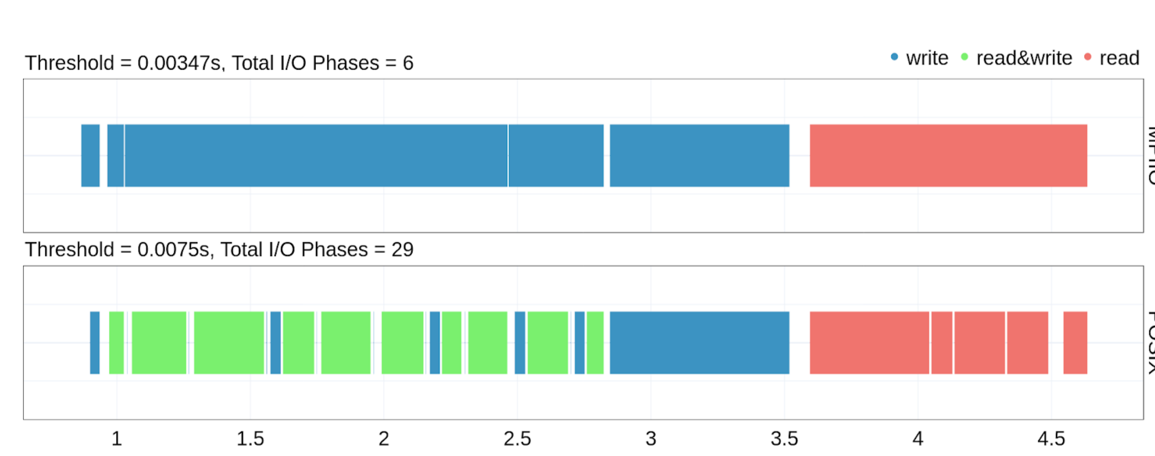


DXT Explorer generates meaningful interactive visualizations and a set of recommendations based on the detected I/O bottlenecks using Darshan DXT I/O trace

github.com/hpc-io/dxt-explorer



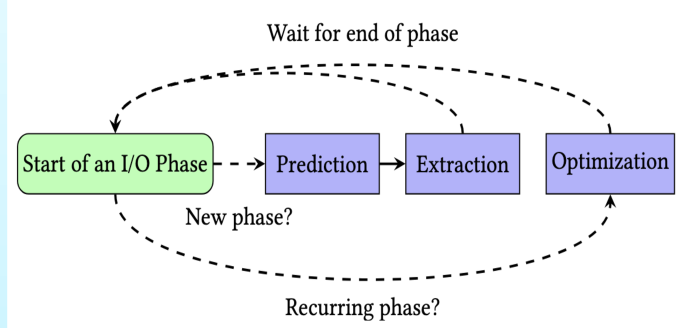
DXT-Explorer also focuses on other facets of I/O behavior such as **transfer sizes** and **spatial locality** of requests. Combined, they provide a clear picture of the I/O access pattern and help identify root causes of performance problems.



Interactive I/O phases visualization in MPI-I/O and POSIX layers

RUNTIME I/O OPTIMIZATION

Propose a framework called ***SmartIO*** that combines three components (**prediction, extraction, optimization**) together into an end-to-end runtime workflow.



No **prior profiling.** ***No*** **model training.** ***No*** **exhaustive parameter searches!**

Prediction

Uses context-free grammars to detect recurring patterns and predict future I/O calls from the current point of execution

Extraction

Extracts I/O behavior from predicted calls

Library	Insight	Function	Key Parameter	Inter-Process Comm.
HDF5	Dataset creation plist ID	H5Pcreate	H5P_class_t type	❌
	File access plist ID	H5Fcreate	hid_t access_id	❌
	File name	H5Fcreate	const char* name	❌
	File operation	N/A	N/A	✅
MPI-IO	Collective write	MPI_File_write_at_all	N/A	✅
	Independent write	MPI_File_write_at	N/A	✅
	Collective read	MPI_File_read_at_all	N/A	✅
	Independent read	MPI_File_read_at	N/A	✅
POSIX	Transfer size	write/read	size_t count	❌
	Spatial locality	write/read	off_t offset	❌

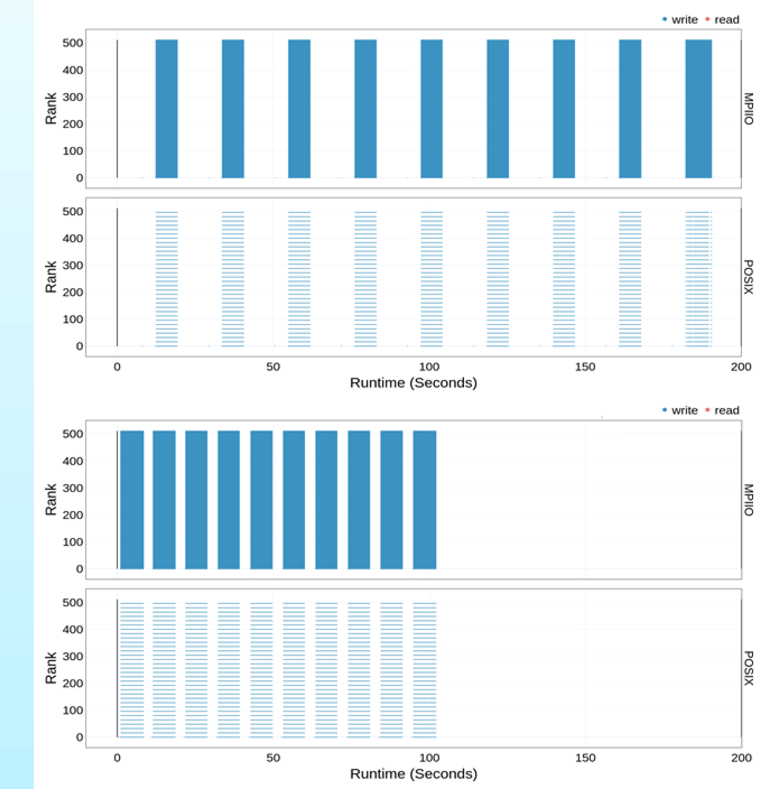
Optimization

Applies runtime tuning rules for HDF5, ROMIO, Lustre using a rules-based system

Optimization	Rule	Evaluation
HDF5 Data Transfer Mode	1. Use Independent I/O for sequential reads/writes to a shared file	❌
	2. Use Collective I/O for random or non-sequential reads/writes to a shared file	❌
	3. For file-per-process configurations, the data transfer mode should be left unchanged	✅
HDF5 Alignment	4. Set alignment between 1–16MB if transfer size < 16MB, otherwise set it >= 16MB	❌
HDF5 Metadata Cache	5. Set metadata cache size between 1–16MB if transfer size < 16MB, otherwise set it >= 16MB	❌
ROMIO Collective Buffering	6. Use cb_config_list to increase aggregators per node when performing collective buffering	✅
	7. Disable romio_cb_write for sequential accesses with a shared file, otherwise keep default	✅
ROMIO Data Sieving	8. Keep default values for the data sieving parameters	✅
Lustre Stripe Count	9. For file-per-process configurations, set stripe count to 1	❌
	10. For a shared file, set a progressive stripe layout if Lustre version > 2.10; otherwise, set stripe count according to file size	✅

EXPERIMENTS

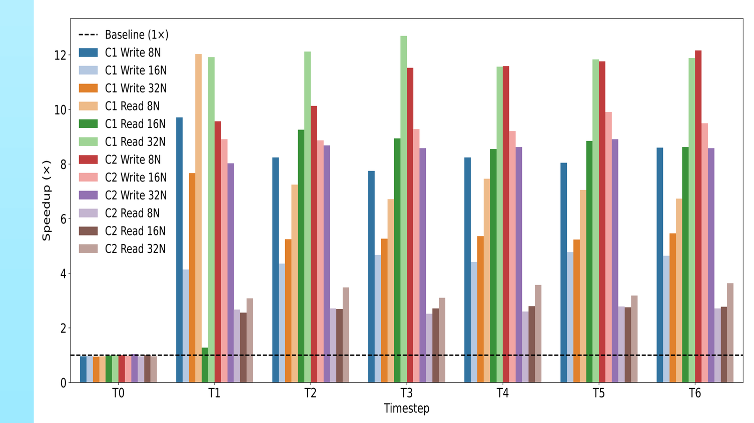
Conducted experiments on **Cori** and **Summit** using ***DXT-Explorer*** with OpenPMD and AMReX



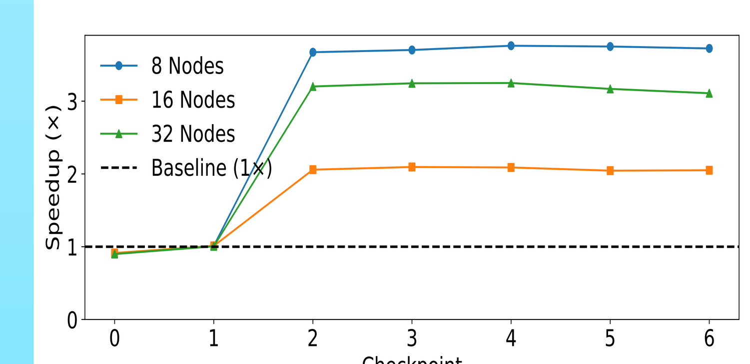
Optimizations

- Added **ASYNC I/O VOL** Connector
- Set the **stripe size** to 16MB

Conducted experiments on **Ruby** and **Lassen** using ***SmartIO*** with IOR and Flash-X



Real-time read and write bandwidth speedup relative to the baseline at each timestep on two systems (C1 and C2) using ***SmartIO*** on IOR



Real-time I/O bandwidth speedup relative to the baseline at each checkpoint on C1 using ***SmartIO*** on Flash-X

CONCLUSION AND FUTURE WORK

- Developed an interactive visualization tool to diagnose I/O bottlenecks at scale.
- Designed a lightweight runtime workflow for on-the-fly I/O tuning without prior profiling.
- Future work: Integrate **LLMs** to learn optimization rules from trace data and research papers.