A Multifaceted Approach to Automated I/O Bottleneck Detection for HPC Workloads

Izzet Yildirim Hariharan Devarajan Anthony Kougkas Xian-He Sun Kathryn Mohror
iyildirim@hawk.iit.edu, hariharandev1@llnl.gov, akougkas@iit.edu, sun@iit.edu, and kathryn@llnl.gov

Lawrence Livermore National Laboratory
ILLINOIS INSTITUTE OF TECHNOLOGY

Abstract

Context

Real-world HPC workloads impose a lot of pressure on storage systems as they are highly data dependent. On the other hand, as a result of recent developments in storage hardware, it is expected that the storage diversity in upcoming HPC systems will grow. This growing complexity in the storage system presents challenges to users, and often results in I/O bottlenecks due to inefficient usage. There have been several studies on reducing I/O bottlenecks. The earliest attempts worked to solve this problem by combining I/O characteristics with expert insight. The recent attempts rely on the performance analysis from the I/O characterization tools. However, the problem is multifaceted with many metrics to consider, hence difficult to do manually, even for experts.

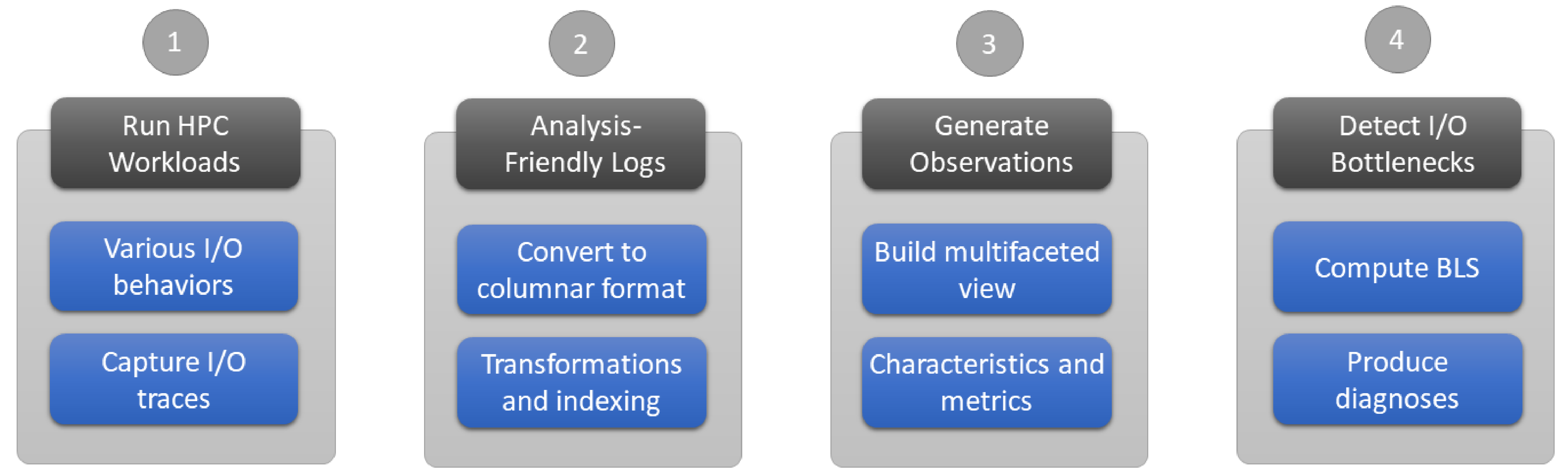
Main Contributions

- A methodology to detect I/O bottlenecks.
- An automated tool, called *Diagnose I/O* (DigIO), which automates the I/O bottleneck detection.
- Demonstration of I/O bottleneck detection.

Poster QR

Tool QR

Overview of DigIO



- Run HPC workloads with various I/O behaviors to simulate real-world scenarios, including CM1, HACC, CosmoFlow, JAG ICF, and Montage.
- Capture I/O traces using Recorder, a multi-level tracing library.

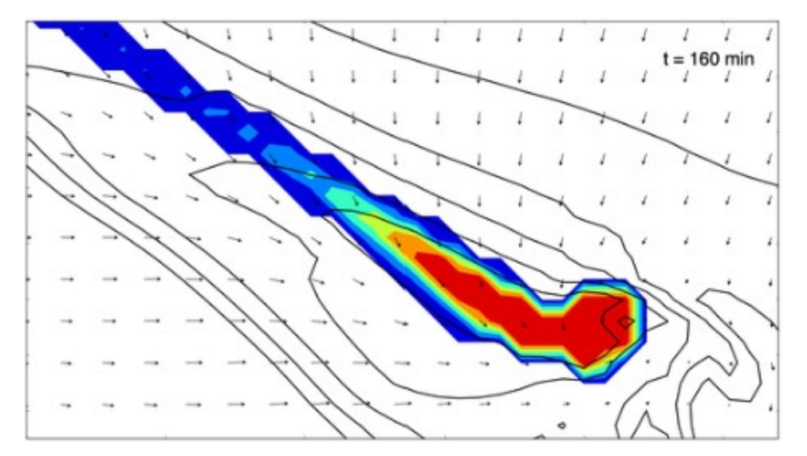
- Convert Recorder logs into an analysis-friendly columnar format–in our case, Parquet.
- Apply necessary transformations to make the logs analysis-ready.
- Repartition and reindex the logs for an efficient analysis.

- Build a multifaceted view of the workload, that accounts for several key I/O metrics.
- Extract I/O characteristics and metrics out of the multifaceted view through distributed analysis.
- Generate observations according the I/O characteristics and metrics for each view.

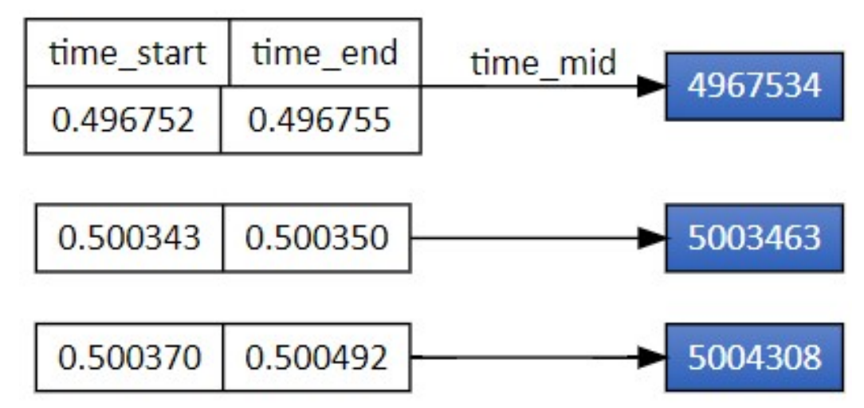
- Compute *Bottleneck Likelihood Score* (BLS) for given observations.
- Pass the observations along with their BLS through a rule-based engine to produce bottleneck diagnoses automatically.
- Output user-friendly text-based diagnoses.

I/O Bottleneck Detection for *CM1* (Time-based*)

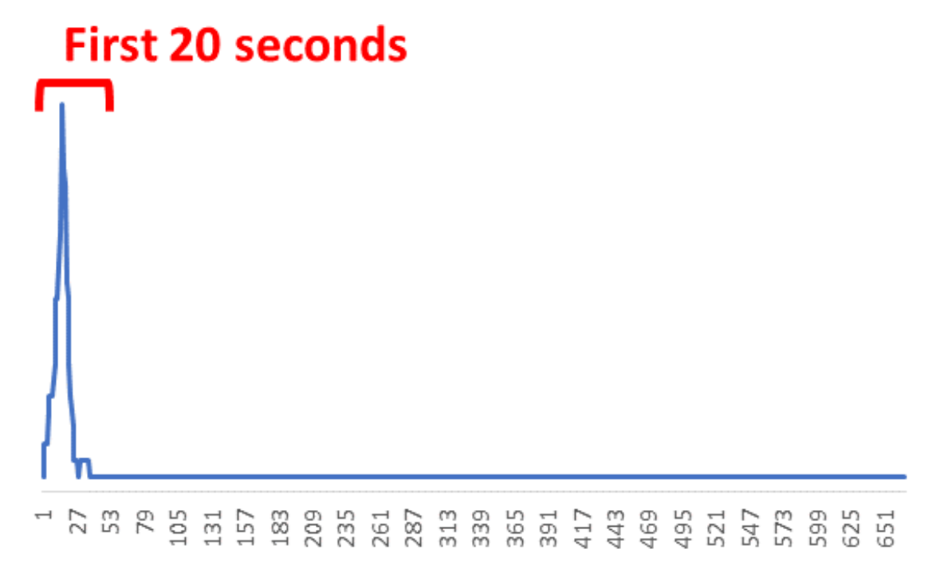
1



2



3



4

During **97.313-194.626**, we suspect an I/O bottleneck with **a BLS of 74.3%** because the application spends a lot of time during I/O operations, but accesses only 124 (8.0%) files

- Atmospheric simulation that models weather phenomena.
- Takes 668 seconds to run and generates around 200MB I/O traces with **27,463 records**.

- Timestamps are converted into microseconds as Dask indexing works faster with non-decimal values.
- For precision, the middle point of two timestamp is used in analysis instead of their range.

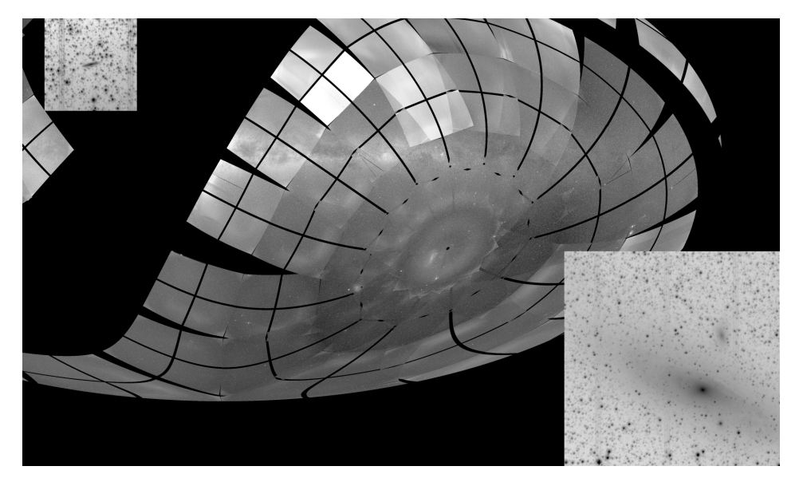
- 94.6% of I/O time is spent during the first 20 seconds.
- Rank 0 performs 100% of the writes on small files.
- Multifaceted analysis takes **22 seconds**.

- Time-based analysis produces diagnoses with specific time ranges.
- Above is an actual text-based diagnosis for CM1.
- **20 diagnoses** produced with BLS between 55-85%.

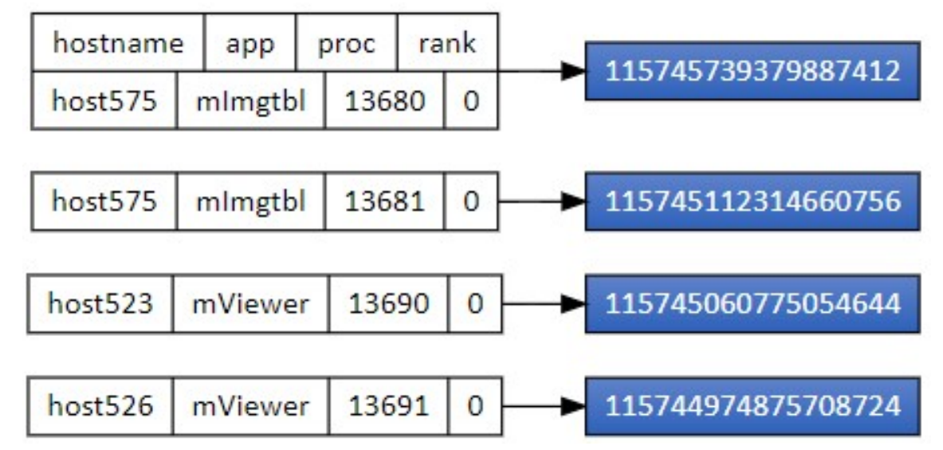
* Multifaceted analysis is done over all filter groups in parallel, yet for brevity, we showcase results from only one of the views for each application.

I/O Bottleneck Detection for *Montage* (Process-based*)

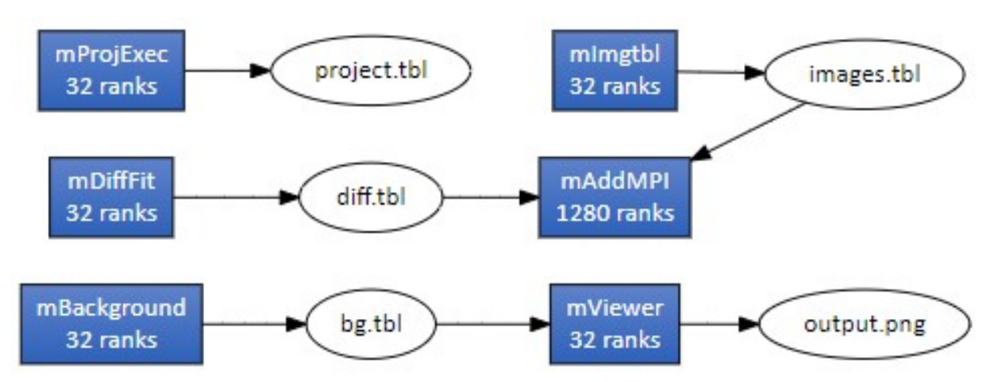
1



2



3



4

For **[process IDs ...]**, there is **a BLS of 61.4%** because the bandwidth is too low (190MB/s), and the transfer size is too small (176KB)

- A toolkit for assembling astronomical images into mosaics.
- Takes 175 seconds to run and generates around 170MB I/O traces with **4,889,291 records**.

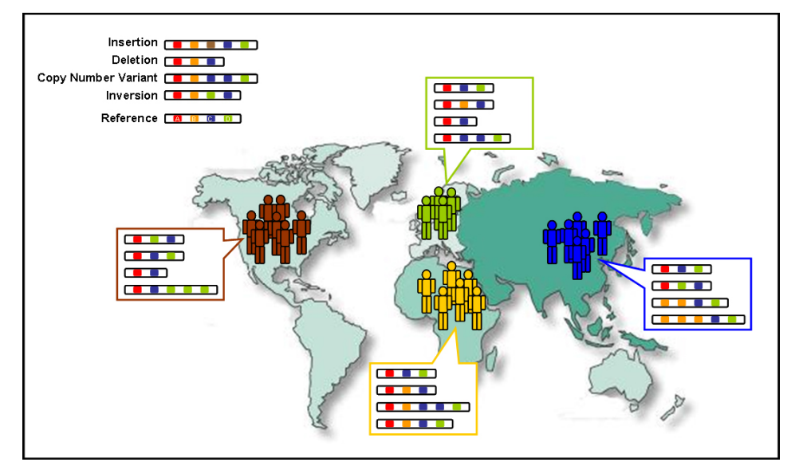
- For efficiency, the process IDs are hashed with respect to their node addresses and hostnames.
- Allows us to analyze groups of process IDs effectively.

- 1280 ranks performs 3.2M read operations and 1.6M write operations.
- Average bandwidth across the I/O operations is low (around 3MB/s).
- Multifaceted analysis takes **42 seconds**.

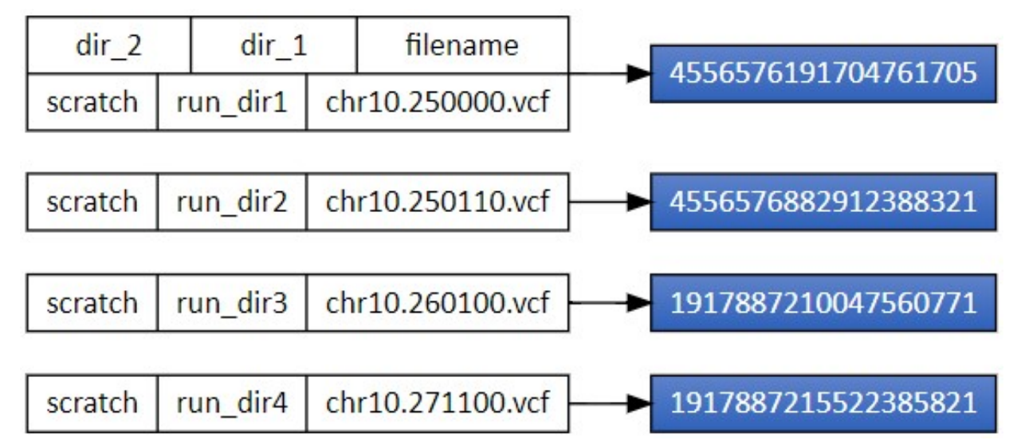
- Process-based analysis produces diagnoses with specific nodes or apps.
- Analysis iterates per-node or per-app basis.
- **14 diagnoses** produced with BLS between 52-61.4%.

I/O Bottleneck Detection for *Genome* (File-based*)

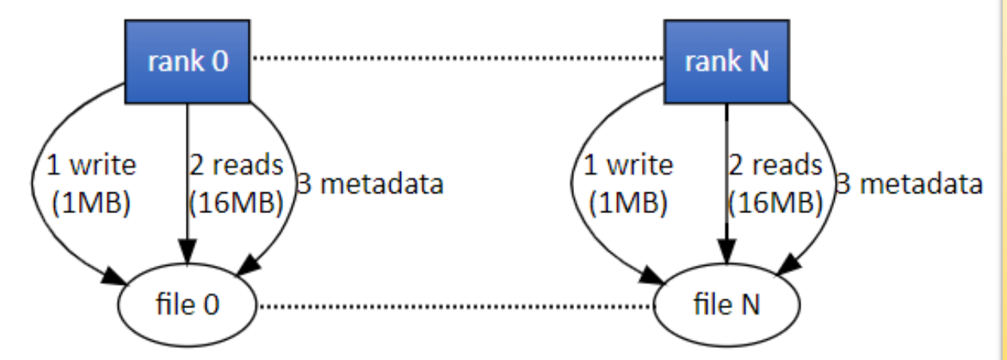
1



2



3



4

In **[file directories ...]**, we find an I/O bottleneck with **a BLS of 67.1%** as the application does the most of I/O operations (38% ops), and the transfer size is too small (113KB)

- A comprehensive description of human genetic variation through sequencing multiple individuals.
- Takes 7 hours to run and generates around 55GB I/O traces with **9,073,970 records**.

- The filenames are hashed with respect to folder hierarchy.
- Allows us to analyze file directories effectively.

- All read operations are 16MB and all write operations are 1MB.
- There are 21m files and all of them are accessed file-per-process basis.
- Multifaceted analysis takes **12 minutes**.

- File-based analysis produces diagnoses with specific folder hierarchy.
- Analysis iterates per-directory basis.
- **30 diagnoses** produced with BLS between 54-67.1%.

Conclusions

1. We demonstrate our methodology and our tool DigIO that use multifaceted views of I/O data to identify I/O bottlenecks.
2. We showcase that applying an automated multifaceted analysis is a complex task and DigIO is still able to detect I/O bottlenecks in seconds or minutes depending on the size of the I/O traces.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract LLNL-POST-838770. Also, the material is based upon work supported by the National Science Foundation under Grant no. NSF OAC-2104013, OCI-1835764, and CSR-1814872.