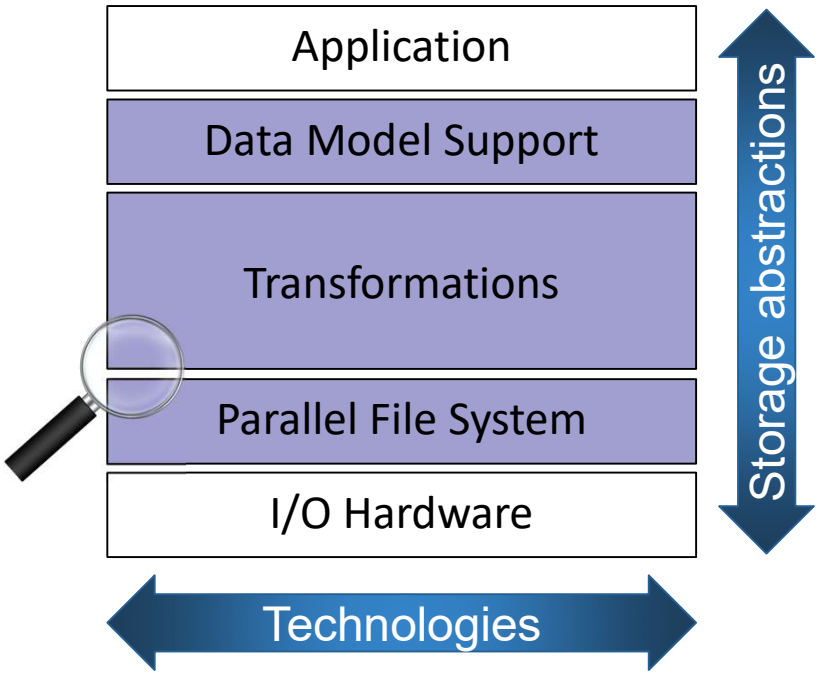
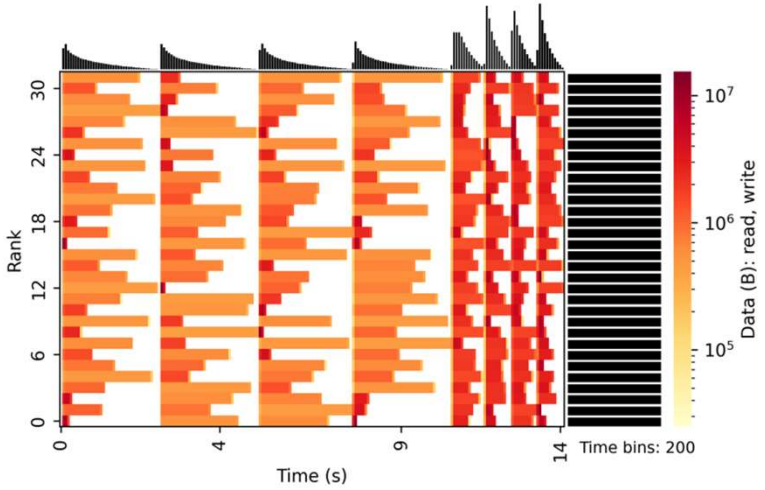
# LOW-LEVEL I/O INSTRUMENTATION

- Darshan provides in-depth instrumentation of the lower layers of traditional HPC I/O stack:
  - **MPI-IO** parallel I/O interface
  - **POSIX** file system interface
  - **STDIO** buffered stream I/O interface
  - **Lustre** striping parameters
- Captures fixed set of statistics, properties, and timing info for each file accessed using these interfaces
- Informs on key I/O performance characteristics of foundational components of the HPC I/O stack



# LOW-LEVEL I/O INSTRUMENTATION

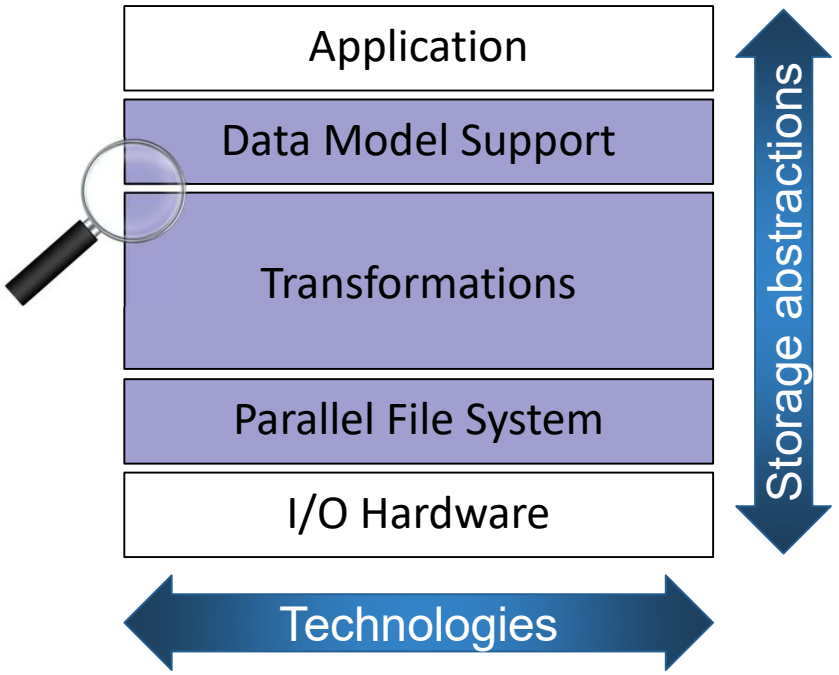
- Beyond its traditional capture mode, Darshan offers key features for obtaining finer-grained details of low-level I/O activity:
  - ➢ **Heatmap module**: captures histograms of I/O activity at each process using a fixed size histogram
    - Available for POSIX, MPI-IO, and STDIO interfaces by default in 3.4+ versions of Darshan
  - ➢ **DXT modules**: captures full I/O traces at each process using a configurable buffer size
    - Available for POSIX and MPI-IO modules
    - Enabled using `DXT_ENABLE_IO_TRACE` environment variable




Heatmaps showcase application I/O intensity across time, ranks, and interfaces – helpful for identifying hot spots, I/O and compute phases, etc.

# HIGH-LEVEL I/O LIBRARY INSTRUMENTATION

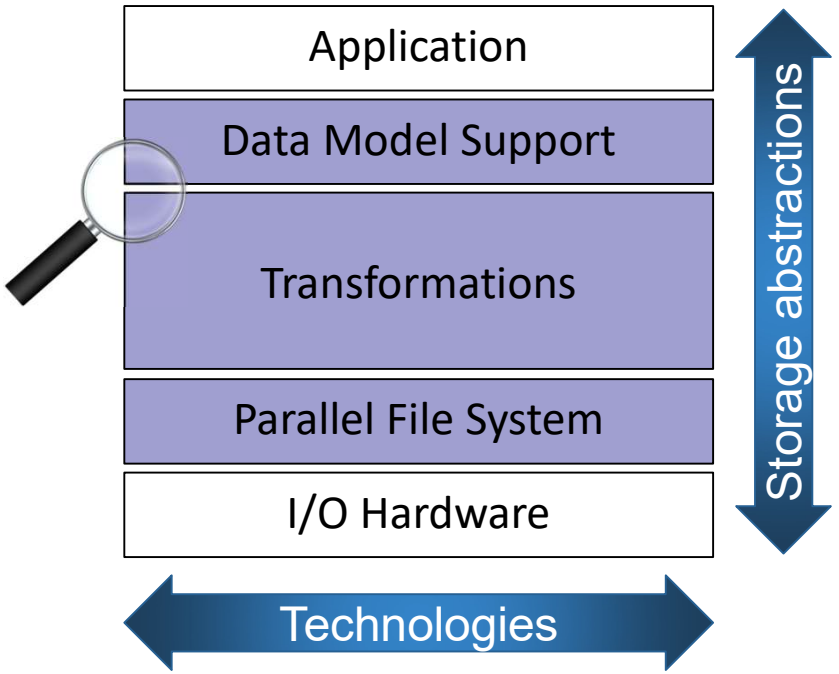
- Darshan similarly provides in-depth instrumentation of popular high-level I/O libraries for HPC
  - **HDF5**: detailed instrumentation of accesses to HDF5 files and datasets available starting in 3.2+ versions
  - **PnetCDF**: detailed instrumentation of accesses to PnetCDF files and variables available starting in 3.4.1+ versions
- Full-stack characterization allows deeper understanding of app usage of I/O libraries, as well as underlying performance characteristics for these usage patterns



# HIGH-LEVEL I/O LIBRARY INSTRUMENTATION

- Darshan similarly provides in-depth instrumentation of popular high-level I/O libraries for HPC
  - **HDF5**: detailed instrumentation of accesses to HDF5 files and datasets available starting in 3.2+ versions
  - **PnetCDF**: detailed instrumentation of accesses to PnetCDF files and variables available starting in 3.4.1+ versions

PnetCDF module contributed by Wei-Keng Liao (NWU)



# HDF5 APPLICATION INSTRUMENTATION EXAMPLE

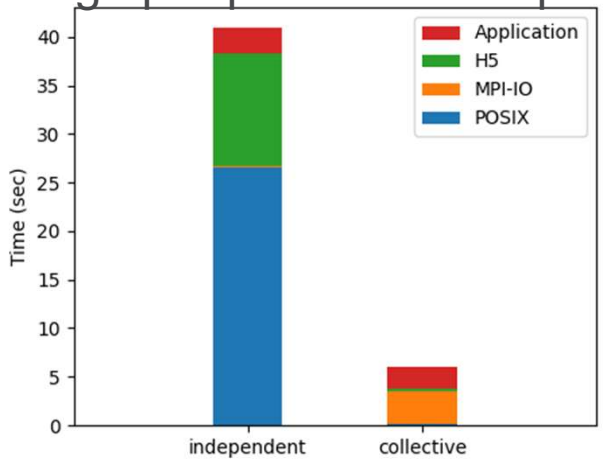
- The MACSio[1] benchmark evaluates behavior of multi-physics I/O workloads using different I/O backends, including HDF5
  - ➢ We instrumented using Darshan's HDF5 module to see what insights we could gain into performance characteristics of independent and collective I/O configurations

b/w: **~30 MB/sec**

**POSIX** I/O dominates, **H5** incurs non-negligible overhead forming this workload

Negligible time spent in **MPI-IO**

Average per-process time spent in I/O




b/w: **~290 MB/sec**

**H5** and **POSIX** incur minimal overhead for this workload

**MPI-IO** collective I/O algorithm dominates

1. https://github.com/LLNL/MACSio

# PYDARSHAN LOG ANALYSIS FRAMEWORK

- Darshan has traditionally offered only the C-based darshan-util library and a handful of corresponding tools to users for log file analysis
  - Complicates development of custom Darshan analysis tools
- PyDarshan developed to simplify the interfacing of analysis tools with log data
  - Use Python CFFI module to define Python bindings to the native darshan-utils C API
  - Expose Darshan log data as dictionaries, pandas dataframes, and NumPy arrays
- PyDarshan should provide a richer ecosystem for development of Darshan log analysis tools, either by end users or by the Darshan team

Available via PyPI or Spack:
- "`pip install darshan`"
- "`spack install py-darshan`"

PyDarshan development led by Jakob Luttgau (UTK), Tyler Reddy and Nik Awtrey (LANL)

# PYDARSHAN JOB SUMMARY TOOL

- PyDarshan includes a new job summary tool that is replacing the original `darshan-job-summary.pl` script
  - ➢ Generates detailed HTML reports summarizing application I/O behavior using different plots, graphs, and statistics
  - ➢ Builds off popular Python libraries like `matplotlib` (plotting), `seaborn` (plotting), and `mako` (HTML templating)
- Users can generate summary reports for a given Darshan log file using the following command:
  - ➢ '**`python -m darshan summary <path_to_log_file>`**'
  - ➢ Generates an output HTML report describing job's I/O behavior

# PYDARSHAN JOB SUMMARY TOOL

## Detailed job metadata

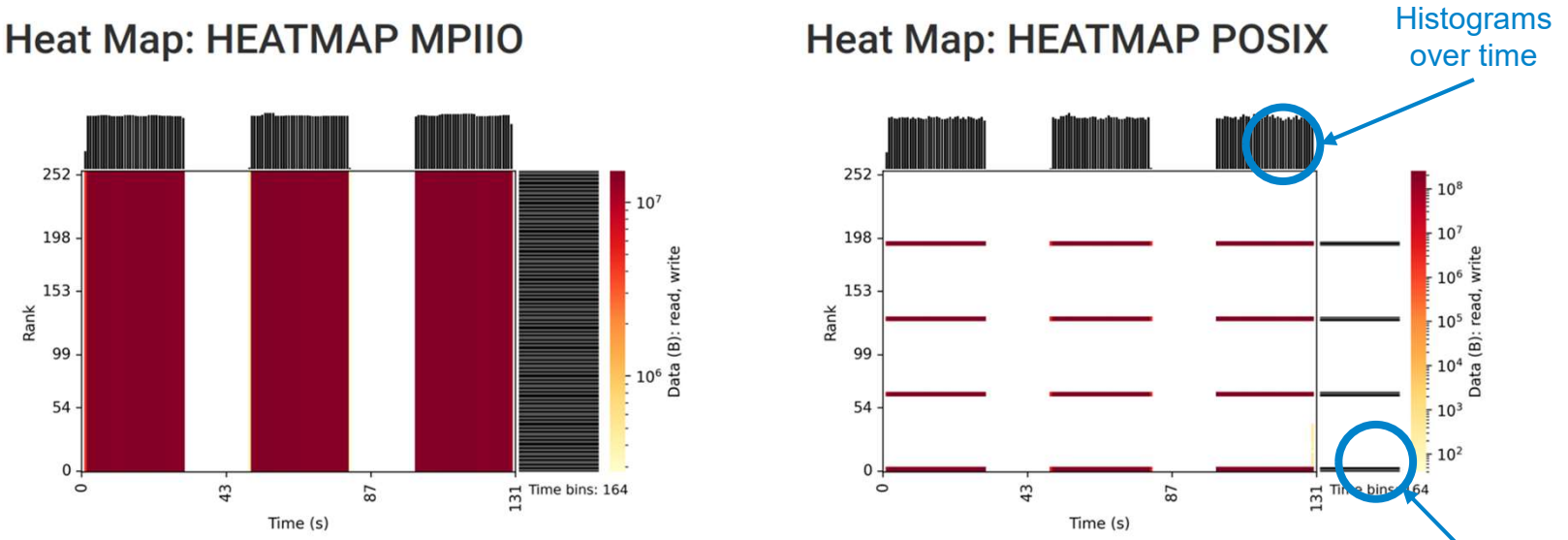
### Job Summary

| | |
|---|---|
| **Job ID** | 6048613 |
| **User ID** | 69628 |
| **# Processes** | 256 |
| **Run time (s)** | 130.6011 |
| **Start Time** | 2023-03-13 20:51:05 |
| **End Time** | 2023-03-13 20:53:16 |
| **Command Line** | /global/homes/s/ssnyder/software/h5bench/install/bin//h5bench_write /pscratch/sd/s/ssnyder/bench-out/7a24c1e9-6048613 /h5bench.cfg /pscratch/sd/s/ssnyder/bench-out/test.h5 |

https://github.com/hpc-io/h5bench

# PYDARSHAN JOB SUMMARY TOOL

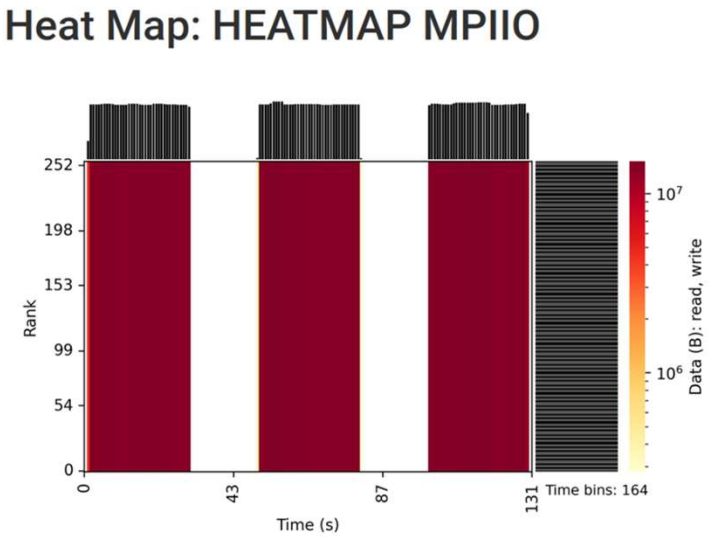
## Heatmaps for visualizing I/O activity

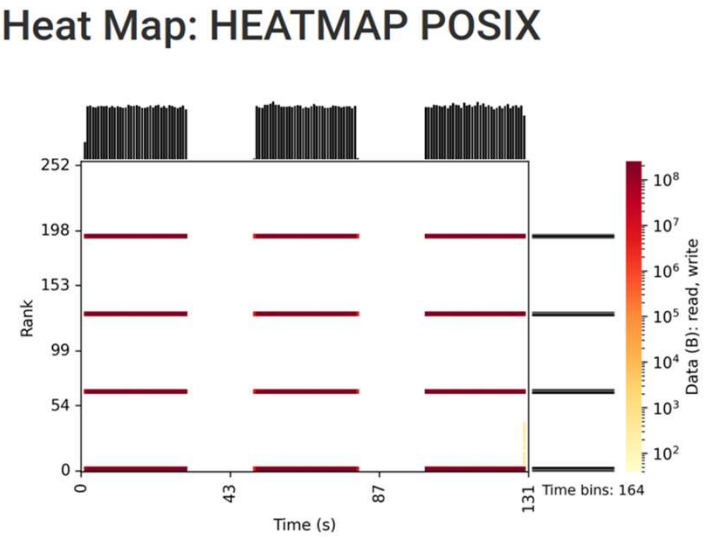



Analyzing I/O behavior over time, ranks, and interfaces can offer key insights into application I/O behavior.

# PYDARSHAN JOB SUMMARY TOOL

## Heatmaps for visualizing I/O activity

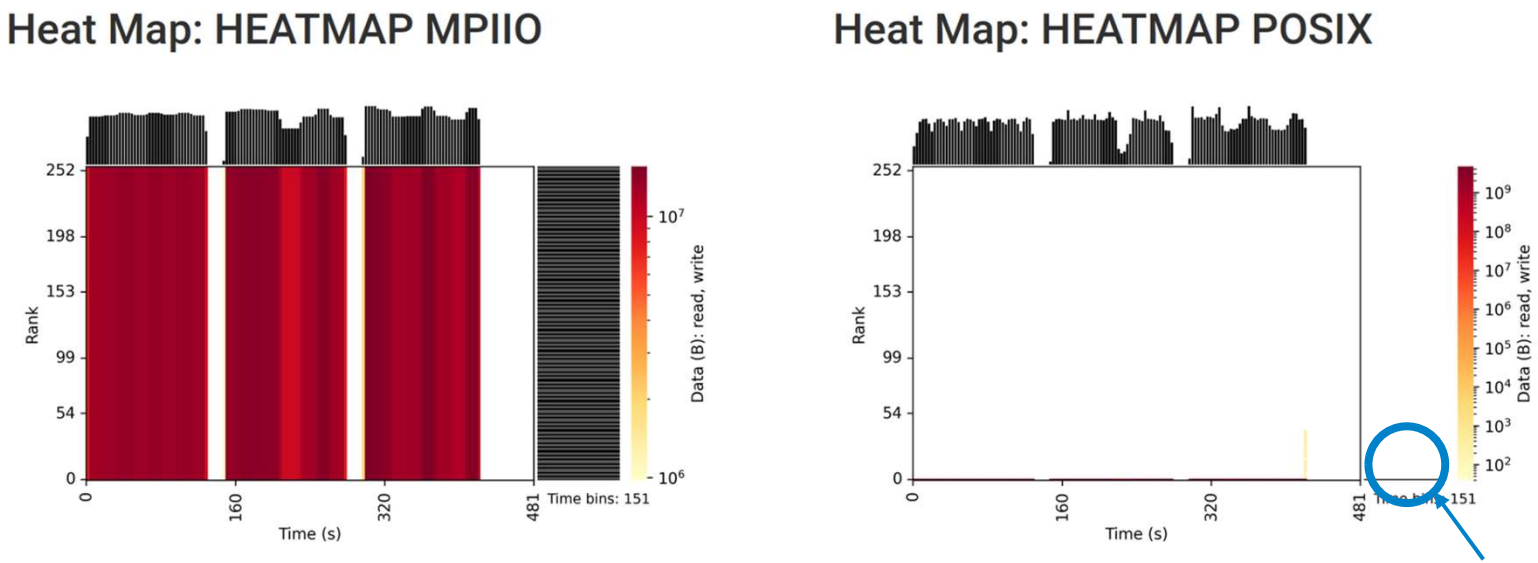



This example heatmap illustrates a typical MPI-IO collective I/O pattern. All MPI ranks perform MPI-IO operations (left), but only a subset of “aggregators” access the file via POSIX operations (right).

# PYDARSHAN JOB SUMMARY TOOL

## Heatmaps for visualizing I/O activity



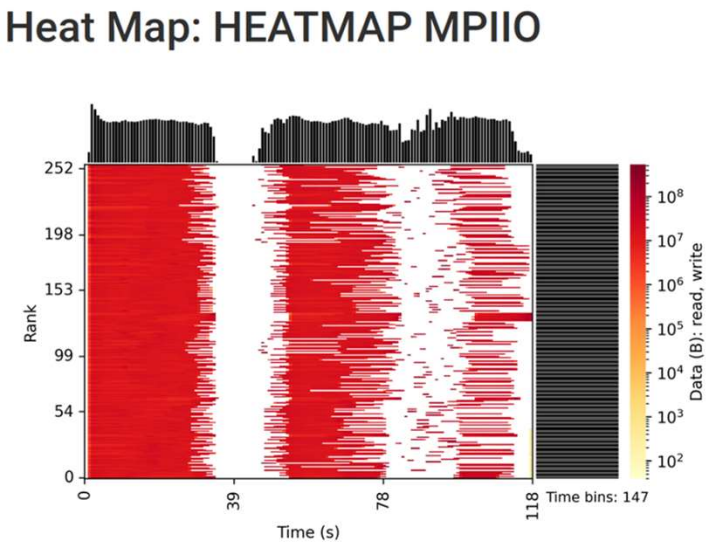

Heatmaps can help quickly detect common pitfalls.

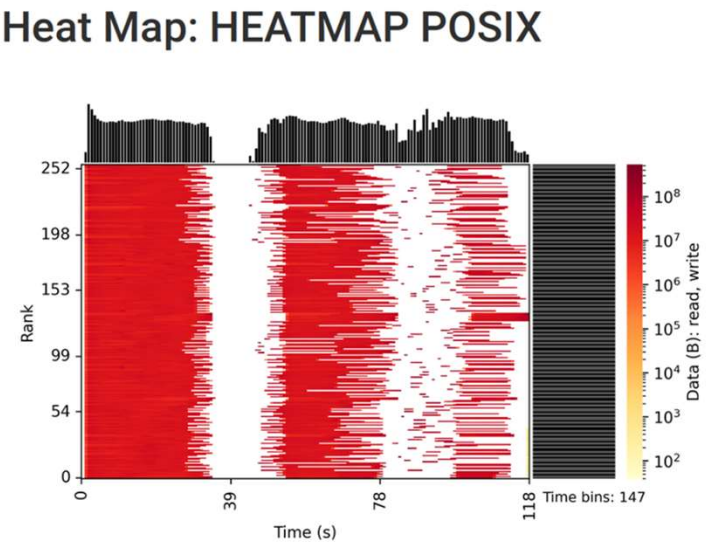
All I/O funneled through rank 0

*Oops, I forgot to tell Lustre to use more than one stripe.*

# PYDARSHAN JOB SUMMARY TOOL

## Heatmaps for visualizing I/O activity



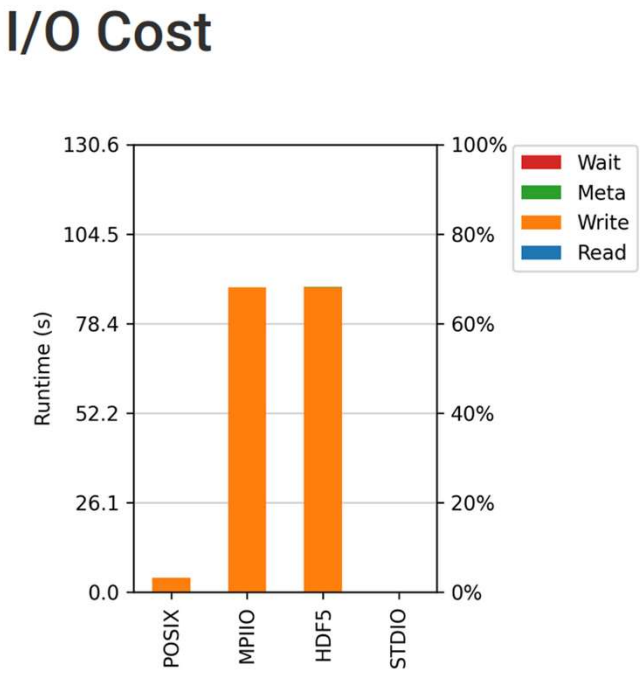




Heatmaps can help quickly detect common pitfalls.

*I could have sworn I enabled collective I/O in HDF5.*

# PYDARSHAN JOB SUMMARY TOOL

## Cross-module I/O comparisons

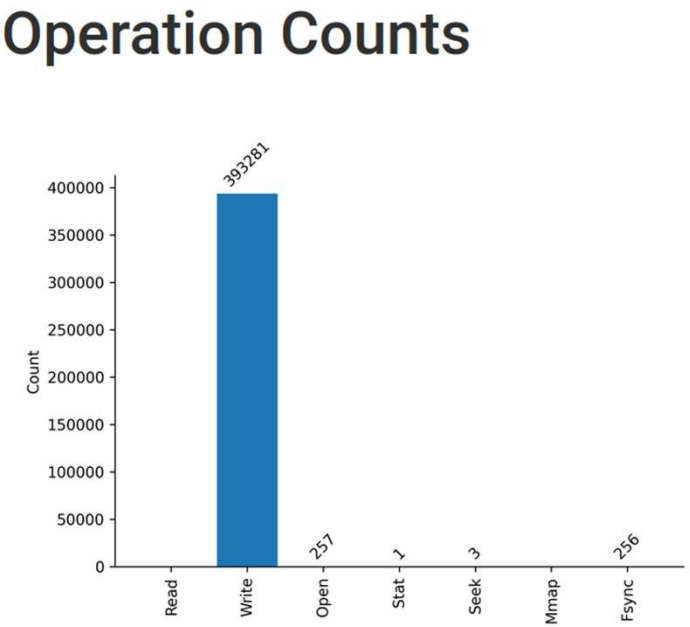



On average, how much time was spent reading, writing, and doing metadata across main I/O interfaces?

If mostly non-I/O (i.e., compute), limited opportunities for I/O tuning.

# PYDARSHAN JOB SUMMARY TOOL

## Per-module I/O statistics



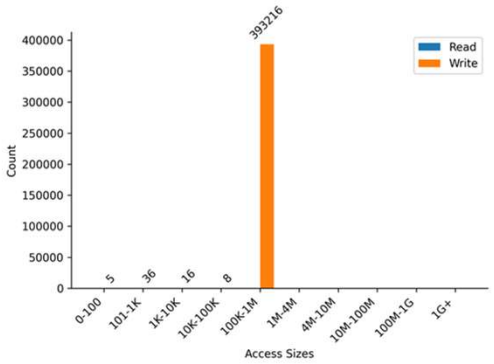

What were the relative totals of different I/O operations across key interfaces?

Lots of metadata operations (open, stat, seek, etc.) could be a sign of poorly performing I/O.

# PYDARSHAN JOB SUMMARY TOOL

## Per-module I/O statistics

### Access Sizes




### Common Access Sizes

| Access Size | Count |
|---|---|
| 1048576 | 393192 |
| 272 | 24 |
| 1040328 | 6 |
| 1044424 | 6 |

Access size distributions and common access sizes are provided to better understand general file access patterns.

In general, larger access sizes perform better with most storage systems.

# AVAILABLE DARSHAN ANALYSIS TOOLS

- Documentation: http://www.mcs.anl.gov/research/projects/darshan/docs/darshan-util.html
- Officially supported tools
  - **darshan-job-summary.pl**: Creates PDF with graphs for initial analysis
  - **darshan-summary-per-file.sh**: Similar to above, but produces a separate PDF summary for every file opened by application
  - **darshan-parser**: Dumps all information into text format
    - For example, darshan-parser user_app_numbers.darshan | grep write
    - Useful for building your own analysis
- Third-party tools (incomplete list!)
  - **darshan-ruby**: Ruby bindings for darshan-util C library https://xgitlab.cels.anl.gov/darshan/darshan-ruby
  - **HArshaD**: Easily find and compare Darshan logs https://kaust-ksl.github.io/HArshaD/
  - **pytokio**: Detect slow Lustre OSTs, create Darshan scoreboards, etc. https://pytokio.readthedocs.io/
  - DXT Explorer: visualize detailed "extended tracing" data https://github.com/hpc-io/dxt-explorer
  - Drishti: a "darshan coach": https://github.com/hpc-io/drishti-io