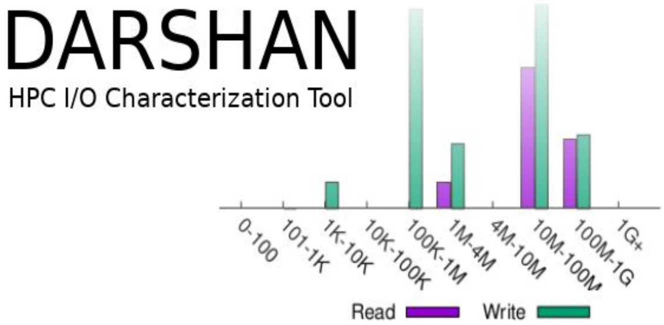
February 1, 2023

Enabling Insights Into HPC Application I/O Behavior With Darshan



Shane Snyder
ssnyder@mcs.anl.gov
Argonne National Laboratory

HSF Software Developer Tools and Packaging Working Group Meeting

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Understanding and improving HPC I/O

- The ability to characterize and understand application I/O workloads is critical to ensuring efficient use of an evolving and increasingly complex HPC I/O stack
  - Deep layers of coordinating I/O libraries and entirely new-to-HPC storage paradigms (e.g., object storage)
  - Emerging storage hardware (e.g., PMEM) and storage architectures (e.g., burst buffers)

- I/O analysis tools are invaluable in helping to navigate this complexity and to better understand I/O
  - Characterize I/O behavior of individual jobs to inform tuning decisions
  - Characterize job populations to better understand system-wide I/O stack usage and optimize deployments

Darshan: A tool for HPC I/O understanding

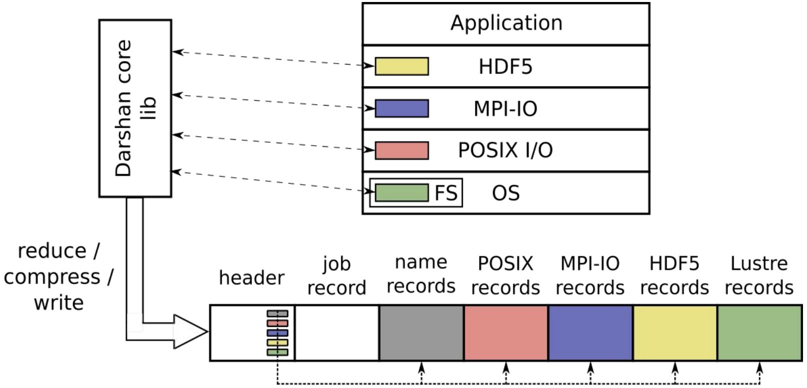
What is Darshan?

- Darshan is a lightweight I/O characterization tool that captures concise views of HPC application I/O behavior
  - Produces a summary of I/O activity for each instrumented job
    - Counters, histograms, timers, & statistics
    - If requested by user, full I/O traces

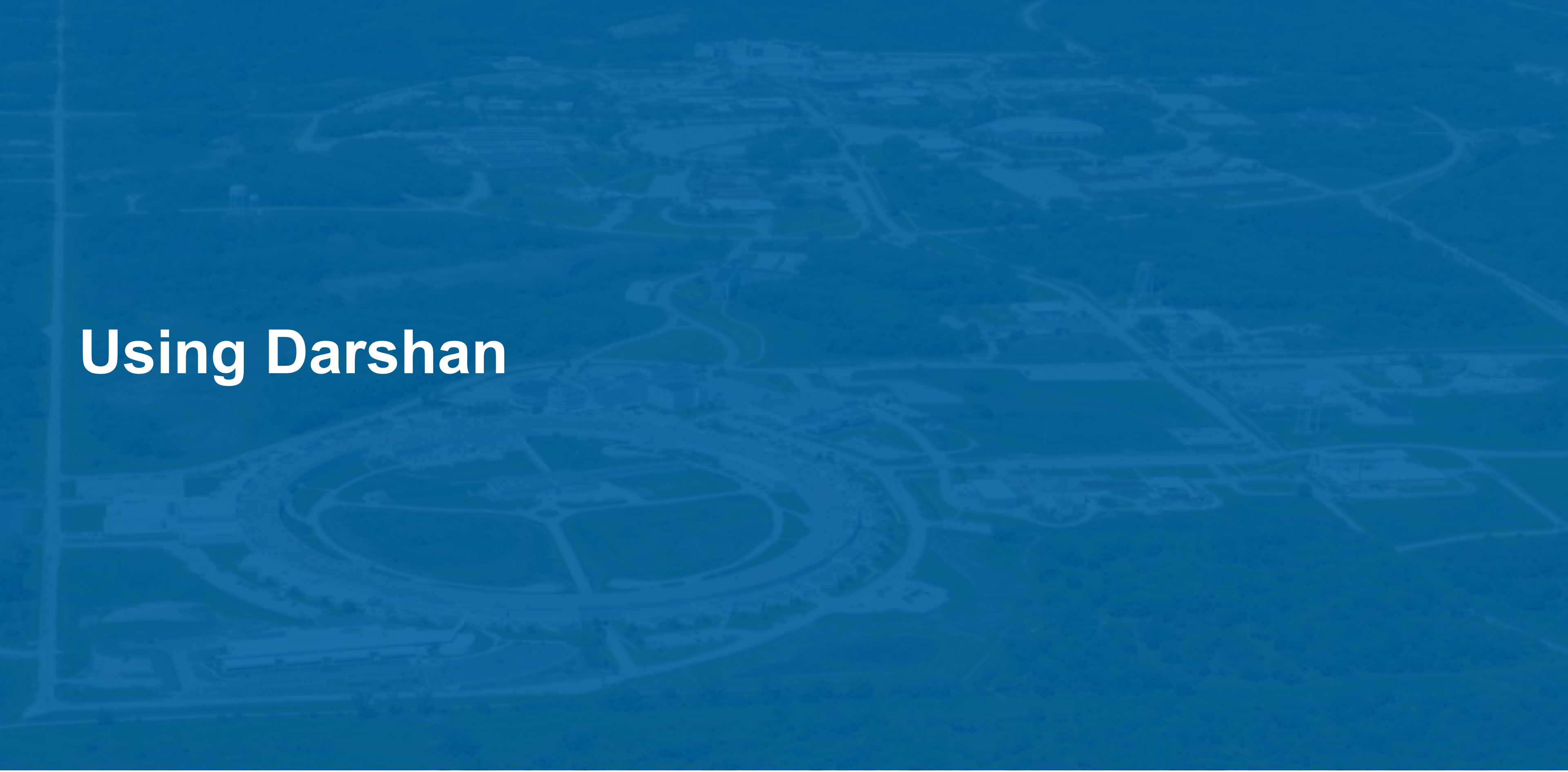
- Widely available
  - Deployed (and commonly enabled by default) at many HPC facilities around the world
- Easy to use
  - No code changes required to integrate Darshan instrumentation
  - Negligible performance impact; just “leave it on”
- Modular
  - Adding instrumentation for new I/O interfaces or storage components is straightforward

How does Darshan work?

- ❖ Darshan records file access statistics for each process as app executes
- ❖ At app shutdown, collect, aggregate, compress, and write log data
- ❖ After job completes, analyze Darshan log data
  - ➢ `darshan-job-summary` - provides a summary PDF characterizing application I/O behavior
  - ➢ `darshan-parser` - provides complete text-format dump of all counters in a log file
  - ➢ *`PyDarshan`* - Python analysis module for Darshan logs



- ❖ Originally designed for MPI applications, but in recent Darshan versions (3.2+) any dynamically-linked executable can be instrumented
  - ➢ In MPI mode, a log is generated for each *app*
  - ➢ In non-MPI mode, a log is generated for every *process*

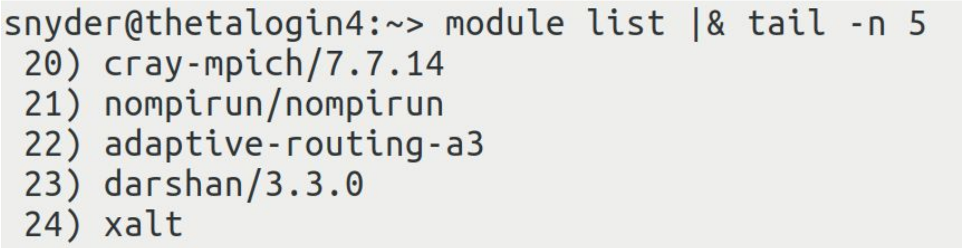


Using Darshan

Instrumenting apps with Darshan

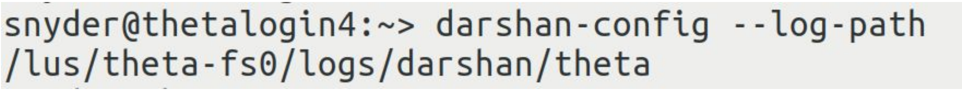
Traditional usage on HPC platforms

- On many HPC platforms (e.g., ALCF Theta, NERSC Cori & Perlmutter, OLCF Summit), Darshan is already installed and enabled by default
  - **Just compile and run your apps like normal**
  - Logs are written to a central repository for all users when the app terminates



```
snyder@thetalogin4:~> module list |& tail -n 5
 20) cray-mpich/7.7.14
 21) nompirun/nompirun
 22) adaptive-routing-a3
 23) darshan/3.3.0
 24) xalt
```

Darshan 3.3.0 is enabled by default on ALCF Theta



```
snyder@thetalogin4:~> darshan-config --log-path
/lus/theta-fs0/logs/darshan/theta
```

‘`darshan-config --log-path`’ command can be used to find output log directory. Directory is further organized into year/month/day subdirectories.

Log file name includes username, app name, and job ID for easy identification, e.g.: snyder_ior_id12345…

Instrumenting apps with Darshan

Traditional usage on HPC platforms

- ❖ On many HPC platforms (e.g., ALCF Theta, NERSC Cori & Perlmutter, OLCF Summit), Darshan is already installed and enabled by default
  - ➢ **Just compile and run your apps like normal**
  - ➢ Logs are written to a central repository for all users when the app terminates

Important caveats related to non-MPI usage:

- Requires dynamically-linked executables
- Non-MPI mode must be explicitly enabled via env variable
  - `export DARSHAN_ENABLE_NONMPI=1`
- Some systems may have dated Darshan versions that don't properly support non-MPI mode

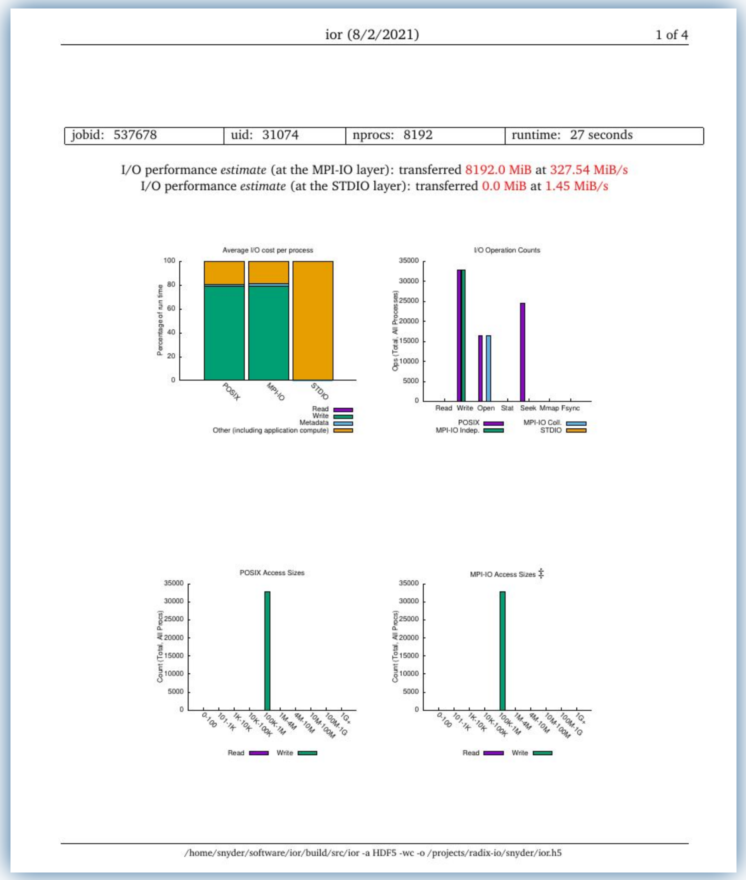
Instrumenting apps with Darshan

Installing and using your own Darshan tools

- In some circumstances, it may be necessary to roll your own install
  - Darshan not installed or lacking necessary features
  - Need to build Darshan in specific software environments (e.g., containers with old compilers)
- Beyond installing from source, Darshan is also available on Spack
  - *darshan-runtime*: runtime instrumentation library linked with application
  - *darshan-util*: log analysis utilities
  - E.g., “**`spack install darshan-runtime`**”
- Once installed, users can `LD_PRELOAD` the darshan-runtime library
  - Output logs are written to directory pointed to by `DARSHAN_LOG_DIR_PATH` environment variable (defaults to `$HOME`)

Analyzing Darshan logs

- ❖ After locating your log, the `darshan-job-summary` script is a useful starting point for visualizing application I/O behavior:
  - ➢ "**`darshan-job-summary.pl <input_log>`**" produces a PDF with same name as input log
  - ➢ Contains useful graphs, tables, and performance estimates describing application I/O behavior



ior (8/2/2021) 1 of 4

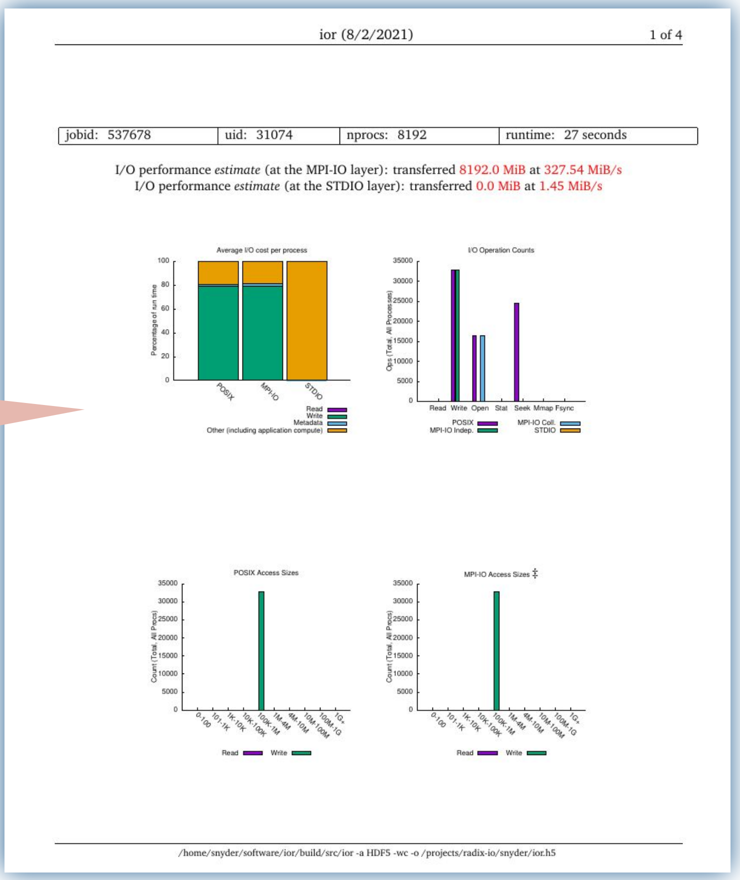
jobid: 537678	uid: 31074	nprocs: 8192	runtime: 27 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 8192.0 MiB at 327.54 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 1.45 MiB/s

/home/snyder/software/ior/build/src/ior -a HDF5 -wc -o /projects/radix-io/snyder/ior.h5

Analyzing Darshan logs

Note: This `darshan-job-summary.pl` tool will soon be deprecated by a new, more comprehensive Python-based Darshan summary tool – more on this coming soon!



ior (8/2/2021) 1 of 4

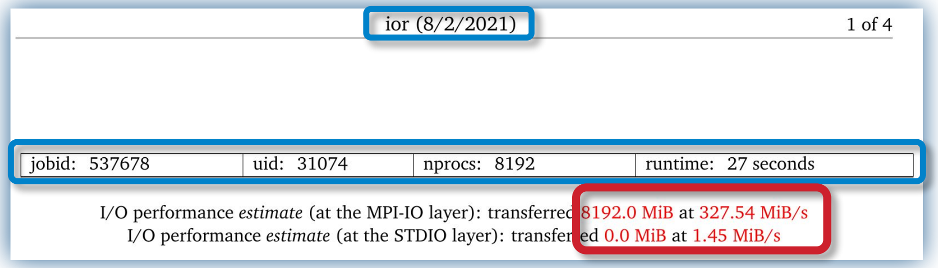
jobid: 537678	uid: 31074	nprocs: 8192	runtime: 27 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 8192.0 MiB at 327.54 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 1.45 MiB/s

/home/snyder/software/ior/build/src/ior -a HDF5 -wc -o /projects/radix-io/snyder/ior.h5

Analyzing Darshan logs

- After locating your log, the `darshan-job-summary` script is a useful starting point for visualizing application I/O behavior:
  - “**`darshan-job-summary.pl <input_log>`**” produces a PDF with same name as input log
  - Contains useful graphs, tables, and performance estimates describing application I/O behavior



ior (8/2/2021) 1 of 4

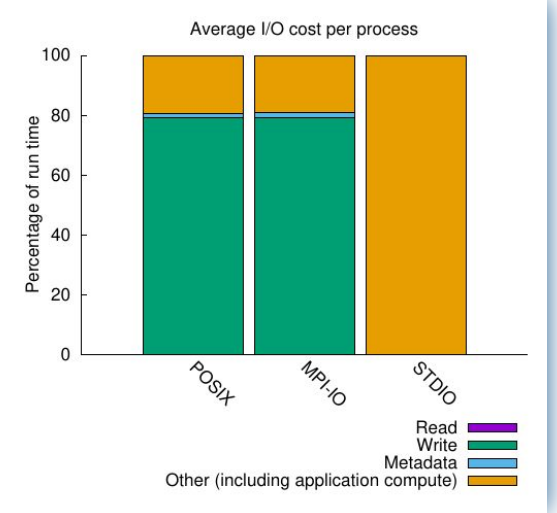
jobid: 537678	uid: 31074	nprocs: 8192	runtime: 27 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 8192.0 MiB at 327.54 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 1.45 MiB/s

Job metadata and performance estimates

Analyzing Darshan logs

- After locating your log, the `darshan-job-summary` script is a useful starting point for visualizing application I/O behavior:
  - “**`darshan-job-summary.pl <input_log>`**” produces a PDF with same name as input log
  - Contains useful graphs, tables, and performance estimates describing application I/O behavior

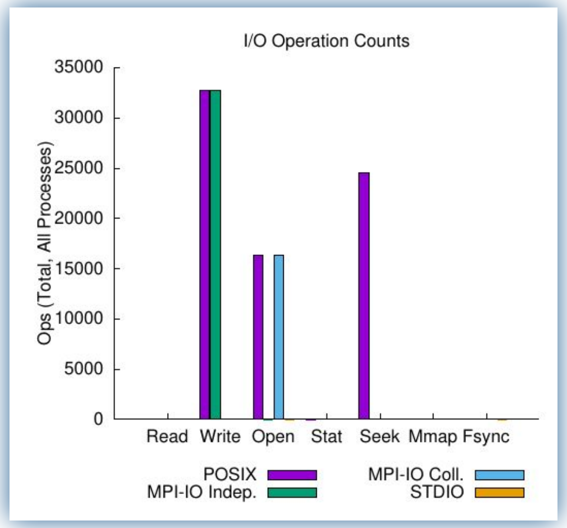


Across main I/O interfaces, how much time was spent reading, writing, doing metadata, or computing?

If mostly compute, limited opportunities for I/O tuning

Analyzing Darshan logs

- After locating your log, the `darshan-job-summary` script is a useful starting point for visualizing application I/O behavior:
  - “**`darshan-job-summary.pl <input_log>`**” produces a PDF with same name as input log
  - Contains useful graphs, tables, and performance estimates describing application I/O behavior



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

Key Darshan instrumentation capabilities

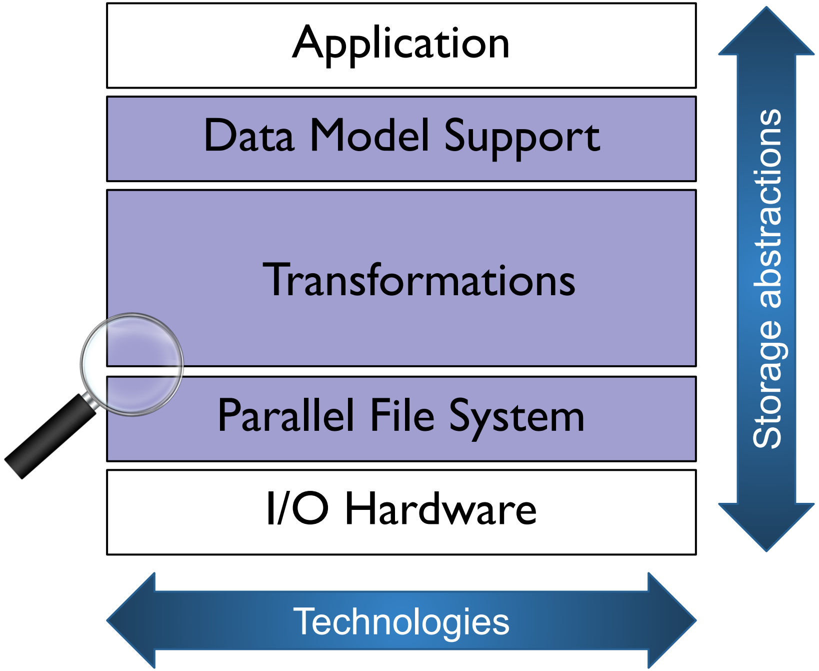
U.S. DEPARTMENT OF ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY

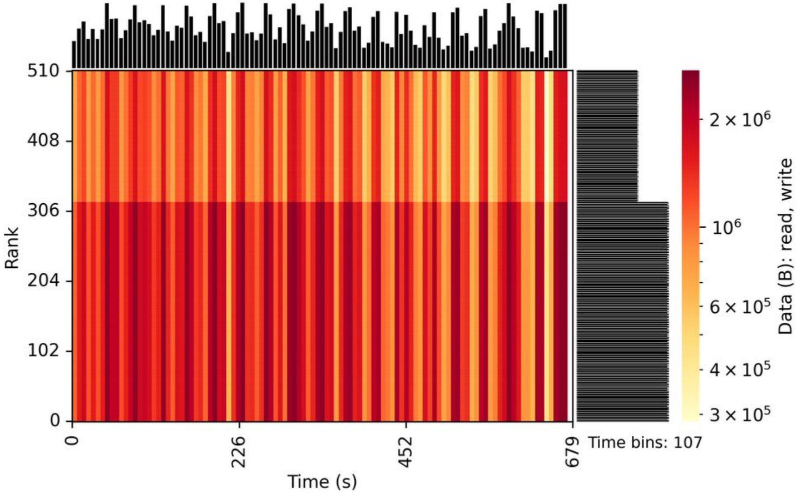
Low-level I/O instrumentation

- 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** file system 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

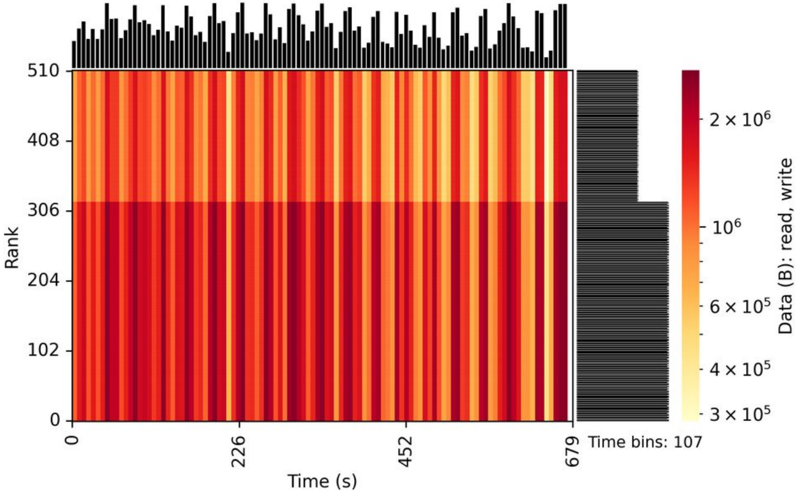
- Beyond its traditional capture mode, Darshan offers features for obtaining finer-grained details of low-level I/O activity:
  - **Heatmap module**: captures histograms of I/O activity at each process using a fixed size histogram
    - Available for POSIX, MPI-IO, and STDIO interfaces by default in 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.

Low-level I/O instrumentation

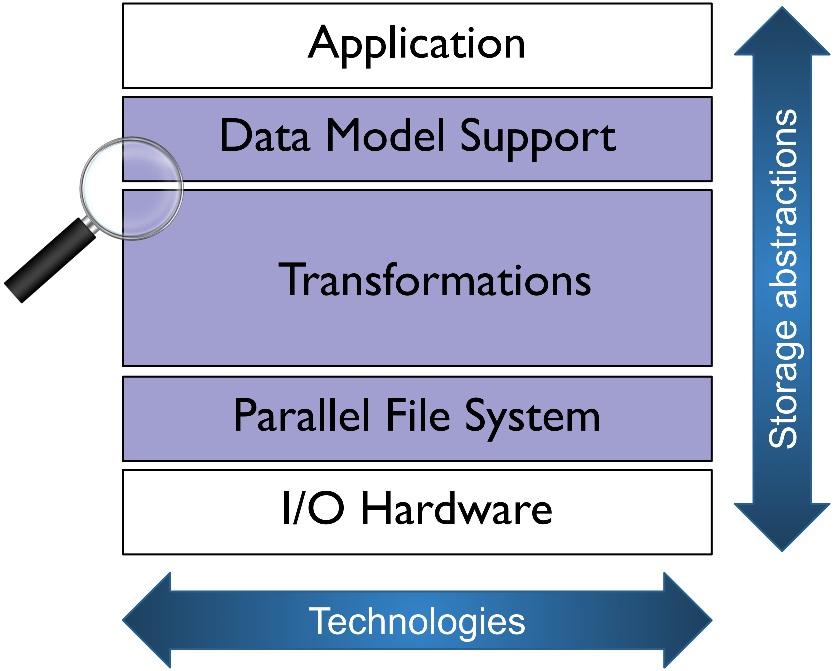
- Beyond its traditional capture mode, Darshan offers features for obtaining finer-grained details of low-level I/O activity:
  - **Heatmap module**: captures histograms of I/O activity at each process using a fixed size histogram
    - Available for POSIX, MPI-IO, and STDIO interfaces by default in 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



These heatmaps could similarly be used to show I/O intensity across a set of processes involved in an HEP workflow, rather than ranks in an MPI app

High-level I/O library instrumentation

- Darshan similarly provides in-depth instrumentation of HDF5 and Parallel netCDF, popular high-level I/O libraries for HPC
- HDF5 support is of particular interest, given its gaining traction in different HEP contexts
  - Darshan provides detailed instrumentation of accesses to HDF5 files and datasets in 3.2+ versions
- Full-stack characterization allows deeper understanding of app usage of I/O libraries, as well as underlying performance characteristics for these usage patterns



HDF5 application instrumentation example

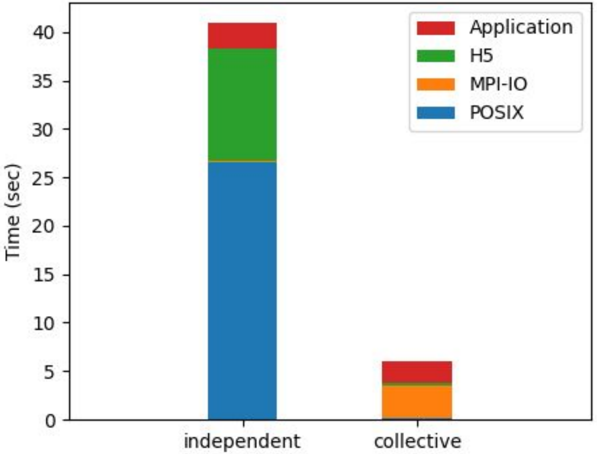
- The MACSio[1] benchmark evaluates behavior of multi-physics I/O workloads using different I/O backends, including HDF5
  - We instrumented using Darshan's HDF5 module to see what insights we could gain into performance characteristics of independent and collective I/O configurations

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

New Darshan log analysis capabilities

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
  - ➢ Implementing customized analysis tasks can become extremely cumbersome
- ❖ 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 enables a richer ecosystem for development of Darshan log analysis tools, by the Darshan team and by end users

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)

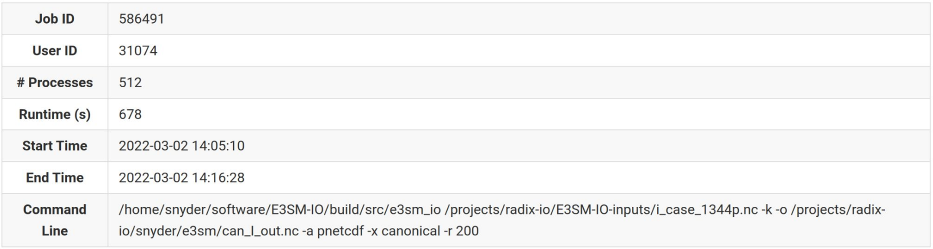
PyDarshan job summary tool

- PyDarshan includes a new job summary tool that will soon replace the `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

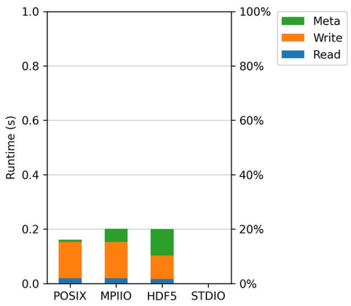
Detailed job metadata

Job Summary



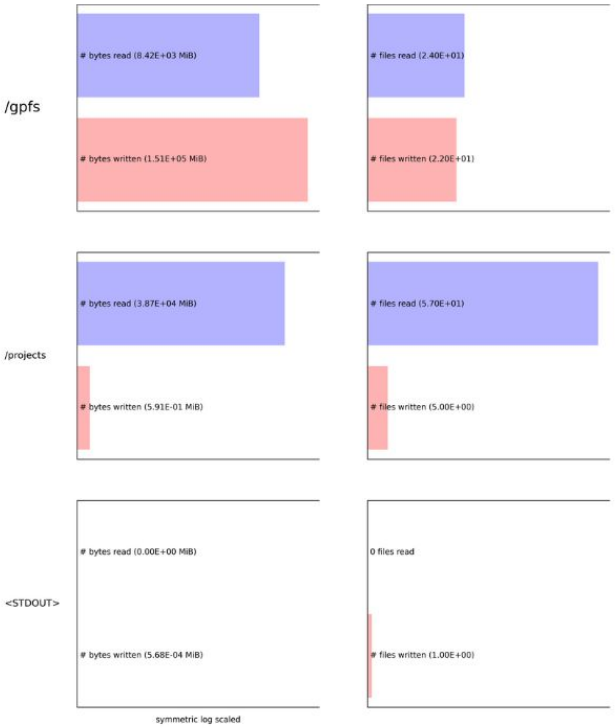
Job ID	586491
User ID	31074
# Processes	512
Runtime (s)	678
Start Time	2022-03-02 14:05:10
End Time	2022-03-02 14:16:28
Command Line	/home/snyder/software/E3SM-IO/build/src/e3sm_io /projects/radix-io/E3SM-IO-inputs/i_case_1344p.nc -k -o /projects/radix-io/snyder/e3sm/can_I_out.nc -a pnetcdf -x canonical -r 200

I/O Cost



I/O cost for all APIs

Data Access by Category



Total files and bytes read/written to different categories (mount points, standard streams, etc.)

Darshan analysis of HEP workflows

Thanks to Rui Wang (ANL) for
ATLAS Athena analysis!

Darshan usage in HEP contexts

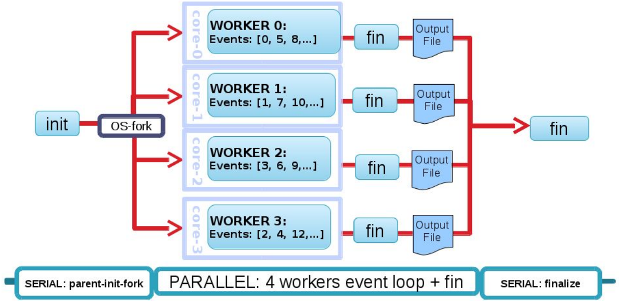
- *HEP-CCE IOS project*: Investigate how to utilize Darshan to understand and improve the I/O behavior of HEP workflows
  - What are the performance characteristics of different HEP I/O workloads?
  - How does HEP software interact with HPC I/O libraries and storage systems? Can these interactions be optimized?
- Our studies have motivated a couple of important improvements to Darshan
  - Proper instrumentation of forked processes
    - Darshan library now detects when a fork occurs and resets instrumentation state on all child processes to start from a clean slate
  - Runtime library configuration
    - Gives user fine-grained runtime control over instrumentation scope (i.e., what interfaces and what files to instrument) and library memory usage

ATLAS offline software – Athena

Various Athena Modes

- **AthenaMP (multi-Process)+standalone merging – Run2 original**
  - Independent parallel workers are forked from main process with shared memory allocation
  - Each worker produces its own outputs and merged later via a post-processing merge process
- **AthenaMP+SharedWriter (multi-Process) – Run2**
  - A shared writer process does all the output writes
  - Reduce time on single thread merging process
- **AthenaMP+SharedWriter (parallelCompression) – Run3**
  - Uses parallel compression to reduce the time increment when moving to higher No. of process
- **AthenaMT (multi-thread)**
  - Gaudi task scheduler maps tasks to kernel threads
  - Shared single pool of heap memory

Schematic View of ATLAS AthenaMP

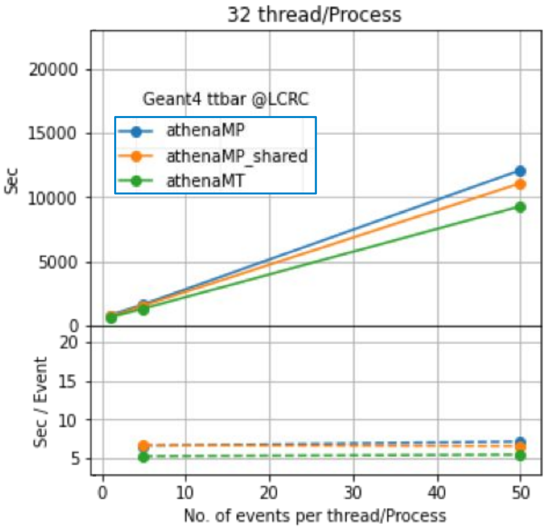


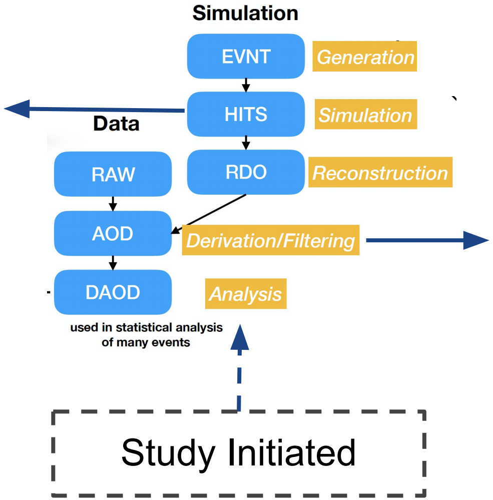
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults

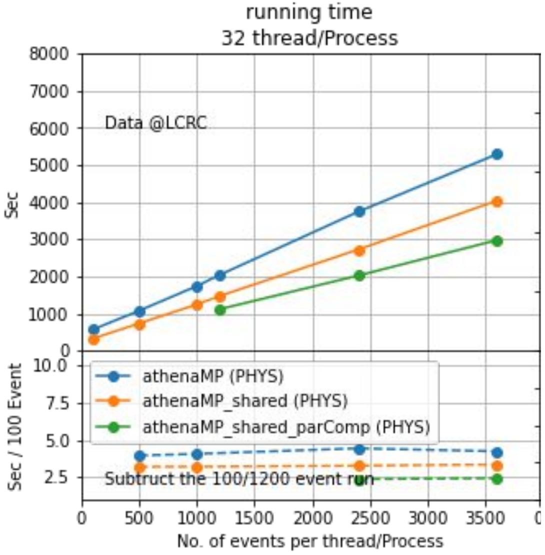
Athena I/O monitoring

- ❖ MC Simulation – CPU intensive
  - ➢ AthenaMP+Standalone merging
  - ➢ AthenaMP+SharedWriter
  - ➢ AthenaMT

- ❖ Derivation (DAOD) production – I/O intensive
  - ➢ AthenaMP+Standalone merging
  - ➢ AthenaMP+SharedWriter
  - ➢ AthenaMP+SharedWriter (parallel compression)







Running on 1 node with 36 cores

Athena I/O monitoring

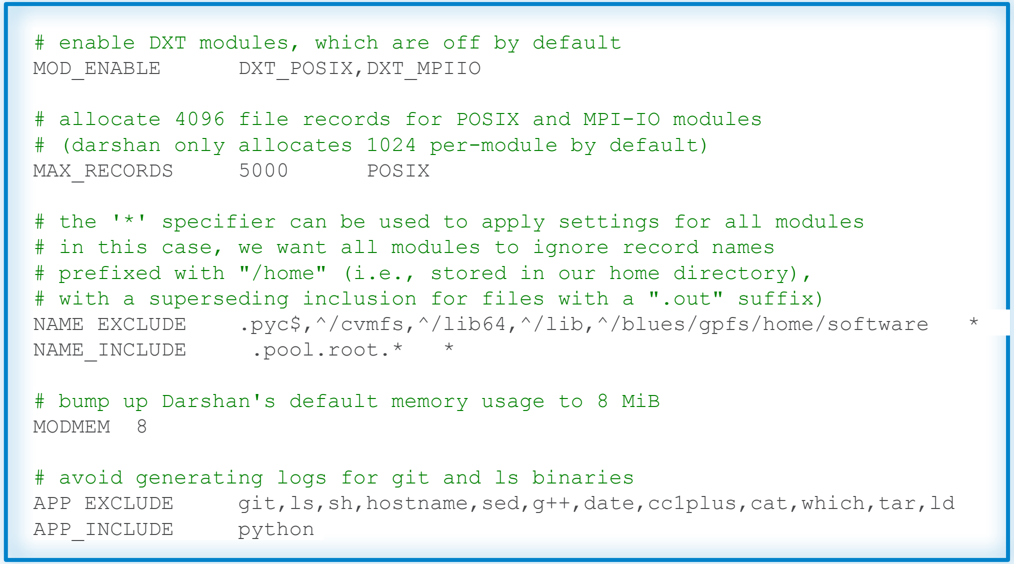
❖ Use Darshan as the I/O monitoring tool for Atlas HPC workflow to gain deeper insights into I/O patterns of Athena

Use `LD_PRELOAD` to interpose Darshan instrumentation in Athena

```
Derivation_tf.py …… --athenaopts='
--preloadlib=$DARSHAN_BASE_DIR/lib/
libdarshan.so'
```

```
> head log.EVNTtoHITS
11:00:45 Thu Oct  6 11:00:45 CDT 2022
11:00:45 Preloading
/lcrc/group/ATLAS/users/rwang/Argonne_computing/PPS-CCE/dar
shan/build_darshan/dev-fork-child-issue786/lib/libdarshan.so
11:00:45 ##############################
11:00:45 ##### DARSHAN CONFIG #####
11:00:45 ##############################
```

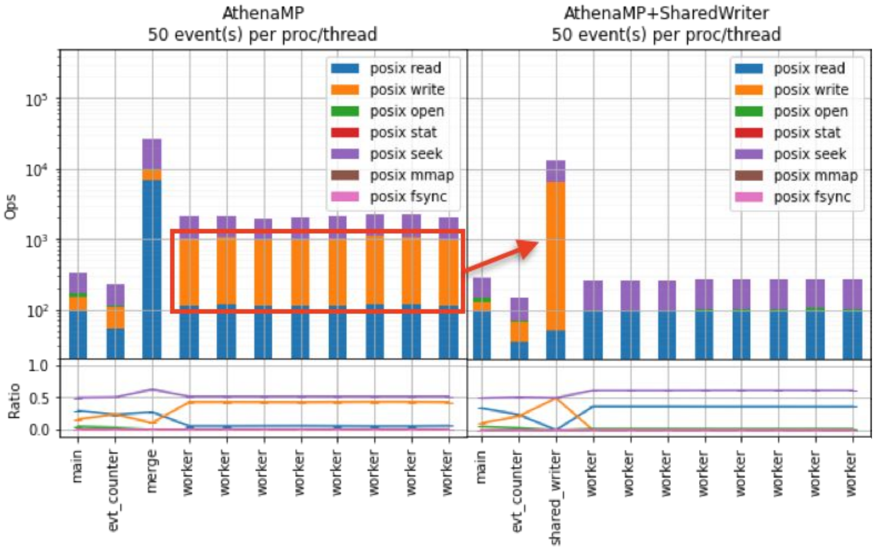
Use custom Darshan configuration to exclude `/cvmfs` activities in runtime environment



```
# enable DXT modules, which are off by default
MOD_ENABLE      DXT_POSIX,DXT_MPIIO

# allocate 4096 file records for POSIX and MPI-IO modules
# (darshan only allocates 1024 per-module by default)
MAX_RECORDS     5000      POSIX

# the '*' specifier can be used to apply settings for all modules
# in this case, we want all modules to ignore record names
# prefixed with "/home" (i.e., stored in our home directory),
# with a superseding inclusion for files with a ".out" suffix)
NAME_EXCLUDE    .pyc$,^/cvmfs,^/lib64,^/lib,^/blues/gpfs/home/software   *
NAME_INCLUDE     .pool.root.*   *

# bump up Darshan's default memory usage to 8 MiB
MODMEM  8

# avoid generating logs for git and ls binaries
APP_EXCLUDE     git,ls,sh,hostname,sed,g++,date,cc1plus,cat,which,tar,ld
APP_INCLUDE     python
```

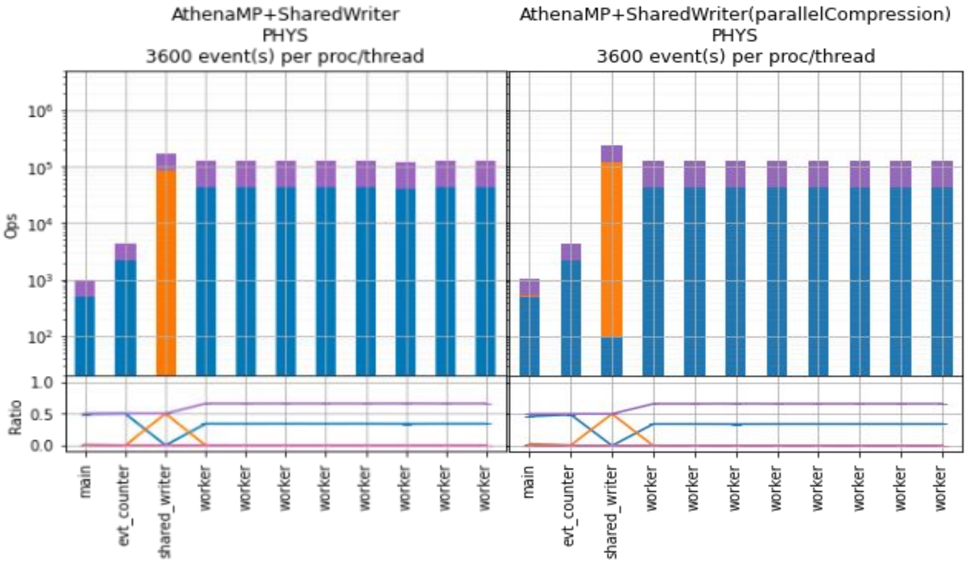
Darshan POSIX I/O analysis

Simulation



- In AthenaMP each worker writes, while a standalone merge process reads all output file of each worker then write to a single file
- In SharedWriter, a single process writes on behalf of workers

DAOD production



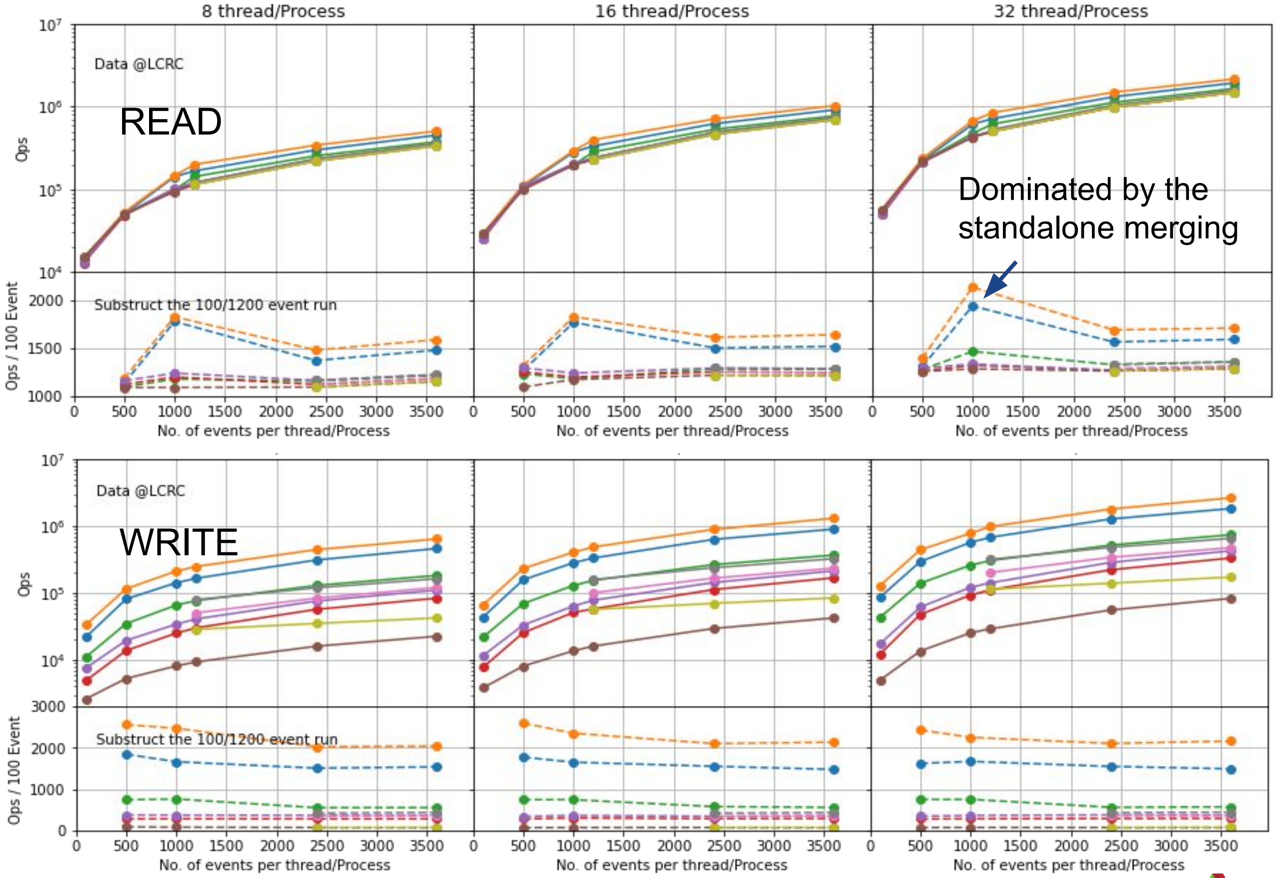
- Additional reads in the shared writer process when using parallel compression

Darshan POSIX I/O analysis

- *Parallel Compression is disabled for < 1K process*
- *Chunk size=100*

DAOD production

- **PHYS**: AOD data model with reduced trigger, MC truth and tracking info
- **PHYSLITE:** event with calibrated objects, further reduced list of variables from PHYS
- **PHYS-PHYSLITE:** producing PHYS then PHYSLITE in a train (default for ATLAS production)



READ

WRITE

Dominated by the standalone merging

What’s next for Darshan?

Ongoing Darshan development activities

- Instrumentation of DAOS libraries
  - ALCF Aurora will feature Intel's DAOS storage system, a first-of-a-kind object-based storage system for large-scale HPC platforms
  - Darshan will implement instrumentation for DAOS file and object interfaces to better understand how apps and I/O middleware make use of this new paradigm
- Continued development efforts on log analysis tools
  - Refining new PyDarshan log analysis framework
  - Recommendations, warnings, and other feedback based on observed I/O patterns
  - Analysis tools for workflows (i.e., multiple Darshan logs created by multiple job steps)

Wrapping up

- ❖ Darshan is an invaluable tool for HPC application scientists, facilities, and I/O researchers for better understanding application I/O behavior
  - ➢ Detailed instrumentation of application access to multiple layers of the HPC I/O stack
  - ➢ Helpful tools for extracting salient data from Darshan logs and summarizing for users
- ❖ Ongoing efforts from the Darshan team and the HEP community to leverage Darshan for better understanding/improving HEP I/O behavior on HPC systems!
- ❖ Please reach out with any questions, comments, or feedback!

- ❖ Darshan website, docs: https://www.mcs.anl.gov/research/projects/darshan/
- ❖ Source code, issue tracking: https://github.com/darshan-hpc/darshan
- ❖ Darshan-users mailing list: darshan-users@lists.mcs.anl.gov

Acknowledgement

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. This research also used resources and data generated from resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.