

Darshan for HEP applications

Rui Wang^{1*}, Shane Snyder^{1**}, Douglas Benjamin², Zhihua Dong², Patrick Gartung³, and Kenneth Herner³

¹Argonne National Laboratory (US)^{***}

²Brookhaven National Laboratory (US)

³Fermi National Accelerator Laboratory (US)

Abstract. Modern HEP workflows must manage increasingly large and complex data collections. HPC facilities may be employed to help meet these workflows' growing data processing needs. However, a better understanding of the I/O patterns and underlying bottlenecks of these workflows is necessary to meet the performance expectations of HPC systems.

Darshan is a lightweight I/O characterization tool that captures concise views of HPC application I/O behavior. It intercepts application I/O calls at runtime, records file access statistics for each process, and generates log files detailing application I/O access patterns.

Typical HEP workflows include event generation, detector simulation, event reconstruction, and subsequent analysis stages. A study of the I/O behavior of the ATLAS simulation and filtering stage, and the CMS simulation workflow using Darshan is presented, including insights into the I/O operations and data access size.

1 Introduction

Modern HEP workflows are increasingly scaled and complex for various data processing and analysis purposes. They used to run on large computing farms or the worldwide Grid. However, data processing needs keep growing rapidly. Employing HPC facilities may help meet these needs of the workflows so that the time required to make new scientific insights can be reduced.

I/O behavior is one of HEP workflows' most significant limiting factors, especially as the collected event data size grows and varies in order of magnitude. Therefore, the ability to instrument and monitor the I/O behavior of the HEP workflows could be critical to characterize and understand their I/O patterns and underlying bottlenecks. Solving the potential bottlenecks would help the HEP workflows meet the HPC systems' performance expectations.

2 Darshan

Darshan[1] is an I/O characterization tool developed specifically for the instrumentation of HPC applications, utilizing a lightweight, modular, and transparent design for users. To avoid

*e-mail: Rui.Wang@cern.ch

**e-mail: ssnyder@mcs.anl.gov

***This work was supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics, High Energy Physics Center for Computational Excellence (HEP-CCE).

perturbing applications, Darshan uses a bounded amount of memory and defers all expensive log writing operations until the application shutdown. A modular software architecture and file format eases the extension of the tool to account for new sources of I/O instrumentation data (e.g., a new I/O library). Utilizing Darshan requires no modifications to the user's source code, enabling it to be transparently integrated into applications with minimal effort on behalf of users. These design principles have led to Darshan being commonly deployed at HPC facilities to automatically characterize the I/O behavior of production applications.

As the application executes, Darshan *instrumentation modules* are responsible for instrumenting I/O calls and capturing other I/O characterization data from various components of the I/O subsystem. Currently implemented instrumentation modules include:

- *POSIX* and *STDIO*: instrument low-level file I/O interfaces
- *MPI-IO*: instrument MPI's parallel I/O interface
- *HDF5*: instrument the HDF5 library, commonly used for managing scientific data
- *Lustre*: capture Lustre file system striping parameters
- *Heatmap*: capture per-process histograms of I/O activity (read/write volumes) over time for various I/O interfaces (POSIX, MPI-IO, STDIO)
- *DXT[2]*: optional modules to capture full I/O traces for POSIX and MPI-IO interfaces

No specific ROOT-IO instrumentation module has been implemented yet. The ROOT I/O activities shown in Sec. 3 are captured via POSIX interface calls.

For default modules (i.e., not DXT), Darshan captures a bounded set of statistics for each file accessed by the application, including counters (e.g., operation counts, total bytes moved), timing information (e.g., total time spent doing read, write, and metadata operations), and other data. If explicitly enabled by users, DXT tracing modules can capture fine-grained I/O traces that may enable deeper insights into I/O behavior. When the application is shutting down, the Darshan core library retrieves instrumentation records from active modules, compresses the records, and writes them in a portable log file format. The generated Darshan log file can then be analyzed using Darshan analysis tools, especially using PyDarshan [3], a recently developed Python framework for extracting, summarizing, and visualizing log data.

Several enhancements have been made to Darshan for HEP use cases. First, Darshan has been modified to enable its use outside message-passing interface (MPI) applications. Breaking Darshan's dependence on MPI allows its use in new contexts, including HEP workflows, which have traditionally not been MPI-based. Second, Darshan has added a capability for instrumenting forked processes since some HEP workflows use forked children processes to split and parallelize workloads. Lastly, Darshan has improved its runtime configuration abilities, allowing users to allocate more memory for instrumentation or to focus instrumentation on specific files.

In an upcoming release, Darshan is adding the ability to instrument interfaces for the Intel DAOS storage system. This new-to-HPC object-based storage system will be deployed on the upcoming Aurora system at the Argonne Leadership Computing Facility (ALCF). DAOS provides appealing I/O performance characteristics, so detailed instrumentation is critical to understanding how applications (including HEP workflows) use this novel storage system. PyDarshan is also being refactored to support aggregation and visualization of Darshan data across multiple logs generated by the stages or forked processes of a HEP workflow as detailed in Sec. 3.

3 Case study of the HEP workflow I/O

The HEP workflow is designed to process data fast and repeatably. Simulation or detector data reconstruction is commonly broken into a few stages: (1) Generation (generate MC events for simulation) (2) Simulation (simulate the generated MC events/digitize the raw detector readout) (3) Reconstruction (reconstruct the physics objects and the events) (4) Filtering (dropping events in a disk-to-disk copy) (5) Analysis (physics analysis by users) as shown in Fig. 1.

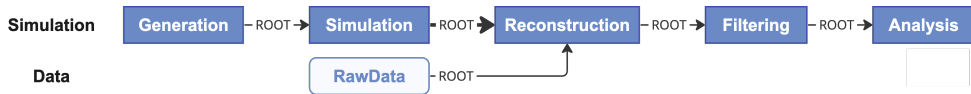


Figure 1: Major stages of a typical HEP workflow for simulated and collected data processing.

This case study used two workflows developed by Collider Physics for data processing.

- ATLAS software

- ATLAS offline software Athena

The ATLAS experiment [4] at the Large Hadron Collider (LHC) at CERN uses Athena [5] as its main simulation, reconstruction, and analysis software framework, and ROOT [6], a generic experiment-independent C++ toolkit, for its I/O and storage.

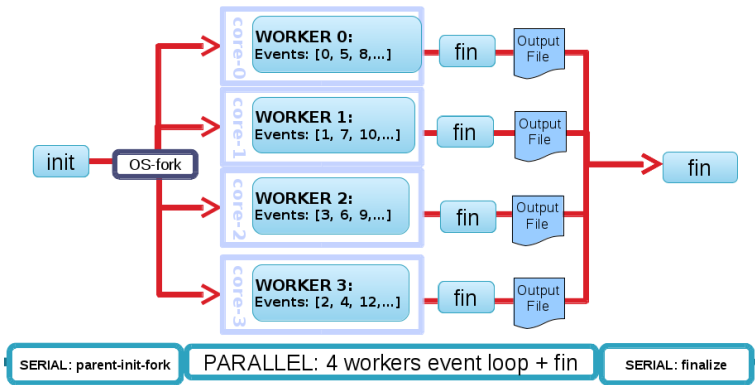
Athena was initially written to run serially. It was extended to support multi-process parallelism for Run 2, so-called AthenaMP [7], where independent parallel workers are forked from the main process with shared memory allocation as shown in Fig. 2. In the early implementation, each worker produced its own output file, with these files eventually read and merged via a serial standalone merge process. To address the inefficiency caused by re-reading and serially merging worker output, a SharedWriter process has been designed to be executed alongside the other workers, retrieving the output data objects from the workers' memory and merging them on the fly. In the meantime, as the hardware is moving to have not only more cores but also more threads, the framework is in transit to AthenaMT [9] to support multi-threading in Run3. The AthenaMT uses the Gaudi task scheduler to map tasks to kernel threads. The tasks share a single pool of heap memory.

- ATLAS xAOD analysis

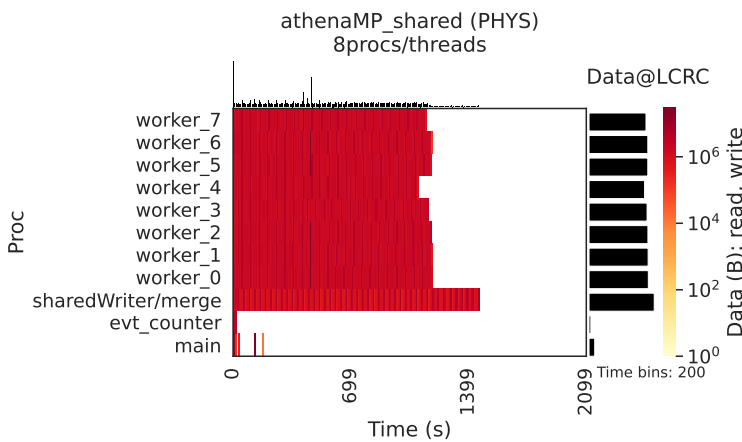
Starting from Run2, ATLAS uses xAOD (Analysis Object Data) [10], a ROOT-readable event data model, for both reconstructed object storage and analysis. A subset of the events is selected with tailored reconstruction information stored in derived AOD (DAOD) formats for specific physics analysis. Two analysis formats in Run3 are the DAOD_PHYS and DAOD_PHYSLITE, where the latter contains already calibrated physics objects.

- CMS software CMSSW

The CMS collaboration [11] at the LHC at CERN uses CMSSW [12], an overall collection of software built around a Framework, an Event Data Model (EDM), and Services needed by the simulation, calibration and alignment, and reconstruction modules that process event data for physics analysis. The framework uses the job-specific configuration file to configure the workflow stages at run time and ROOT for its I/O and storage. The



(a) Atlas AthenaMP Schematic view



(b) Data access Heatmap

Figure 2: (2a) shows the Atlas AthenaMP Schematic view [8]. (2b) shows an example of the data access visualization of the main and the eight forked worker processes in an AthenaMP run.

framework supports multi-threading. The EDM is a C++ object container for all RAW and reconstructed data related to a particular collision in ROOT format.

The ATLAS and CMS workflow stages that have been looked at are summarized in Table 1. Three stages have been explored individually under the ATLAS workflow: the simulation stage (CPU-intensive), the filtering stage (I/O-intensive), and the analysis stage (ROOT-based). For CMS, the generation, simulation, reconstruction, and filtering stages were explored within a single job. These jobs have also been repeated on various hardware and filesystems. Table 1 provides a few typical runs' running time, read/write data, and POSIX transfer rate.

Understanding the breakdown of the I/O operations by types such as read, write, open, stat, seek, mmap, and fsync for each workflow stage helps gain insights into the software and various job configurations. ATLAS and CMS workflows share similar general I/O behavior

Stage	Workflow	Job config (events * threads)	Running time (s)	I/O read/write (MB)	POSIX transfer (MB/s)
Gen	CMS	100 * 16	680	~0.007 / ~193	~49
Sim	ATLAS	50 * 8	7267	~314 / ~380	~90
	CMS	100 * 16	1648	~188 / ~6831	~266.6
Rec	CMS	100 * 16	1019	~5110 / ~1857	~128.9
Filtering	ATLAS	3600 * 8	3800 (main) 1908 (worker)	~1300 / ~0.32 (worker) ~3.13 / ~485 (writer)	~53.1 (main) ~326.8 (worker)
	CMS	100 * 16	383	~431 / ~25	~321.4
Ana	ATLAS	405K * 1	1319	~6709 / ~106	~84.5

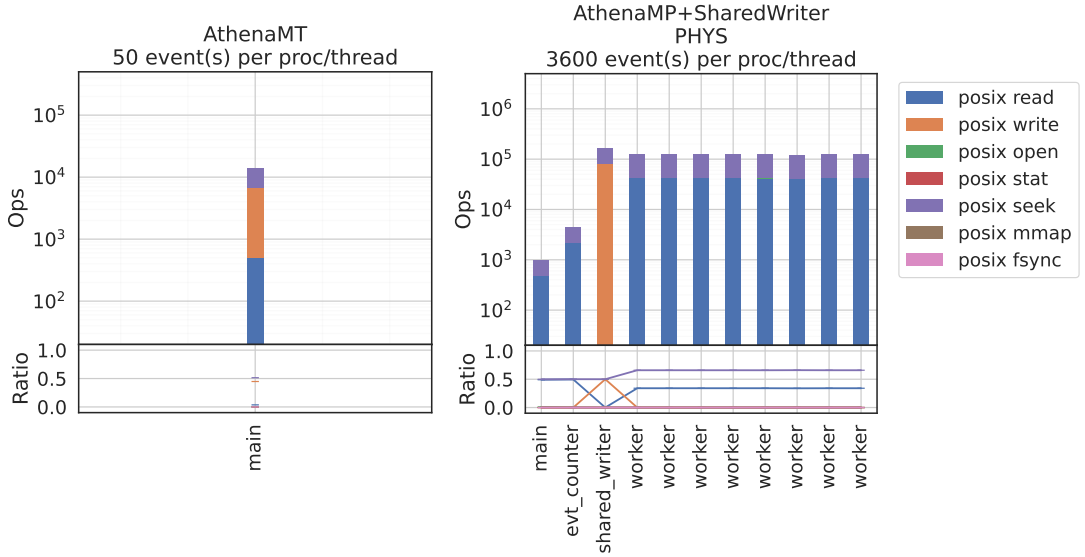
Table 1: This table summarizes the stages explored under ATLAS and CMS workflows. The ATLAS simulation and derivation jobs ran on the Broadwell nodes on LCRC at ANL with GPFS shared filesystem. The ATLAS analysis job ran on the SDCC nodes at BNL with GPFS shared filesystem. The CMS Generation, simulation, reconstruction, and filtering job ran on the Haswell nodes on Cori at NERSC with SSD and Lustre shared filesystem.

characteristics over stages. There are an equal number of writes and seeks in Generation, Simulation (Fig. 3a, 3c), Reconstruction, and SharedWriter process under the ATLAS Filtering stage (Fig. 3b). These write operations are sequential or consecutive (eg. Fig. 4a). The rest of the stages, the CMS Filtering stage, the worker process under the ATLAS Filtering stage, and the ATLAS Analysis stage have much more seeks than reads (Fig. 3b, 3d). Among these reads, the operations are majorly sequential (eg. Fig 4b), which means less random access to data.

The access size of the reads and writes has also been looked into. Compared to access sizes commonly used to access HPC storage systems, all stages mainly do small read/write at $O(1KB)$ (Eg. Fig. 5a, 5b), except for the ATLAS Analysis stage where the reads are mainly at $O(100KB)$ (Eg. Fig. 5c). This could be a potential bottleneck in the ATLAS and CMS workflow. The small reads may be related to ROOT TTreeCache vector I/O support on certain file systems. The ROOT data sieving concept (overread) might be used for this. On the other hand, the TFileCacheWrite offered by ROOT could help minimize the small writes. However, both should be studied by the experiments.

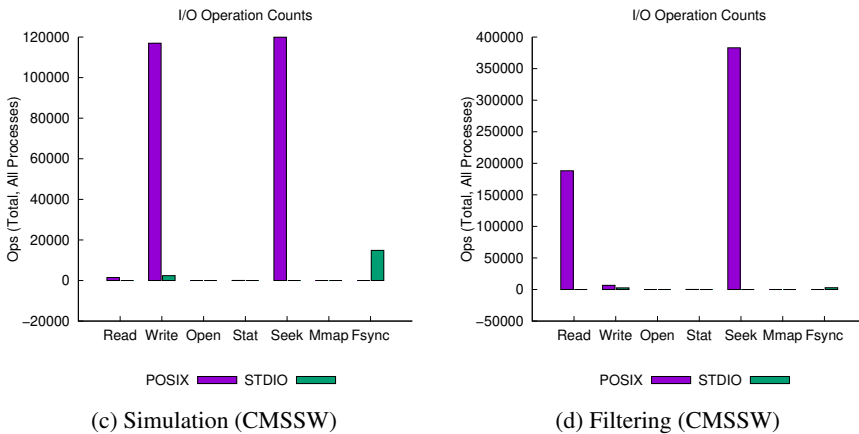
4 Conclusion

The I/O activities of various stages of the ATLAS and CMS workflows have been characterized at scale using Darshan, including the amount of data movement in various phases, the ROOT I/O patterns, and the sizes of the data access. This characterization data can be used to guide performance optimizations that better align with established best practices for HPC I/O. Small access size has been found in both workflows. It could be a potential bottleneck that should be paid attention to for future HEP workflow developments. In addition to the ATLAS and CMS workflow, the Deep Underground Neutrino Experiment (DUNE) workflow at Fermilab [13] has also been looked into. The studies demonstrate that Darshan is a lightweight tool that can assist in understanding the I/O behavior of HEP workflows. With Darshan, a better understanding of the HEP workflow I/O patterns will guide the further tuning of the ROOT and HDF5 (used for DUNE Raw data storage) I/O patterns to better inform storage capabilities requirements at HPC facilities, uncover the I/O bottlenecks in current workflows when deployed at scale on both CPU and GPU platforms, and provide recommendations for data format and access patterns for future HEP workloads.



(a) Simulation (AthenaMT)

(b) Filtering (AthenaMP)



(c) Simulation (CMSSW)

(d) Filtering (CMSSW)

Figure 3: The number of operations (read, write, open, stat, seek, mmap, and fsync) of ATLAS&CMS simulation and filtering jobs. (3a) and (3b) only include POSIX operations on the input and output files with the fraction of operations shown in the bottom panels. (3c) and (3d) include POSIX and STDIO operations on the input, output, and configuration files.

This work was supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics, and High Energy Physics Center for Computational Excellence (HEP-CCE). This work is in part supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357; in part supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale

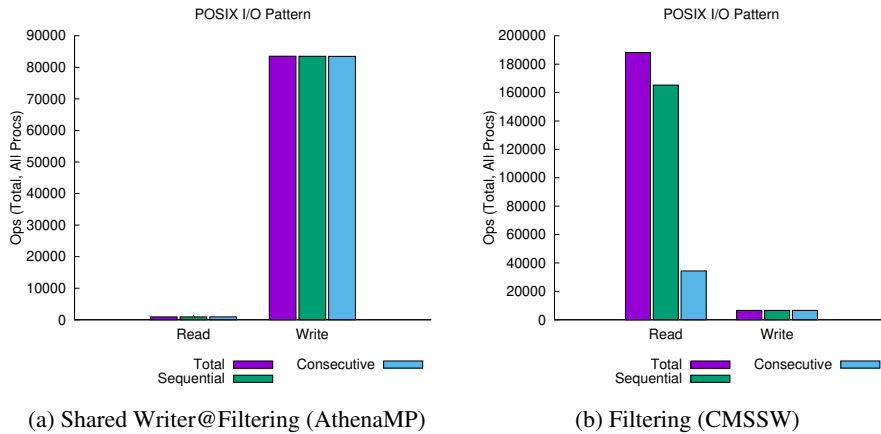
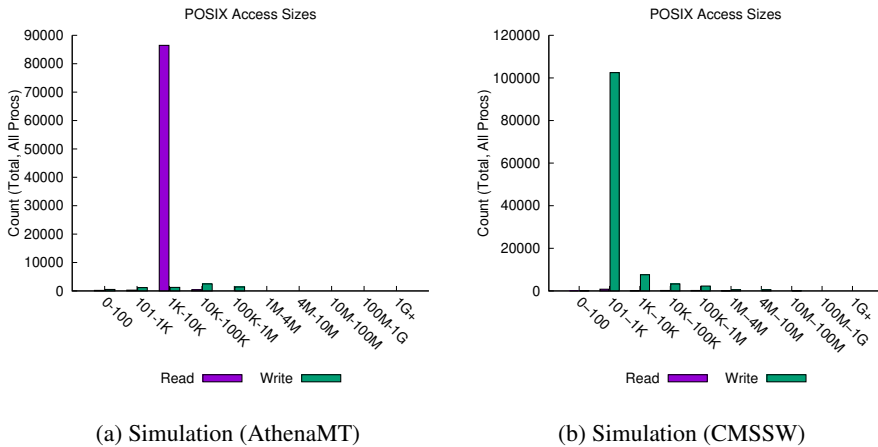


Figure 4: Sequential I/O: next access came somewhere after the last one in the file. Consecutive I/O: next access starts with the byte immediately following the last access.

computing imperative; and in part supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program. This research used resources at the Argonne Leadership Computing Facility (ALCF), Argonne Laboratory Computing Resource Center (LCRC), NERSC, and BNL Scientific Data and Computing Center (SDCC).

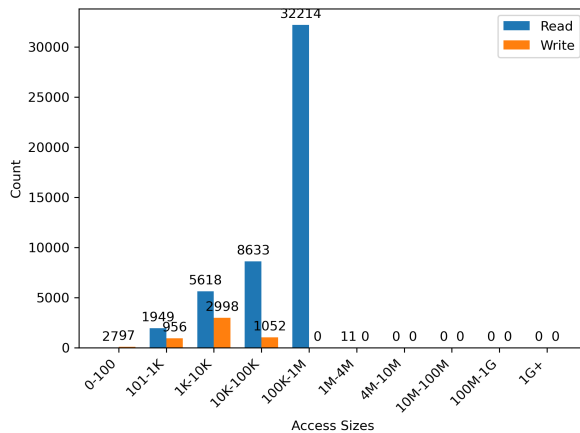
References

- [1] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, K. Riley, *24/7 characterization of petascale I/O workloads*, in *2009 IEEE International Conference on Cluster Computing and Workshops* (IEEE, 2009), pp. 1–10
- [2] C. Xu, S. Snyder, O. Kulkarni, V. Venkatesan, P. Carns, S. Byna, R. Sisneros, K. Chadalavada, *DXT: Darshan eXtended Tracing* (2019), <https://www.osti.gov/biblio/1490709>
- [3] *PyDarshan*, <https://pypi.org/project/darshan/>
- [4] ATLAS Collaboration, *JINST* **3**, S08003 (2008)
- [5] Tech. rep., CERN, Geneva (2021), all figures including auxiliary figures are available at <https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-SOFT-PUB-2021-001>, <https://cds.cern.ch/record/2767187>
- [6] R. Brun, F. Rademakers, P. Canal, A. Naumann, O. Couet, L. Moneta, V. Vassilev, S. Linev, D. Piparo, G. GANIS et al., *Root-project/root: V6.18/02*, Zenodo (2019), <https://doi.org/10.5281/zenodo.3895860>
- [7] P. Calafura, C. Leggett, R. Seuster, V. Tsulaia, P.V. Gemmeren, o.b.o.t.A. Collaboration, *Journal of Physics: Conference Series* **664**, 072050 (2015)
- [8] *AthenaMP Schematic*, <https://twiki.cern.ch/twiki/pub/AtlasPublic/ComputingandSoftwarePublicResults>



(a) Simulation (AthenaMT)

(b) Simulation (CMSSW)



(c) xAOD analysis

Figure 5: The POSIX read/write access sizes.

- [9] C. Leggett, J. Baines, T. Bold, P. Calafiura, S. Farrell, P. van Gemmeren, D. Malon, E. Ritsch, G. Stewart, S. Snyder et al., *Journal of Physics: Conference Series* **898**, 042009 (2017)
- [10] A. Buckley, T. Eifert, M. Elsing, D. Gillberg, K. Koeneke, A. Krasznahorkay, E. Moyse, M. Nowak, S. Snyder, P. van Gemmeren et al., *Journal of Physics: Conference Series* **664**, 072045 (2015)
- [11] C. Collaboration, S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic et al., *Jinst* **3**, S08004 (2008)
- [12] P. Elmer, B. Hegner, L. Sexton-Kennedy, *Experience with the CMS event data model*, in *Journal of Physics: Conference Series* (IOP Publishing, 2010), Vol. 219, p. 032022
- [13] A. Abed Abud et al. (DUNE) (2022), 2210.15665