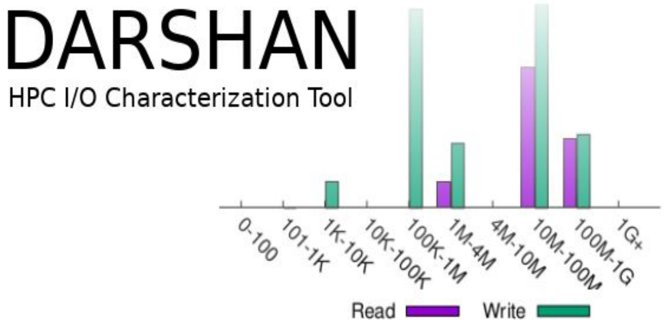
December 18, 2023

Argonne NATIONAL LABORATORY

Analyzing I/O behavior of HEP workflows with Darshan



Doug Benjamin (ANL), Patrick Gartung (FNAL), Ken Herner (FNAL), Shane Snyder (ANL), Rui Wang (ANL)

HEP-CCE All-hands meeting

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?

Two primary components:

1. Darshan runtime library

- Instrumentation modules: lightweight wrappers (interposed at link or run time) intercept application I/O calls and record statistics about file accesses
  - File records are stored in bounded, compact memory on each process

- Core library: aggregate statistics when the application exits and generate a log file
  - Collect, filter, compress records and write a single summary file for the job

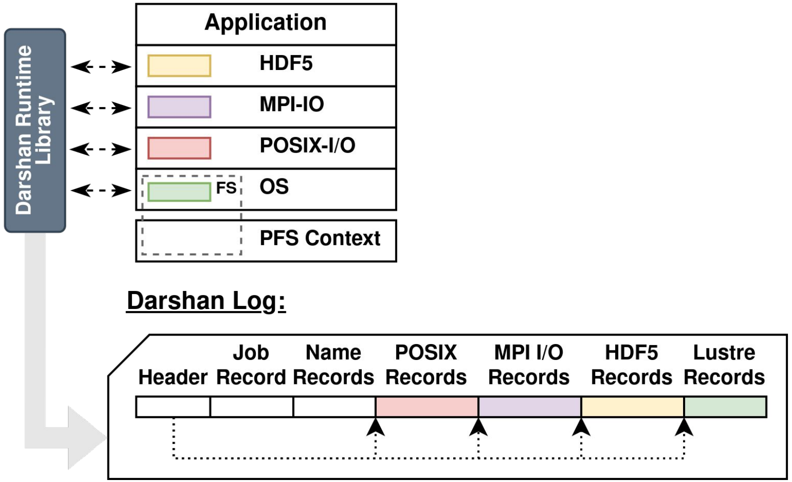


Figure courtesy Jakob Luettgau (UTK)

How does Darshan work?

Two primary components:

2. Darshan log analysis tools

- Tools and interfaces to inspect and interpret log data
  - PyDarshan command line utilities like the new job summary tool
  - Python APIs for usage in custom tools, Jupyter notebooks, etc.
  - Legacy C-based tools/library

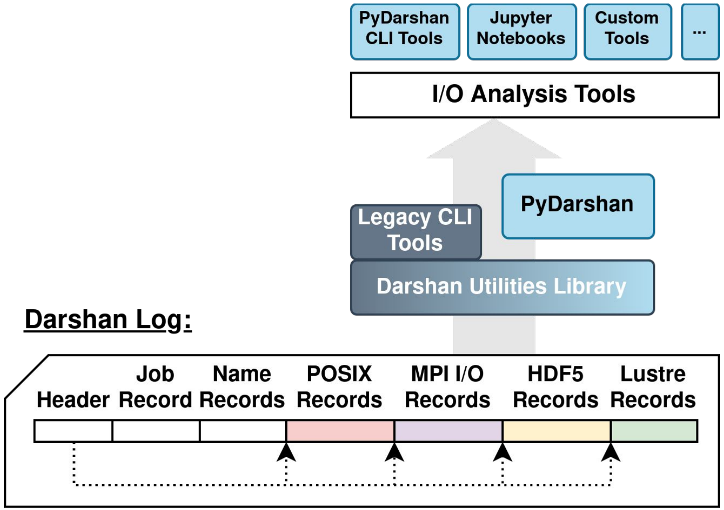


Figure courtesy Jakob Luettgau (UTK)

Darshan enhancements from HEP-CCE

Darshan enhancements for HEP use case

- Handling of `fork()` (AthenaMP)
  - Forked processes inherit a copy of parent process’s memory – including all Darshan library instrumentation state
    - Child process logs inaccurate as they include all pre-fork parent I/O
  - Modifications made to Darshan library to resolve this:
    - Mechanism to reset a process’s instrumentation state
    - Use pthread_atfork() function to define handler that resets Darshan state on fork children

```
The pthread_atfork() function registers fork handlers that are to
be executed when fork(2) is called by this thread.  The handlers
are executed in the context of the thread that calls fork(2).

Three kinds of handler can be registered:

*  prepare specifies a handler that is executed before fork(2)
   processing starts.

*  parent specifies a handler that is executed in the parent
   process after fork(2) processing completes.

*  child specifies a handler that is executed in the child
   process after fork(2) processing completes.
```

Darshan enhancements for HEP use case

- Detailed runtime library configuration
  - HEP Python frameworks access tons of files, many irrelevant for I/O analysis (shared libraries, headers, compiled Python byte code, etc.)
  - Darshan users need more control over memory limits and instrumentation scope
  - Comprehensive runtime library configuration integrated into Darshan
    - Total and per-module memory limits
    - File name patterns to ignore
    - Application name patterns to ignore

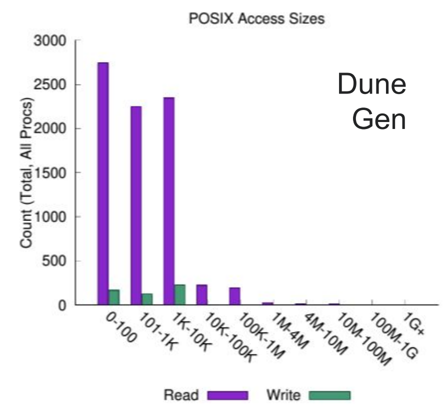
```
# allocate 4096 file records for POSIX and MPI-IO modules
# (darshan only allocates 1024 per-module by default)
MAX_RECORDS     5000      POSIX

# 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
```

Analysis of HEP workflows with Darshan – ATLAS, CMS & DUNE



Case study: ATLAS & CMS workflow

ATLAS EXPERIMENT

Broadwell on LCRC@ANL
GPFS
SDCC@BNL
Lustre

Talk + proceeding @ CHEP2023

Simulation Ge ... sis

Data

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)	~1300 / ~0.32 (worker)	~53.1 (main)
			1908 (worker)	~3.13 / ~485 (writer)	~326.8 (worker)
	CMS	100 * 16	383	~431 / ~25	~321.4
Ana	ATLAS	405K * 1	1319	~6709 / ~106	~84.5

CMS

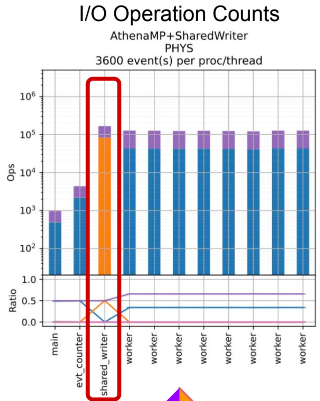
Haswell on Cori @Nersc
SSD + Lustre
100 events, 16 threads

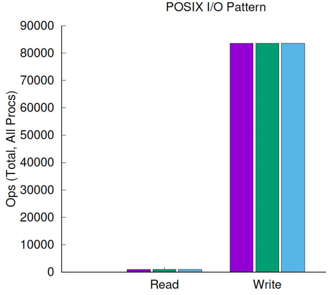
- **Multi-thread**
- **Multi-process**
- **Serial**

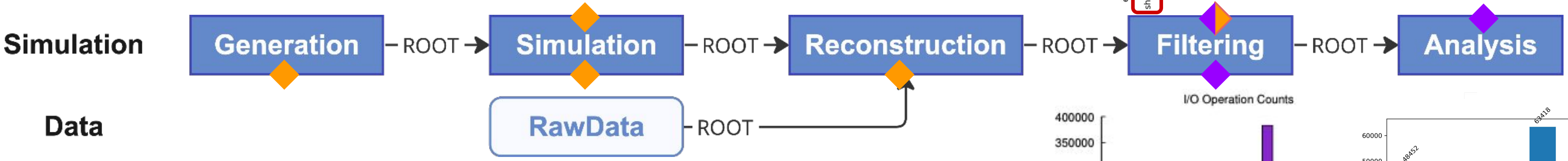
Case study: I/O operations

Broadwell on LCRC@ANL
GPFS
SDCC@BNL
GPFS

- **Equal number of writes/seeks**
  - Generation & Simulation & Reconstruction & SharedWriter process in Filtering stage at ATLAS (marked)
- **Equal sequential & consecutive I/O**

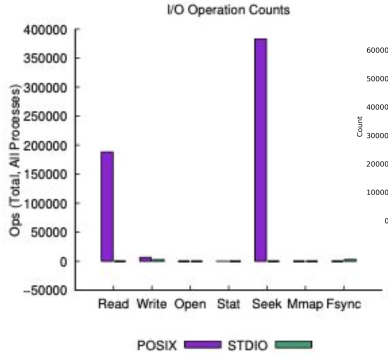


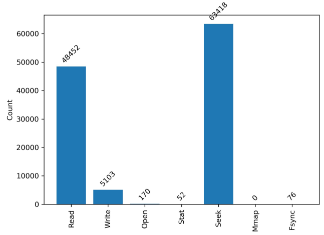




Haswell on Cori @Nersc
SSD + Lustre
100 events, 16 threads

- **Seeks > reads**
  - Filtering stage (worker process at ATLAS)
  - Analysis stage
- **Sequential > consecutive I/O**

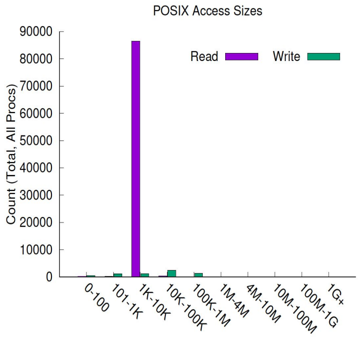


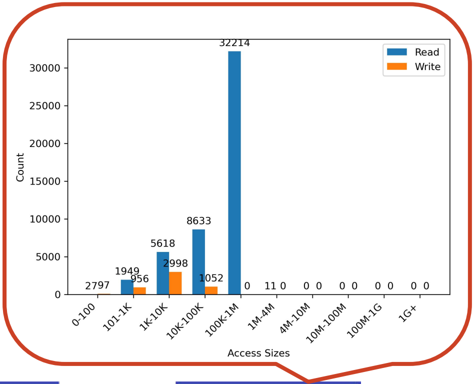


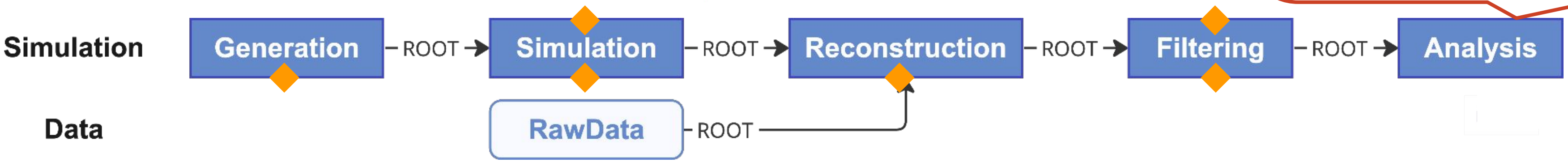
Sequential – next access came somewhere after the last one in the file
Consecutive – next access starts with the byte immediately following the last access

Case study: Access size

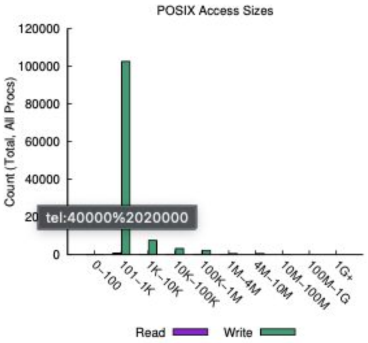
Broadwell on LCRC@ANL
GPFS
SDCC@BNL
GPFS







Haswell on Cori @Nersc
SSD + Lustre
100 events, 16 threads



Small reads/writes at O(1KB)

- All stages (marked) except ATLAS Analysis which is at O(100KB)
- Related to ROOT TTreeCache vector I/O support on certain FSes
- **Potential bottleneck need to be paid attention to for future HEP workflow developments**
- ROOT has a data sieving concept (overread) that might be taken advantage of

Deliver Darshan as a tool for HEP

Fixes/enhancements to common software and experiments frameworks

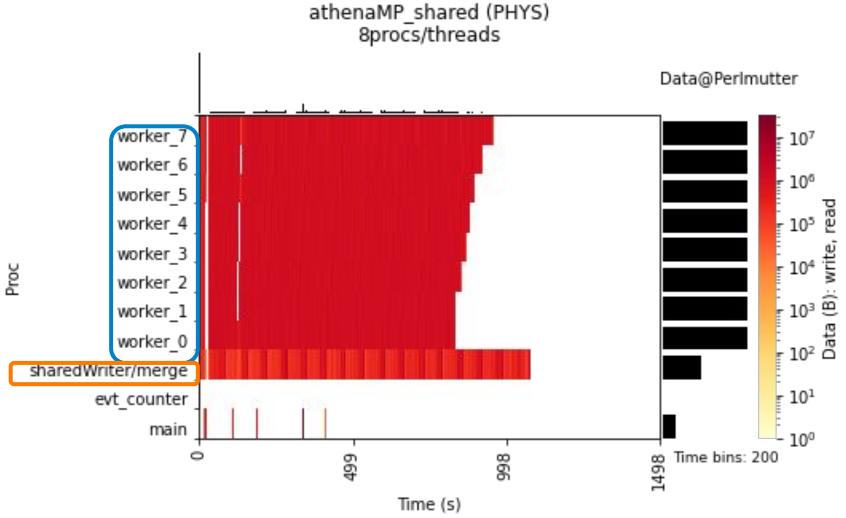
- Darshan included fork-safety and better filtering for I/O.
- ROOT serialization bottleneck was fixed.
- Patch to resolve the Athena library issue on DSO loading hooks which cause PyRoot crash when running with Darshan

Available in CVMFS

- Installed in ATLAS ALRB as an external tool
- Can be load and used out of box

Workflow I/O characterization

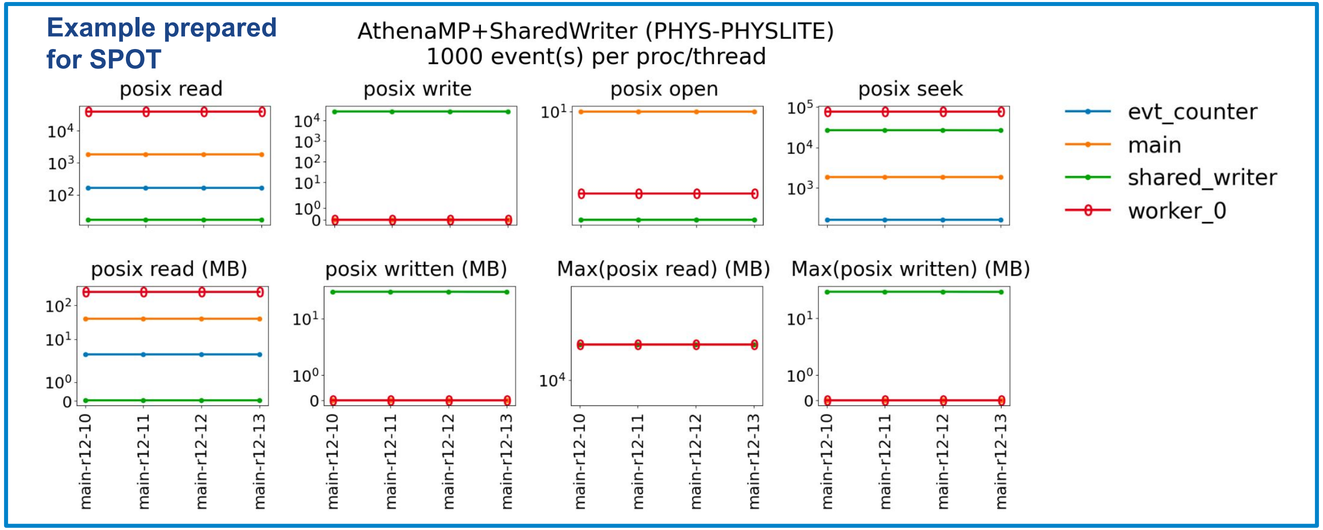
- Capture MPI and HDF I/O
- GPU workflows Benchmarking (DUNE)
- Darshan with container (SULI project)
- Monitor analysis workflows to better understand optimal storage parameter for data products



Heatmap visualization of multiprocess data processing workflow (AthenaMP). 8 **workers** read input data, while a **shared writer** process writes all worker output data from shared memory.

Adding Darshan to SPOT Toolkit

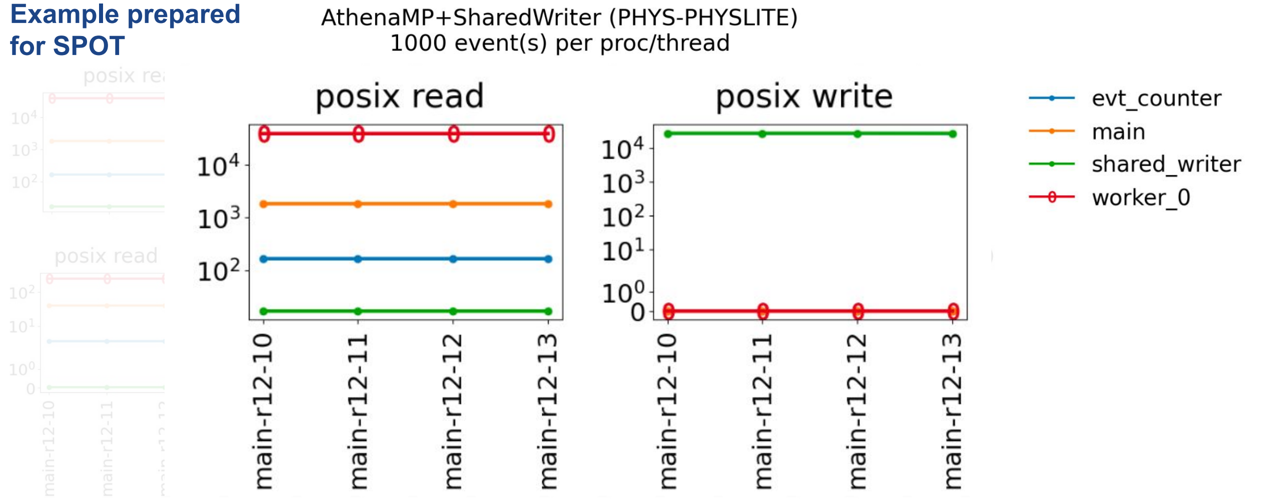
- Software performance monitoring between releases at ATLAS – SPOT
  - Guiding the evolution of the software and EDM to optimize performance in technical performance, resource usage needs and usability for analysis
  - Monitoring the performance of the software, including the transient and persistent event data models
  - **Darshan adds abilities of insight on forked processes in time & detailed data access of specific file(s)**
    - Detailed usage to be explored with the SPOT



Adding Darshan to SPOT Toolkit

- Software performance monitoring between releases at ATLAS – SPOT
  - Guiding the evolution of the software and EDM to optimize performance in technical performance, resource usage needs and usability for analysis
  - Monitoring the performance of the software, including the transient and persistent event data models
  - **Darshan adds abilities of insight on forked processes in time & detailed data access of specific file(s)**
    - Detailed usage to be explored with the SPOT

Example prepared for SPOT



Student Engagement

Yanli Lyu (Summer intern from NMSU)

- Refining PyDarshan log analysis framework for workflow analysis

Adhithya Vijayakumar (2023 summer SULI from Texas A&M)

- I/O monitoring for portable HPC applications (report)
  - Darshan with container

Nehemyah Green (Summer student from IIT)

- Working on Ceph, learn Darshan with Ken

Yanli Lyu
New Mexico State University

Adhithya Vijayakumar
Texas A&M University Physics