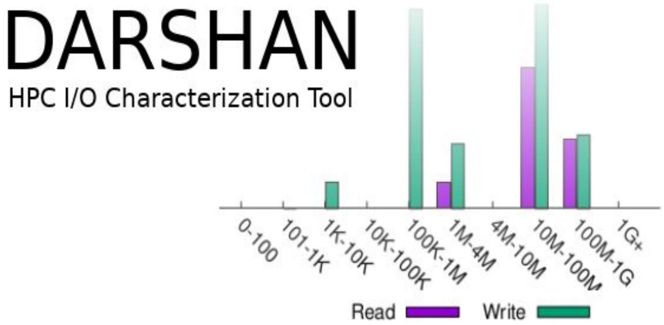
Analyzing HEP workflow I/O behavior with Darshan



Douglas Benjamin[2], Patrick Gartung[3], Kenneth Herner[3], Shane Snyder[1], Rui Wang[1], Zhihua Zhang[2]

1. Argonne National Laboratory
2. Brookhaven National Laboratory
3. Fermi National Accelerator Laboratory

HEP-CCE AHM, April ‘23

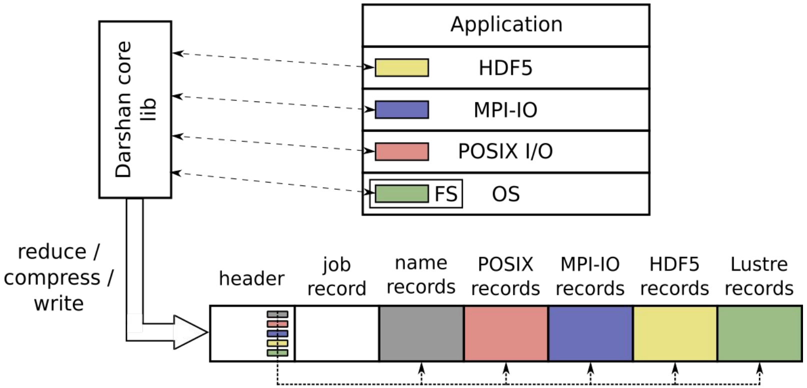
U.S. DEPARTMENT OF ENERGY

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

Darshan background

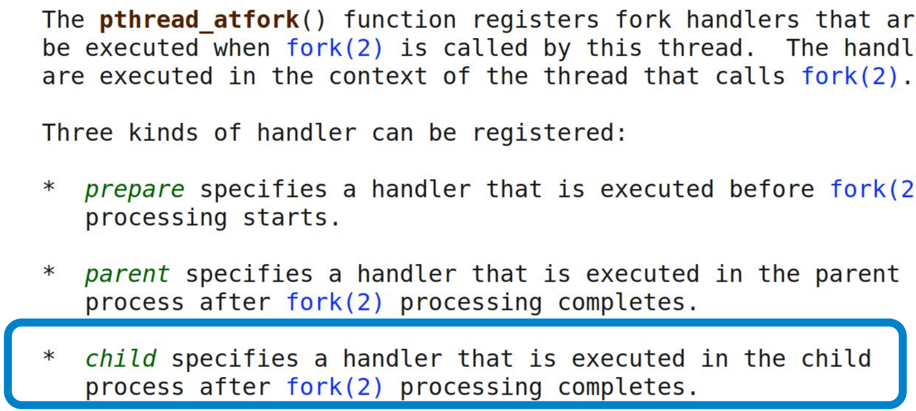
- Darshan is a lightweight I/O characterization tool that captures concise views of HPC application I/O behavior
  - Produces a summary of I/O activity for each instrumented job
    - Counters, histograms, timers, & statistics
    - If requested by user, full I/O traces

- *Widely available* – Deployed (and commonly enabled by default) at many HPC facilities
- *Easy to use* – no code changes required
- *Modular* – straightforward to add new instrumentation sources



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

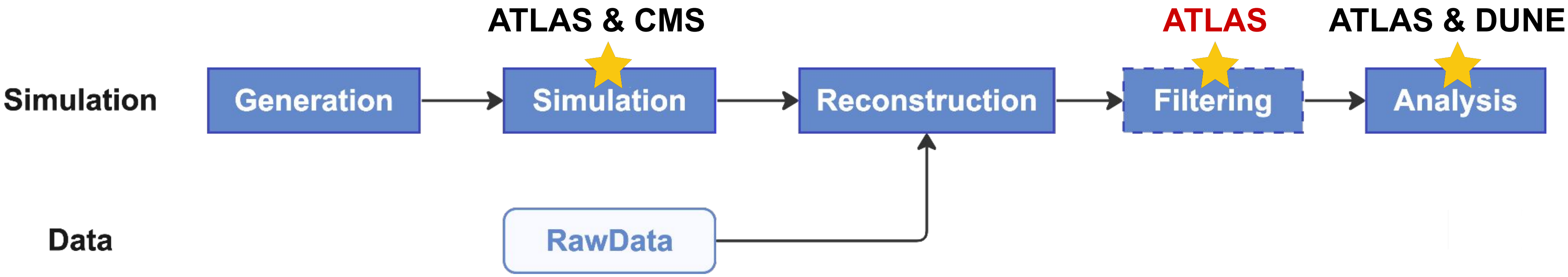
- Detailed runtime library configuration
  - HEP Python frameworks access tons of files, many irrelevant for I/O analysis (shared libraries, headers, compiled Python byte code, etc.)
  - Darshan users need more control over memory limits and instrumentation scope
  - Comprehensive runtime library configuration 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
```

Darshan for HEP

Characterize of various workflow stages at scale to gain insight on the I/O patterns

- Guide the further tuning of the I/O patterns to better inform storage capabilities requirements at facilities
- Uncover the I/O bottlenecks in current workflows when deployed at scale
- Provide recommendations for data format and access patterns for future HEP workloads



Runtime configs and examples are collected under HEP-CCE IOS repository

ATLAS offline software – Athena

Serial Athena **Run1**

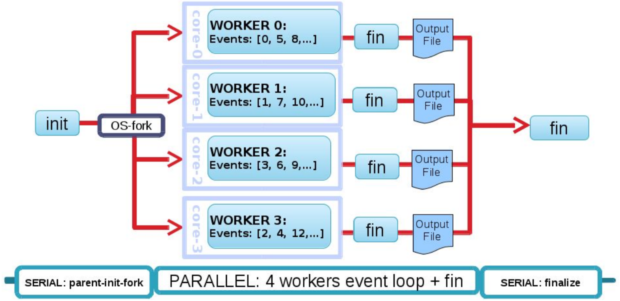
Multi-Process **Run2 – Run3**

- **AthenaMP+standalone merging**
  - 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**
  - A shared writer process does all the output writes
  - Reduce time on single thread merging process
- **AthenaMP+sharedWriter (parallelCompression)**
  - Using parallel compression to reduce the time increment when moving to higher No. of process

Multi-thread **Run3**

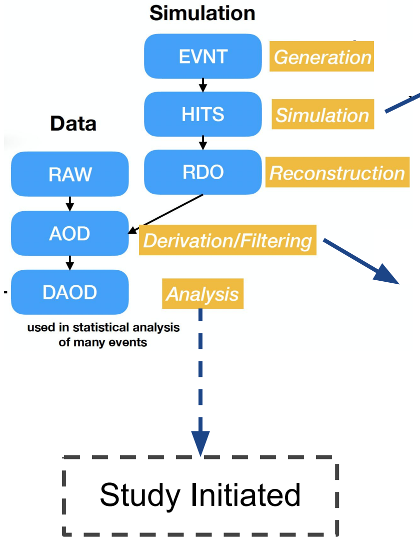
- **AthenaMT**
  - Gaudi task scheduler maps task to kernel threads
  - Shared single pool of heap memory

Schematic View of ATLAS AthenaMP



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

Athena I/O characterization



Simulation

EVNT

Generation

HITS

Simulation

Data

RAW

RDO

Reconstruction

AOD

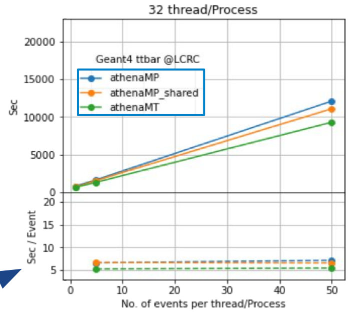
Derivation/Filtering

DAOD

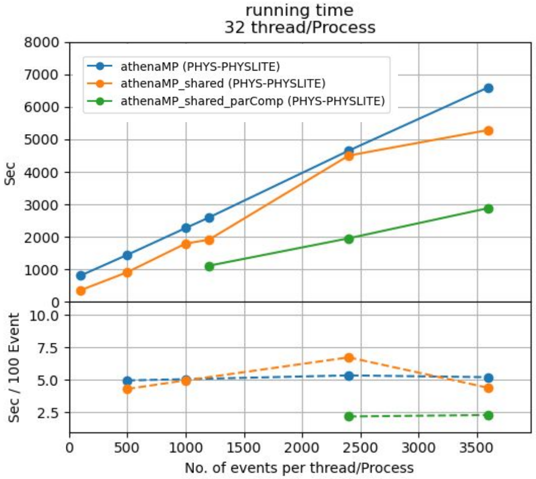
Analysis

used in statistical analysis of many events

Study Initiated



32 thread/Process



running time
32 thread/Process

MC Simulation – CPU intensive

Report @ Oct. 2022 AHM

- AthenaMP+Standalone merging
- AthenaMP+SharedWriter
- AthenaMT

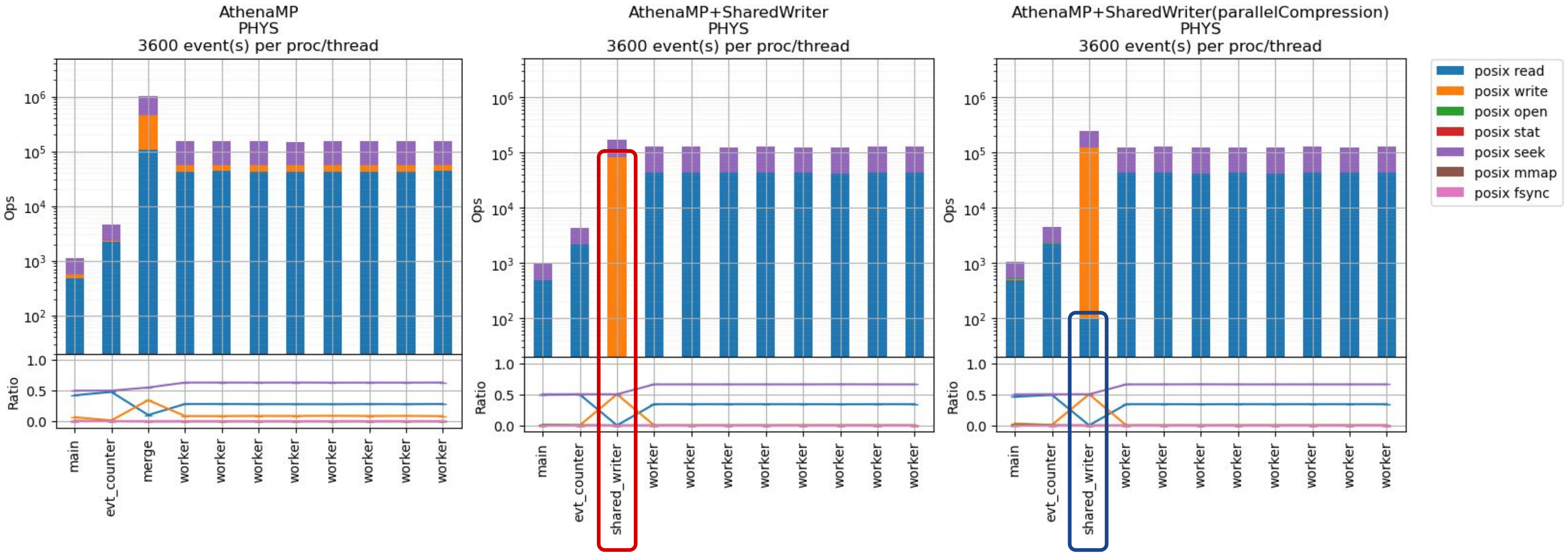
Derivation (DAOD) production – I/O intensive

- AthenaMP+Standalone merging
- AthenaMP+SharedWriter
- AthenaMP+SharedWriter (parallel compression)
  - Enabled only for >1K process

xAOD Analysis

- Athena (serial)

DAOD Production – POSIX operations



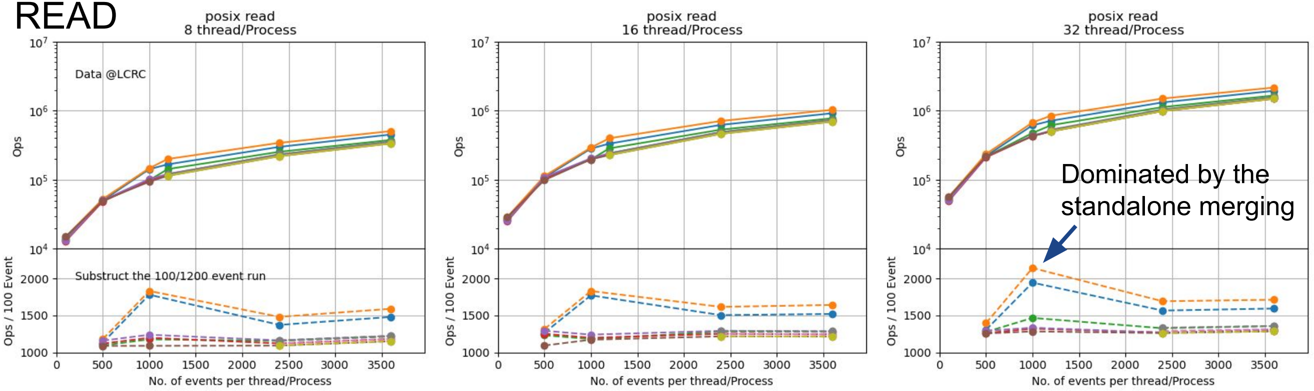
- Standalone merging reads all output file of each worker then write to a single file
- SharedWirter take over all the writes each worker does
- Additional reads in the shared writer process when using parallel compression

DAOD Production – POSIX operations

- **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)

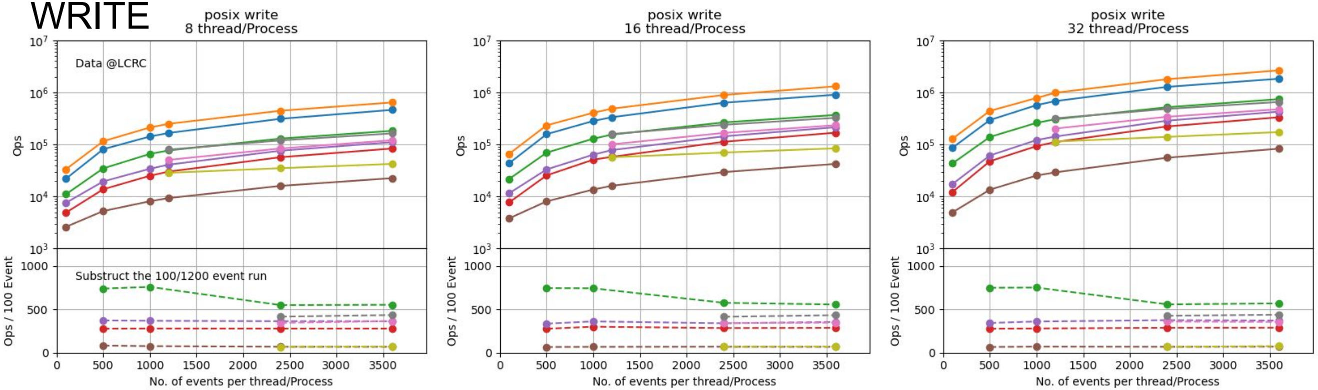
Auto flush: 100 events

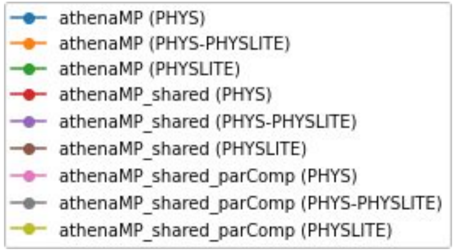
READ



Dominated by the standalone merging

WRITE

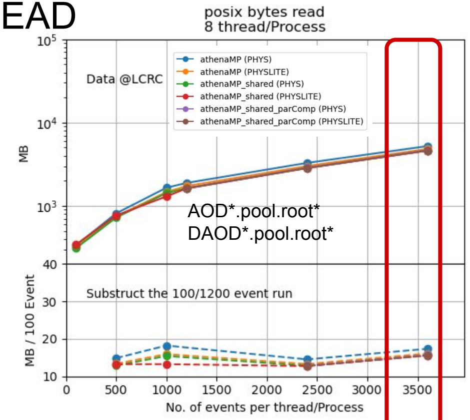



athenaMP (PHYS)
athenaMP (PHYS-PHYSLITE)
athenaMP (PHYSLITE)
athenaMP_shared (PHYS)
athenaMP_shared (PHYS-PHYSLITE)
athenaMP_shared (PHYSLITE)
athenaMP_shared_parComp (PHYS)
athenaMP_shared_parComp (PHYS-PHYSLITE)
athenaMP_shared_parComp (PHYSLITE)

DAOD Production – Read/Write

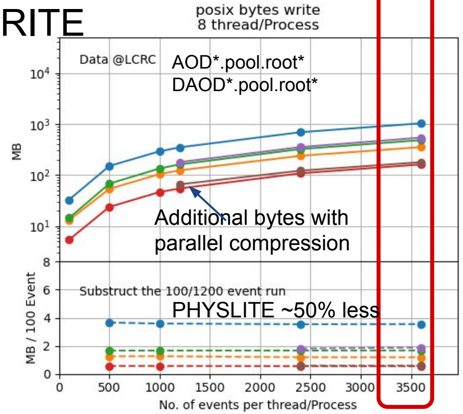
Auto flush: 100 events

READ



AOD*.pool.root*
DAOD*.pool.root*

WRITE

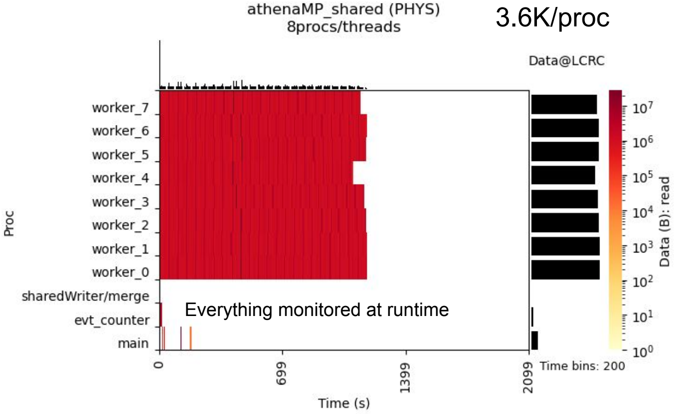


AOD*.pool.root*
DAOD*.pool.root*

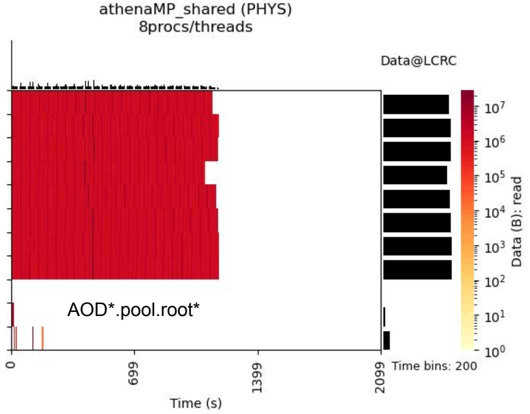
Additional bytes with parallel compression

PHYSLITE ~50% less

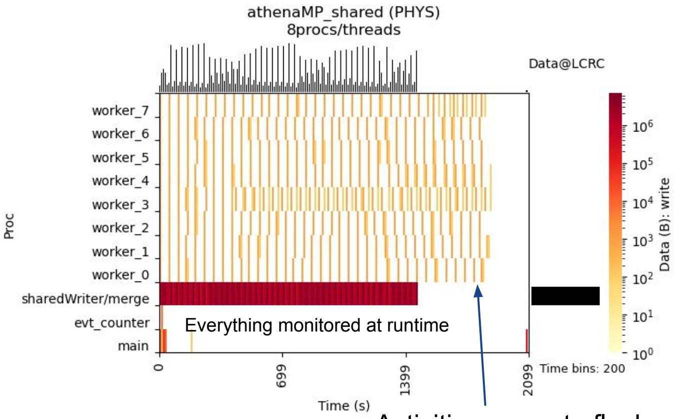
3.6K/proc



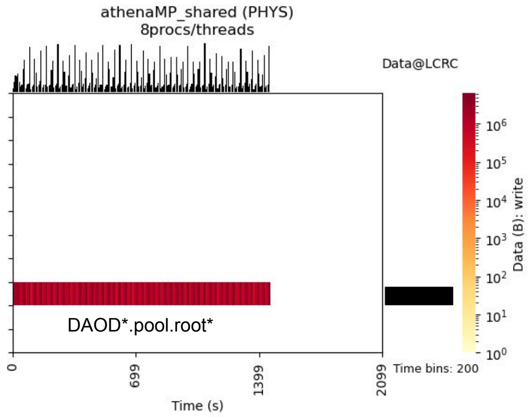
Everything monitored at runtime



AOD*.pool.root*



Everything monitored at runtime



DAOD*.pool.root*

Activities per auto flush

ATLAS Workflow monitoring

Darshan has been installed in ATLAS ALRB as an external tool available from CVMFS

- `lsetup darshan`
- Work out of box when proper log path been provided
- Relocatable preferred for the future release
  - No issue found in the current build with the darshan tools

Plan

- Add to pilot
  - Job could have Darshan enabled during submission
- Customized runtime config example for each stage
- Monitoring plots
  - Input, output & condition data

```
> export
ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
> source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh -3
> lsetup darshan

******************************************************************************
Requested:  darshan ...
 Setting up darshan 3.4.2-fix1-x86_64-centos7 ...
>>>>>>>>>>>>>>>>>>>>>>>>>>> Information for user
<<<<<<<<<<<<<<<<<<<<<<<<<<<
 darshan:
   DARSHAN_LOGDIR is set to
/lcrc/group/ATLAS/users/rwang/Argonne_computing/PPS-CCE/darshan/darshan_test/athena
    Or you can 'export DARSHAN_LOGDIR=<path>' to customize the
log path.
   You must 'export LD_PRELOAD=$DARSHAN_LD_PRELOAD' to
enable instrumentation
    of applications.
******************************************************************************
```

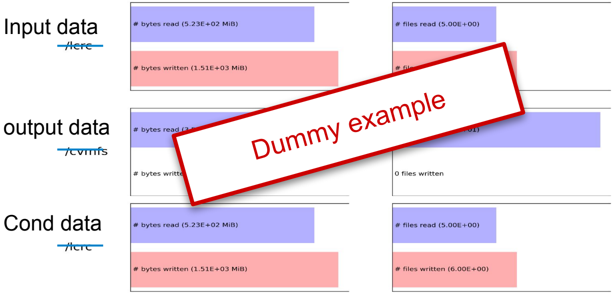
ATLAS Software performance monitoring

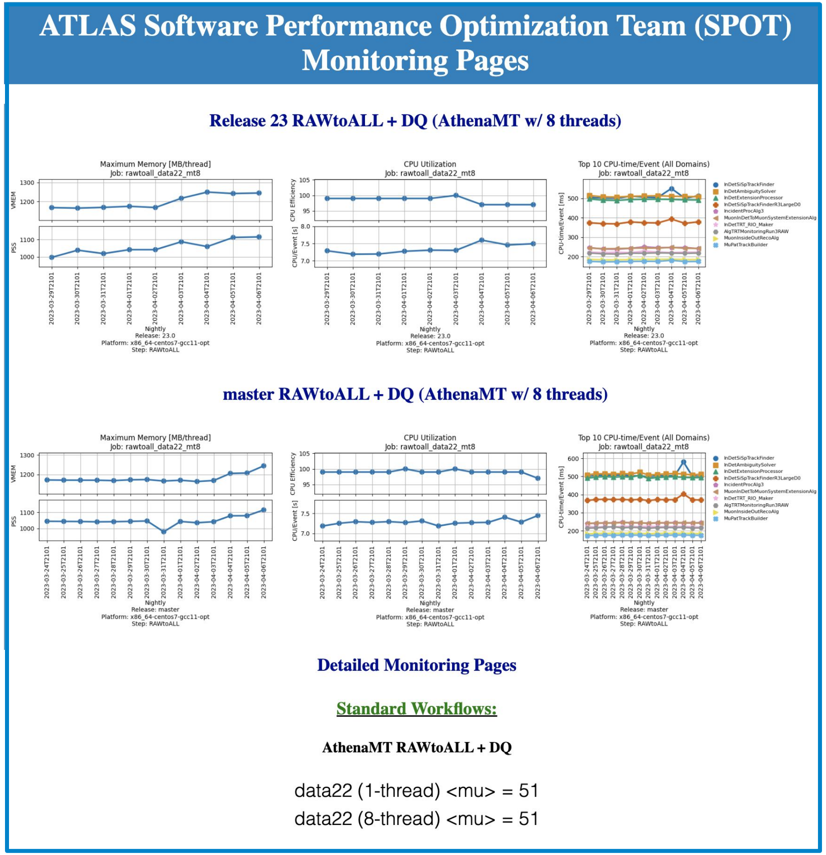
Release change in the software could make large impact on the performance

- ATLAS SPOT monitoring the performance of the software, including the transient and persistent event data models
- Guiding the evolution of the software and EDM in order to optimize performance in its multiple aspects: technical performance, resource usage needs and usability for analysis

Plan

- Design and add Darshan test to SPOT
  - SPOT use `prmon` to trace the overall performance
  - Darshan provides insight on forked processes in time & detailed data access of specific file(s)





ATLAS Software Performance Optimization Team (SPOT) Monitoring Pages

Release 23 RAWtoALL + DQ (AthenaMT w/ 8 threads)

master RAWtoALL + DQ (AthenaMT w/ 8 threads)

Detailed Monitoring Pages

Standard Workflows:

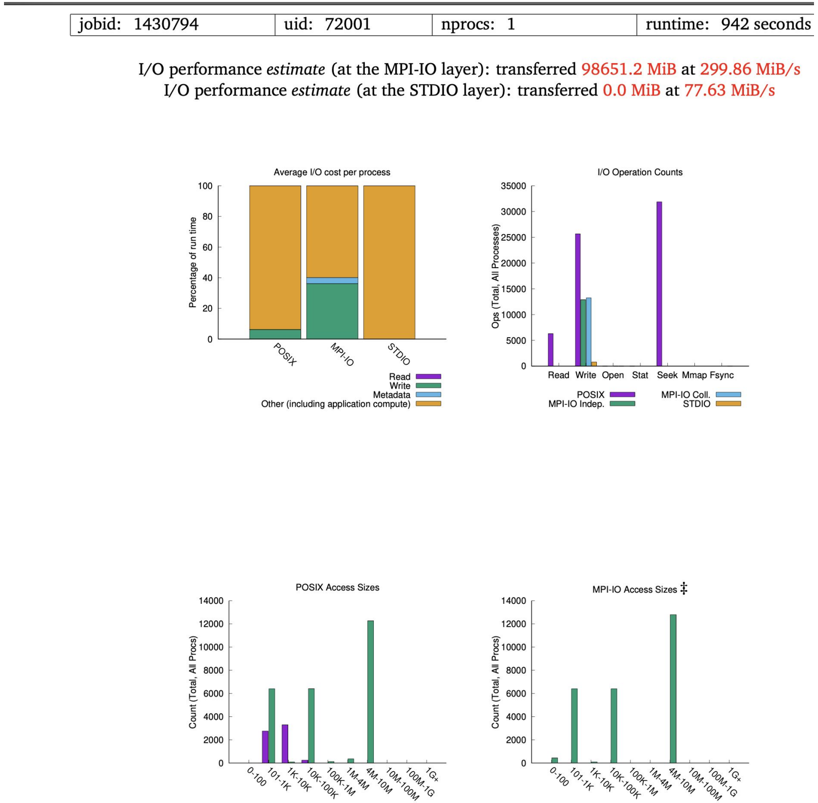
AthenaMT RAWtoALL + DQ

data22 (1-thread) <mu> = 51
data22 (8-thread) <mu> = 51

https://atlaspmb.web.cern.ch/atlaspmb/

CMS plan

- Currently working on capturing MPI and HDF IO info in Darshan logs for mpi_threaded_test_io
    - Captures POSIX and MPI IO info but only in non-MPI mode – Each rank is treated as a process.



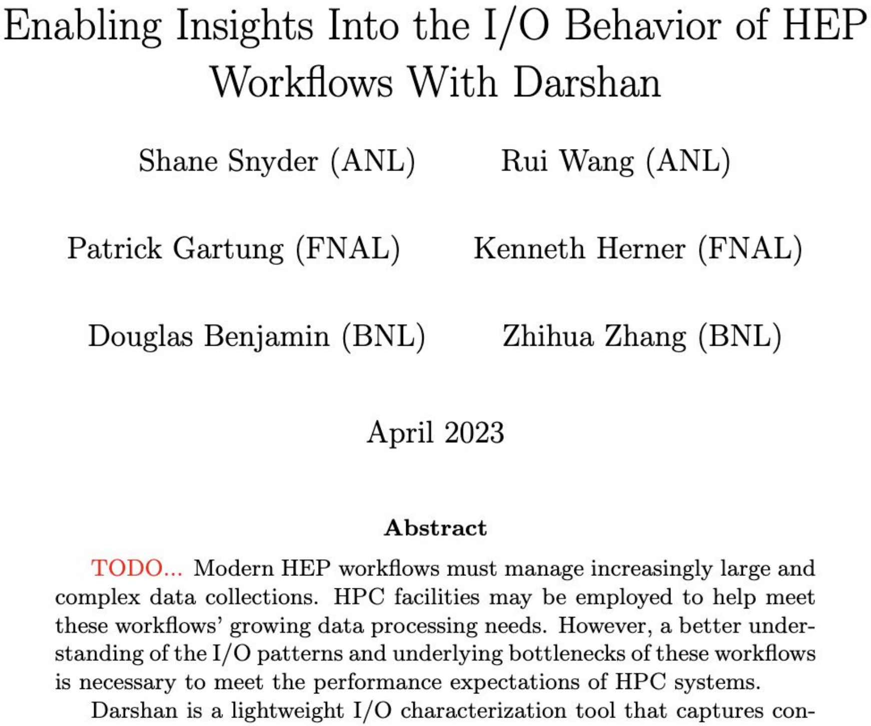
jobid: 1430794	uid: 72001	nprocs: 1	runtime: 942 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 98651.2 MiB at 299.86 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 77.63 MiB/s

DUNE plan

- ❖ Currently working on benchmarking some workflows that make use of GPUs in part
- ❖ Might need some development effort to capture the I/O in this case (doesn’t go through an intermediate file) but would be interesting to see
- ❖ Currently using ROOT files, but will start looking at algos that read in HDF5 files in the intermediate term (ProtoDUNE Run II DAQ will generate HDF5)

Darshan white paper



Enabling Insights Into the I/O Behavior of HEP Workflows With Darshan

Shane Snyder (ANL) Rui Wang (ANL)

Patrick Gartung (FNAL) Kenneth Herner (FNAL)

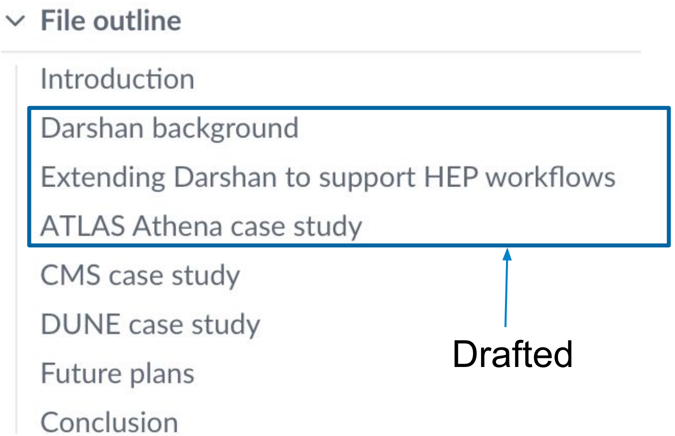
Douglas Benjamin (BNL) Zhihua Zhang (BNL)

April 2023

Abstract

TODO... 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 con-



File outline

- Introduction
- Darshan background
- Extending Darshan to support HEP workflows
- ATLAS Athena case study
- CMS case study
- DUNE case study
- Future plans
- Conclusion

Drafted

https://www.overleaf.com/project/64246f4b882e40db87f8d53f

Next steps for Darshan

- Instrumentation of Intel DAOS I/O libraries
  - Upcoming exascale system at Argonne, Aurora, will feature a new-to-HPC object-based storage system
  - Appealing performance characteristics for I/O middleware (e.g., HDF5 and ROOT) that can effectively leverage storage model
  - **File-based module complete, native object-based module underway**

- Darshan analysis tools for workflows
  - Refactor PyDarshan code to more easily allow aggregation and visualization of Darshan data across multiple logs (e.g., multiple logs generated by the steps of an HEP workflow)
  - **Planning underway, aim to push on this development this summer with a student**