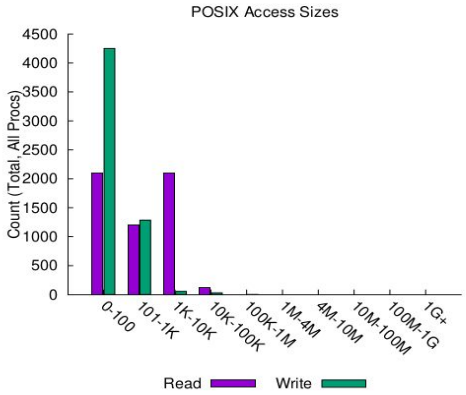
October 16, 2020

Argonne NATIONAL LABORATORY

Characterizing and understanding the behavior of HDF5 I/O workloads with Darshan

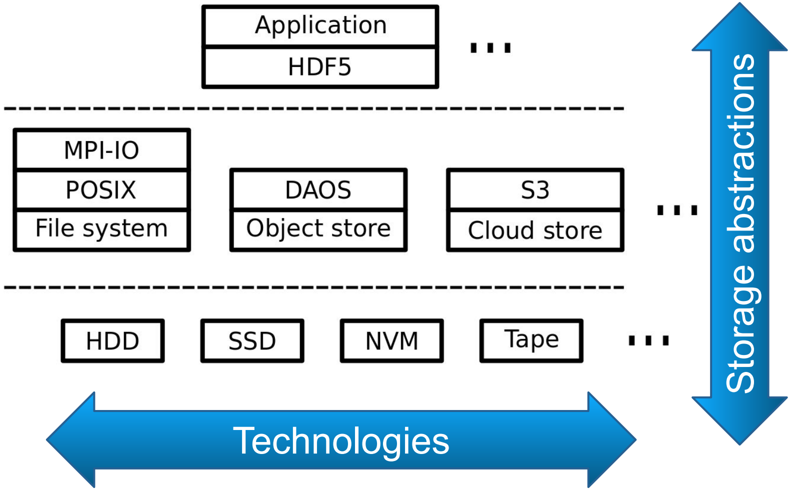


Shane Snyder
Argonne National Laboratory

HDF Users Group (HUG) ’20

Motivation

- ❖ HDF5 offers a convenient abstraction for large data collections, but it can be difficult to understand how it interacts with lower layers of the I/O stack that most impact performance
  - ➢ Users may not adequately understand the linkage between their I/O workloads and attained performance
- ❖ Instrumentation of HDF5 I/O workloads can be critical to understanding and improving their use of storage resources
  - ➢ This data can inform tuning decisions of individual users, or to better understand broader HDF5 usage in the wild



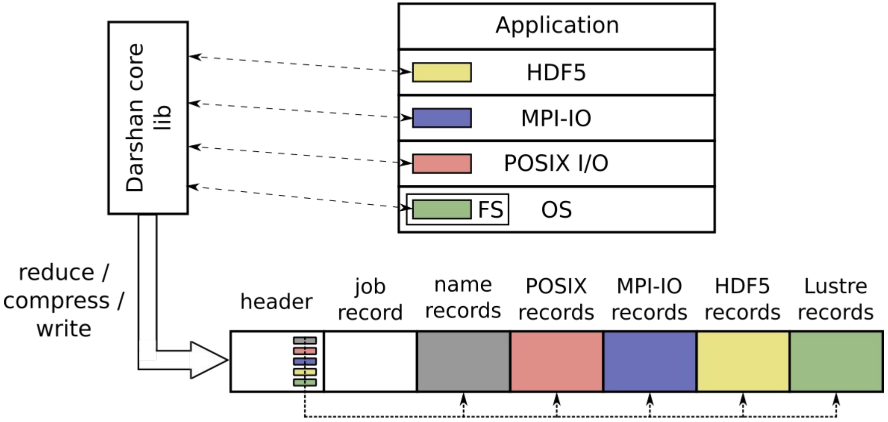
Darshan: An application-centric I/O characterization tool

Darshan background

- ❖ Darshan is a lightweight I/O characterization tool that captures concise views of application I/O behavior
  - ➢ For each instrumented job, produce a summary of I/O activity for each file accessed
    - ■ Counters, histograms, timers, & statistics
    - ■ Full I/O traces (if requested)
- ❖ Widely available
  - ➢ Deployed (and typically enabled by default!) at many production computing facilities
- ❖ 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

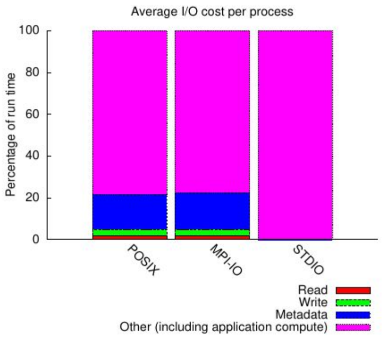
How does Darshan work?

- ❖ Darshan inserts application I/O instrumentation at link-time (for static executables) or at runtime (for dynamic executables)
  - ➢ Darshan has traditionally depended on MPI, but recent versions (3.2.0+) can also instrument serial applications (only for dynamically-linked executables)
- ❖ As app executes, Darshan records file access statistics for each process
  - ➢ Per-process memory usage is bounded to limit runtime overheads
- ❖ At app shutdown, collect, compress, and write log data
  - ➢ For MPI applications, use collective operations to reduce shared file records and write log data

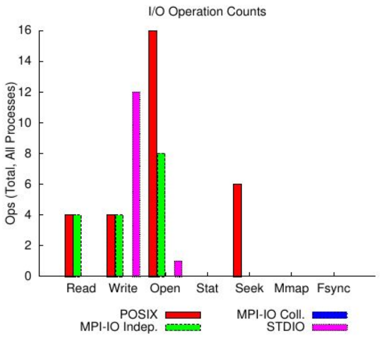


Analyzing Darshan logs

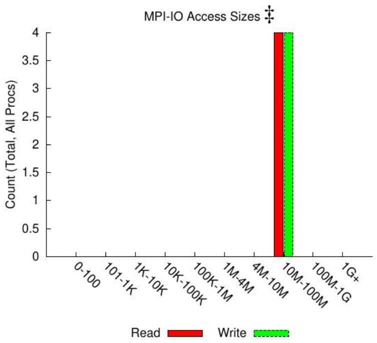
❖ With a log generated, Darshan offers command line analysis tools for inspecting log data
  - `darshan-parser` - provides complete text-format dump of all counters in a log file
  - `darshan-job-summary` - provides a summary PDF characterizing application I/O behavior



I/O operation costs across different I/O interfaces



I/O operation counts across different I/O interfaces



I/O access size ranges used by application

Integrating HDF5 support into Darshan

Darshan HDF5 instrumentation

- To provide a deeper understanding of HDF5 I/O workloads, we have developed a detailed instrumentation module for Darshan[1] that characterizes I/O behavior from HDF5 file- (H5F) and dataset-level (H5D) perspectives
  - Characterize dataset properties, access patterns, organization within files, etc.

- This data not only characterizes an application’s usage of the HDF5 library, but can help contextualize HDF5 I/O behavior with that of lower layers of the I/O stack (e.g., MPI-IO or POSIX layers) that Darshan also instruments
  - Do high-level HDF5 dataset accesses decompose efficiently into underlying MPI-IO and POSIX file system accesses?
  - If not, what optimizations (e.g., collective I/O, chunking) make most sense?

1. Available starting in Darshan version 3.2.0

Darshan HDF5 instrumentation

- H5F instrumentation highlights:
  - Operation counts
    - open/create
    - flush
  - MPI-IO usage
  - Metadata timing

```
#<module>   <rank>  <record id> <counter>    <value> <file na
H5F -1  11831850109748558379    H5F_OPENS    8   /home/shane/
H5F -1  11831850109748558379    H5F_FLUSHES 0   /home/shane/
H5F -1  11831850109748558379    H5F_USE_MPIIO   1   /home/sh
H5F -1  11831850109748558379    H5F_F_OPEN_START_TIMESTAMP
H5F -1  11831850109748558379    H5F_F_CLOSE_START_TIMESTAMP
H5F -1  11831850109748558379    H5F_F_OPEN_END_TIMESTAMP
H5F -1  11831850109748558379    H5F_F_CLOSE_END_TIMESTAMP
H5F -1  11831850109748558379    H5F_F_META_TIME 0.019466
```

Darshan HDF5 instrumentation

- H5D instrumentation highlights:
  - Operation counts:
    - open/create
    - read/write
    - flush
  - Total bytes read/written
  - Access size histograms
  - Dataspace selection types
    - Regular hyperslab
    - Irregular hyperslab
    - Points
  - Dataspace total dimensions, points
  - Chunking parameters
  - MPI-IO collective usage
  - Deprecated function usage
  - Read, write, and metadata timing

```
#<module>   <rank>  <record id> <counter>   <value>
H5D -1  7600138186531619366 H5D_OPENS   8   /home/sh
H5D -1  7600138186531619366 H5D_READS   16  /home/sh
H5D -1  7600138186531619366 H5D_WRITES  16  /home/sh
H5D -1  7600138186531619366 H5D_FLUSHES 0   /home/sh
H5D -1  7600138186531619366 H5D_BYTES_READ  4194304
H5D -1  7600138186531619366 H5D_BYTES_WRITTEN   4194
H5D -1  7600138186531619366 H5D_RW_SWITCHES 4   /hom
H5D -1  7600138186531619366 H5D_REGULAR_HYPERSLAB_SE
xt4
H5D -1  7600138186531619366 H5D_IRREGULAR_HYPERSLAB_
xt4
H5D -1  7600138186531619366 H5D_POINT_SELECTS   0
H5D -1  7600138186531619366 H5D_MAX_READ_TIME_SIZE
H5D -1  7600138186531619366 H5D_MAX_WRITE_TIME_SIZE
H5D -1  7600138186531619366 H5D_SIZE_READ_AGG_0_100
```

A Darshan+HDF5 example

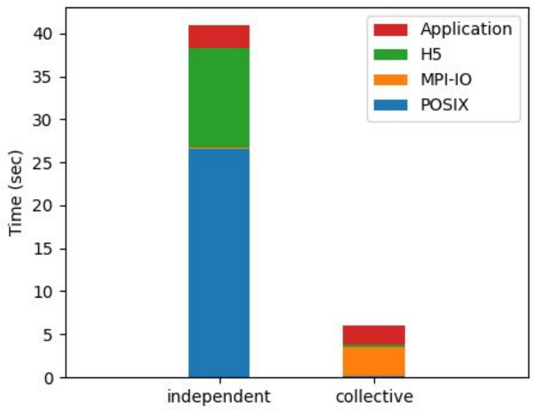
- Using the MACSio[1] HDF5 plugin, run a couple of simple examples demonstrating the types of insights HDF5 I/O instrumentation can enable
  - 60-process (5-node) single shared file, 3d mesh, write roughly 1 GiB of cumulative H5D data
  - Compare performance of collective and independent I/O configurations

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

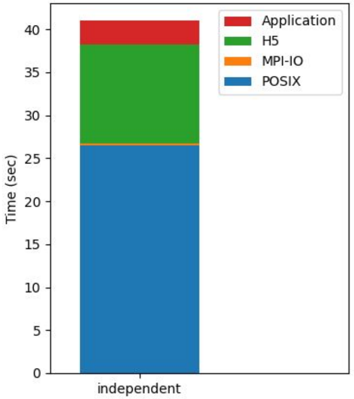
A Darshan+HDF5 example

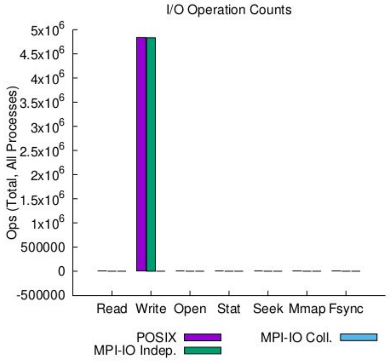
b/w: **~30 MB/sec**

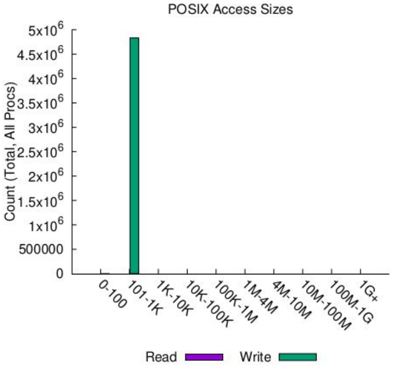
POSIX I/O dominates, **H5** incurs non-negligible overhead forming this workload

Negligible time spent in **MPI-IO**

Average per-process time spent in I/O







Nearly 5 million **POSIX** writes, all less than 1KB in size -- challenging workload for a parallel file system

Number of **MPI-IO** writes same as **POSIX** writes -- no transformations at **MPI-IO** layer

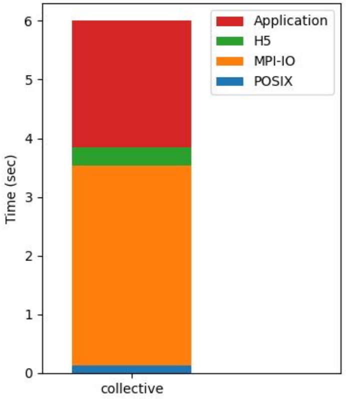
A Darshan+HDF5 example

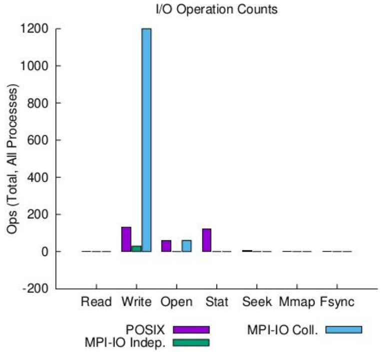
b/w: **~290 MB/sec**

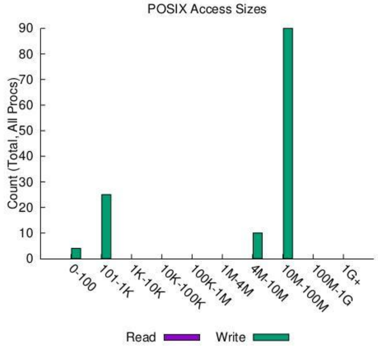
H5 and POSIX incur minimal overhead for this workload

MPI-IO collective I/O algorithm dominates

Average per-process time spent in I/O







Considerable reduction in number of POSIX writes, with some accesses in the O(10 MB) range

Notice there are still some MPI-IO independent writes for HDF5 metadata

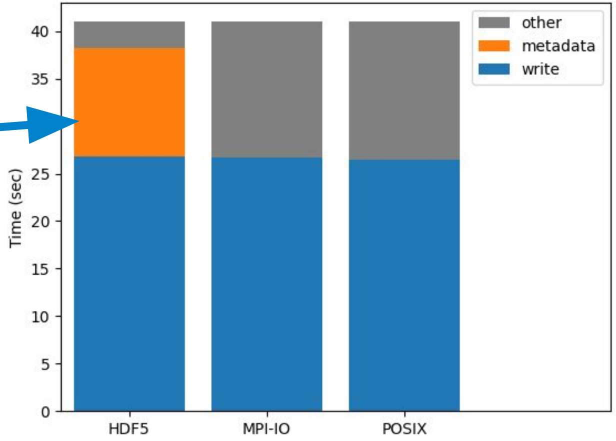
A Darshan+HDF5 example

This graph provides a slight variation on previous graphs showing relative costs of different types of I/O operations (write and metadata) within different APIs

More than 99% of HDF5 metadata time spent in H5F-level functions instrumented by Darshan

- ➢ H5F metadata cost can be completely attributed to file creation/close for this workload
- ➢ This H5F metadata cost does not translate to metadata costs at other layers, yet it seems unlikely this ~10 seconds is just due to the writing of HDF5 metadata at file open/close?

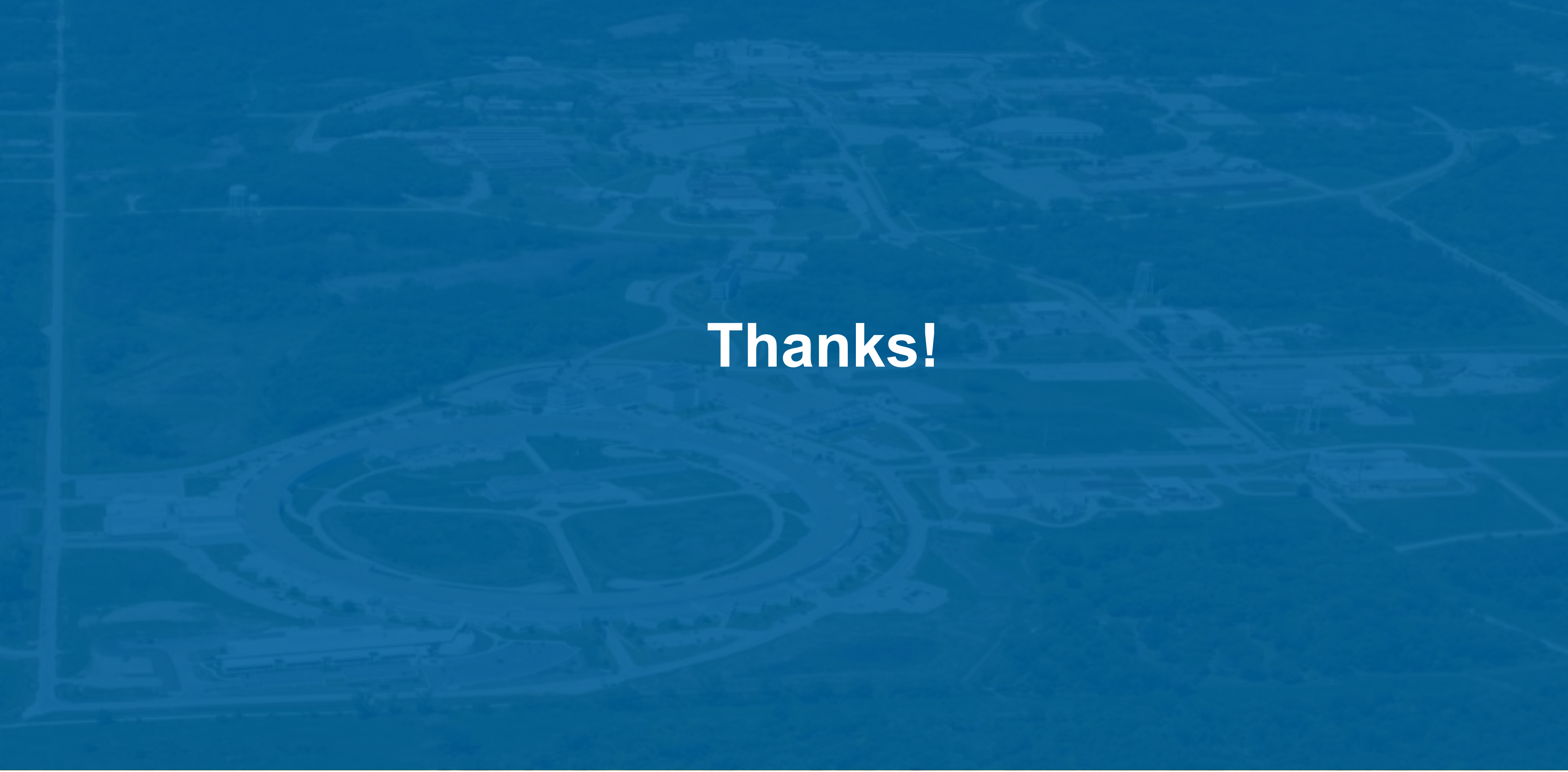
Average per-process I/O cost at different API levels



Independent I/O

Wrapping up

- Integrating HDF5 support into the Darshan I/O characterization tool enables a better understanding of HDF5 application I/O workloads and their interaction with underlying storage layers
  - This instrumented HDF5 data can be used in Darshan analysis tools to assist users in detecting inefficiencies in application I/O behavior and to inform their tuning decisions
- While we have already released a Darshan version with HDF5 support, it’s not too late to make an impact -- we’d love to hear more from the HDF community!
  - What else should we instrument? What are effective ways of visualizing this data?
- Darshan website: https://www.mcs.anl.gov/research/projects/darshan/
- Darshan-users mailing list: darshan-users@lists.mcs.anl.gov
- Source code, issue tracking: https://xgitlab.cels.anl.gov/darshan/darshan



Thanks!