2024 HDF5 USER GROUP (HUG) MEETING

U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB

Scientific Data Division

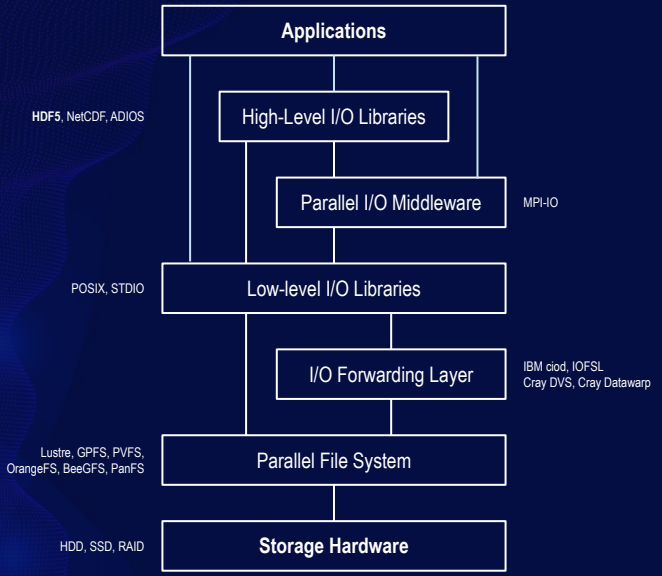
Drishti VOL

THE PERFORMANCE PROFILING AND TRACING HDF5 VOL CONNECTOR

JEAN LUCA BEZ <jlbez@lbl.gov>

COMPLEX I/O STACK!

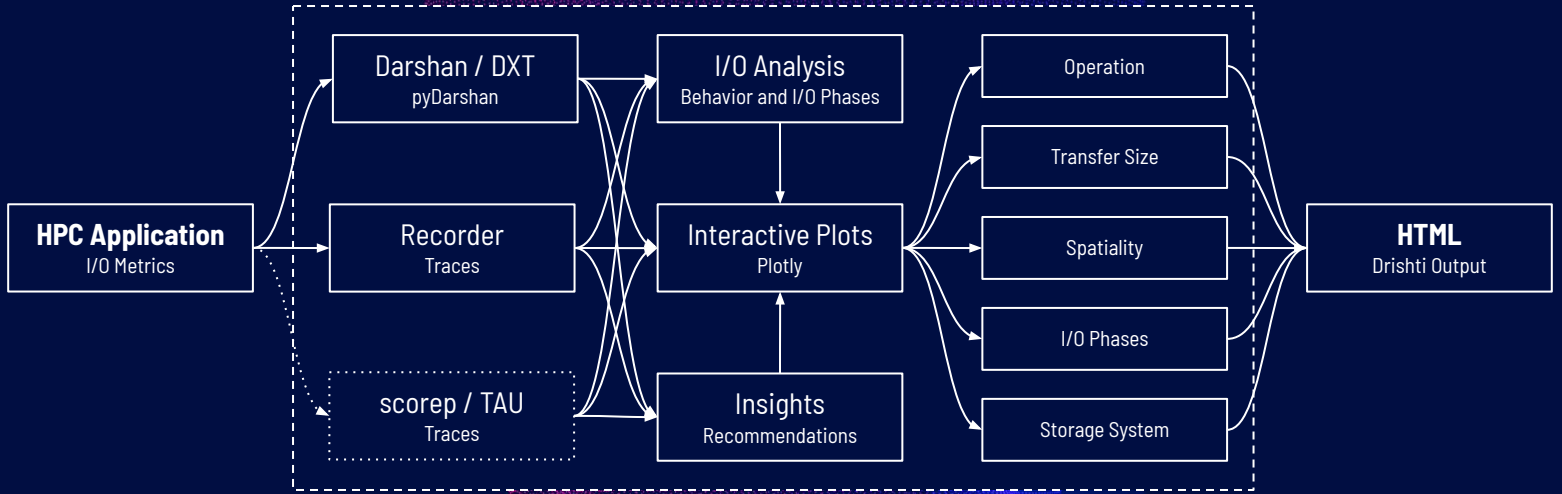
- Using the HPC I/O stack efficiently is a **tricky problem**
- **Interplay of factors** can affect I/O performance
- Various **optimizations** techniques available
- Plethora of tunable **parameters**
- Each layer brings a new set of parameters



WHAT IS THE PROBLEM?

- There is still a gap between profiling and tuning
- How to convert I/O metrics to **meaningful information**?
  - **Visualize** characteristics, behavior, and bottlenecks
  - **Detect** root causes of I/O bottlenecks
  - **Map** I/O bottlenecks into actionable items
  - **Guide** end-user to tune I/O performance
- Towards a **cross-layer** I/O profiling exploration

Drishti


HPC Application
I/O Metrics
Darshan / DXT
pyDarshan
Recorder
Traces
scorep / TAU
Traces
I/O Analysis
Behavior and I/O Phases
Interactive Plots
Plotly
Insights
Recommendations
Operation
Transfer Size
Spatiality
I/O Phases
Storage System
HTML
Drishti Output

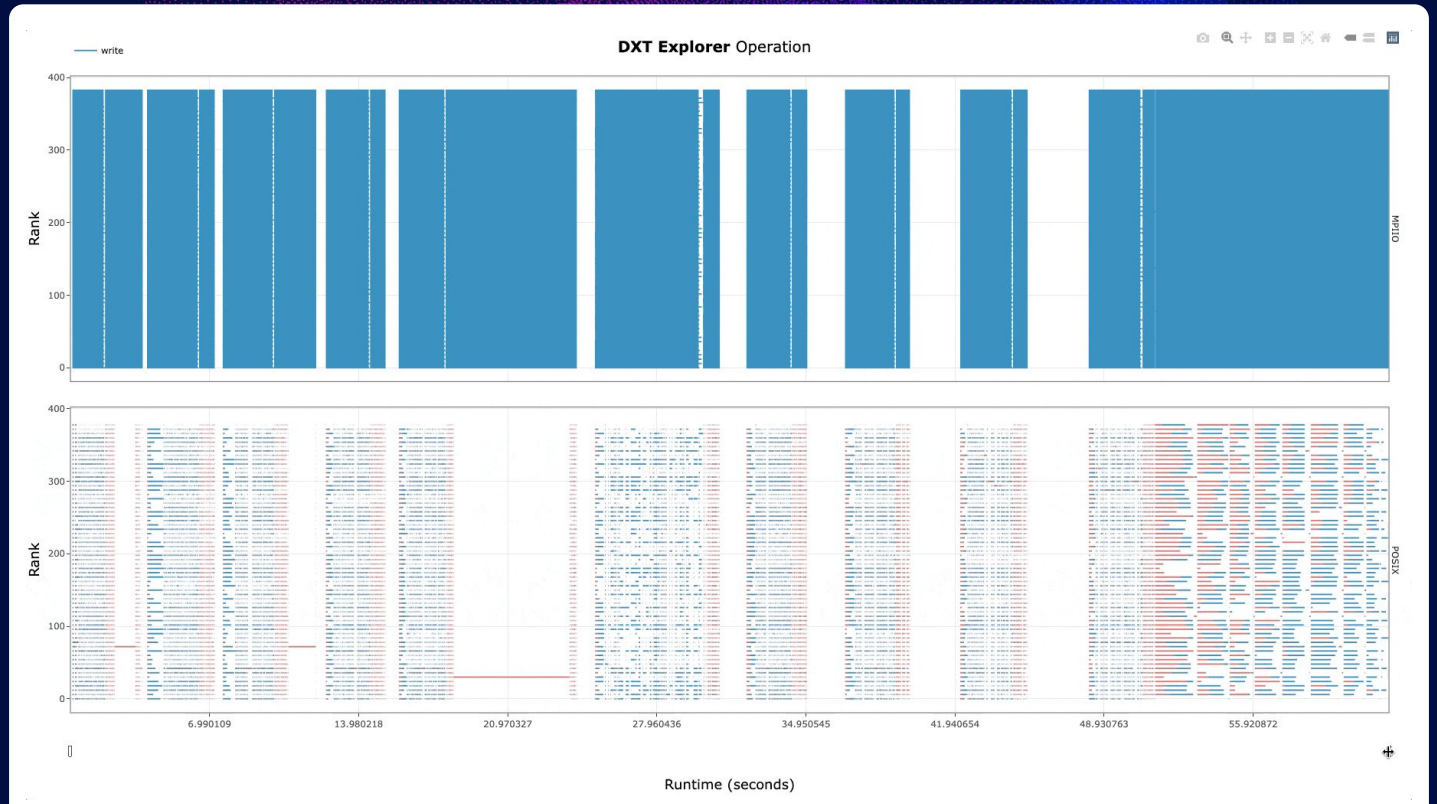
DARSHAN DXT

- Darshan collects **I/O profiling**
- It **aggregates** information
- **Extended tracing** mode (**DXT**)
  - Fine grain view of I/O behavior
  - POSIX or MPI-IO, read/write
  - Rank, segment, offset, size
  - Start and end timestamp

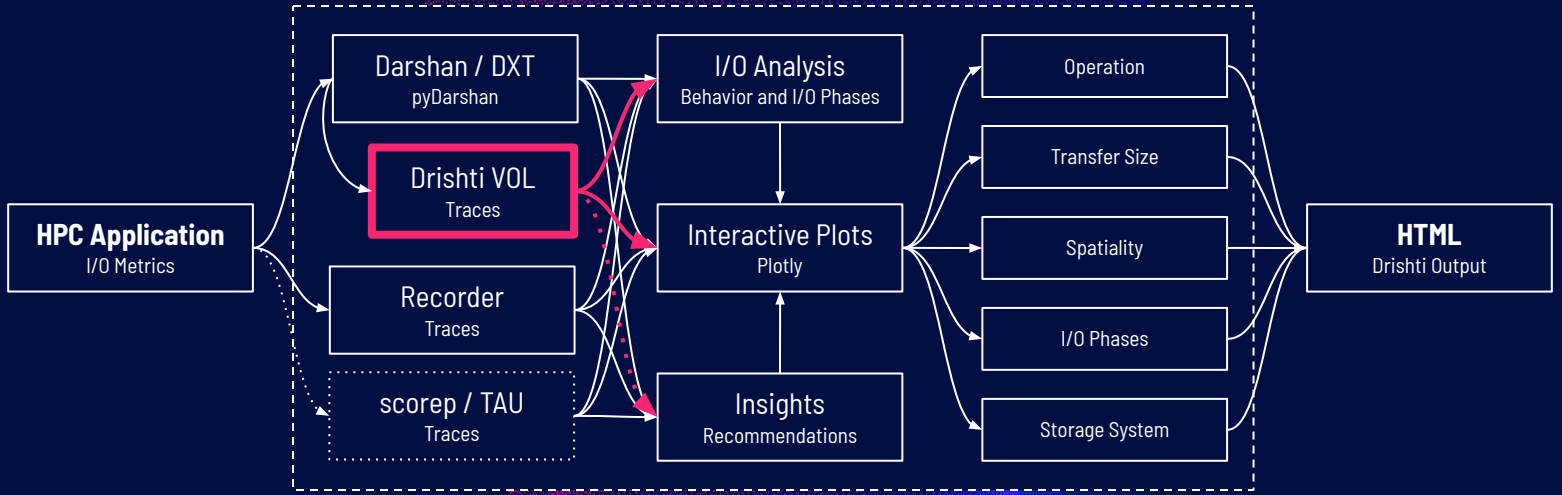
```
# ***************************************************
# DXT_POSIX module data
# ***************************************************

# DXT, file_id: 13771918696892050919, file_name:
/gpfs/alpine/csc300/scratch/houjun/Flash-X-apr8.gcc/FLASH_IO_hdf5_1.10.6/2366525/flash.par
# DXT, rank: 0, hostname: d11n01
# DXT, write_count: 0, read_count: 3
# DXT, mnt_pt: /gpfs/alpine, fs_type: gpfs
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)
 X_POSIX       0   read        0               0            783      0.0110      0.0110
 X_POSIX       0   read        1             783              0      0.0111      0.0111
 X_POSIX       0   read        2             783              0      0.0111      0.0111

# DXT, file_id: 17855743881390289785, file_name:
/gpfs/alpine/csc300/scratch/houjun/Flash-X-apr8.gcc/FLASH_IO_hdf5_1.10.6/2366525/flash.log
# DXT, rank: 0, hostname: d11n01
# DXT, write_count: 62, read_count: 0
# DXT, mnt_pt: /gpfs/alpine, fs_type: gpfs
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)
 X_POSIX       0  write        0               0            4105      0.0518      0.0527
 X_POSIX       0  write        1            4105            4141      0.0530      0.0530
 X_POSIX       0  write        2            8246            4127      0.0532      0.0532
 X_POSIX       0  write        3           12373            4097      0.0534      0.0547
...
```


write
DXT Explorer Operation
Rank
400
300
200
100
0
MPIIO
POSIX
6.990109
13.980218
20.970327
27.960436
34.950545
41.940654
48.930763
55.920872
Runtime (seconds)

Drishti


HPC Application
I/O Metrics
Darshan / DXT
pyDarshan
Drishti VOL
Traces
Recorder
Traces
scorep / TAU
Traces
I/O Analysis
Behavior and I/O Phases
Interactive Plots
Plotly
Insights
Recommendations
Operation
Transfer Size
Spatiality
I/O Phases
Storage System
HTML
Drishti Output

WHAT ABOUT HDF5?

- Focus on:
  - The aspects upon which the user might have control
  - Dynamic user metadata defined by the attributes API
  - Operations that go through the VOL
- Ensured **timestamps** (microseconds) from Darshan DXT and VOL will match
- Implemented as a **passthrough** VOL connector
- Designed for use in combination with **Darshan DXT**
- VOL traces stored in **memory** and persisted in a **file-per-process** at the end

github.com/hpc-io/vol-drishti

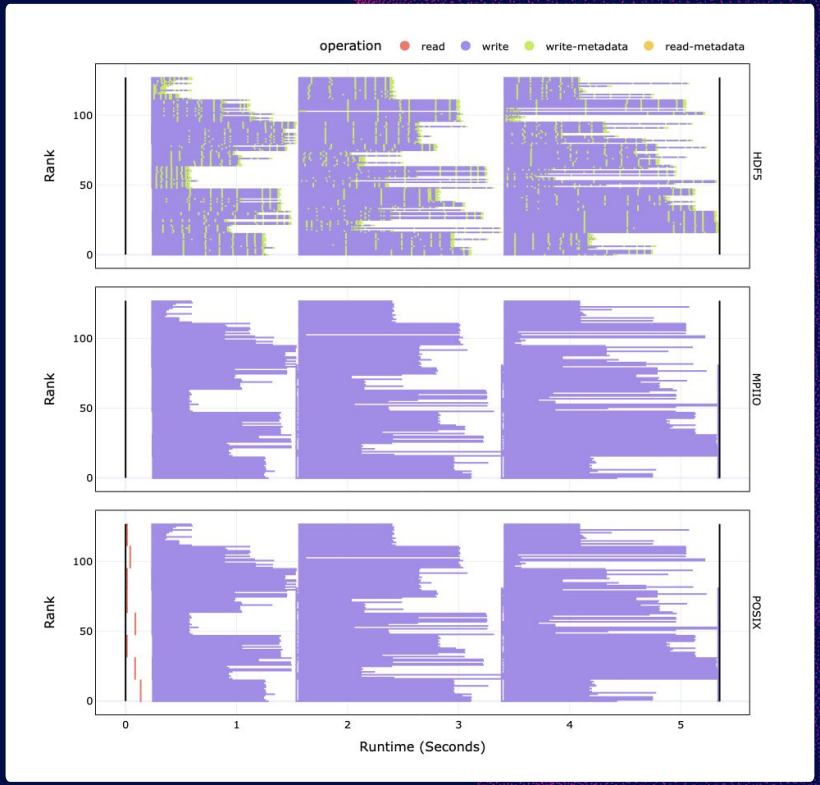
DRISHTI VOL

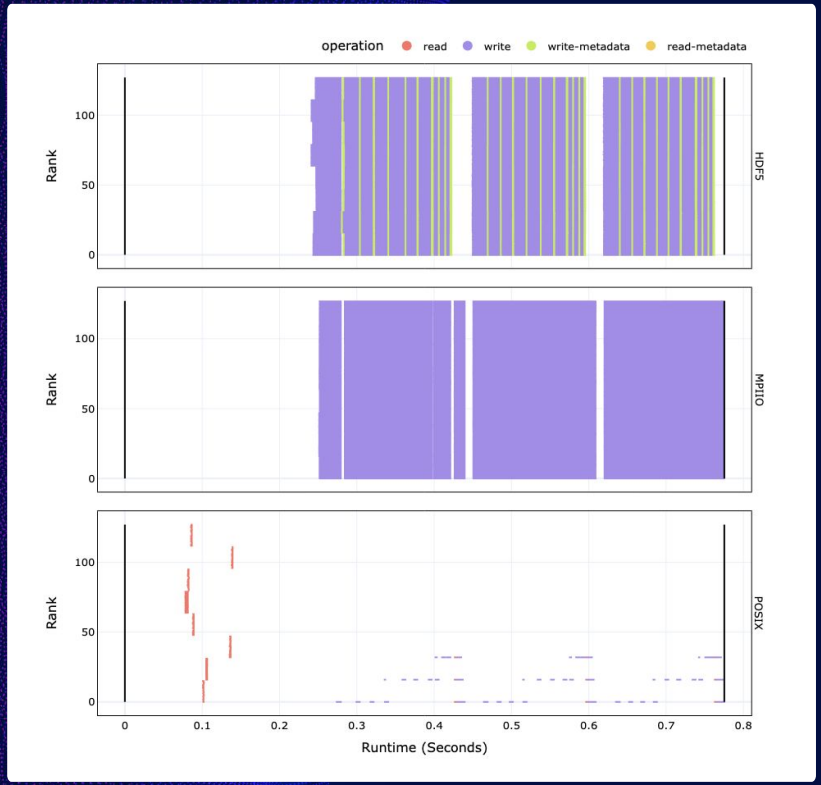
	Operation	File Operations	Drishti-VOL
Datasets	H5Dcreate	✅*	⛔
	H5Dopen	⛔	⛔
	H5Dwrite	✅	✅
	H5Dread	✅	✅
	H5Dclose	⛔	⛔
Attributes	H5Acreate	⛔	⛔
	H5Aopen	⛔	⛔
	H5Awrite	✅	✅
	H5Aread	✅	✅
	H5Aclose	⛔	⛔

* H5Dcreate could result in I/O operations if file space allocation is set

CROSS LAYER EXPLORATION

HDF5 VOL CONNECTOR





OVERHEAD? YES, BUT...

- Our target is exploratory/debug **small-scale** runs to replicate the issue
  - Tracing is **expensive**!
- **Non-negligible** overhead of collecting data from multiple layers of the I/O stack

	Runtime (seconds)			Overhead (Min. %)	Combined Log/Trace
	Min.	Median	Max.		
Baseline	5.99	7.52	8.62	-	-
+ Darshan	6.59	8.03	8.57	+9.62	35.88 KB
+ DXT	6.76	7.53	8.51	+3.03	38.88 MB
+ VOL	7.09	8.73	11.19	+4.88	41.69 MB

CROSS LAYER EXPLORATION

SOURCE CODE

AMREX

DARSHAN | 3 critical issues | 2 warnings | 8 recommendations

- 57 files (2 use STDIO, 1 use POSIX, 10 use MPI-IO)
- Application is write operation intensive (99.98% writes vs. 0.02% reads)
- Application is write size intensive (100.00% write vs. 0.00% read)

- High number (491640) of small write requests (< 1MB)
  - 99.99% of all write requests
  - Observed in 10 files:
    - plt00007.h5 with 49164 (10%) small write requests
      - 1 rank made small write requests to "plt00007.h5"
        - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
        - /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
        - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
        - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
    - plt00004.h5 with 49164 (10%) small write requests:
      - 1 rank made small write requests to "plt00004.h5"
        - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
        - /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
        - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
        - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
  - Recommended action:
    - Consider buffering write operations into larger, contiguous ones
    - Since the application uses MPI-IO, consider using collective I/O calls to aggregate requests into larger, contiguous ones (e.g., MPI_File_write_all() or MPI_File_write_at_all())

```
SOLUTION EXAMPLE SNIPPET
MPI_File_open(MPI_COMM_WORLD, "out.txt", MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
MPI_File_write_all(fh, &buffer, size, MPI_CHAR, &s);
```

- Detected data transfer imbalance caused by stragglers
  - Observed in 10 shared file:
    - plt00007.h5 with a load imbalance of 100.00%
      - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
      - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
      - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
      - /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
    - plt00004.h5 with a load imbalance of 100.00%
      - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
      - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
      - /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
      - /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
  - Recommended action:

E3SM

- High number (10878) of small read requests (< 1MB)
  - 100% of all read requests
  - Observed in 1 files:
    - map_f_case_16p.h5 with 49164 (10%) small read requests
      - 1 rank made small write requests to "map_f_case_16p.h5"
        - /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
        - /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
        - /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
        - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
  - Recommended action:
    - Consider buffering read operations into larger, contiguous ones
    - Since the application uses MPI-IO, consider using collective I/O calls to aggregate requests into larger, contiguous ones (e.g., MPI_File_write_all() or MPI_File_write_at_all())
- High number (4122) of random read operations (< 1MB)
  - 37.89% of all read requests
  - Observed in 1 files:
    - Below is the backtrace for these calls
      - 1 rank made small write requests to "map_f_case_16p.h5"
        - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
        - /h5bench/e3sm/src/cases/var_wr_case.cpp: 448
        - /h5bench/e3sm/src/e3sm_io_core.cpp: 97
        - /h5bench/e3sm/src/e3sm_io.c: 563
        - /h5bench/e3sm/src/drivers/e3sm_io_driver_h5blob.cpp: 254
        - /h5bench/e3sm/src/cases/e3sm_io_case.cpp: 136
  - Recommended action:
    - Consider changing your data model to have consecutive or sequential reads
- Application uses MPI-IO and issues 10877 (100.00%) independent read calls
  - 10877 (100.0%) of independent reads in "map_f_case_16p.h5"
  - Observed in 1 files:
    - Below is the backtrace for these calls
      - /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
      - /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
      - /h5bench/e3sm/src/drivers/e3sm_io_driver_hdf5.cpp: 552
      - /h5bench/e3sm/src/read_decomp.cpp: 253
  - Recommended action:
    - Consider using collective read operations and set one aggregator per compute node (e.g. MPI_File_read_all() or MPI_File_read_at_all())

CROSS LAYER EXPLORATION

SOURCE CODE

AMREX

```
DARSHAN | 3 critical issues | 2 warnings | 8 recommendations

▶ 57 files (2 use STDIO, 1 use POSIX, 10 use MPI-IO)
▶ Application is write operation intensive (99.98% writes vs. 0.02% reads)
▶ Application is write size intensive (100.00% write vs. 0.00% read)

▶ High number (491640) of small write requests (< 1MB)
    ▶ 99.99% of all write requests
    ▶ Observed in 10 files:
        ▶ plt00007.h5 with 49164 (10%) small write requests
            ▶ 1 rank made small write requests to "plt00007.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
                ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
        ▶ plt00004.h5 with 49164 (10%) small write requests:
            ▶ 1 rank made small write requests to "plt00004.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
                ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
    ▶ Recommended action:
        ▶ Consider buffering write operations into larger, contiguous ones
        ▶ Since the application uses MPI-IO, consider using collective I/O calls
          to aggregate requests into larger, contiguous ones
          (e.g., MPI_File_write_all() or MPI_File_write_at_all())

          SOLUTION EXAMPLE SNIPPET

          MPI_File_open(MPI_COMM_WORLD, "out.txt", MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
          MPI_File_write_all(fh, &buffer, size, MPI_CHAR, &s);

▶ Detected data transfer imbalance caused by stragglers
    ▶ Observed in 10 shared file:
        ▶ plt00007.h5 with a load imbalance of 100.00%
            ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
            ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
        ▶ plt00004.h5 with a load imbalance of 100.00%
            ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
            ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
    ▶ Recommended action:
```

E3SM

```
▶ High number (10878) of small read requests (< 1MB)
    ▶ 100% of all read requests
    ▶ Observed in 1 files:
        ▶ map_f_case_16p.h5 with 49164 (10%) small read requests
            ▶ 1 rank made small write requests to "map_f_case_16p.h5"
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
                ▶ /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
    ▶ Recommended action:
        ▶ Consider buffering read operations into larger, contiguous ones
        ▶ Since the application uses MPI-IO, consider using collective I/O calls
          to aggregate requests into larger, contiguous ones
          (e.g., MPI_File_write_all() or MPI_File_write_at_all())
▶ High number (4122) of random read operations (< 1MB)
    ▶ 37.89% of all read requests
    ▶ Observed in 1 files:
        ▶ Below is the backtrace for these calls
            ▶ 1 rank made small write requests to "map_f_case_16p.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
                ▶ /h5bench/e3sm/src/cases/var_wr_case.cpp: 448
                ▶ /h5bench/e3sm/src/e3sm_io_core.cpp: 97
                ▶ /h5bench/e3sm/src/e3sm_io.c: 563
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver_h5blob.cpp: 254
                ▶ /h5bench/e3sm/src/cases/e3sm_io_case.cpp: 136
    ▶ Recommended action:
        ▶ Consider changing your data model to have consecutive or sequential reads
▶ Application uses MPI-IO and issues 10877 (100.00%) independent read calls
    ▶ 10877 (100.0%) of independent reads in "map_f_case_16p.h5"
    ▶ Observed in 1 files:
        ▶ Below is the backtrace for these calls
            ▶ /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
            ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
            ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver_hdf5.cpp: 552
            ▶ /h5bench/e3sm/src/read_decomp.cpp: 253
    ▶ Recommended action:
        ▶ Consider using collective read operations and set one aggregator per compute node
          (e.g. MPI_File_read_all() or MPI_File_read_at_all())
```

CROSS LAYER EXPLORATION

SOURCE CODE

AMREX

```
DARSHAN | 3 critical issues | 2 warnings | 8 recommendations

▶ 57 files (2 use STDIO, 1 use POSIX, 10 use MPI-IO)
▶ Application is write operation intensive (99.98% writes vs. 0.02% reads)
▶ Application is write size intensive (100.00% write vs. 0.00% read)

▶ High number (491640) of small write requests (< 1MB)
    ▶ 99.99% of all write requests
    ▶ Observed in 10 files:
        ▶ plt00007.h5 with 49164 (10%) small write requests
            ▶ 1 rank made small write requests to "plt00007.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
                ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
        ▶ plt00004.h5 with 49164 (10%) small write requests:
            ▶ 1 rank made small write requests to "plt00004.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S:122
                ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp:380
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
                ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
    ▶ Recommended action:
        ▶ Consider buffering write operations into larger, contiguous ones
        ▶ Since the application uses MPI-IO, consider using collective I/O calls
          to aggregate requests into larger, contiguous ones
          (e.g., MPI_File_write_all() or MPI_File_write_at_all())

        SOLUTION EXAMPLE SNIPPET
        MPI_File_open(MPI_COMM_WORLD, "out.txt", MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
        MPI_File_write_all(fh, &buffer, size, MPI_CHAR, &s);

▶ Detected data transfer imbalance caused by stragglers
    ▶ Observed in 10 shared file:
        ▶ plt00007.h5 with a load imbalance of 100.00%
            ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
            ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
        ▶ plt00004.h5 with a load imbalance of 100.00%
            ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 134
            ▶ /h5bench/amrex/Tests/HDF5Benchmark/main.cpp: 24
            ▶ /h5bench/amrex/Src/Extern/HDF5/AMReX_PlotFileUtilHDF5.cpp: 516
    ▶ Recommended action:
```

E3SM

```
▶ High number (10878) of small read requests (< 1MB)
    ▶ 100% of all read requests
    ▶ Observed in 1 files:
        ▶ map_f_case_16p.h5 with 49164 (10%) small read requests
            ▶ 1 rank made small write requests to "map_f_case_16p.h5"
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver.cpp: 120
                ▶ /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
    ▶ Recommended action:
        ▶ Consider buffering read operations into larger, contiguous ones
        ▶ Since the application uses MPI-IO, consider using collective I/O calls
          to aggregate requests into larger, contiguous ones
          (e.g., MPI_File_write_all() or MPI_File_write_at_all())
▶ High number (4122) of random read operations (< 1MB)
    ▶ 37.89% of all read requests
    ▶ Observed in 1 files:
        ▶ Below is the backtrace for these calls
            ▶ 1 rank made small write requests to "map_f_case_16p.h5"
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
                ▶ /h5bench/e3sm/src/cases/var_wr_case.cpp: 448
                ▶ /h5bench/e3sm/src/e3sm_io_core.cpp: 97
                ▶ /h5bench/e3sm/src/e3sm_io.c: 563
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver_h5blob.cpp: 254
                ▶ /h5bench/e3sm/src/cases/e3sm_io_case.cpp: 136
    ▶ Recommended action:
        ▶ Consider changing your data model to have consecutive or sequential reads
▶ Application uses MPI-IO and issues 10877 (100.00%) independent read calls
    ▶ 10877 (100.0%) of independent reads in "map_f_case_16p.h5"
    ▶ Observed in 1 files:
        ▶ Below is the backtrace for these calls
                ▶ /h5bench/e3sm/src/e3sm_io.c: 539 (discriminator 5)
                ▶ /home/abuild/rpmbuild/BUILD/glibc-2.31/csu/../sysdeps/x86_64/start.S: 122
                ▶ /h5bench/e3sm/src/drivers/e3sm_io_driver_hdf5.cpp: 552
                ▶ /h5bench/e3sm/src/read_decomp.cpp: 253
    ▶ Recommended action:
        ▶ Consider using collective read operations and set one aggregator per compute node
          (e.g. MPI_File_read_all() or MPI_File_read_at_all())
```

KEY TAKEAWAYS

- Drishti:
  - **Detects** root causes based on metrics
  - **Maps** them to I/O bottlenecks, and
  - **Recommends** actionable items to tune I/O performance
- Drishti uses > 30 triggers to check how an application is accessing data
- Drishti can ingest **multiple sources** of I/O metrics (e.g., Darshan, DXT, Recorder, scorep)
- Drishti VOL connector enhances the reports with **HDF5**-related metrics

github.com/hpc-io/vol-drishti

Drishti

github.com/hpc-io/drishti

JEAN LUCA BEZ <jlbez@lbl.gov>, SUREN BYNA <byna.1@osu.edu>
HAMMAD ATHER, YANKUN XIA, JOEL TONY