INTERACTIVE I/O LOG ANALYSIS

JEAN LUCA BEZ, SUREN BYNA
<jlbez@lbl.gov>

DXT EXPLORER

- Darshan is a popular tool to collect **I/O profiling**
- It **aggregates** information to provide insights
- **Extended tracing** mode (DXT)

```
export DXT_ENABLE_IO_TRACE=1
```

  - Fine grain view of the I/O behavior
  - POSIX or MPI-IO, read/write
  - Rank, segment, offset, request size
  - Start and end timestamp

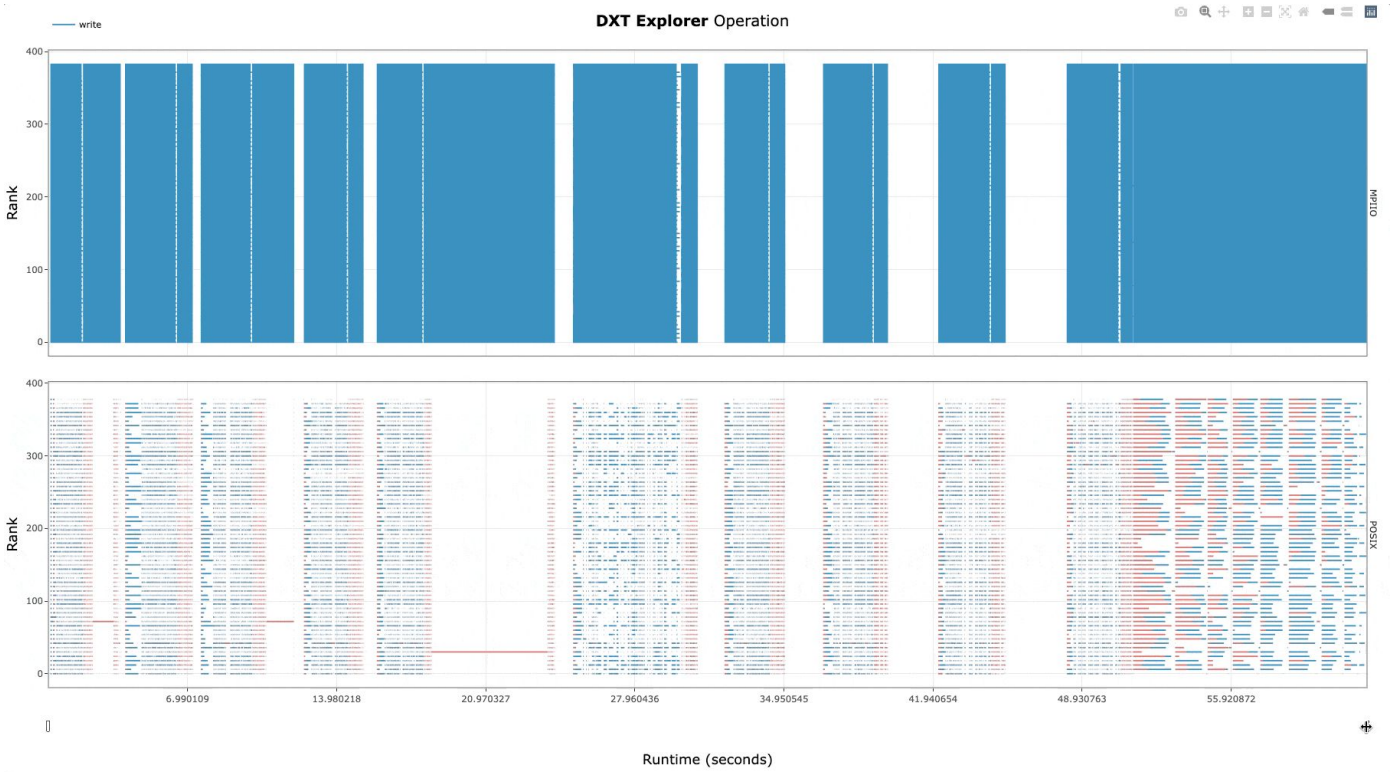
```
# ************************************************
# DXT_POSIX module data
# ************************************************

# 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
...
```

DXT EXPLORER

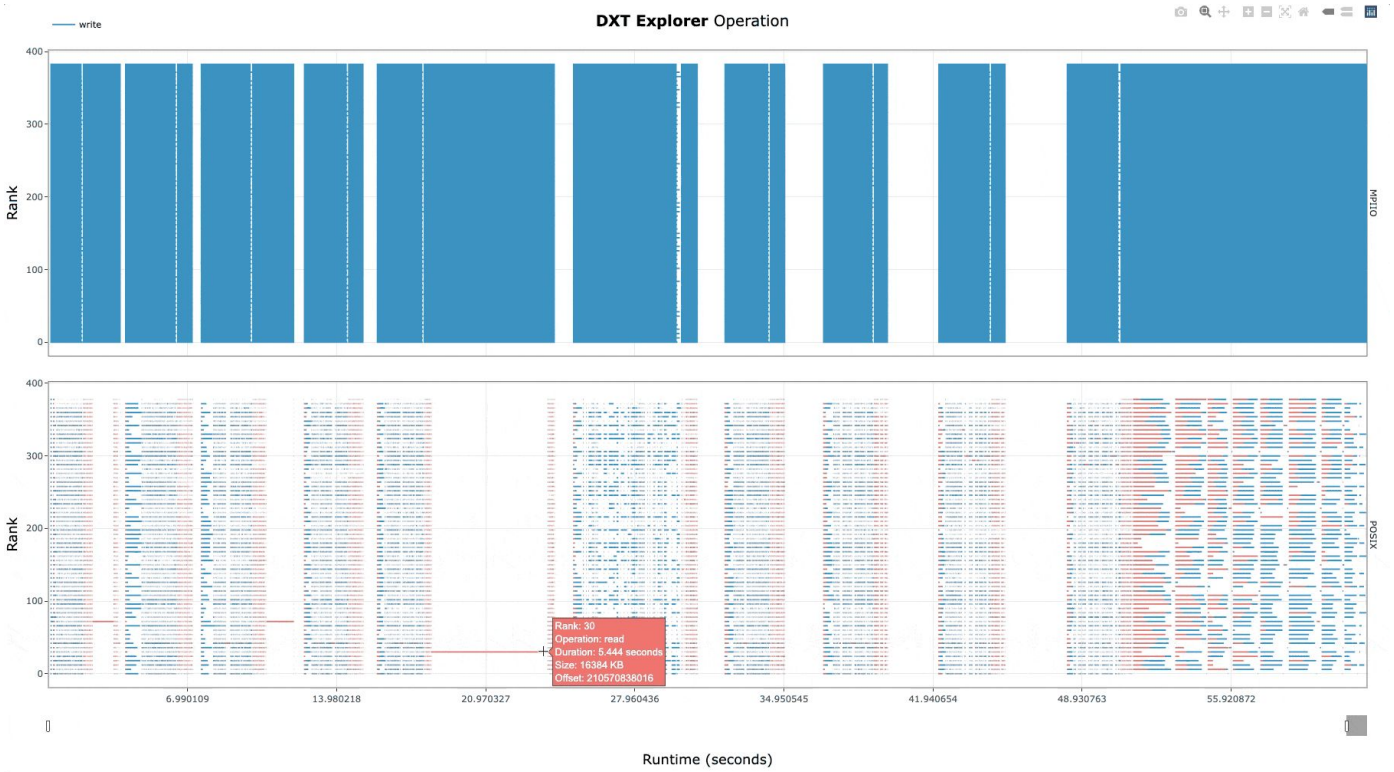
- No tool to visualize and explore yet
- Static plots have **limitations**
- **Features** we seek:
  - Observe POSIX and MPI-IO together
  - Zoom-in/zoom-out in time and subset of ranks
  - Contextual information about I/O calls
  - Focus on operation, size, or spatiality
- By visualizing the application behavior, we are **one step closer** to optimize the application

github.com/hpc-io/dxt-explorer

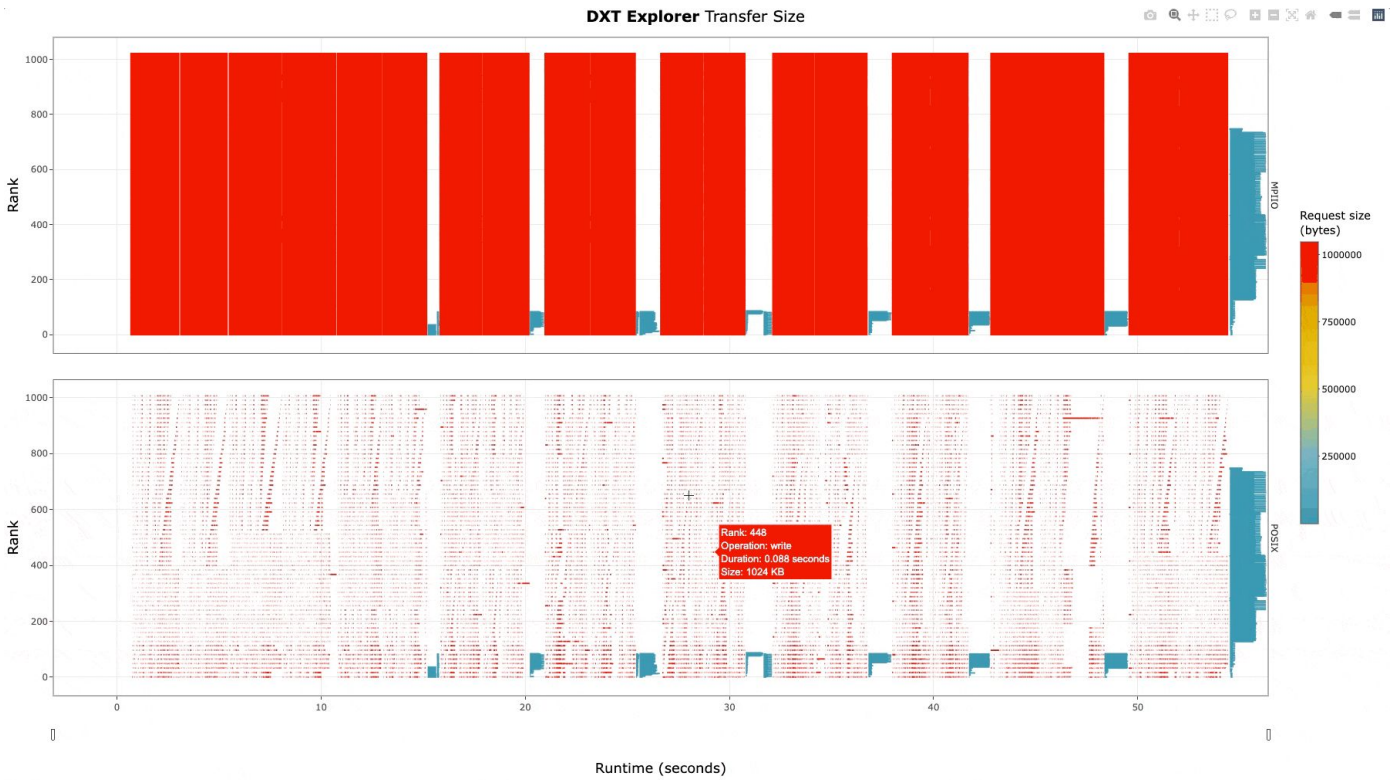
docker pull hpcio/dxt-explorer



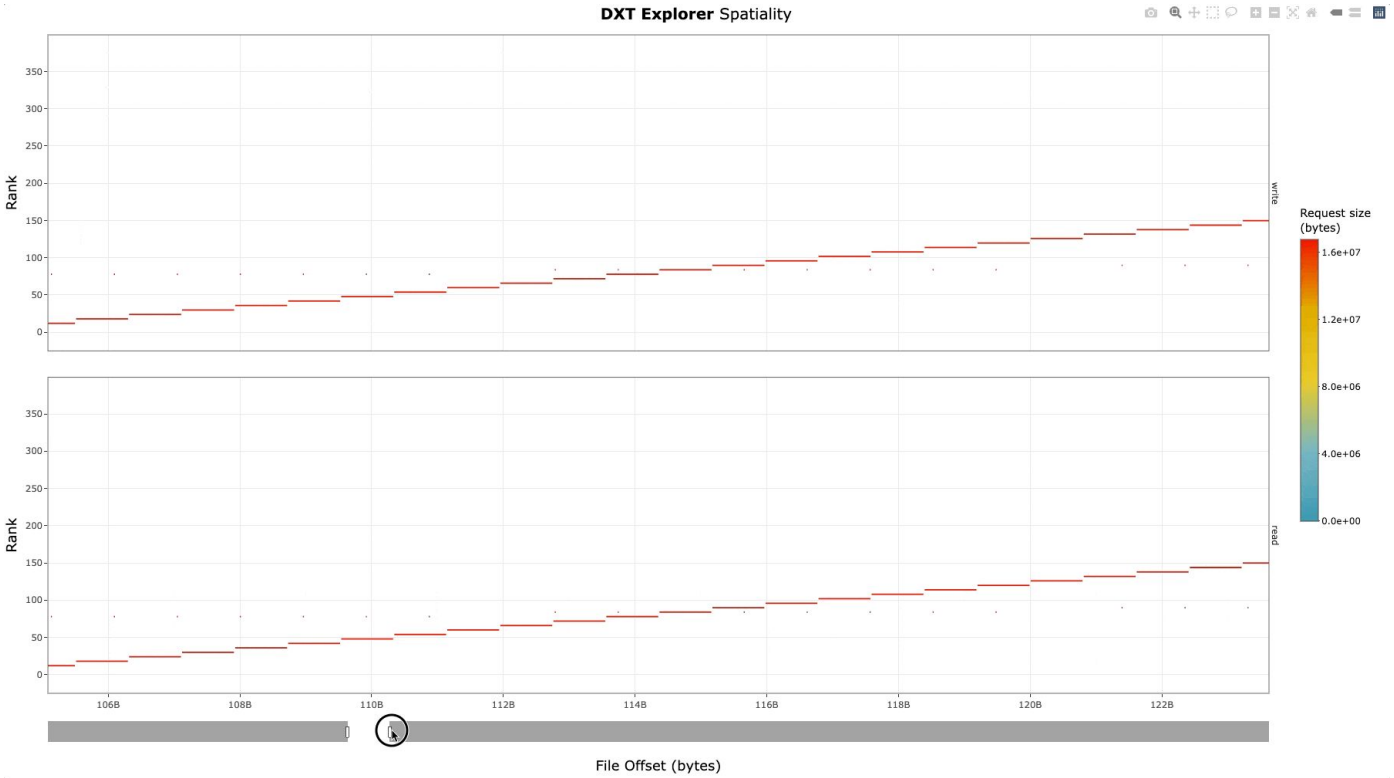
Explore the timeline by **zooming in and out** and observing how the **MPI-IO** calls are translated to the **POSIX** layer. For instance, you can use this feature to detect stragglers.



Visualize relevant information in the **context** of **each I/O call**
(rank, operation, duration, request size, and OSTs if Lustre) by hovering over a given operation.



Explore the **operations by size** in POSIX and MPI-IO. You can, for instance, identify small or metadata operations from this visualization.



Explore the **spatiality** of accesses in file by each rank with **contextual** information.
Understand how each rank is accessing each file.


```
dxt-explorer [-h] [-t] [-s] [-d] [-l] [--start START] [--end END] [--from START_RANK] [--to END_RANK] [-v] darshan

DXT Explorer:
positional arguments:
  darshan               Input .darshan file
optional arguments:
  -h, --help            show this help message and exit
  -t, --transfer        Generate an interactive data transfer explorer
  -s, --spatiality      Generate an interactive spatiality explorer
  -d, --debug           Enable debug mode
  -l, --list            List all the files with trace
  --start START         Report starts from X seconds (e.g., 3.7) from beginning of the job
  --end END             Report ends at X seconds (e.g., 3.9) from beginning of the job
  --from START_RANK     Report start from rank N
  --to END_RANK         Report up to rank M
  -v, --version         show program's version number and exit
```

INSTALL

```
# Install DXT Explorer on your local machine
$ pip install dxt-explorer


# On NERSC systems you can also use the container version with Shifter
$ shifter --image=docker:hpcio/dxt-explorer
```

HANDS-ON

```
# Download some files for the hands-on exercise
$ wget https://github.com/jeanbez/dxt-sample-logs/raw/main/samples-openpmd.tar.gz
$ tar zxvf samples-openpmd.tar.gz

# Run dxt-explorer with the provided .darshan DXT traces
$ dxt-explorer --debug samples/REPLACE_WITH_FILE_NAME.darshan
# On NERSC systems you can also use the container version with Shifter
$ shifter --image=docker:hpcio/dxt-explorer

# Download the files for local interactive exploration on your browser!
```

I/O INSIGHTS FOR ALL

JEAN LUCA BEZ, SUREN BYNA
<jlbez@lbl.gov>

Drishti

- There is still a **gap** between **profiling** and **tuning**
- Drishti: from I/O profiles to **meaningful** information
  - **Detect** root causes of I/O bottlenecks
  - **Map** I/O bottlenecks into actionable items
  - **Guide** end-user to tune their application's I/O performance
- 4 levels of triggers
- > 30 triggers are checked for each .darshan log

github.com/hpc-io/drishti

docker pull hpcio/drishti

Drishti

Level	Description
HIGH	High probability of harming I/O performance.
WARN	Detected issues that could cause a significant negative impact on the I/O performance. The confidence of these recommendations is low as available metrics might not be sufficient to detect application design, configuration, or execution choices.
OK	Best practices have been followed.
INFO	Relevant information regarding application configuration.

Drishti

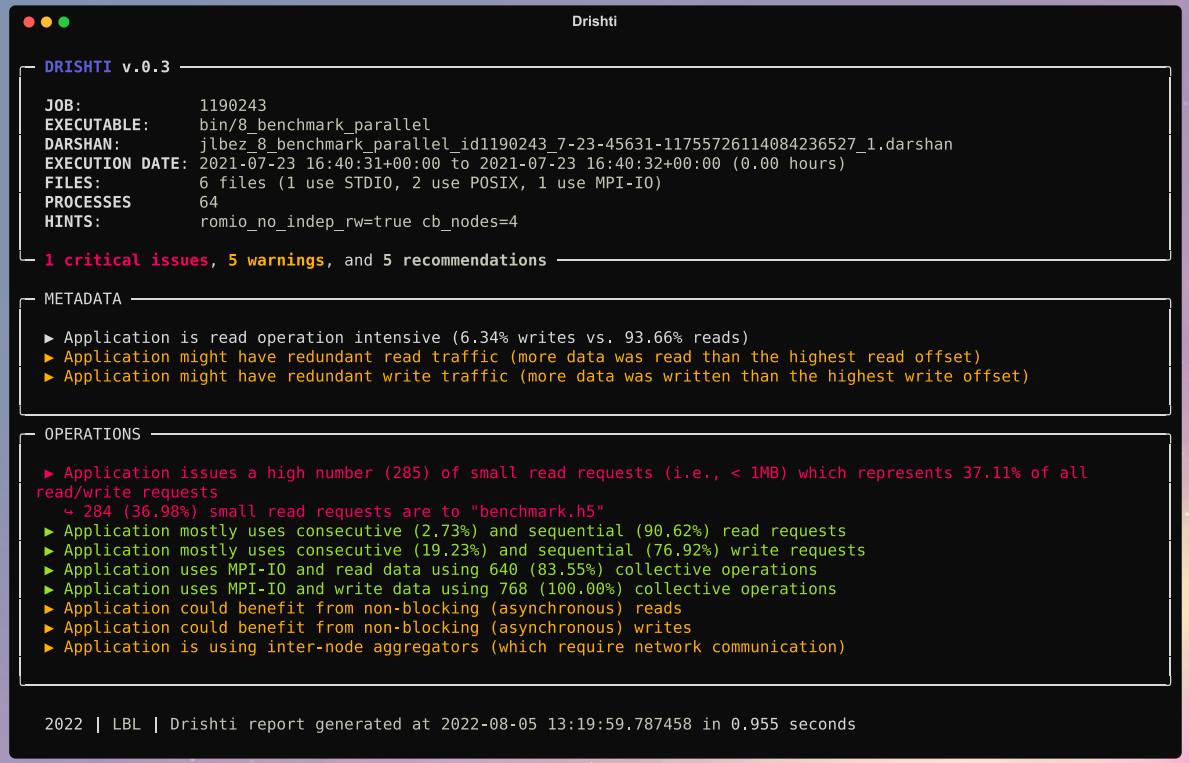
```
usage: drishti [-h] [--issues] [--html] [--svg] [--verbose] [--code] darshan

Drishti:

positional arguments:
  darshan     Input .darshan file

optional arguments:
  -h, --help  show this help message and exit
  --issues    Only displays the detected issues and hides the recommendations
  --html      Export the report as an HTML page
  --svg       Export the report as an SVG image
  --verbose   Display extended details for the recommendations
  --code      Display insights identification code
```

Overall information about the Darshan log and execution

Number of critical issues, warning, and recommendations

Drishti checks metrics for **over 30 triggers**

Highlight the **file** that triggered the issue



Current version only checks **profiling** metrics

Severity based on certainty and impact: **high**, **medium**, **low**, info

Multiple output formats: textual, SVG, HTML

Provides **actionable** feedback for users

Drishti

```
DRISHTI v.0.3

  JOB:            1190243
  EXECUTABLE:     bin/8_benchmark_parallel
  DARSHAN:        jlbez_8_benchmark_parallel_id1190243_7-23-45631-11755726114084236527_1.darshan
  EXECUTION DATE: 2021-07-23 16:40:31+00:00 to 2021-07-23 16:40:32+00:00 (0.00 hours)
  FILES:          6 files (1 use STDIO, 2 use POSIX, 1 use MPI-IO)
  PROCESSES       64
  HINTS:          romio_no_indep_rw=true cb_nodes=4

1 critical issues, 5 warnings, and 5 recommendations

METADATA

  ▶ Application is read operation intensive (6.34% writes vs. 93.66% reads)
  ▶ Application might have redundant read traffic (more data was read than the highest read offset)
  ▶ Application might have redundant write traffic (more data was written than the highest write offset)

OPERATIONS

  ▶ Application issues a high number (285) of small read requests (i.e., < 1MB) which represents 37.11% of all
read/write requests
    ↪ 284 (36.98%) small read requests are to "benchmark.h5"
    ↪ Recommendations:
      ↪ Consider buffering read operations into larger more contiguous ones
      ↪ Since the appplication already uses MPI-IO, consider using collective I/O calls (e.g. MPI_File_read_all() or
MPI_File_read_at_all()) to aggregate requests into larger ones
  ▶ Application mostly uses consecutive (2.73%) and sequential (90.62%) read requests
  ▶ Application mostly uses consecutive (19.23%) and sequential (76.92%) write requests
  ▶ Application uses MPI-IO and read data using 640 (83.55%) collective operations
  ▶ Application uses MPI-IO and write data using 768 (100.00%) collective operations
  ▶ Application could benefit from non-blocking (asynchronous) reads
    ↪ Recommendations:
      ↪ Since you use MPI-IO, consider non-blocking/asynchronous I/O operations (e.g., MPI_File_iread(),
MPI_File_read_all_begin/end(), or MPI_File_read_at_all_begin/end())
  ▶ Application could benefit from non-blocking (asynchronous) writes
    ↪ Recommendations:
      ↪ Since you use MPI-IO, consider non-blocking/asynchronous I/O operations (e.g., MPI_File_iwrite(),
MPI_File_write_all_begin/end(), or MPI_File_write_at_all_begin/end())
  ▶ Application is using inter-node aggregators (which require network communication)
    ↪ Recommendations:
      ↪ Set the MPI hints for the number of aggregators as one per compute node (e.g., cb_nodes=32)

2022 | LBL | Drishti report generated at 2022-08-05 13:20:19.715639 in 0.996 seconds
```

Drishti can check for HDF5 usage to **fine tune** the recommendations

Sample **code solutions** are provided

OPERATIONS

▶ Application issues a high number (285) of small read requests (i.e., < 1MB) which represents 37.11% of all read/write requests
↪ 284 (36.98%) small read requests are to "benchmark.h5"
↪ **Recommendations:**
↪ Consider buffering read operations into larger more contiguous ones
↪ Since the appplication already uses MPI-IO, consider using collective I/O calls (e.g. MPI_File_read_all() or MPI_File_read_at_all()) to aggregate requests into larger ones

Solution Example Snippet

```
1 MPI_File_open(MPI_COMM_WORLD, "output-example.txt", MPI_MODE_CREATE|MPI_MODE_RDONLY, MPI_INFO_NULL,
2 ...
3 MPI_File_read_all(fh, &buffer, size, MPI_INT, &s);
```

▶ Application mostly uses consecutive (2.73%) and sequential (90.62%) read requests
▶ Application mostly uses consecutive (19.23%) and sequential (76.92%) write requests
▶ Application uses MPI-IO and read data using 640 (83.55%) collective operations
▶ Application uses MPI-IO and write data using 768 (100.00%) collective operations
▶ Application could benefit from non-blocking (asynchronous) reads
↪ **Recommendations:**
↪ Since you use MPI-IO, consider non-blocking/asynchronous I/O operations (e.g., MPI_File_iread(), MPI_File_read_all_begin/end(), or MPI_File_read_at_all_begin/end())

Solution Example Snippet

```
1  MPI_File fh;
2  MPI_Status s;
3  MPI_Request r;
4  ...
5  MPI_File_open(MPI_COMM_WORLD, "output-example.txt", MPI_MODE_CREATE|MPI_MODE_RDONLY, MPI_INFO_NULL
6  ...
7  MPI_File_iread(fh, &buffer, BUFFER_SIZE, n, MPI_CHAR, &r);
8  ...
9  // compute something
10 ...
11 MPI_Test(&r, &completed, &s);
12 ...
13 if (!completed) {
14     // compute something
15
16     MPI_Wait(&r, &s);
17 }
```

▶ Application could benefit from non-blocking (asynchronous) writes
↪ **Recommendations:**
↪ Since you use MPI-IO, consider non-blocking/asynchronous I/O operations (e.g., MPI_File_iwrite(), MPI_File_write_all_begin/end(), or MPI_File_write_at_all_begin/end())

Solution Example Snippet

```
10 ...
11 MPI_Test(&r, &completed, &s);
12 ...
13 if (!completed) {
14     // compute something
15
16     MPI_Wait(&r, &s);
17 }
```

▶ Application is using inter-node aggregators (which require network communication)
  ↳ **Recommendations:**
    ↳ Set the MPI hints for the number of aggregators as one per compute node (e.g., cb_nodes=32)

Solution Example Snippet

```
 1 # ------------------------------ #
 2 # MPICH                         #
 3 # ------------------------------ #
 4 export MPICH_MPIIO_HINTS="*:cb_nodes=16:cb_buffer_size=16777216:romio_cb_write=enable:romio_ds_wri
 5
 6 # * means it will apply the hints to any file opened with MPI-IO
 7 # cb_nodes        ---> number of aggregator nodes, defaults to stripe count
 8 # cb_buffer_size  ---> controls the buffer size used for collective buffering
 9 # romio_cb_write  ---> controls collective buffering for writes
10 # romio_cb_read   ---> controls collective buffering for reads
11 # romio_ds_write  ---> controls data sieving for writes
12 # romio_ds_read   ---> controls data sieving for reads
13
14 # to visualize the used hints for a given job
15 export MPICH_MPIIO_HINTS_DISPLAY=1
16
17 # ------------------------------ #
18 # OpenMPI / SpectrumMPI (Summit)  #
19 # ------------------------------ #
20 export OMPI_MCA_io=romio321
21 export ROMIO_HINTS=./my-romio-hints
22
23 # the my-romio-hints file content is as follows:
24 cat $ROMIO_HINTS
25
26 romio_cb_write enable
27 romio_cb_read  enable
28 romio_ds_write disable
29 romio_ds_read  disable
30 cb_buffer_size 16777216
31 cb_nodes 8
```

Sample **configurations** are provided

Level	Interface	Detected Behavior	Jobs	Total (%)	Relative* (%)
HIGH	STDIO	High STDIO usage (>10% of total transfer size uses STDIO)	43,120	38.29	52.1
OK	POSIX	High number of sequential read operations (≥ 80%)	38,104	33.84	58.14
OK	POSIX	High number of sequential write operations (≥ 80%)	64,486	57.26	98.39
INFO	POSIX	Write operation count intensive (>10% more writes than reads)	26,114	23.19	39.84
INFO	POSIX	Read operation count intensive (>10% more reads than writes)	23,168	20.57	35.35
INFO	POSIX	Write size intensive (>10% more bytes written then read)	23,568	20.93	35.96
INFO	POSIX	Read size intensive (>10% more bytes read then written)	40,950	36.36	62.48
WARN	POSIX	Redundant reads	14,518	12.89	22.15
WARN	POSIX	Redundant writes	59	0.05	0.09
HIGH	POSIX	High number of small (<1MB) read requests (>10% of total read requests)	64,858	57.59	98.96
HIGH	POSIX	High number of small (<1MB) write requests (>10% of total write requests)	64,552	57.32	98.49
HIGH	POSIX	High number of misaligned memory requests (>10%)	36,337	32.27	55.44
HIGH	POSIX	High number of misaligned file requests (>10%)	65,075	57.79	99.29
HIGH	POSIX	High number of random read requests (>20%)	26,574	23.6	40.54
HIGH	POSIX	High number of random write requests (>20%)	559	0.5	0.85
HIGH	POSIX	High number of small (<1MB) reads to shared-files (>10% of total reads)	60,121	53.39	91.73
HIGH	POSIX	High number of small (<1MB) writes to shared-files (>10% of total writes)	55,414	49.21	84.55
HIGH	POSIX	High metadata time (at least one rank spends >30 seconds)	9,410	8.36	14.35
HIGH	POSIX	Data transfer imbalance between ranks causing stragglers (>15% difference)	40,601	36.05	61.95
HIGH	POSIX	Time imbalance between ranks causing stragglers (>15% difference)	40,533	35.99	61.84

Level	Interface	Detected Behavior	Jobs	Total (%)	Relative* (%)
WARN	MPI-IO	No MPI-IO calls detected from Darshan logs	109,569	97.3 -	
HIGH	MPI-IO	Detected MPI-IO but no collective read operation	169	0.15	5.55
HIGH	MPI-IO	Detected MPI-IO but no collective write operation	428	0.38	14.06
WARN	MPI-IO	Detected MPI-IO but no non-blocking read operations	3,043	2.7	100
WARN	MPI-IO	Detected MPI-IO but no non-blocking write operations	3,043	2.7	100
OK	MPI-IO	Detected MPI-IO and collective read operations	402	0.36	13.21
OK	MPI-IO	Detected MPI-IO and collective write operations	2,592	2.3	85.17
HIGH	MPI-IO	Detected MPI-IO and inter-node aggregators	2,496	2.22	82.02
WARN	MPI-IO	Detected MPI-IO and intra-node aggregators	304	0.27	9.99
OK	MPI-IO	Detected MPI-IO and one aggregator per node	29	0.03	0.95

Drishti

HANDS-ON

```
# Install Drishti on your local machine
$ pip install drishti
# Run Drishti with the provided .darshan DXT traces
$ drishti --verbose samples/REPLACE_WITH_FILE_NAME.darshan


# On NERSC systems you can also use the container version with Shifter
$ shifter --image=docker:hpcio/drishti -- drishti samples/REPLACE_WITH_FILE_NAME.darshan
```