INTERACTIVE I/O LOG ANALYSIS

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Rank</th>
<th>Wt/Rd</th>
<th>Segment</th>
<th>Offset</th>
<th>Length</th>
<th>Start(s)</th>
<th>End(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>read</td>
<td>0</td>
<td>0</td>
<td>783</td>
<td>0.0110</td>
<td>0.0110</td>
</tr>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>read</td>
<td>1</td>
<td>783</td>
<td>0</td>
<td>0.0111</td>
<td>0.0111</td>
</tr>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>read</td>
<td>2</td>
<td>783</td>
<td>0</td>
<td>0.0111</td>
<td>0.0111</td>
</tr>
</tbody>
</table>

# DXT, file_id: 17855743881390289785, file_name: /gpfs/alpine/csc300/scratch/houjun/Flash-X-apr8.gcc/FLASH_IO_hdf5_1.10.6/2366525/flash.log

<table>
<thead>
<tr>
<th>Module</th>
<th>Rank</th>
<th>Wt/Rd</th>
<th>Segment</th>
<th>Offset</th>
<th>Length</th>
<th>Start(s)</th>
<th>End(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>write</td>
<td>0</td>
<td>0</td>
<td>4105</td>
<td>0.0518</td>
<td>0.0527</td>
</tr>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>write</td>
<td>1</td>
<td>4105</td>
<td>4141</td>
<td>0.0530</td>
<td>0.0530</td>
</tr>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>write</td>
<td>2</td>
<td>8246</td>
<td>4127</td>
<td>0.0532</td>
<td>0.0532</td>
</tr>
<tr>
<td>X_POSIX</td>
<td>0</td>
<td>write</td>
<td>3</td>
<td>12373</td>
<td>4097</td>
<td>0.0534</td>
<td>0.0547</td>
</tr>
</tbody>
</table>
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
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:
positionual 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 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
# Download some files for the hands-on exercise


$ 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>
● 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](https://github.com/hpc-io/drishti)

[docker pull hpcio/drishti](https://docker pull hpcio/drishti)
<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>High probability of harming I/O performance.</td>
</tr>
<tr>
<td>WARN</td>
<td>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.</td>
</tr>
<tr>
<td>OK</td>
<td>Best practices have been followed.</td>
</tr>
<tr>
<td>INFO</td>
<td>Relevant information regarding application configuration.</td>
</tr>
</tbody>
</table>

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

---

**Drishti v.0.3**

**JOB:** 1190243

**EXECUTABLE:** bin/d benchmark_parallel

**DARSHAN:** jbez_b benchmark_parallel_std1190243_7-23-45631-11755726114088236527_1.darshan

**EXECUTION DATE:** 2021-07-23 16:49:31-00:00 To 2021-07-23 16:49:32+00:00 (0.00 hours)

**FILES:** 6 files (1 use STIDIO, 2 use POSIX, 1 use MPI-I0)

**PROCESSES:** 64

**MINTS:** romio_no_indep_rw=true cb_nodes=4

---

**1 critical issues, 5 warnings, and 5 recommendations**

---

**META DATA**

- 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
- 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-I0 and read data using 640 (83.55%) collective operations
- Application uses MPI-I0 and write data using 768 (100.00%) collective operations
- Application could benefit from non-blocking (asynchronous) reads
- Application could benefit from non-blocking (asynchronous) writes
- Application is using inter-node aggregators (which require network communication)

---

2022 | LBL | Drishti report generated at 2022-08-05 13:19:59.787458 in 0.955 seconds
Provides actionable feedback for users

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.txt".

- **Recommendations:**
  - Consider buffering read operations into larger more contiguous ones.
  - Since the application already uses MPI-I/O, 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

```c
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 (78.83%) read requests.

- **Recommendations:**
  - Since you use MPI-I/O, 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

```c
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-I/O, 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

```c
...```
Sample configurations are provided

```c
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

```bash
1 # ----------------------------- #
2 # MPICH #
3 # ----------------------------- #
4 export MPICH_MPIIO_HINTS="*:cb_nodes=16:cb_buffer_size=16777216:romio_cb_write=enable:romio_ds_write=enable"
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 OMP_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
```
<table>
<thead>
<tr>
<th>Level</th>
<th>Interface</th>
<th>Detected Behavior</th>
<th>Jobs</th>
<th>Total (%)</th>
<th>Relative* (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>STDIO</td>
<td>High STDIO usage (&gt;10% of total transfer size uses STDIO)</td>
<td>43,120</td>
<td>38.29</td>
<td>52.1</td>
</tr>
<tr>
<td>OK</td>
<td>POSIX</td>
<td>High number of sequential read operations (≥ 80%)</td>
<td>38,104</td>
<td>33.84</td>
<td>58.14</td>
</tr>
<tr>
<td>OK</td>
<td>POSIX</td>
<td>High number of sequential write operations (≥ 80%)</td>
<td>64,486</td>
<td>57.26</td>
<td>98.39</td>
</tr>
<tr>
<td>INFO</td>
<td>POSIX</td>
<td>Write operation count intensive (&gt;10% more writes than reads)</td>
<td>26,114</td>
<td>23.19</td>
<td>39.84</td>
</tr>
<tr>
<td>INFO</td>
<td>POSIX</td>
<td>Read operation count intensive (&gt;10% more reads than writes)</td>
<td>23,168</td>
<td>20.57</td>
<td>35.35</td>
</tr>
<tr>
<td>INFO</td>
<td>POSIX</td>
<td>Write size intensive (&gt;10% more bytes written then read)</td>
<td>23,568</td>
<td>20.93</td>
<td>35.96</td>
</tr>
<tr>
<td>INFO</td>
<td>POSIX</td>
<td>Read size intensive (&gt;10% more bytes read then written)</td>
<td>40,950</td>
<td>36.36</td>
<td>62.48</td>
</tr>
<tr>
<td>WARN</td>
<td>POSIX</td>
<td>Redundant reads</td>
<td>14,518</td>
<td>12.89</td>
<td>22.15</td>
</tr>
<tr>
<td>WARN</td>
<td>POSIX</td>
<td>Redundant writes</td>
<td>59</td>
<td>0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of small (&lt;1MB) read requests (&gt;10% of total read requests)</td>
<td>64,858</td>
<td>57.59</td>
<td>98.96</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of small (&lt;1MB) write requests (&gt;10% of total write requests)</td>
<td>64,552</td>
<td>57.32</td>
<td>98.49</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of misaligned memory requests (&gt;10%)</td>
<td>36,337</td>
<td>32.27</td>
<td>55.44</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of misaligned file requests (&gt;10%)</td>
<td>65,075</td>
<td>57.79</td>
<td>99.29</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of random read requests (&gt;20%)</td>
<td>26,574</td>
<td>23.6</td>
<td>40.54</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of random write requests (&gt;20%)</td>
<td>559</td>
<td>0.5</td>
<td>0.85</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of small (&lt;1MB) reads to shared-files (&gt;10% of total reads)</td>
<td>60,121</td>
<td>53.39</td>
<td>91.73</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High number of small (&lt;1MB) writes to shared-files (&gt;10% of total writes)</td>
<td>55,414</td>
<td>49.21</td>
<td>84.55</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>High metadata time (at least one rank spends &gt;30 seconds)</td>
<td>9,410</td>
<td>8.36</td>
<td>14.35</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>Data transfer imbalance between ranks causing stragglers (&gt;15% difference)</td>
<td>40,601</td>
<td>36.05</td>
<td>61.95</td>
</tr>
<tr>
<td>HIGH</td>
<td>POSIX</td>
<td>Time imbalance between ranks causing stragglers (&gt;15% difference)</td>
<td>40,533</td>
<td>35.99</td>
<td>61.84</td>
</tr>
<tr>
<td>Level</td>
<td>Interface</td>
<td>Detected Behavior</td>
<td>Jobs</td>
<td>Total (%)</td>
<td>Relative* (%)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>WARN</td>
<td>MPI-I0</td>
<td>No MPI-I0 calls detected from Darshan logs</td>
<td>109,569</td>
<td>97.3</td>
<td>-</td>
</tr>
<tr>
<td>HIGH</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 but no collective read operation</td>
<td>169</td>
<td>0.15</td>
<td>5.55</td>
</tr>
<tr>
<td>HIGH</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 but no collective write operation</td>
<td>428</td>
<td>0.38</td>
<td>14.06</td>
</tr>
<tr>
<td>WARN</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 but no non-blocking read operations</td>
<td>3,043</td>
<td>2.7</td>
<td>100</td>
</tr>
<tr>
<td>WARN</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 but no non-blocking write operations</td>
<td>3,043</td>
<td>2.7</td>
<td>100</td>
</tr>
<tr>
<td>OK</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 and collective read operations</td>
<td>402</td>
<td>0.36</td>
<td>13.21</td>
</tr>
<tr>
<td>OK</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 and collective write operations</td>
<td>2,592</td>
<td>2.3</td>
<td>85.17</td>
</tr>
<tr>
<td>HIGH</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 and inter-node aggregators</td>
<td>2,496</td>
<td>2.22</td>
<td>82.02</td>
</tr>
<tr>
<td>WARN</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 and intra-node aggregators</td>
<td>304</td>
<td>0.27</td>
<td>9.99</td>
</tr>
<tr>
<td>OK</td>
<td>MPI-I0</td>
<td>Detected MPI-I0 and one aggregator per node</td>
<td>29</td>
<td>0.03</td>
<td>0.95</td>
</tr>
</tbody>
</table>
# 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