



Visualizing I/O Bottlenecks with DXT Explorer 2.0

Jean Luca Bez, Hammad Ather, Suren Byna

Lawrence Berkeley National Laboratory
jlbez@lbl.gov



BERKELEY LAB
Bringing Science Solutions to the World

How to understand I/O behavior?

- Using the HPC I/O stack **efficiently** is a **tricky problem!**
- Darshan is a popular tool to collect I/O **profiling**
 - It **aggregates** information to provide insights
- Extended **tracing** mode (DXT)
 - Fine grain view of the I/O behavior
 - POSIX or MPI-IO, read/write
 - Rank, segment, offset, request size
 - Start and end timestamp



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



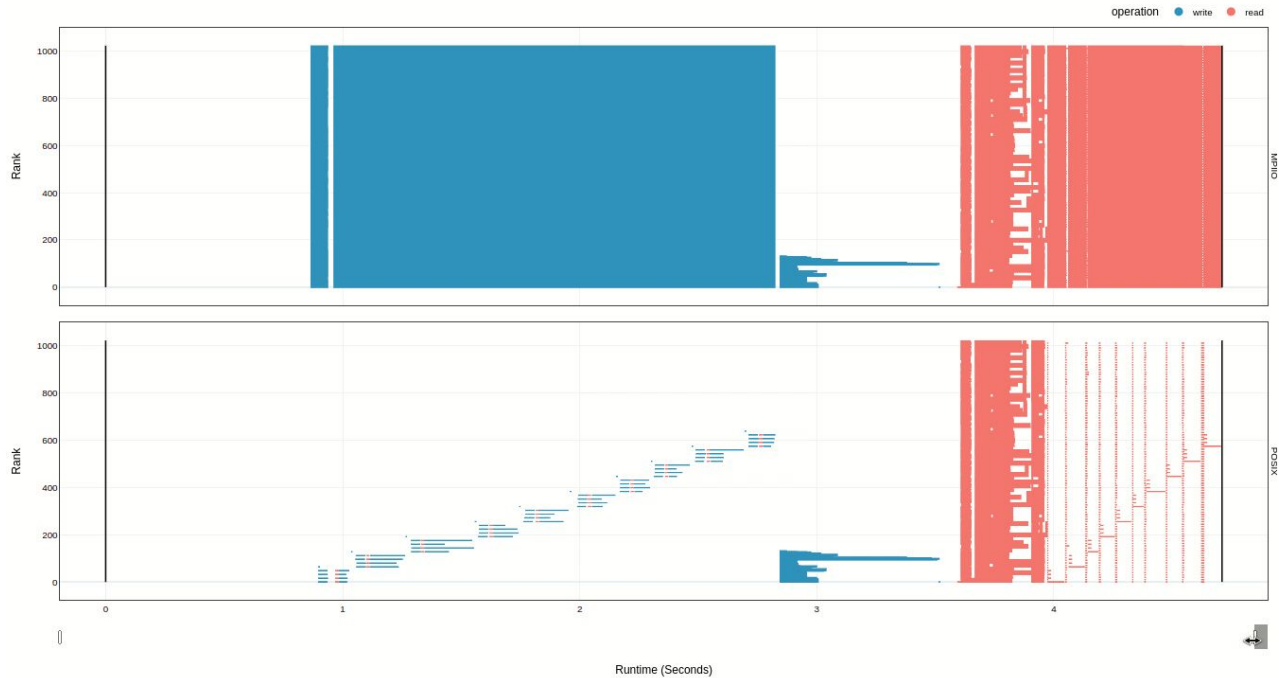
github.com/hpc-io/dxt-explorer



`docker pull hpcio/dxt-explorer`

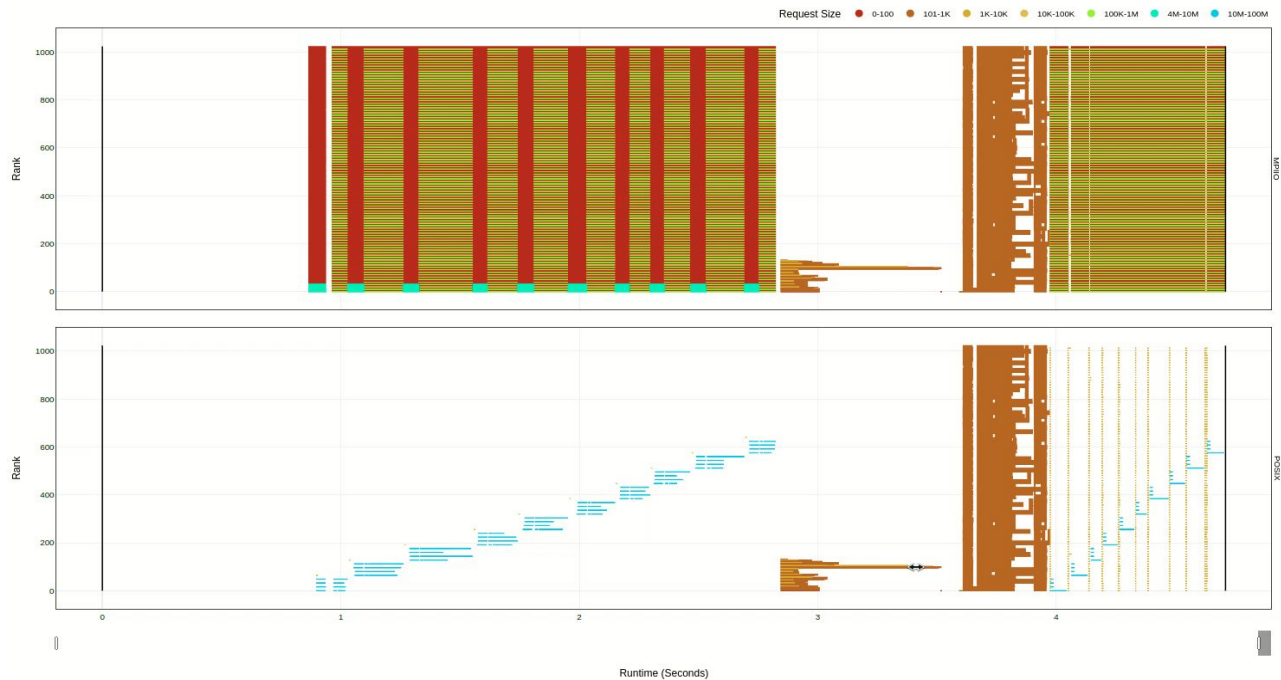


DXT EXPLORER



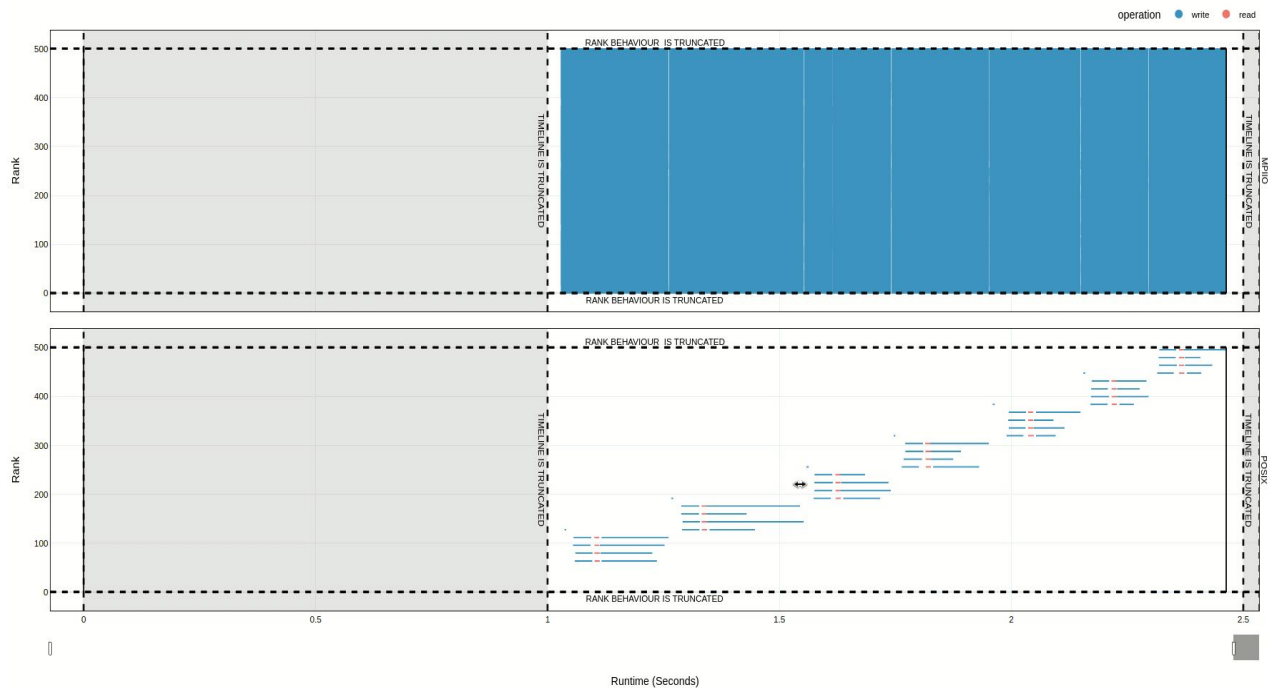
Explore the timeline by **zooming in and out** and observing how the MPI-I/O calls are translated to the POSIX layer. Visualize relevant information in the context of each I/O call (**rank, operation, duration, request size, and OSTs if Lustre**).





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





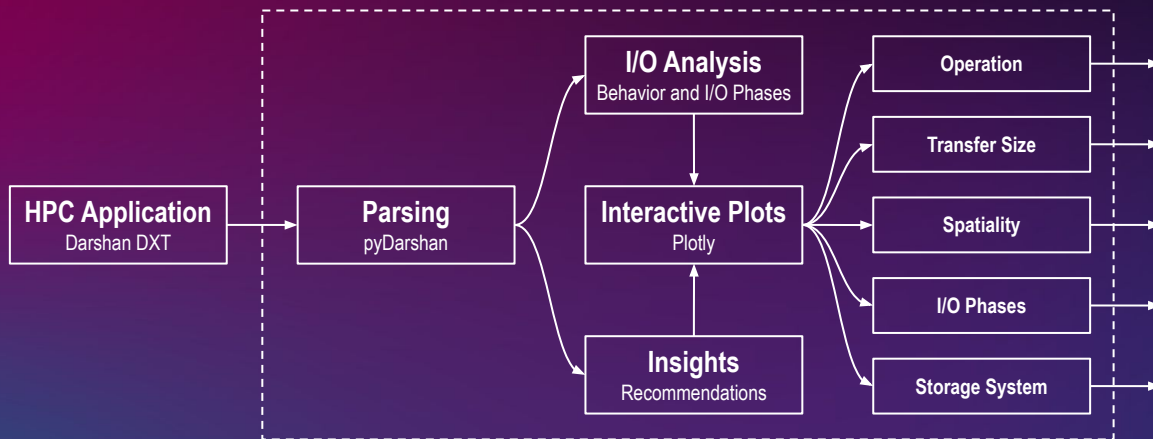
Explore the timeline by **zooming in and out** and observing how the MPI-IO calls are translated to the POSIX layer. **Truncated** (by rank, time, or both) plots help visualize **larger traces**.



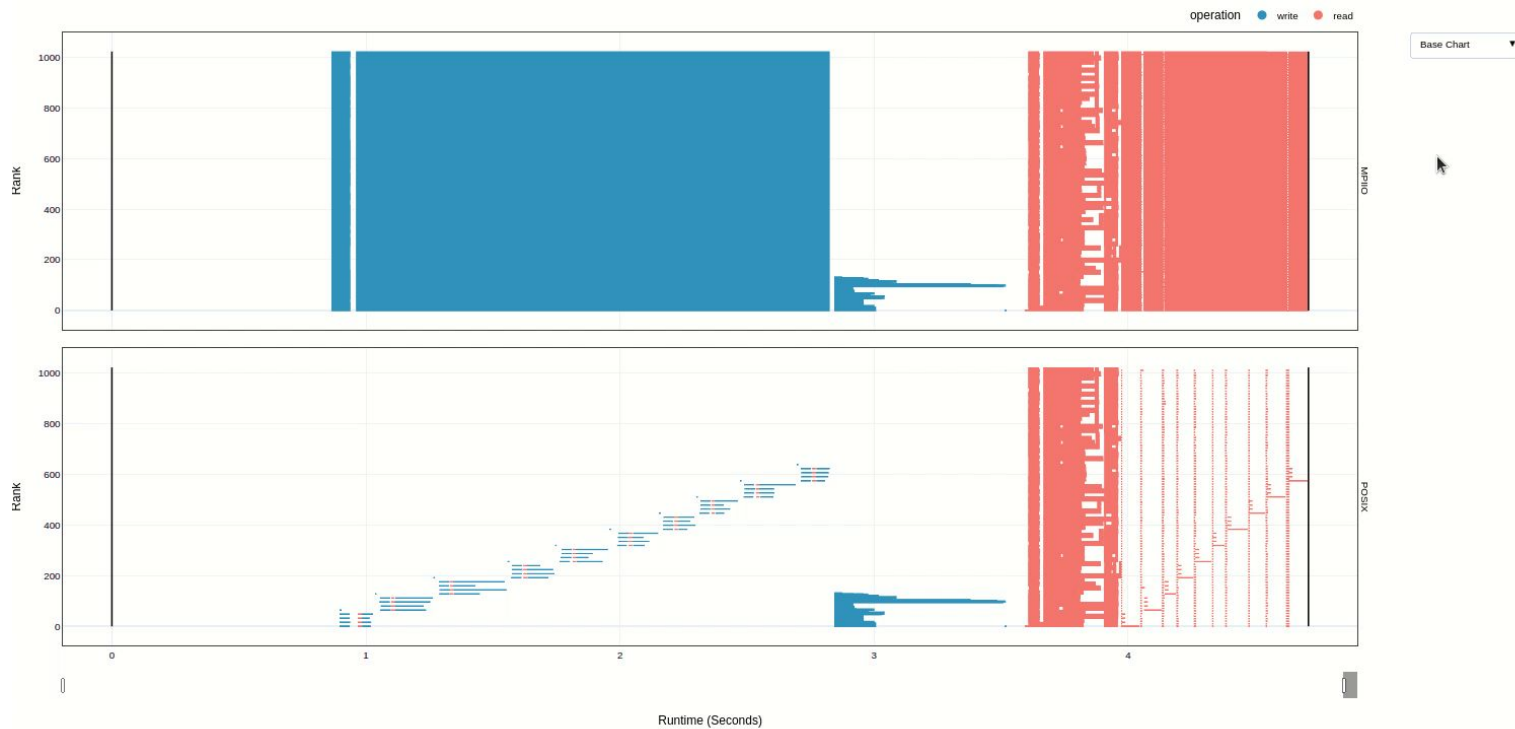


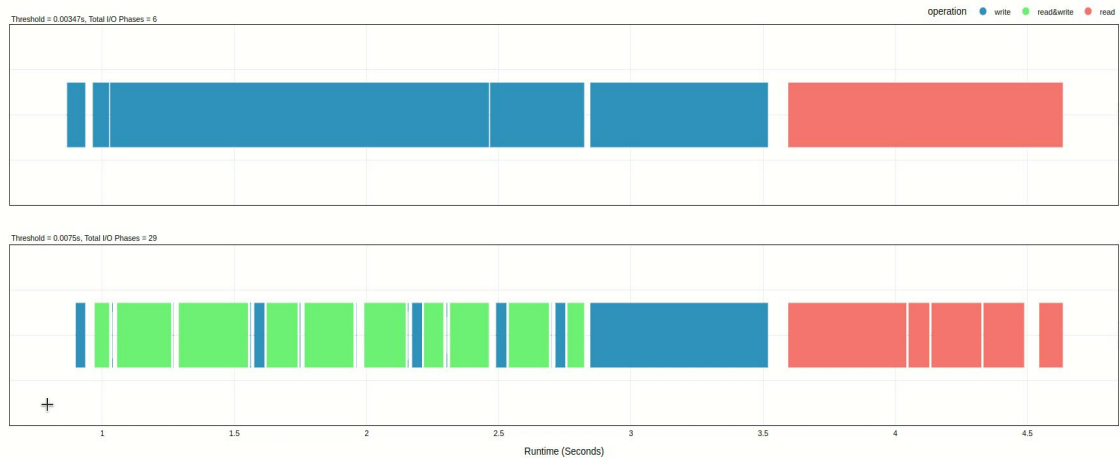
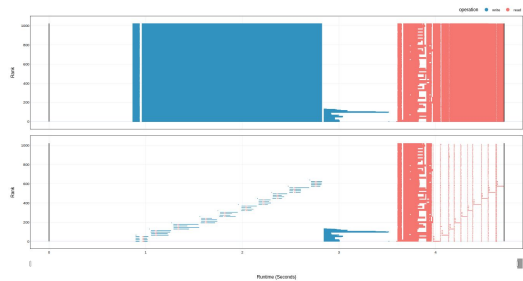
New Features

Coming Soon!



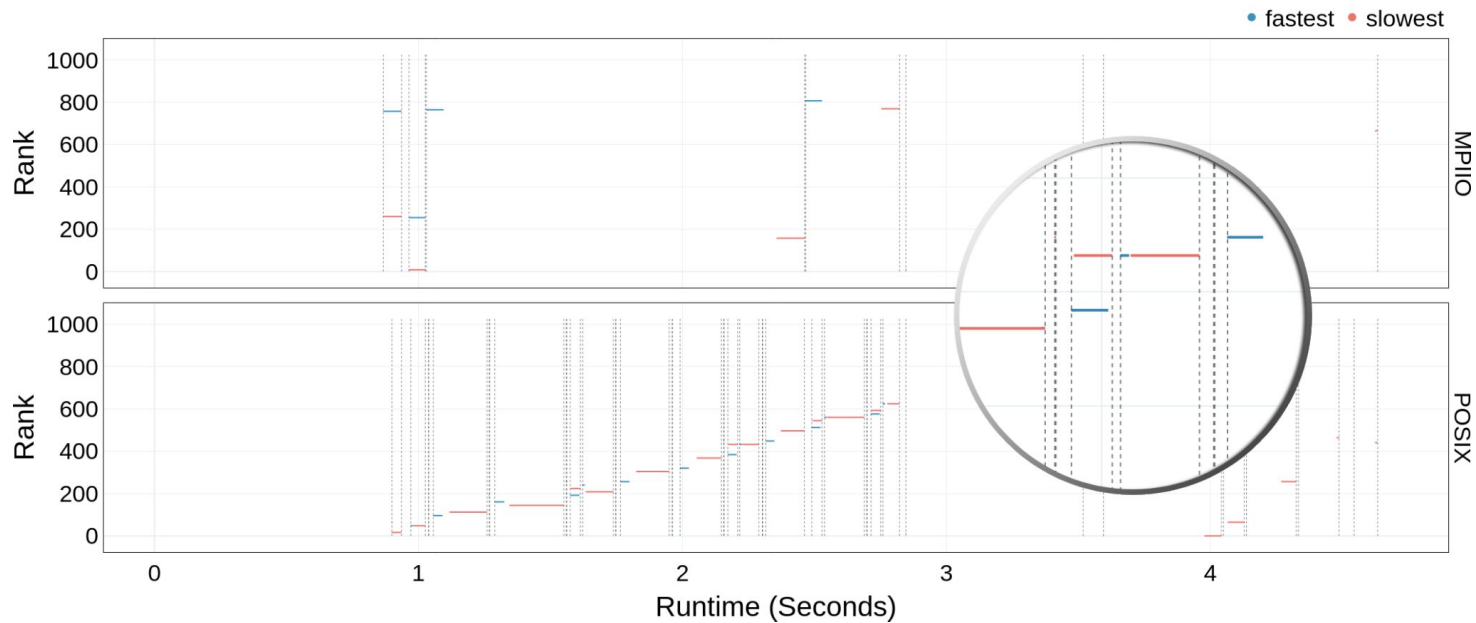
DXT EXPLORER





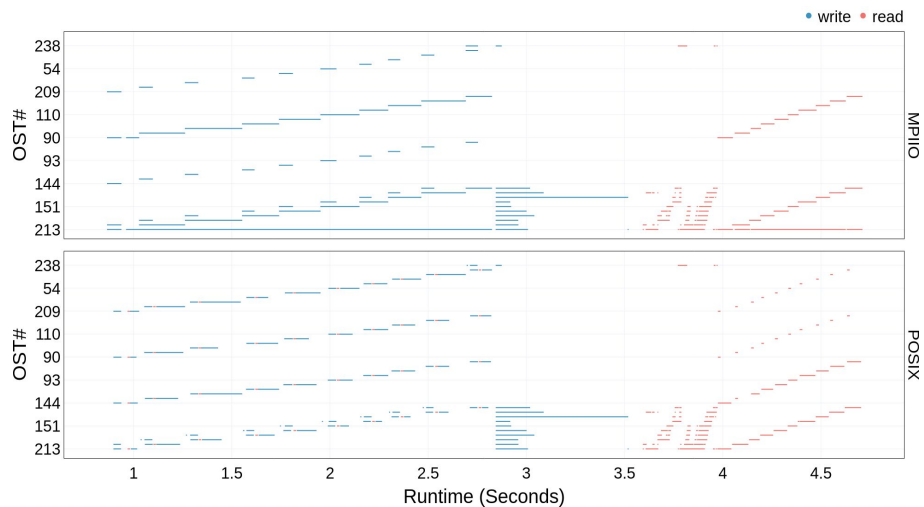
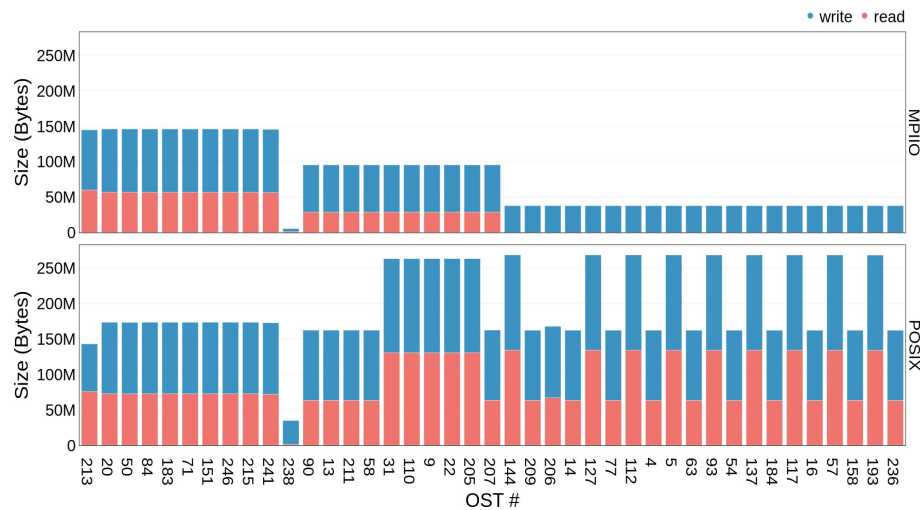
Explore the **I/O phases** detected based on behavior and threshold.





Explore the **stragglers** in the entire execution and the **critical path**.
Upon hovering over a phase, all the information related to the **fastest** and **slowest** rank is shown. The dotted lines are the start and the end of a phase.





Novel interactive visualizations towards exploring **file system usage**.



How to get DXT Explorer?

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

# Run DXT Explorer with the provided .darshan DXT traces
$ dxt-explorer --verbose samples/REPLACE_WITH_FILE_NAME.darshan

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



How to run DXT Explorer?

```
usage: dxt-explorer [-h] [-o OUTPUT] [-p PREFIX] [-t] [-s] [-i] [-oo] [-ot] [-d] [-l] [--start START] [--end END] [--from START_RANK]
[--to END_RANK] [--browser] [-r] [-u] [-st] [-v] darshan
```

DXT Explorer:

positional arguments:

darshan Input .darshan file

optional arguments:

-h, --help show this help message and exit
-o OUTPUT, --output OUTPUT Output directory
-p PREFIX, --prefix PREFIX Output directory
-t, --transfer Generate an interactive data transfer explorer
-s, --spatiality Generate an interactive spatiality explorer
-i, --io_phase Generate an interactive I/O phase explorer
-oo, --ost_usage_operation Generate an interactive OST usage operation explorer
-ot, --ost_usage_transfer Generate an interactive OST usage data transfer size 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
--browser Open the browser with the generated plot
-r, --rank_zero_workload Determine if rank 0 is doing more I/O than the rest of the workload
-u, --unbalanced_workload Determine which ranks have unbalanced workload
-st, --stragglers Determine the 5 percent slowest operations in the time distribution
-v, --version Show program's version number and exit





Visualizing I/O Bottlenecks with DXT Explorer 2.0

Jean Luca Bez, Hammad Ather, Suren Byna
jlbez@lbl.gov



`docker pull hpcio/dxt-explorer`



github.com/hpc-io/dxt-explorer



BERKELEY LAB

Bringing Science Solutions to the World

