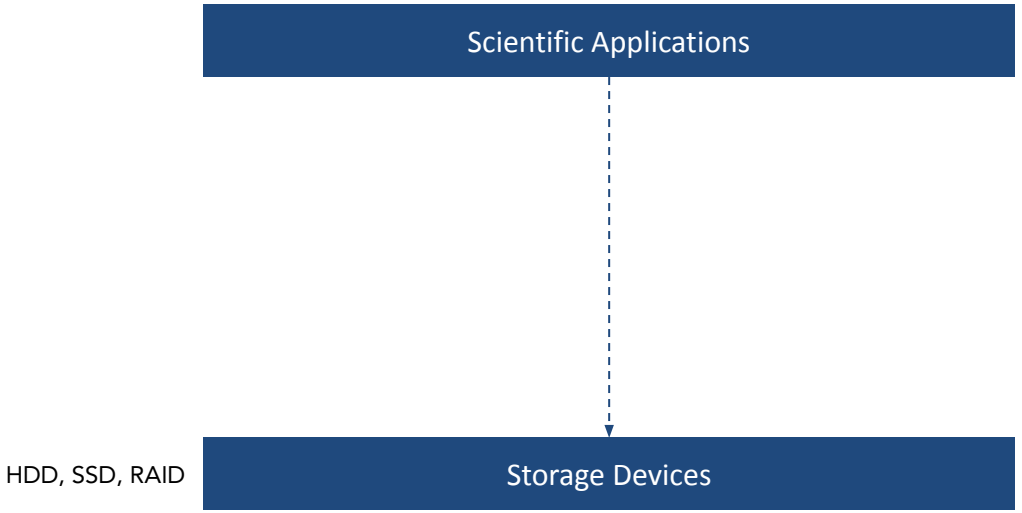
Understanding I/O Behavior with Interactive Darshan Log Analysis

Jean Luca Bez
Suren Byna

EXASCALE COMPUTING PROJECT
National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science

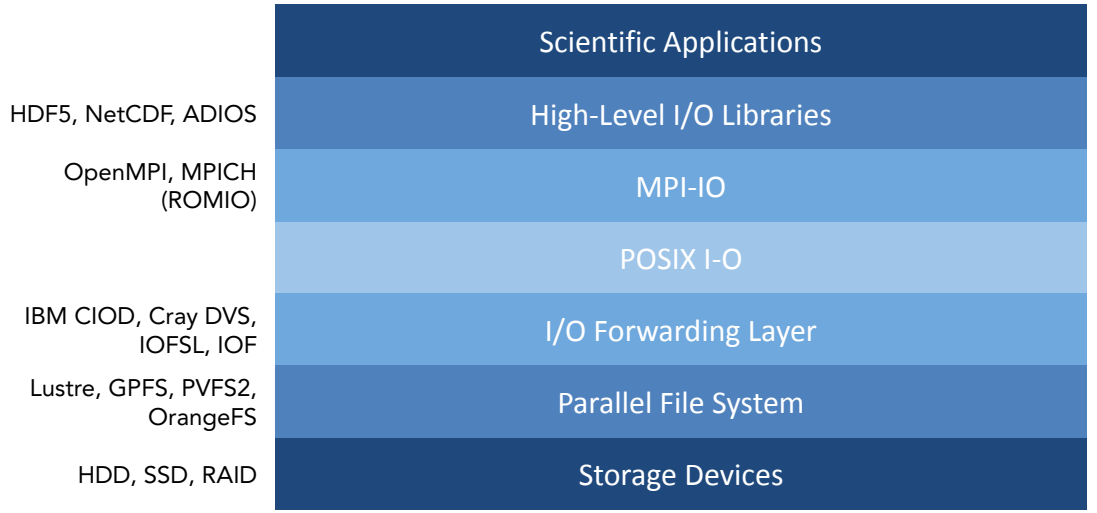
The HPC I/O stack is **complex**!

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



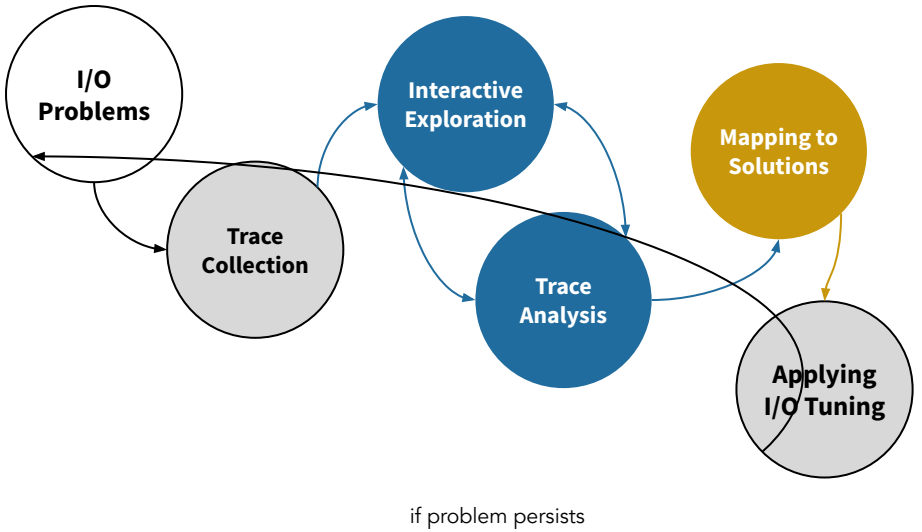
The HPC I/O stack is **complex**!

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



Interactive Exploration!

- Darshan can collect fine grain traces with **DXT**
  - 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
- There is still a lack of translation from I/O bottlenecks to optimizations

DXT EXPLORER

github.com/hpc-io/dxt-explorer

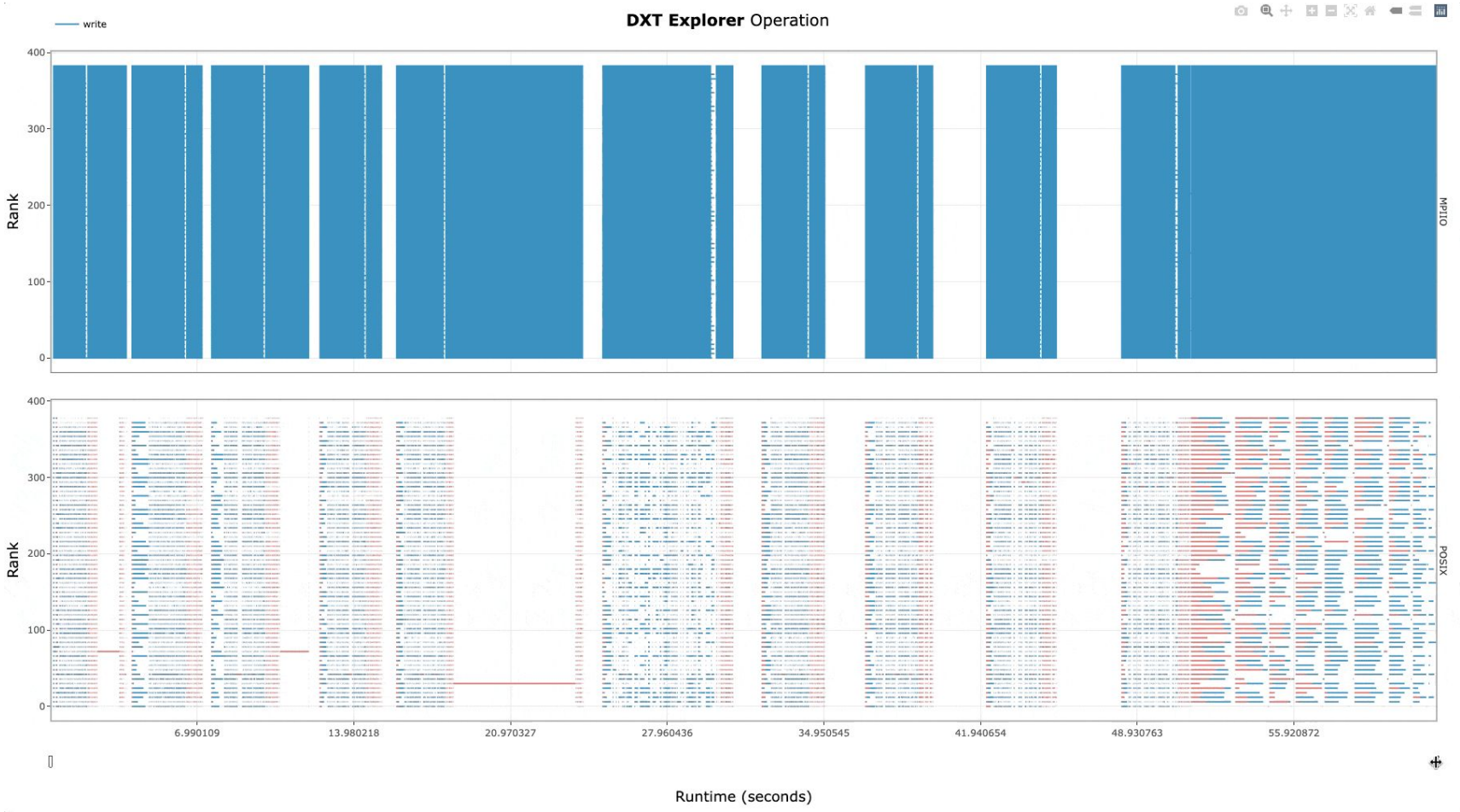
docker pull hpcio/dxt-explorer

Spack recipe coming soon!

What **options** do we have?

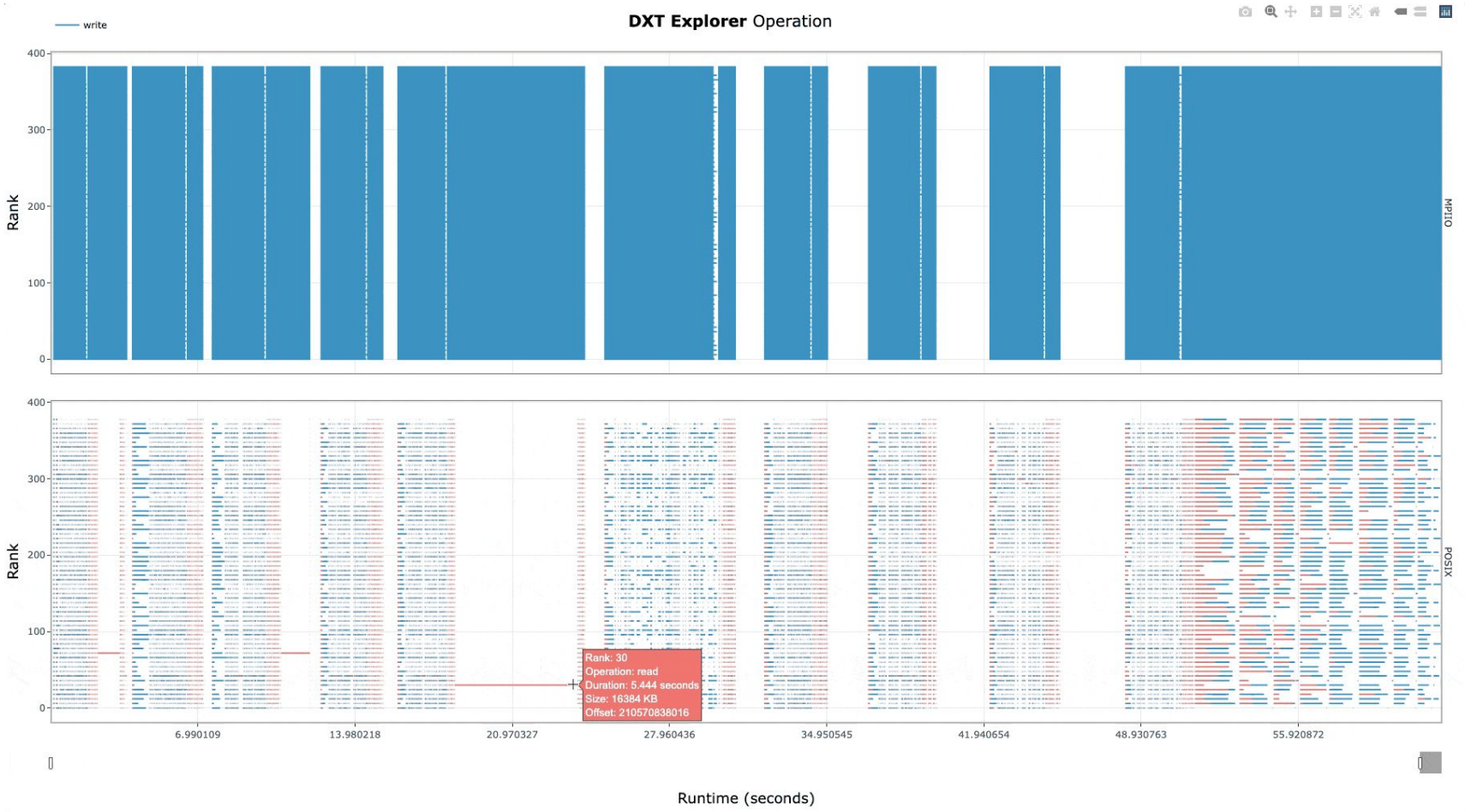
```
usage: dxt-explorer [-h] [-o OUTPUT] [-t] [-s] [-d] [-l] [--start START] [--end END] [--from START_RANK] [--to END_RANK] darshan

DXT Explorer:

positional arguments:
  darshan               Input .darshan file

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Name of the output file
  -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
```

Exploring I/O **operations**…



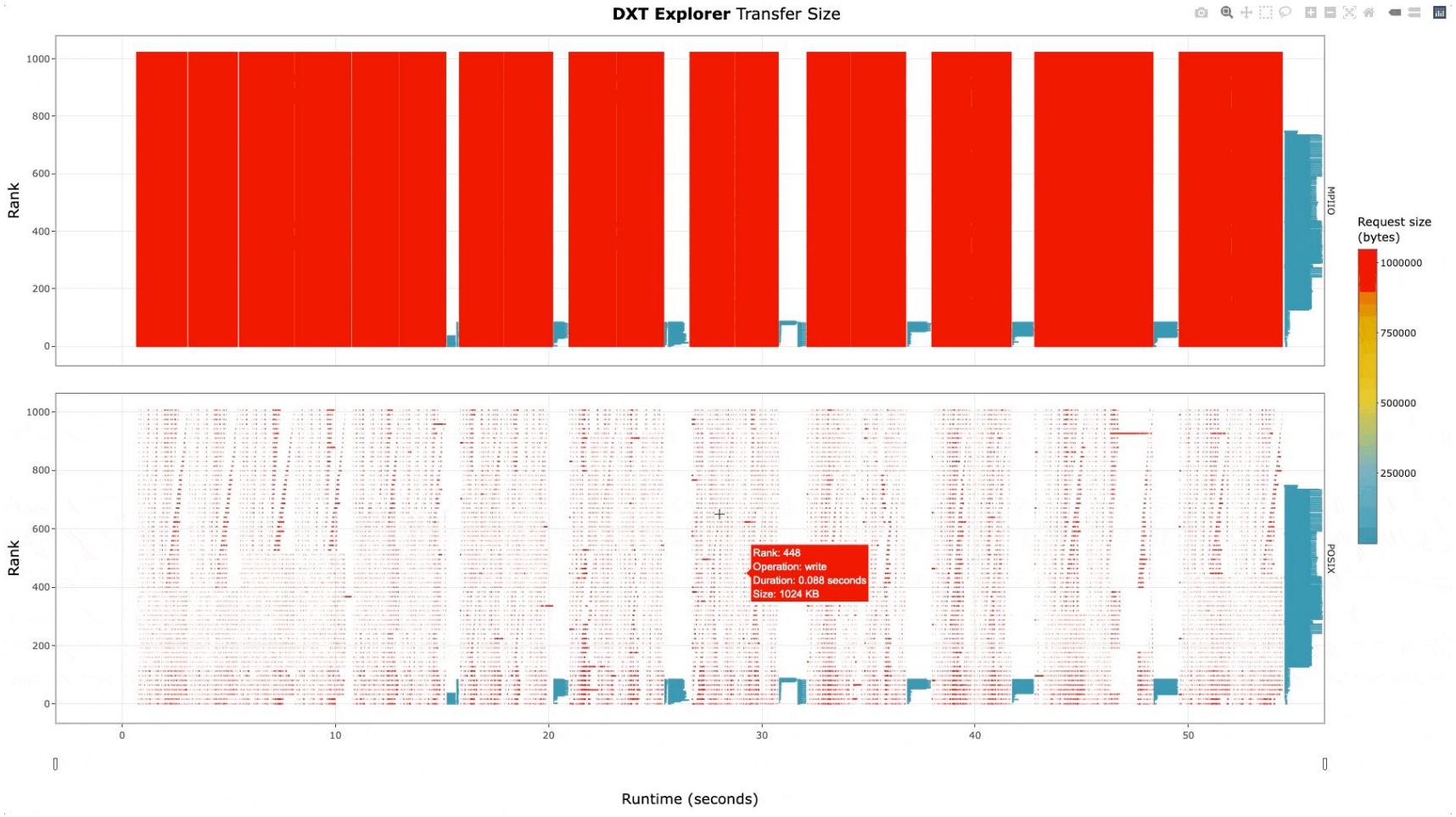
Explore the timeline by **zooming in and out** and observing how the **MPI-IO** calls are translated to the **POSIX** layer. For instance, you can use this feature to detect stragglers.

Context is important!



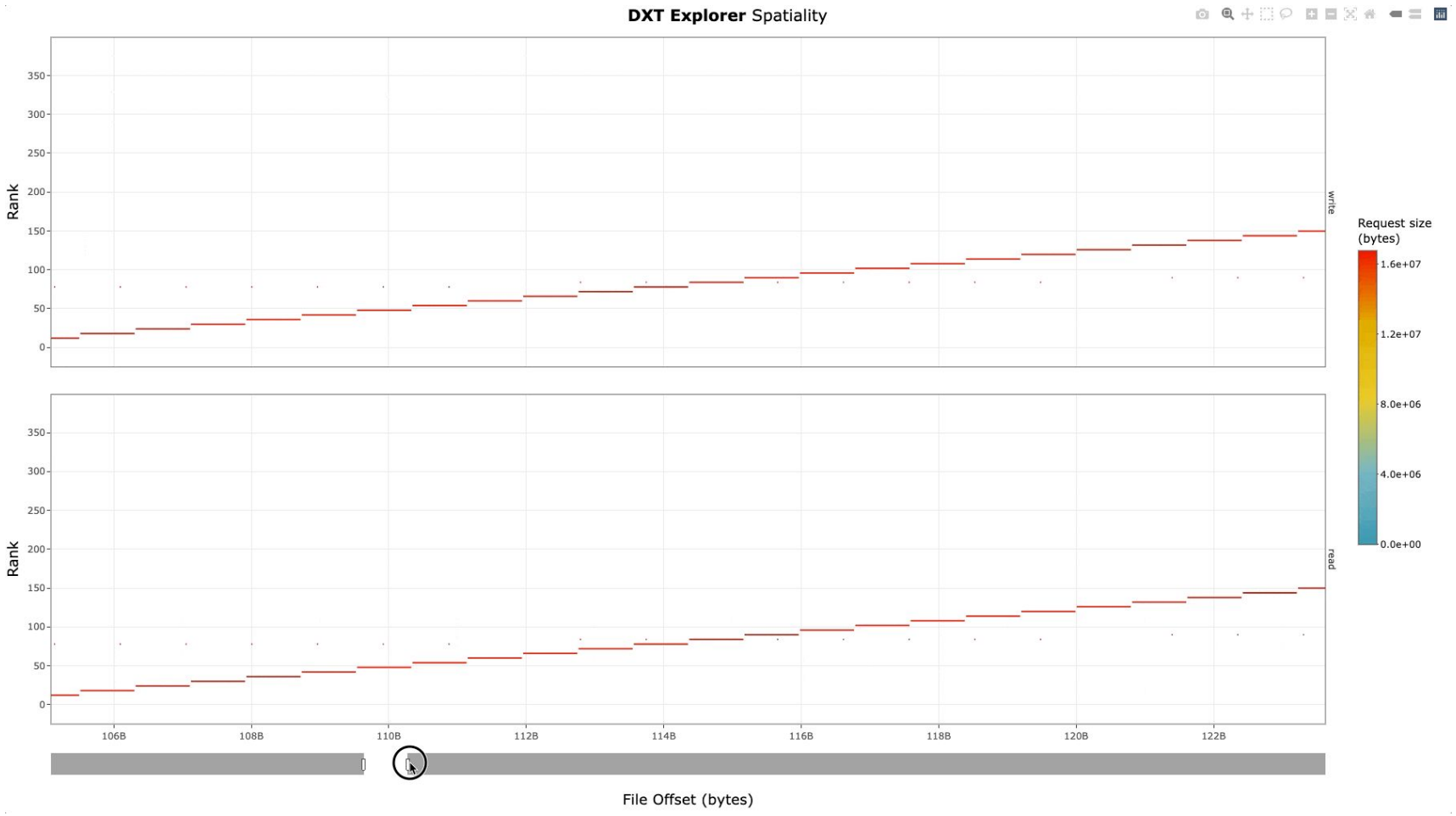
Visualize relevant information in the **context** of **each I/O call** (rank, operation, duration, request size, and OSTs if Lustre) by hovering over a given operation.

Exploring **request sizes**…



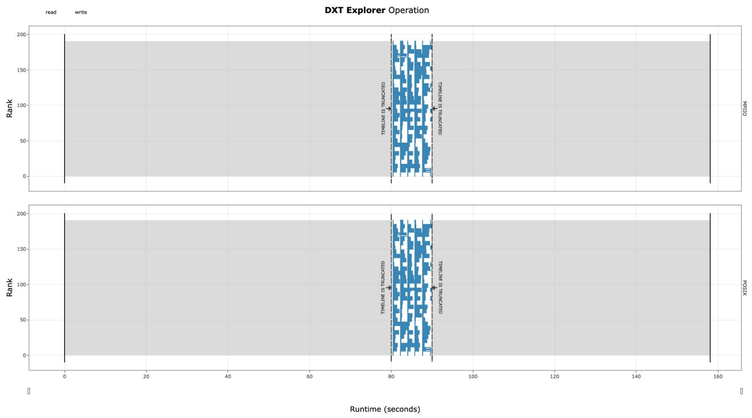
Explore the **operations by size** in POSIX and MPI-IO. You can, for instance, identify small or metadata operations from this visualization.

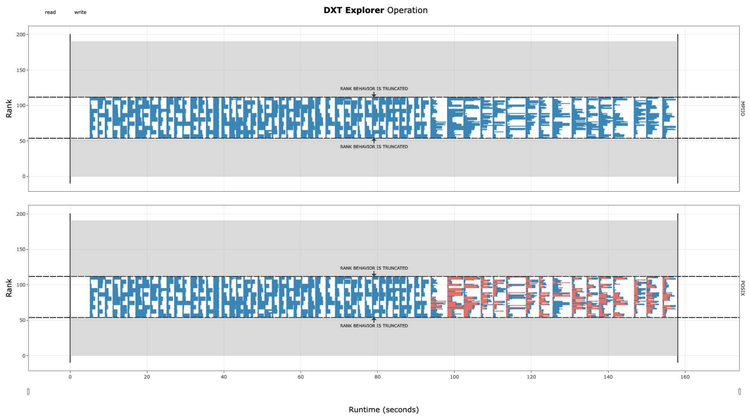
Exploring **spatiality**…

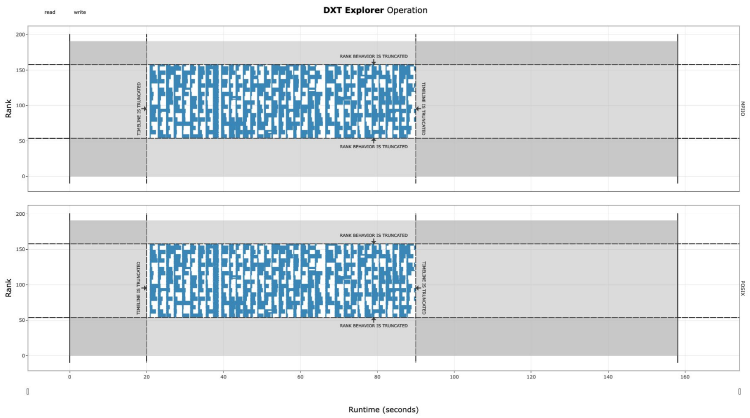


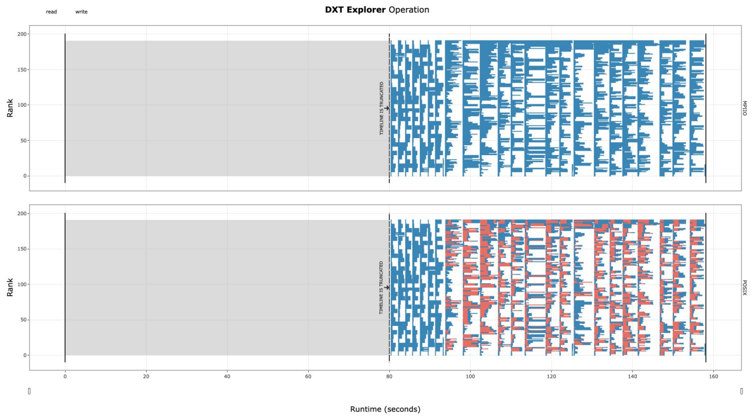
Explore the **spatiality** of accesses in file by each rank with **contextual** information.

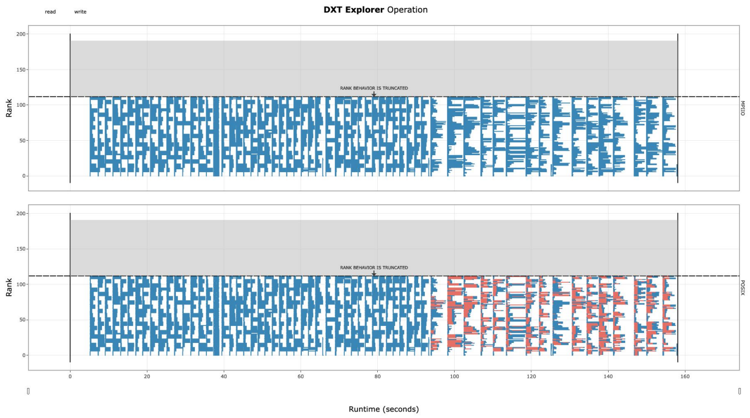
Let’s **focus**!

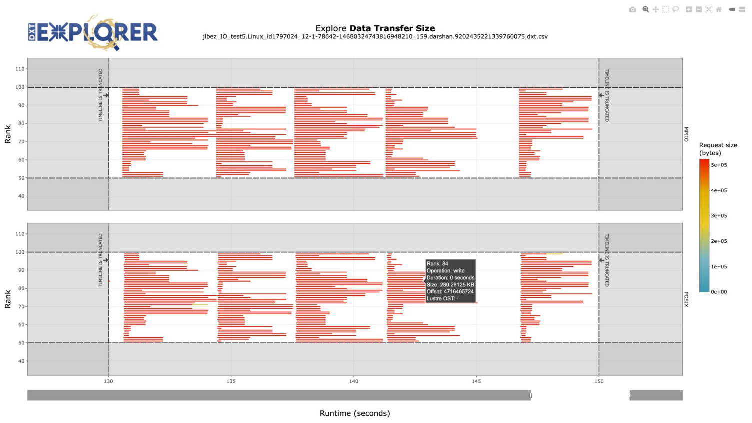

DXT Explorer Operation
read
write
Rank
TIMELINE IS TRUNCATED
MPIIO
POSIX
Runtime (seconds)


DXT Explorer Operation
read
write
Rank
RANK BEHAVIOR IS TRUNCATED
MPIIO
POSIX
Runtime (seconds)

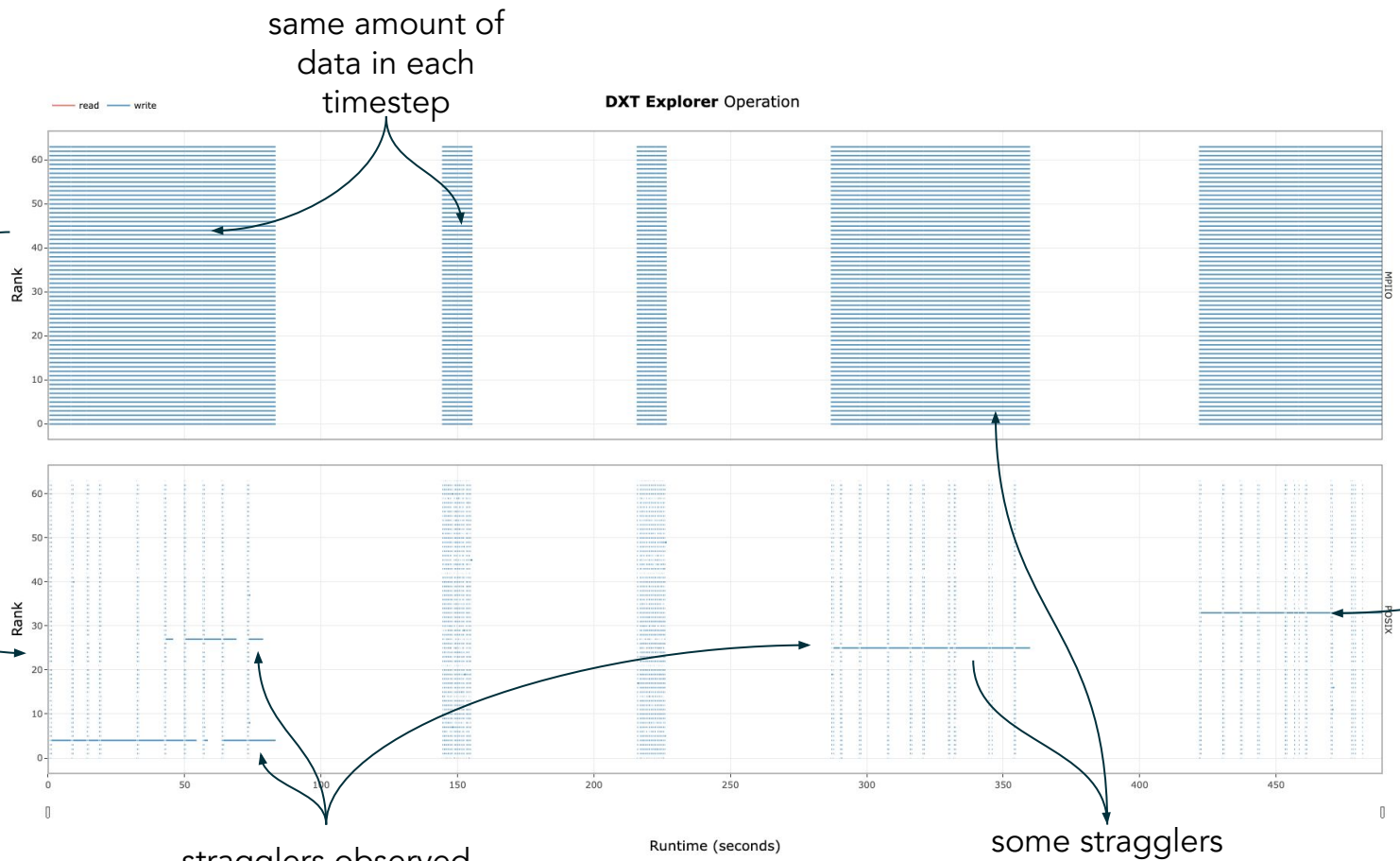

DXT Explorer Operation
read
write
Rank
RANK BEHAVIOR IS TRUNCATED
TIMELINE IS TRUNCATED
MPIIO
POSIX
Runtime (seconds)


DXT Explorer Operation
read
write
Rank
TIMELINE IS TRUNCATED
MPIIO
POSIX
Runtime (seconds)


DXT Explorer Operation
read
write
Rank
RANK BEHAVIOR IS TRUNCATED
MPIIO
POSIX
Runtime (seconds)

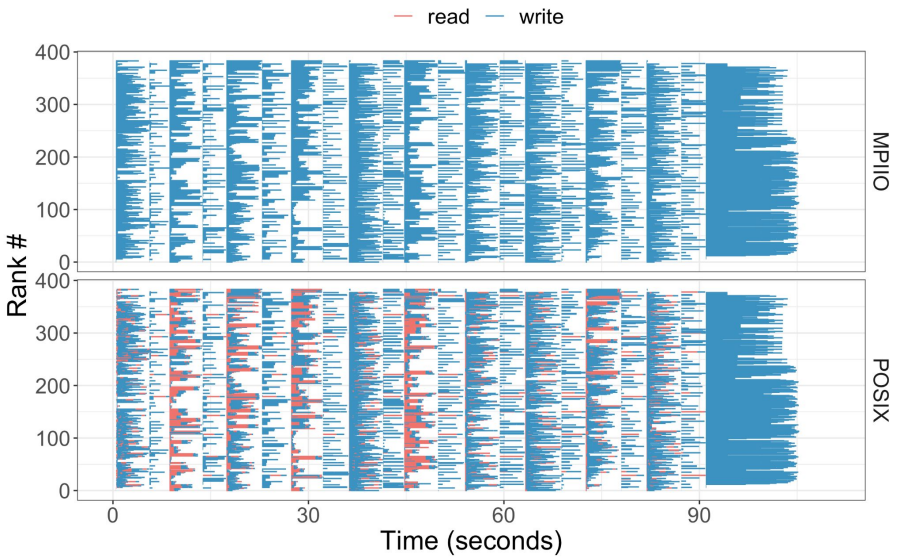

Explore Data Transfer Size
Rank
Request size (bytes)
Runtime (seconds)

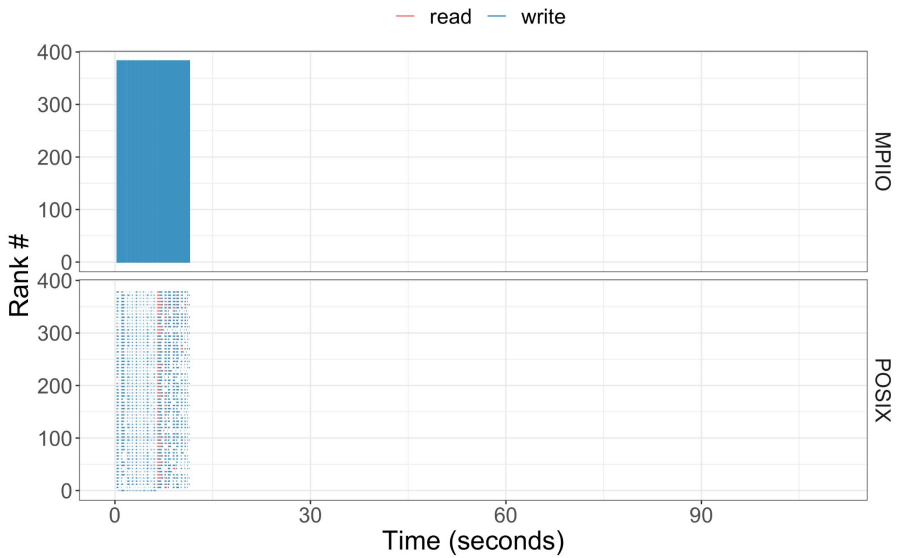
What can we see with **OpenPMD**?

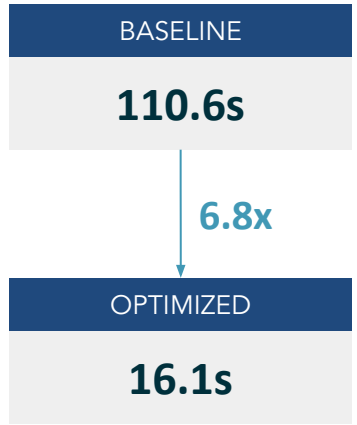


OpenPMD use case

- Collective I/O using ROMIO: **1.54x** speedup
- GPFS large block I/O + HDF5 collective metadata: **+3.8x** speedup
  - Discovered an **issue** with collective metadata introduced in HDF5 1.10.5
- Fix combined with previous optimizations gives a total of **6.8x** speedup from baseline

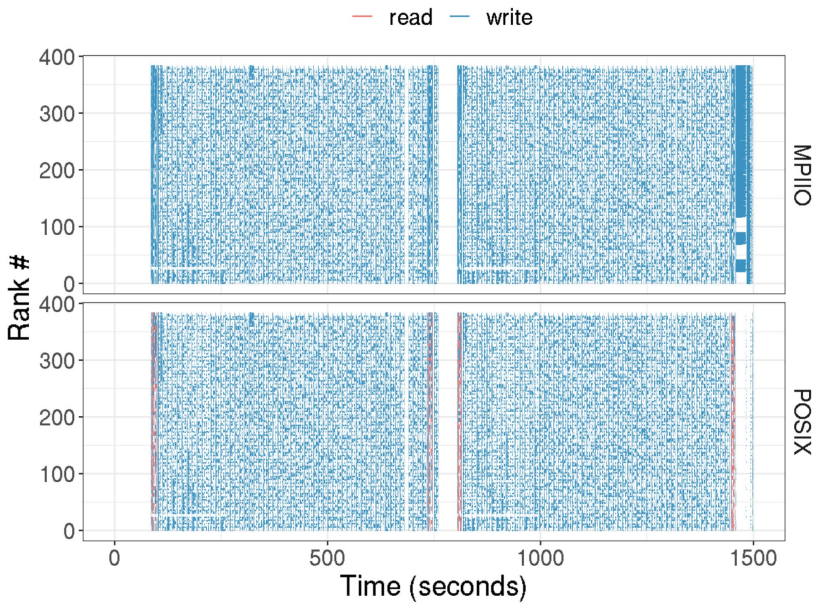


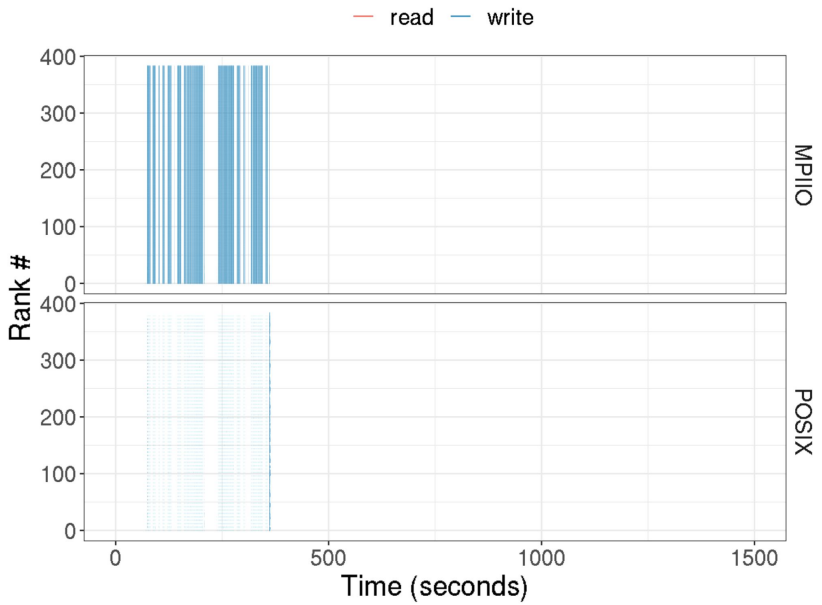


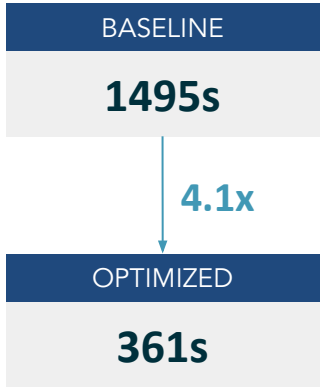


FLASH use case

- 2 checkpoint files (≈2.3TB each) and 2 plot file (≈14GB each)
- FLASH was not using collective MPI-IO calls
- **Optimizations**: collective I/O, HDF5 alignment, and defer metadata flush







Conclusion

- **DXT Explorer**
  - Adds an **interactive** component to **Darshan DXT trace analysis**
  - Moves a **step closer** towards connecting the dots between bottleneck detection and tuning
  - We can only do something about it, if we know something is wrong
  - Our tool is publicly available at **github.com/hpc-io/dxt-explorer**
- There is still the need for **further R&D**
  - How to better report findings to end-users?
  - How to make **automatic recommendations** by mapping problems to tuning options?

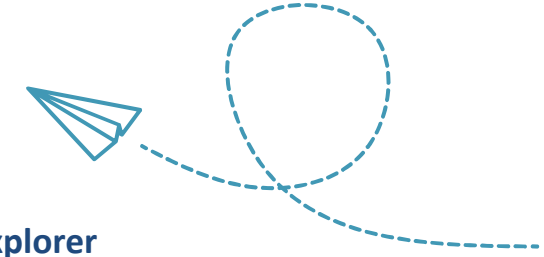
Thank you!

Approved for public release

EXASCALE COMPUTING PROJECT

You can reach us by email:

jlbez@lbl.gov



docker pull hpcio/dxt-explorer

github.com/hpc-io/dxt-explorer

DEMO **bit.ly/dxt-explorer**

National Nuclear Security Administration

BERKELEY LAB
Bringing Science Solutions to the World

OAK RIDGE National Laboratory

Argonne NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY | Office of Science