SC21
St. Louis, MO | science & beyond.

I/O Bottleneck Detection and Tuning: Towards Connecting the Dots using Interactive Log Analysis

Jean Luca Bez, Houjun Tang, Bing Xie, David Williams-Young,
Rob Latham, Rob Ross, Sarp Oral, and Suren Byna

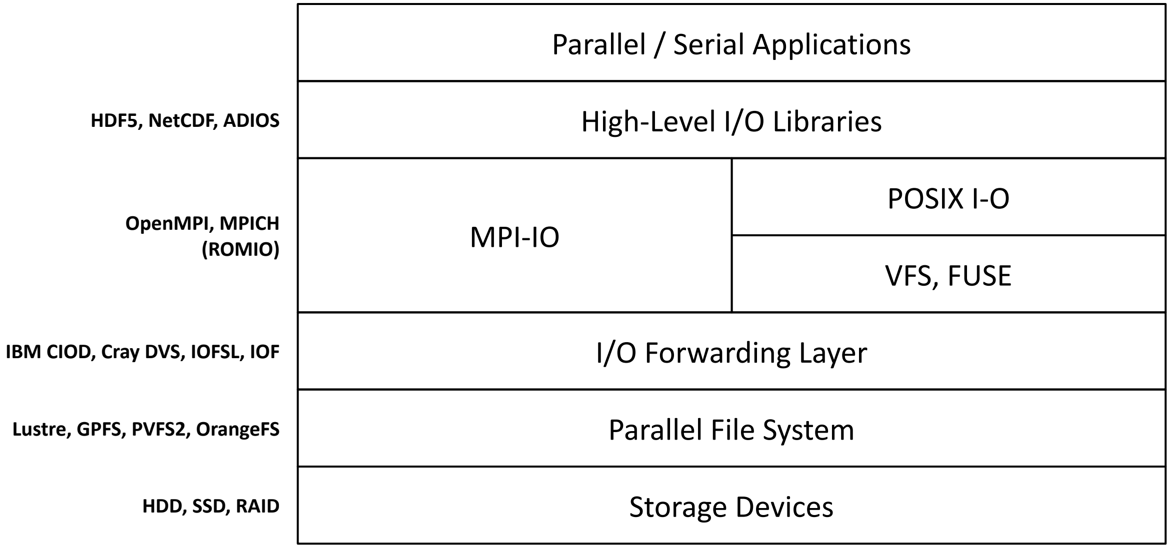
ECP EXASCALE COMPUTING PROJECT
BERKELEY LAB Bringing Science Solutions to the World
OAK RIDGE National Laboratory
Argonne NATIONAL LABORATORY

Agenda

- Overview of the HPC I/O Stack
- Darshan and DXT
- The Missing Dots...
- Interactive Exploration & Optimization
- The DXT Explorer Tool
- DXT Explorer in Practice
- Conclusion

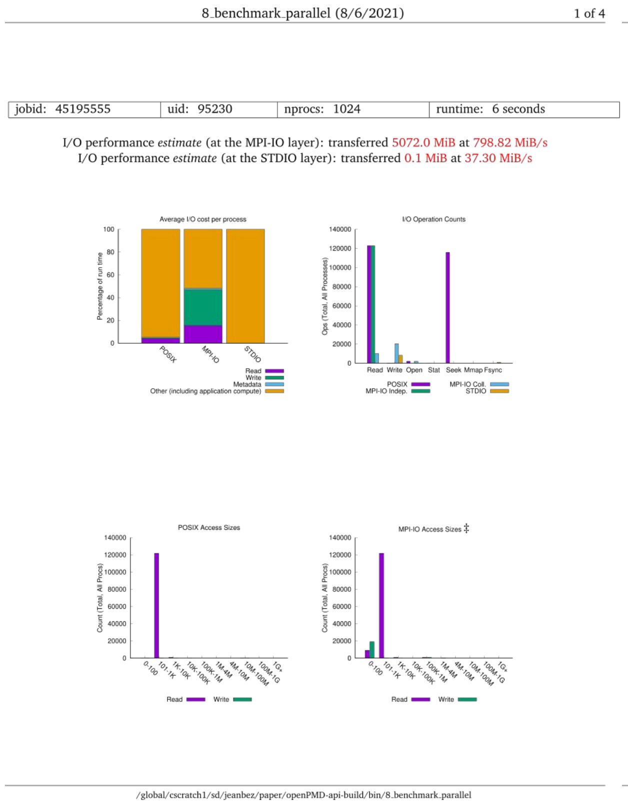
Overview of the HPC I/O Stack

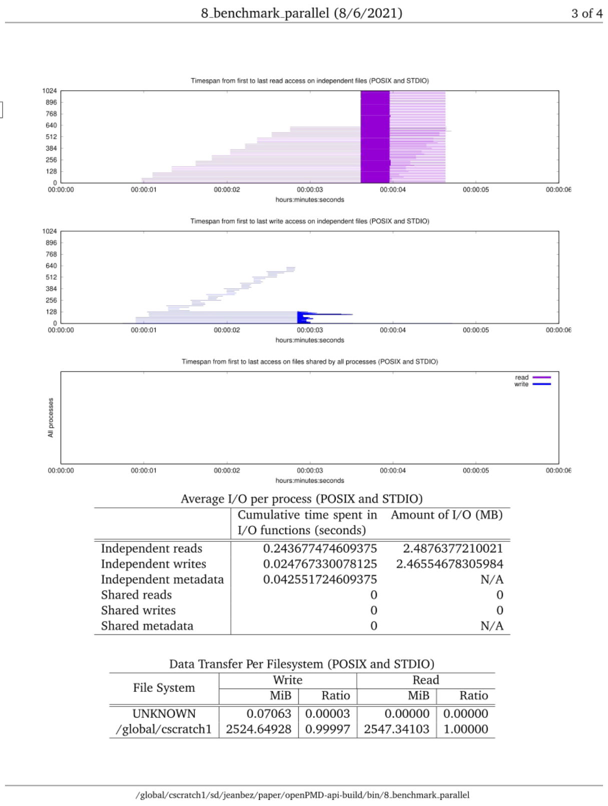
- HPC **I/O stack** is complex (multiple layers)
- Interplay of factors can affect I/O performance
- Various **optimizations techniques** available
- Plethora of **tunable parameters**
  - Each layer brings a new set of parameters
- Using the all layers **efficiently** is a **tricky** problem



Darshan and DXT

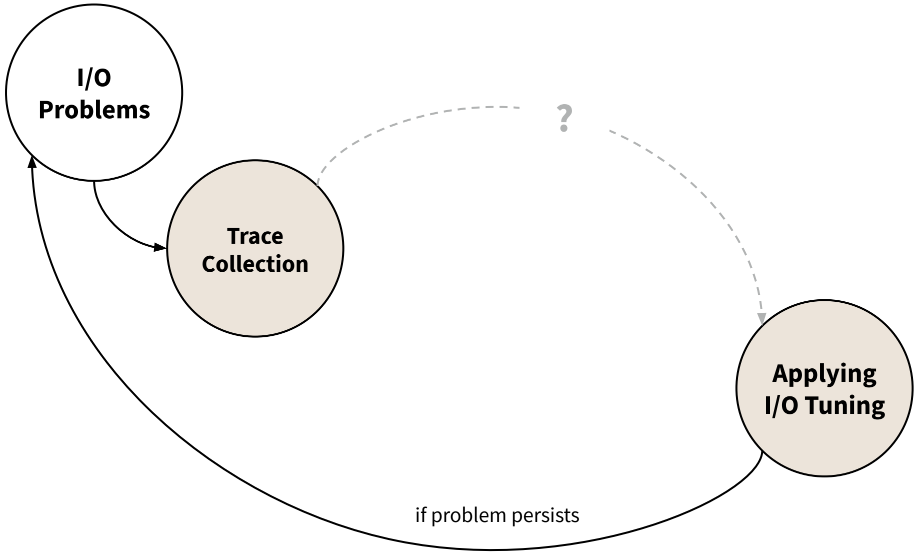
- Darshan is a popular tool to collect **I/O profiling**
- It **aggregates** information to provide insights
- **Extended tracing** mode (DXT)
  - Provide a fine grain view of the application behavior
  - Interface (POSIX or MPI-IO), operation (read/write)
  - Rank, segment, offset, request size
  - Start and end timestamp
- How to **visualize** and extract insights DXT data?
  - Identify I/O bottlenecks
  - Optimize the application



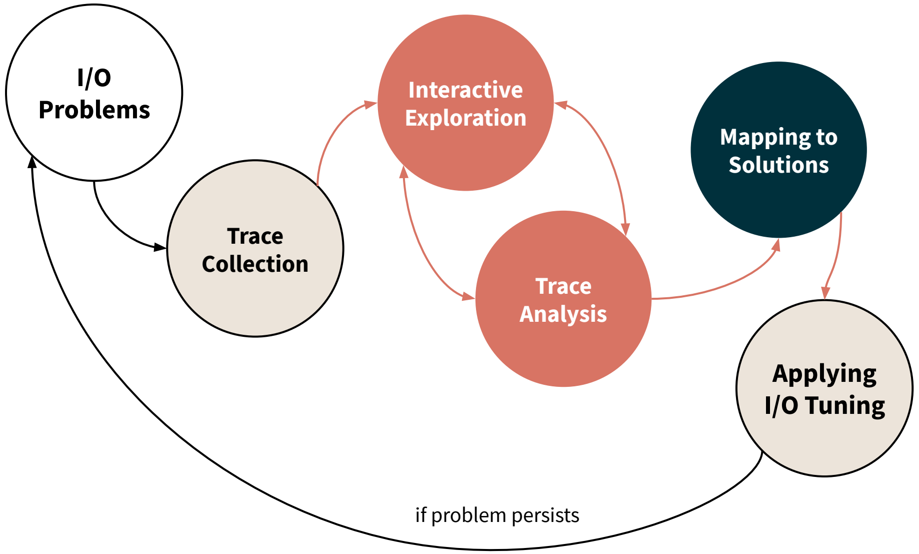


The Missing Dots...

- Lack of knowledge of **available** tunable **options**
- Little **guidance** on **when** to use them
  - In Cori (NERSC) Darshan logs from January 2019 report:
    - 94% of the files used the default 1MB stripe size
    - 36% of the files are striped over a single storage server
  - Collective buffering and data sieving (>20 years ago)
    - Aggregators, placement, and matching to the concurrency at the PFS
- Between a performance bottleneck and its tuning solution, there remain **dots to be connected**...



Interactive Optimization Approach



- Lack of knowledge of **available** tunable **options**
- Little **guidance** on **when** to use them
  - In Cori (NERSC) Darshan logs from January 2019 report:
    - 94% of the files used the default 1MB stripe size
    - 36% of the files are striped over a single storage server
  - Collective buffering and data sieving (>20 years ago)
    - Aggregators, placement, and matching to the concurrency at the PFS
- Between a performance bottleneck and its tuning solution, there remain **dots to be connected**...

The DXT Explorer Tool

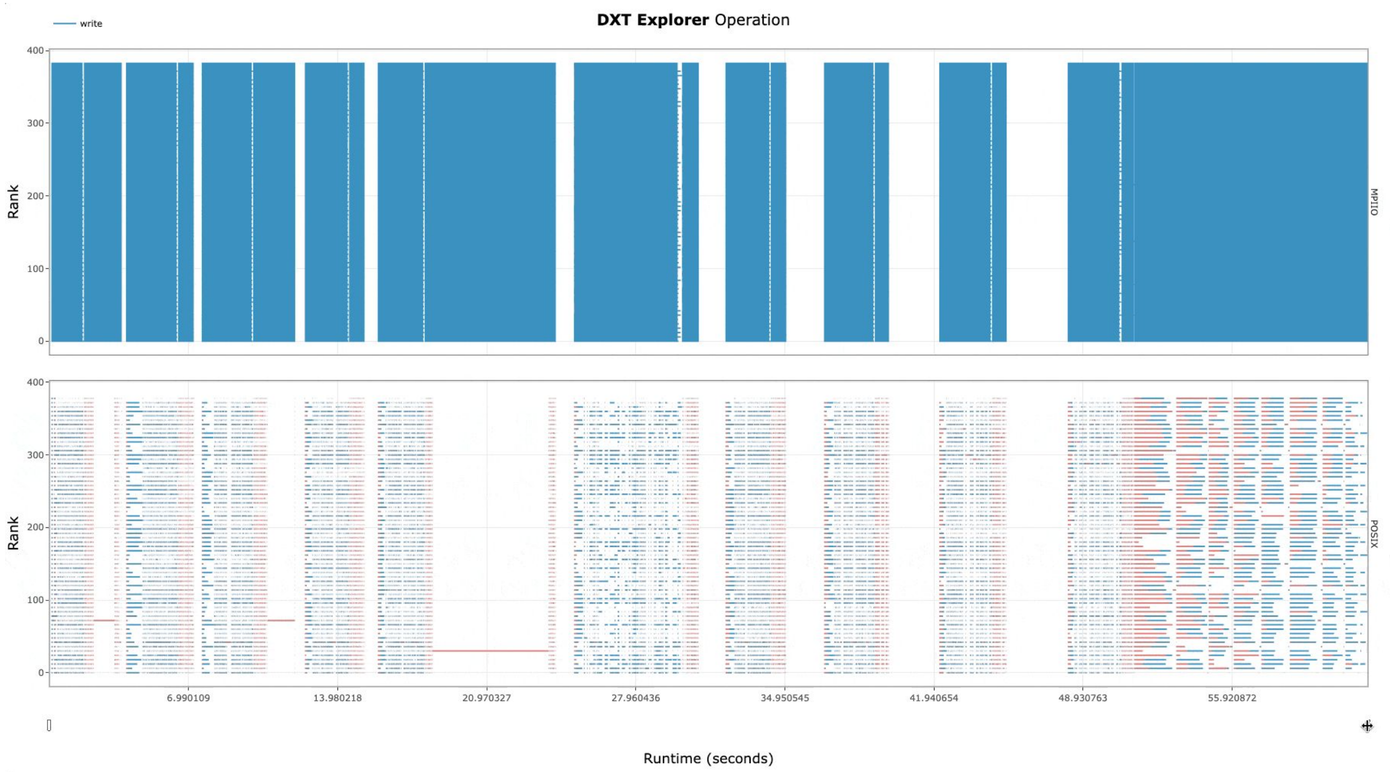
- Darshan can collect fine grain traces with **DXT**
  - **No tool** to visualize and **explore** yet
  - Static plots have **limitations**
- **Features** we seek:
  - Observe POSIX and MPI-IO together
  - Zoom-in/zoom-out in time and subset of ranks
  - Contextual information about I/O calls
  - Focus on operation, size, or spatiality
- By visualizing the application behavior, we are **one step closer** to optimize the application
- There is still a lack of translation from I/O bottlenecks to optimizations

github.com/hpc-io/dxt-explorer

docker pull hpcio/dxt-explorer

DXT Explorer

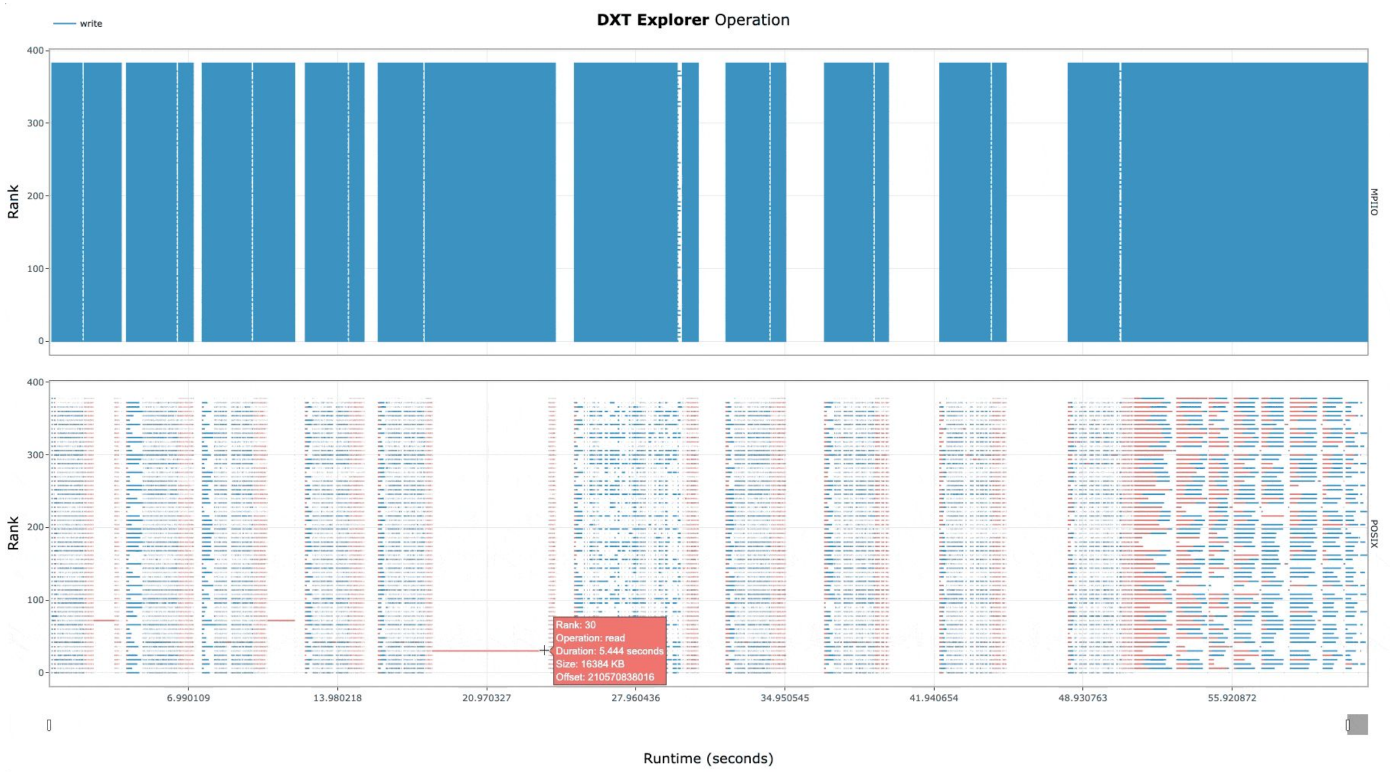
Interactive Features



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.

DXT Explorer

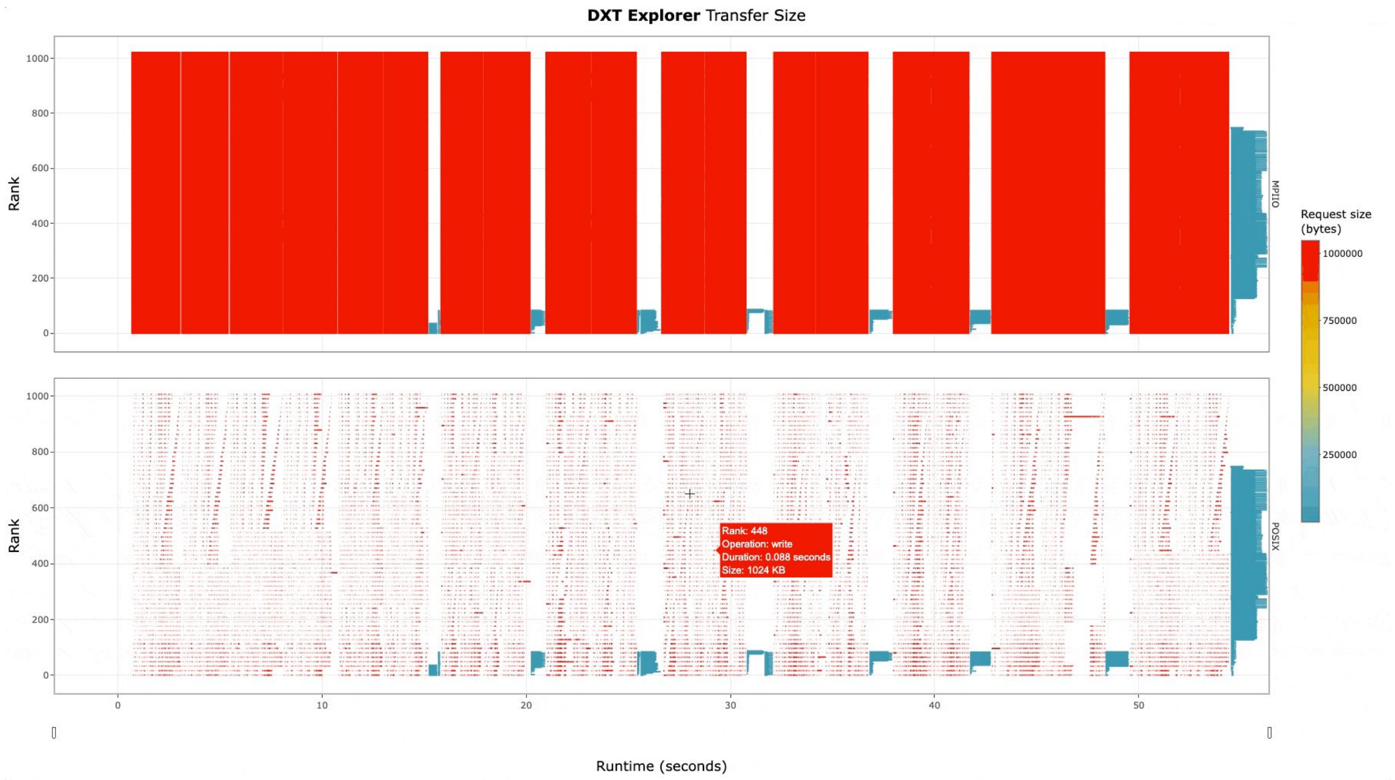
Interactive Features



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.

DXT Explorer

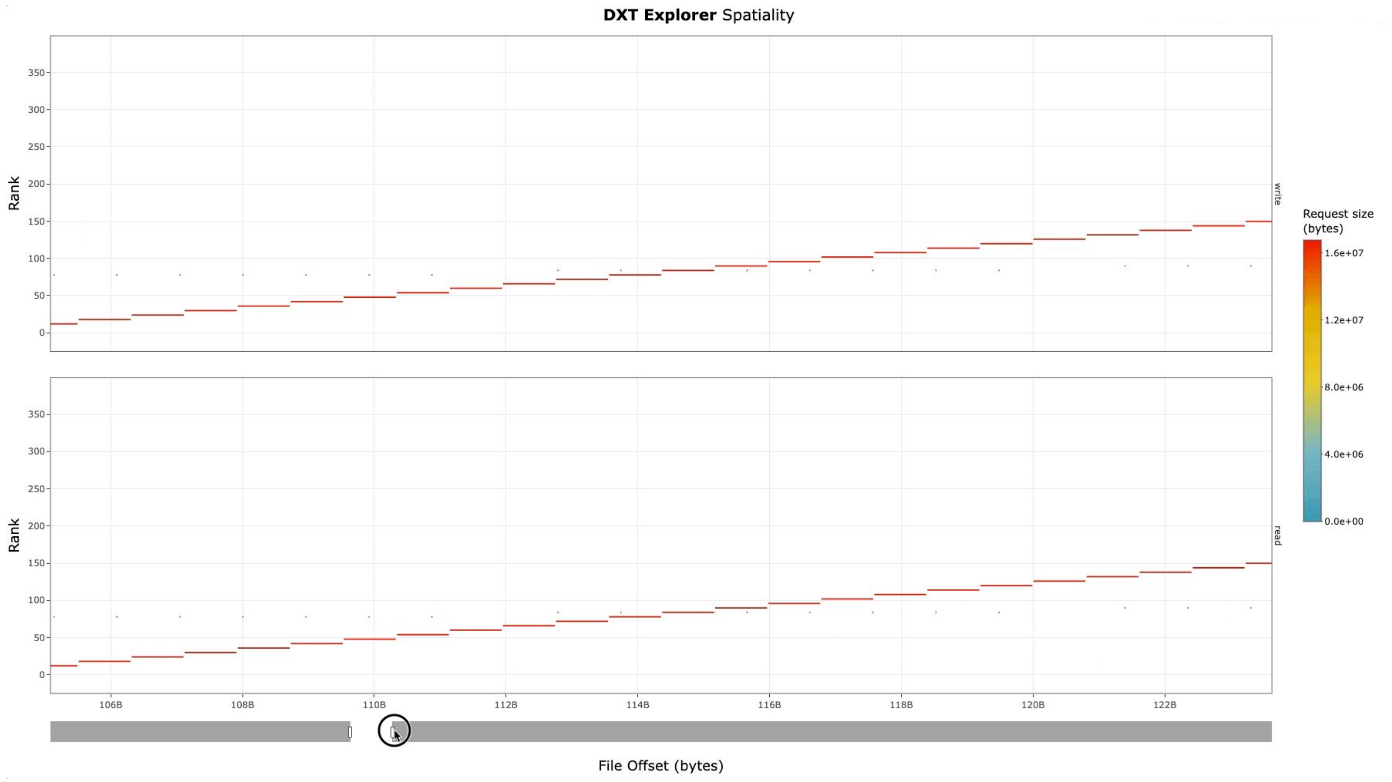
Interactive Features



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

DXT Explorer

Interactive Features



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

DXT Explorer in Practice

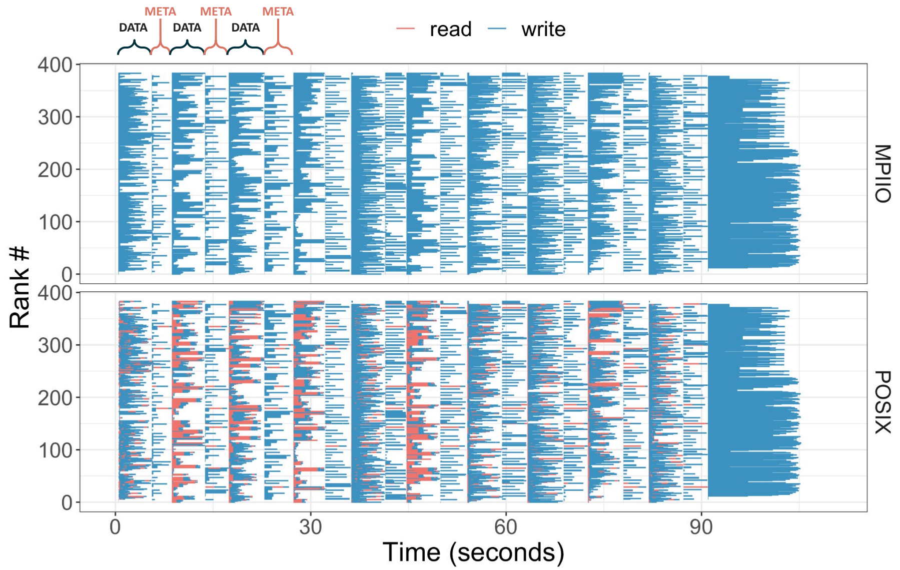
- **Summit** (Oak Ridge) and **Cori** (NERSC) supercomputers
- **Four** application **kernels**, best of five repetitions

I/O Kernel	Context	Summit (OLCF)		Cori (NERSC)	
		Baseline (s)	Optimized (s)	Baseline (s)	Optimized (s)
OpenPMD	Particle and mesh based data	110.6		54.8	
E2E benchmarks	Domain decomposition	15.9		80.0	
Block-cyclic I/O	Linear algebra	-		> 8h	
FLASH-IO	Astrophysics	1495.0		-	

OpenPMD

Baseline

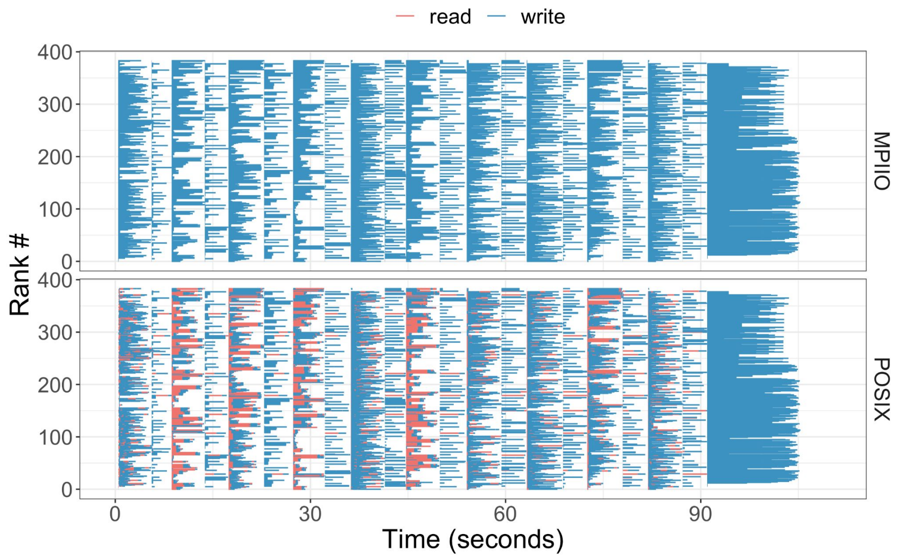
- **Summit** with 64 compute nodes, 6 ranks per node, and a total of 384 MPI ranks
  - Mesh size is [65536 × 256 × 256], 10 iterations, total file size is ≈121GB

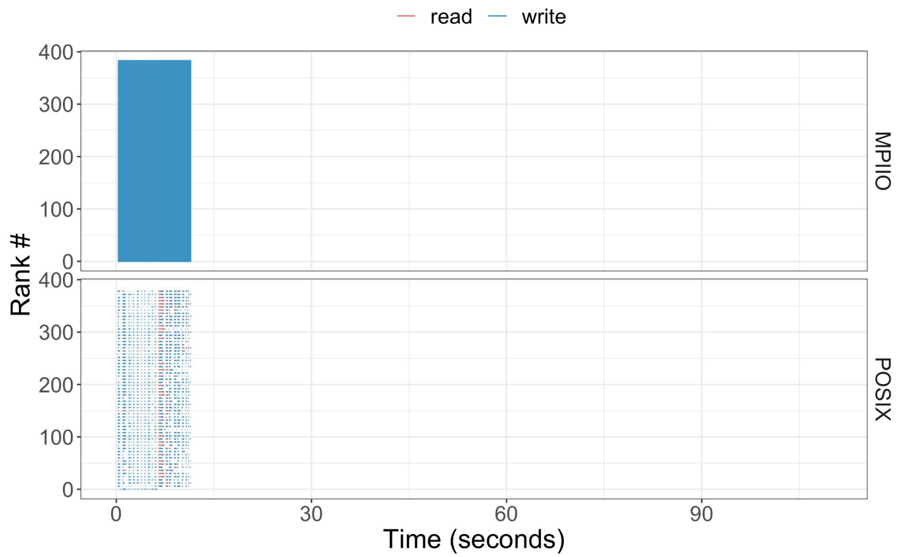


OpenPMD

Optimized

- Collective I/O using **ROMIO** hints with 1 agg/node and 16 MB collective **buffer size** provides **1.54x** speedup
- GPFS **large block** I/O with **HDF5 collective metadata** gives additional **3.8x** speedup
- Collective HDF5 **metadata** were **not actually collective** due to an issue introduced in HDF5 1.10.5
- With **HDF5 1.10.4** combined with previous optimizations gives a total of **6.8x** speedup from baseline





BASELINE

110.6s

6.8x

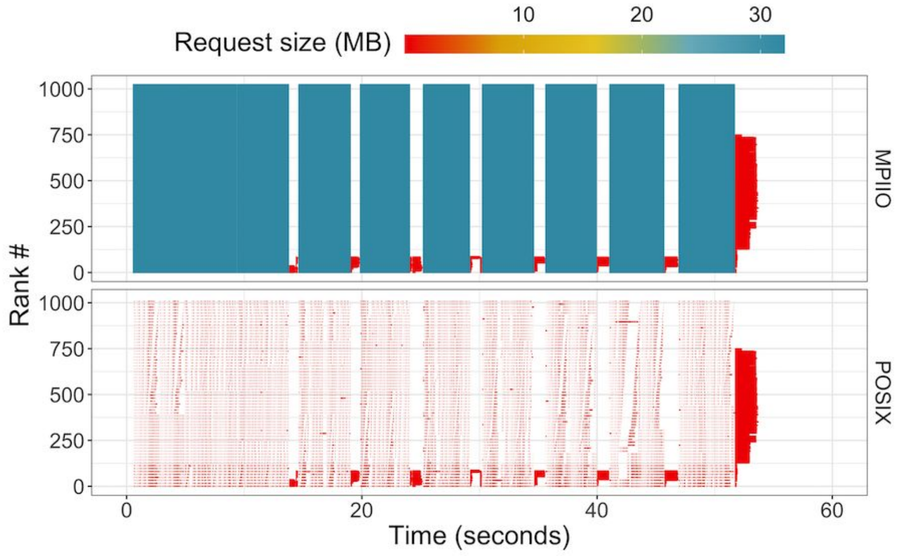
OPTIMIZED

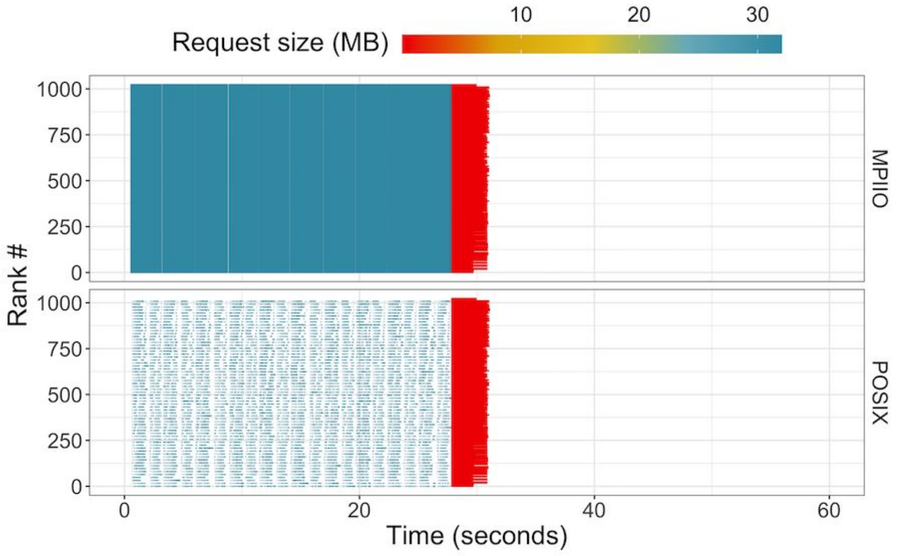
16.1s

OpenPMD

Optimized

- **Cori** with 64 compute nodes, 16 ranks per node, and a total of 1024 MPI ranks
  - Mesh size is [65536 × 256 × 256], 10 iterations, total file size is ≈320GB
- **Collective** HDF5 **metadata** were **not actually collective** due to an issue introduced in HDF5 1.10.5





BASELINE
54.8s

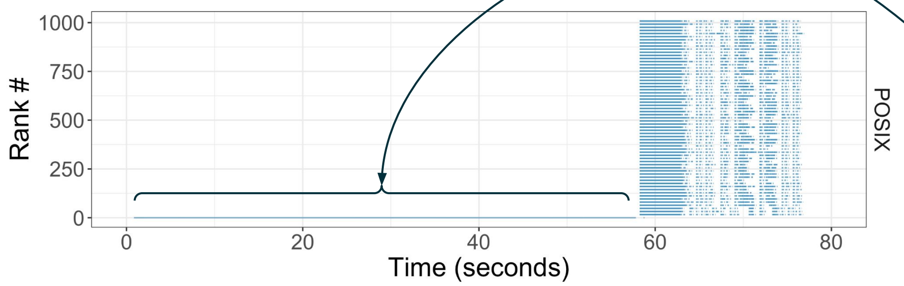
43.8%

OPTIMIZED
30.8s

E2E Benchmarks

Baseline

- **Cori** with 64 compute nodes, 6 ranks per node, and a total of 1024 MPI ranks
  - 1024 processes arranged in a 32 x 32 x 16 distribution, total file size is ≈41GB
- **44%** of the time is taken by rank 0!

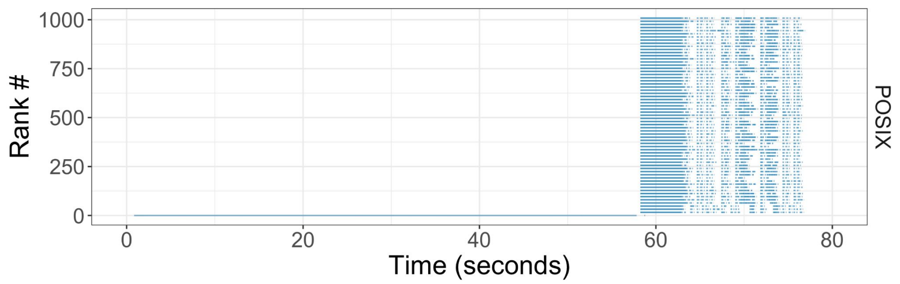


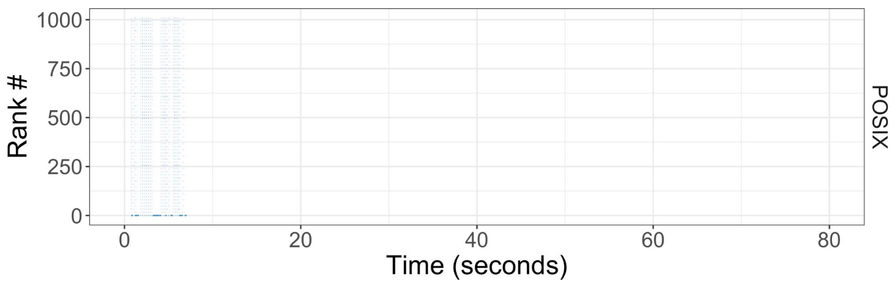
Rank 0 is **sequentially writing fill values** to all of the defined variables (10 in this workload), issuing over 40 thousand write requests with of ≈1MB

E2E Benchmarks

Optimized

- **Cori** with 64 compute nodes, 6 ranks per node, and a total of 1024 MPI ranks
    - 1024 processes arranged in a 32 x 32 x 16 distribution, total file size is ≈41GB
- **44%** of the time is taken by rank 0!
- **Disabling** the data filling (`NC_NOFILL` in NetCDF) translates to **7.3x** speedup





BASELINE

80s

10x

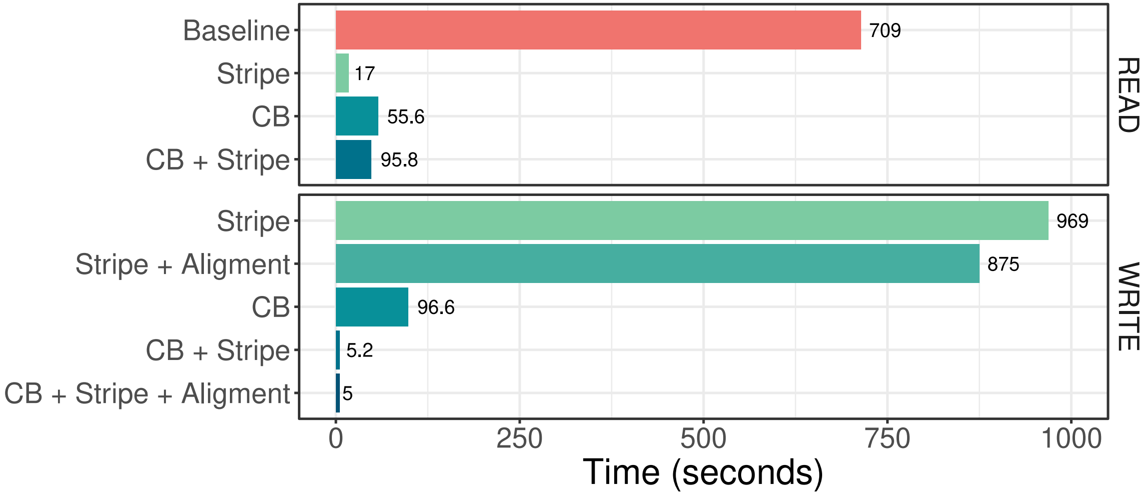
OPTIMIZED

8s

Block-cyclic I/O

Baseline

- **Cori** with 32 compute nodes, 32 ranks per node, and a total of 1024 MPI ranks
  - Square matrix with 81250 x 81250 with FP64 data, total of ≈50GB
  - **Block-cyclic** data structures with 128 x 128 with 1024 processes arranged in a 32 x 32 process grid
- Lustre striping, MPI-IO collective buffering, and HDF5 alignment **optimizations**

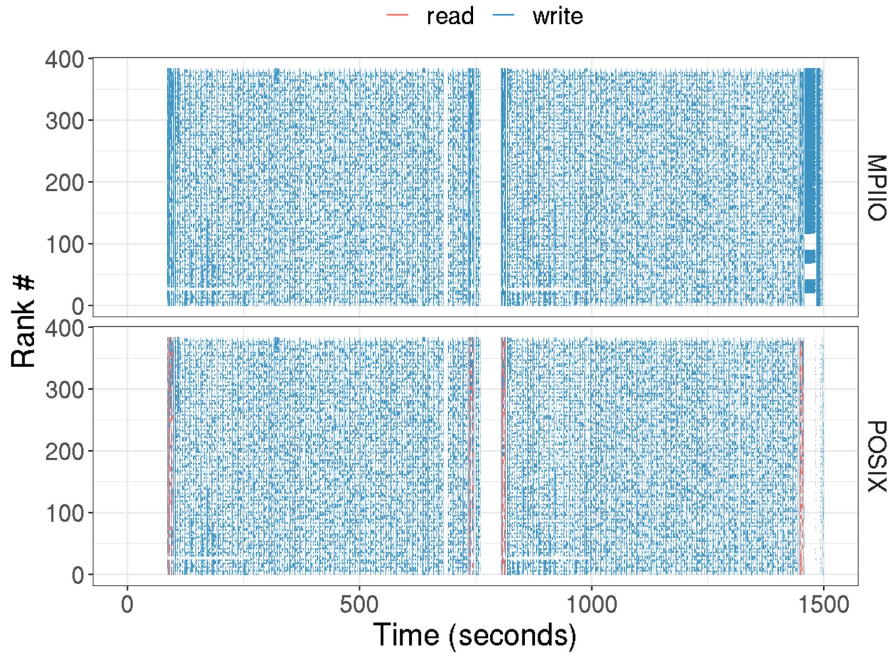


BASELINE	BASELINE
709s	969s ~~8h~~
READ 41x	WRITE 193x
OPTIMIZED	OPTIMIZED
17s	5s

FLASH

Baseline

- **Summit** with 64 compute nodes, 6 ranks per node, and a total of 384 MPI ranks
  - 2 checkpoint files (≈2.3TB each) and 2 plot file (≈14GB each) both using HDF5 backend
- MPI **not** issuing **collective I/O** operations

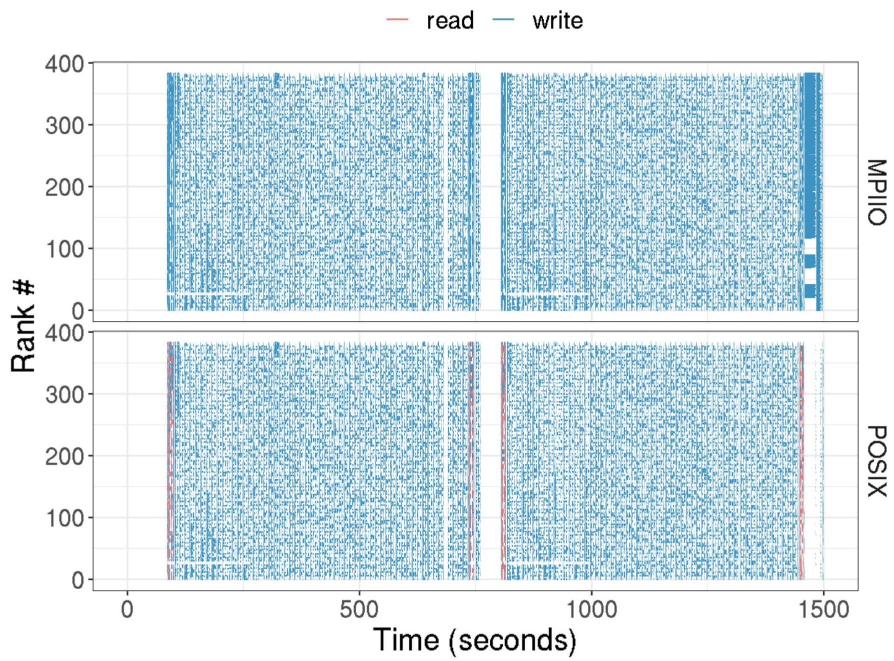


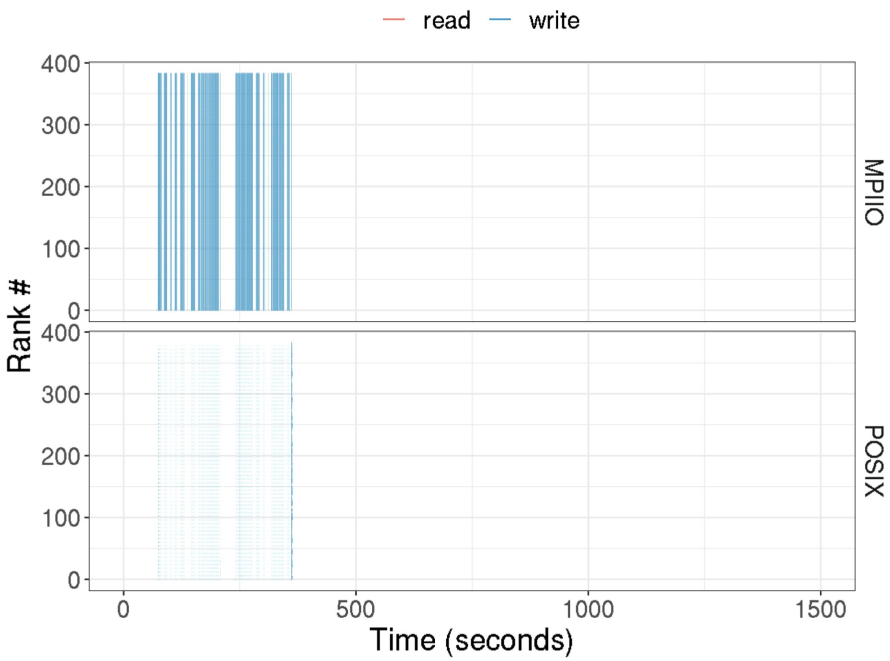
Looking at the **MPI-IO** and **POSIX** levels, each rank is writing its own data

FLASH

Optimized

- Collective I/O using **ROMIO** hints with 1 agg/node and 16 MB collective **buffer size** provides **3.2x** speedup
- Setting the HDF5 **alignment** size to 16 MB provides an additional **1.18**x speedup
- **Deferring** the HDF5 metadata flush provides another **1.1x** speedup





BASELINE

1495s

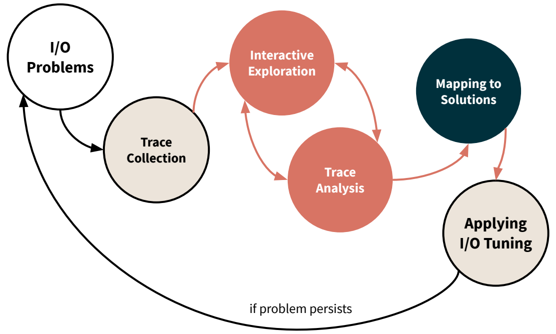
4.1x

OPTIMIZED

361s

Conclusion

- We targeted the gaps between data collection and tuning
  - Seek to identifying bottlenecks and re-shape the I/O behavior
- **DXT Explorer** tool to interactively visualize the I/O behavior
- Case study with four application kernels in two supercomputers



I/O Kernel	Context	Summit (OLCF)		Cori (NERSC)	
		Baseline (s)	Optimized (s)	Baseline (s)	Optimized (s)
OpenPMD	Particle and mesh based data	110.6	16.1	54.8	30.8
E2E benchmarks	Domain decomposition	15.9	1.9	80.0	5.0
Block-cyclic I/O	Linear algebra	-	-	> 8h	22.0
FLASH-IO	Astrophysics	1495.0	361.0	-	-

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
- There is still the need for **further R&D**
  - Tools to better report findings to end-users
  - Automatically mapping performance problems to tuning options, e.g. recommendations

docker pull hpcio/dxt-explorer

github.com/hpc-io/dxt-explorer

jeanbez.gitlab.io/pdsw-2021 (Companion Repository)

You can reach us by email:

jlbez@lbl.gov

docker pull hpcio/dxt-explorer

github.com/hpc-io/dxt-explorer

jeanbez.gitlab.io/pdsw-2021 (Companion Repository)

SC21 St. Louis, MO | science & beyond.

ECP EXASCALE COMPUTING PROJECT

BERKELEY LAB Bringing Science Solutions to the World

OAK RIDGE National Laboratory

Argonne NATIONAL LABORATORY