Exploring I/O Traces with DXT Explorer

Jean Luca Bez
Lawrence Berkeley National Laboratory

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

HDF5 USERS GROUP 2021
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)

```bash
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
- How to visualize and extract insights DXT data?
  - Identify I/O bottlenecks
  - Hint which optimizations we should apply
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
E2E Benchmarks

Baseline

- **Cori** with 64 compute nodes, 16 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 $\approx 41$GB
- **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 $\approx 1$MB
E2E Benchmarks
Optimized

- **Cori** with 64 compute nodes, 16 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 **10x** speedup
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 \(\approx 50\)GB
  - **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**

```
Base Line | Stripe | CB | CB + Stripe | Stripe | Stripe + Aligment | CB | CB + Stripe | CB + Stripe + Aligment
---|---|---|---|---|---|---|---|---
READ | 709s | 17 | 55.6 | 95.8 | 875 | 969 | 5.2 | 5
WRITE | 969s | 969 | 969 | 969 | 969 | 969 | 969 | 969
```

**BASELINE**
- READ: 709s
- WRITE: 969s **8h**

**OPTIMIZED**
- READ: 17s **41x**
- WRITE: 5s **193x**
**FLASH-IO**

**Baseline**

- **Summit** with 64 compute nodes, 6 ranks per node, and a total of 384 MPI ranks
  - 2 checkpoint files ($\approx 2.3$TB each) and 2 plot file ($\approx 14$GB 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-IO

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 1.18x speedup.
- Deferring the HDF5 metadata flush provides another 1.1x speedup.

![Graphs showing I/O performance comparison between baseline and optimized configurations.](image)
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
  - How can we better report findings to end-users?
  - How can we automatically map performance problems to tuning options?
  - How can we provide recommendations?

```
docker pull hpcio/dxt-explorer
```

```
github.com/hpc-io/dxt-explorer
```
Exploring I/O Traces with DXT Explorer

Jean Luca Bez
jlbez@lbl.gov

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

HDF5 USERS GROUP 2021