SC20

Everywhere we are | more than hpc.

GPU Direct IO with HDF5

John Ravi • Quincey Koziol • Suren Byna

Motivation

- With large-scale computing systems are moving towards using GPUs as workhorses of computing

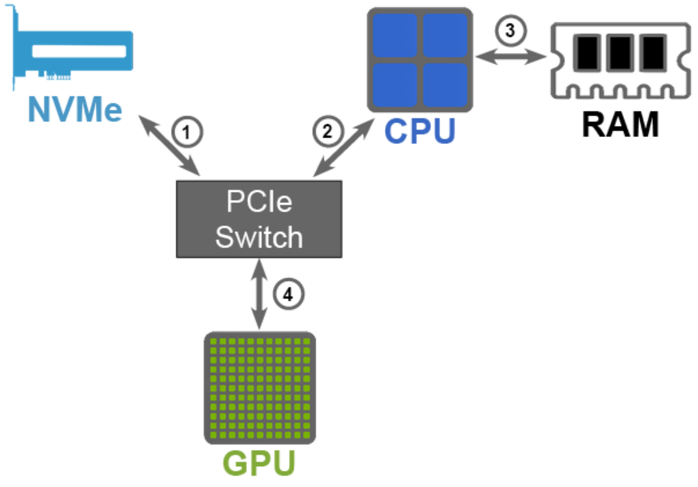
- file I/O to move data between GPUs and storage devices becomes critical

- I/O performance optimizing technologies
    - NVIDIA’s GPU Direct Storage (GDS) - reducing the latency of data movement between GPUs and storage.

- In this presentation, we will talk about a recently developed virtual file driver (VFD) that takes advantage of the GDS technology allowing data transfers between GPUs and storage without using CPU memory as a “bounce buffer”

Traditional Data Transfer without GPUDirect Storage

1. fd = open(“file.txt”, O_RDONLY);
2. buf = malloc(size);
3. pread(fd, buf, size, 0);
4. cudaMalloc(d_buf, size);
5. cudaMemcpy(d_buf, buf, size, cudaMemcpyHostToDevice);

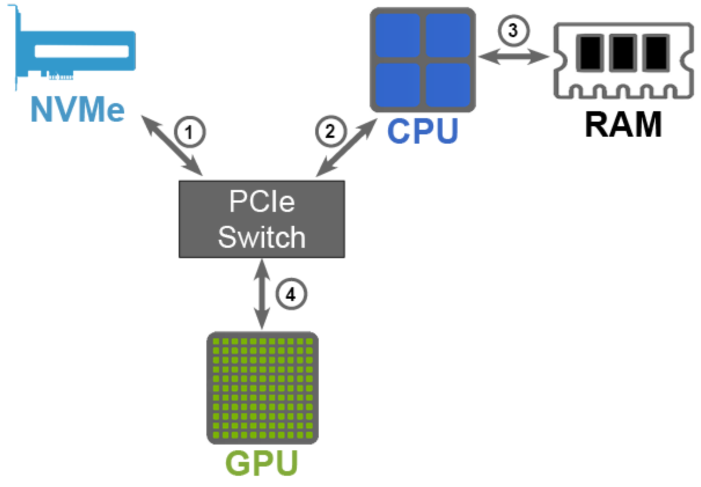


Data Transfer with GPUDirect Storage (GDS)

Traditional Data Transfer

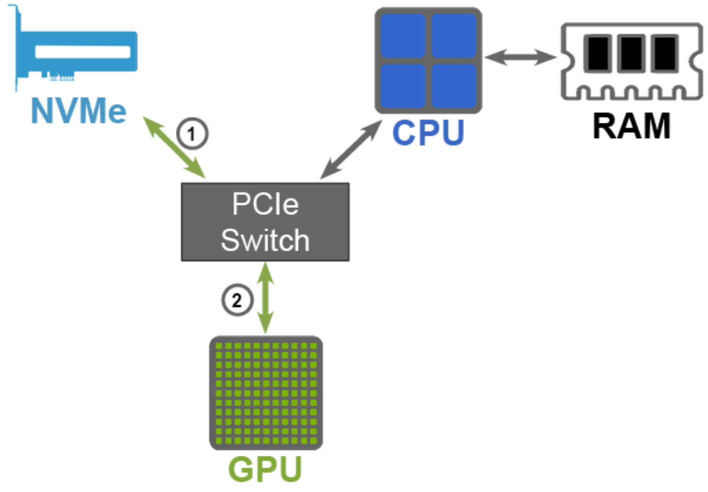
1. fd = open(“file.txt”, O_RDONLY, …);
2. buf = malloc(size);
3. pread(fd, buf, size, 0);
4. cudaMalloc(d_buf, size);
5. cudaMemcpy(d_buf, buf, size, cudaMemcpyHostToDevice);

No need for a “bounce buffer”

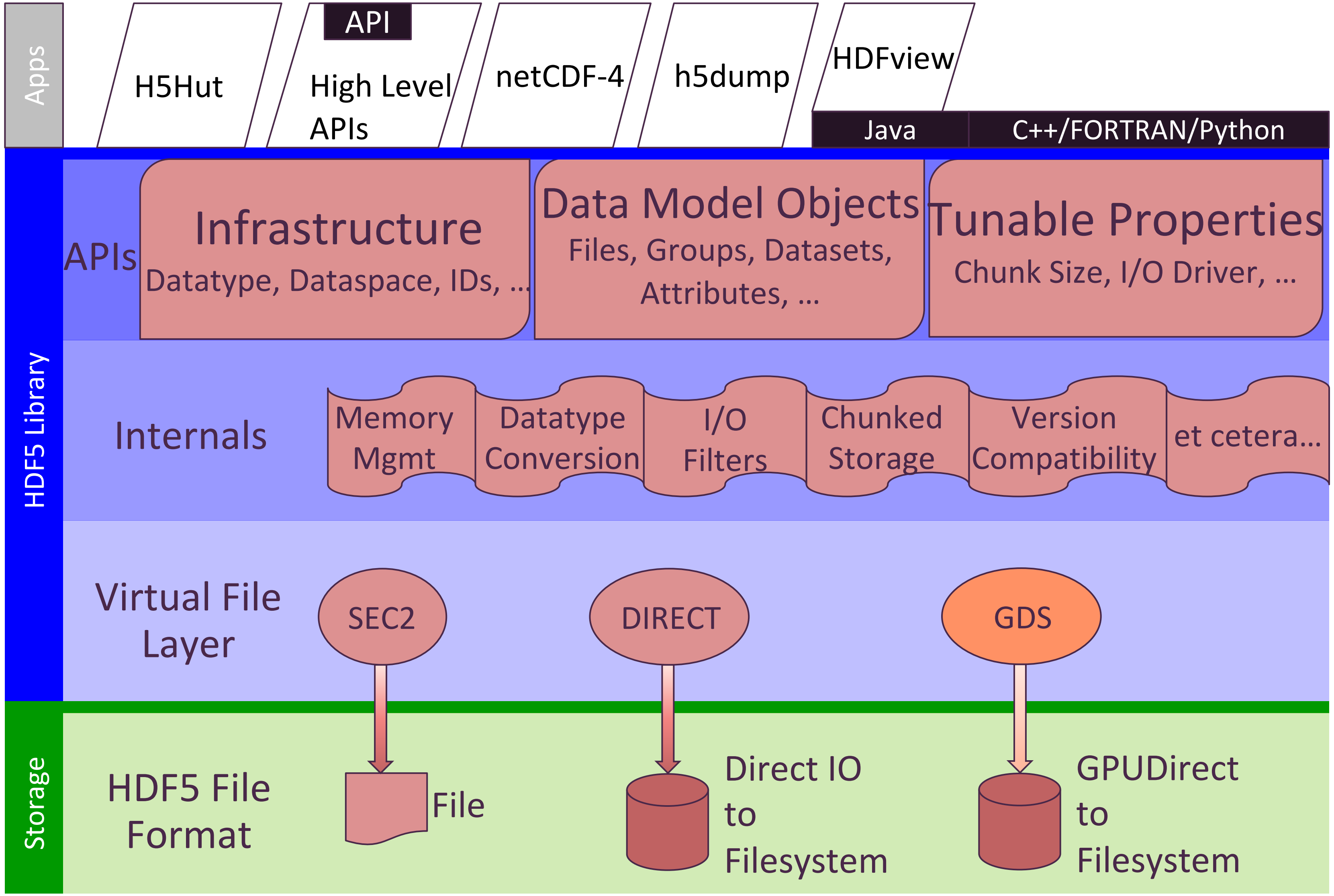


NVIDIA GPUDirect Storage

1. fd = open(“file.txt”, O_RDONLY | O_DIRECT, …);
2. cudaMalloc(d_buf, size);
3. cuFileRead(fhandle, d_buf, size, 0, 0);



HDF5 Virtual File Driver(s)



VFD	Description
SEC2	default driver POSIX file-system functions like read and write to perform I/O to a single file
DIRECT	force data to be written directly to file-system disables OS buffering
GDS	Enable GPUDirect Storage

SC20

HDF5 GDS – Virtual File Driver

- GDS VFD differences from SEC2 VFD
  - File Descriptor is open with O_DIRECT (disables all OS buffering)
  - Read and Write handlers needs to distinguish between CPU (metadata) and GPU memory pointers
  - cuFileDriver needs to be initialized per run
- Some overhead for each I/O call
  - Querying CUDA Runtime for information about memory pointers
  - cuFile buffer registration and deregistration

Experimental Evaluation – Lustre File System

- GDS VFD knobs
  - num_threads – number of pthreads servicing one cuFile request
  - blocksize – transfer size of one cuFile request

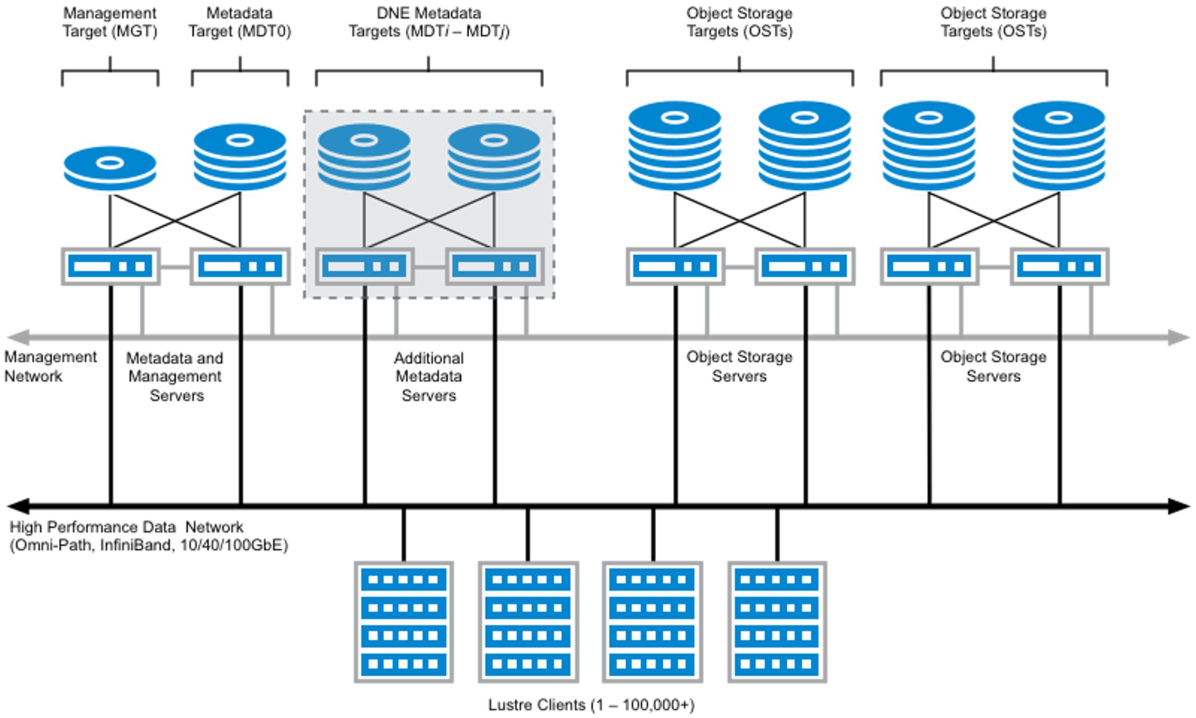


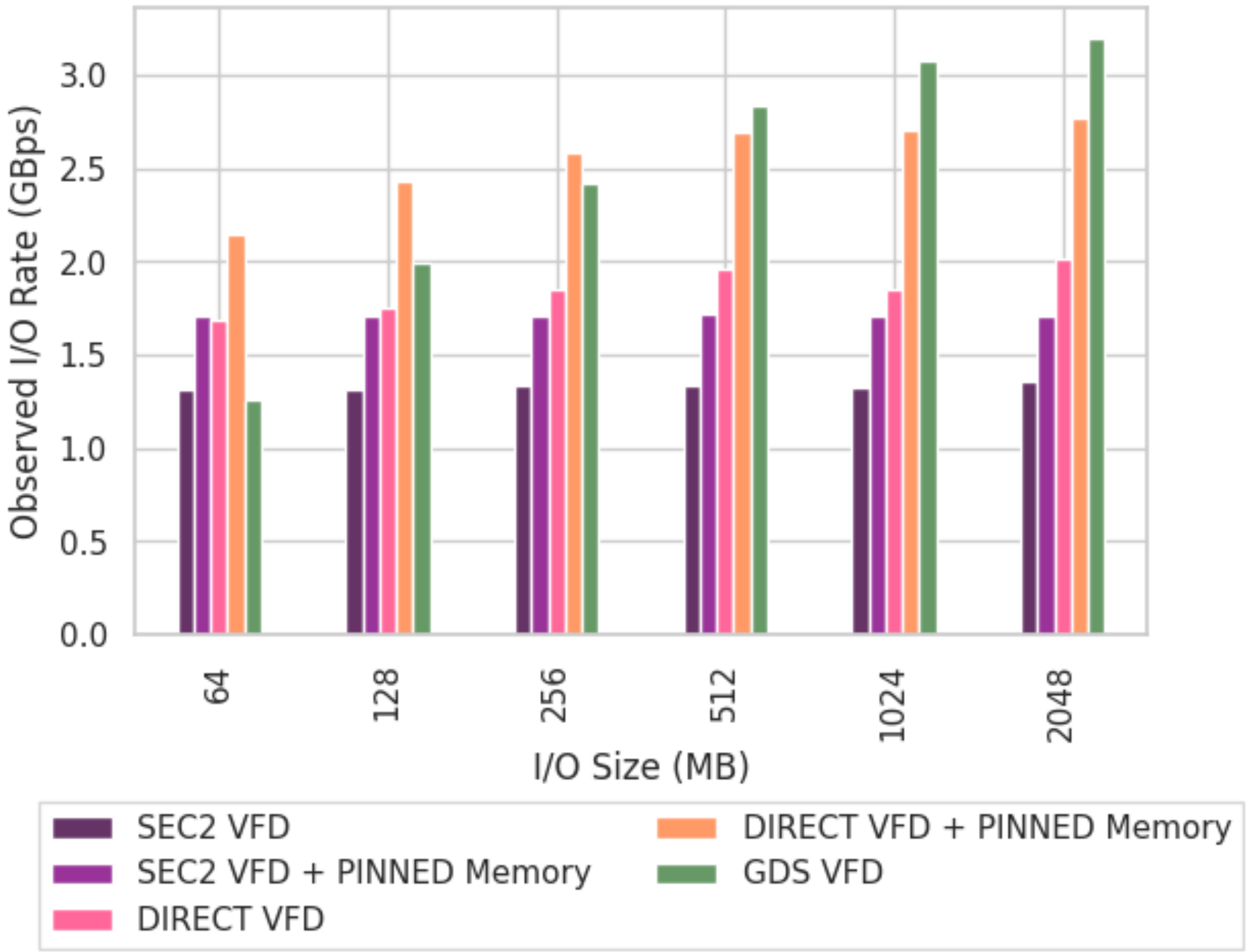
Image Source: https://wiki.lustre.org/Introduction_to_Lustre

Experimental Evaluation

- System Configuration
  - NVIDIA DGX-2
  - 16x Tesla v100
  - 2x Samsung NVMe SM961/PM961 RAID0 (Seq Reads = ~6.4 GB/s, Seq Write = ~3.6 GB/s)
  - Lustre File System (4 OSTs, 1MB strip size)

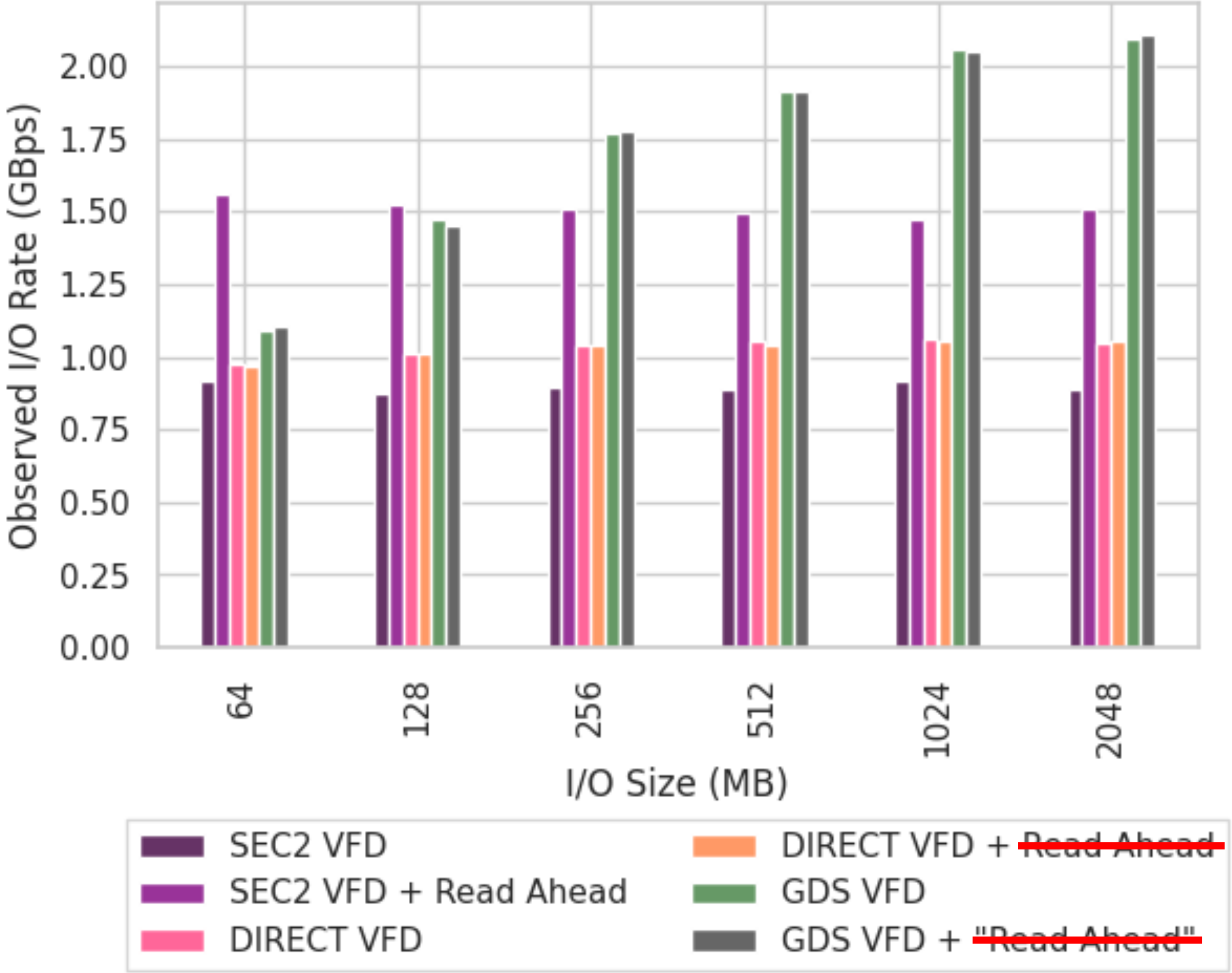
- Benchmarks
  - Local Storage
    - Sequential R/W Rates
  - Lustre File System
    - Multi-threaded Sequential R/W Rates
    - Multi-GPU (one GPU per process, one file per process)

Write Performance – Local Storage



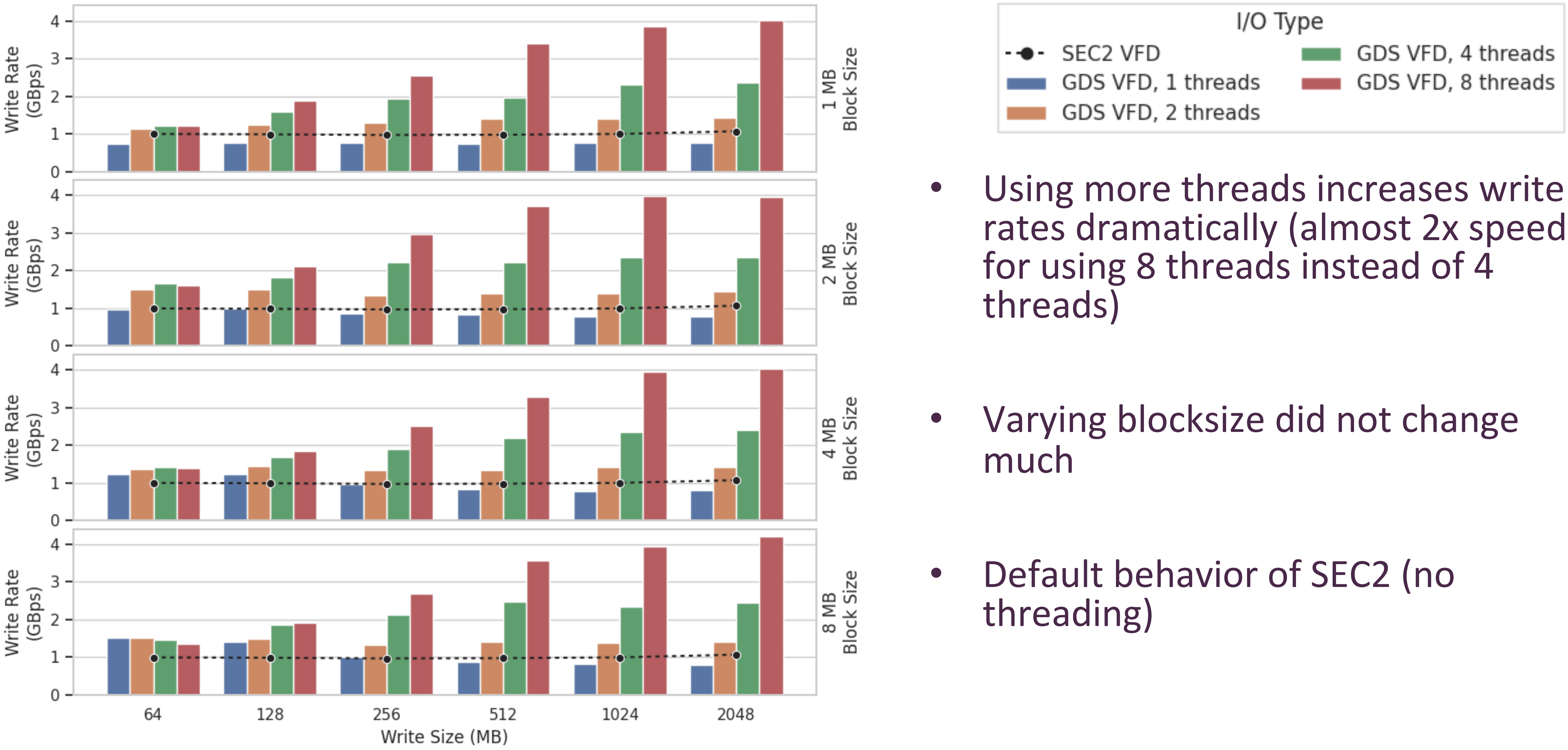
- HDF5 GDS achieves higher write rates for requests greater than 512 MB
- Possible Optimizations:
  - make user specify the location of the memory pointer for each memory transfer
  - cuFile buffer register before I/O call

Read Performance – Local Storage



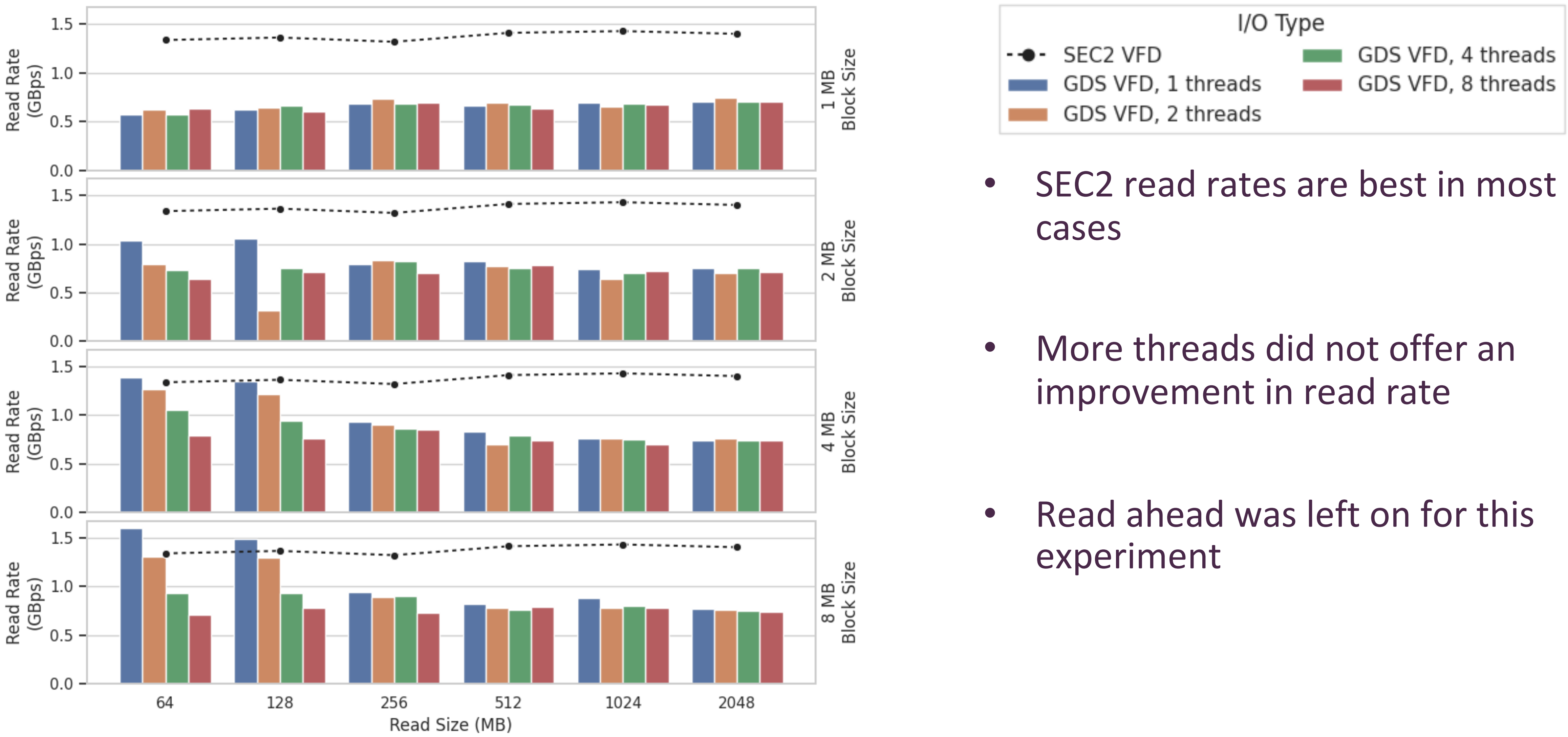
- HDF5 GDS achieves higher read rates for requests greater than 256 MB
- Possible Optimizations:
  - make user specify the location of the memory pointer for each memory transfer
  - cuFile buffer register before I/O call

Multi-Threaded Writes, Single GPU, Lustre File System



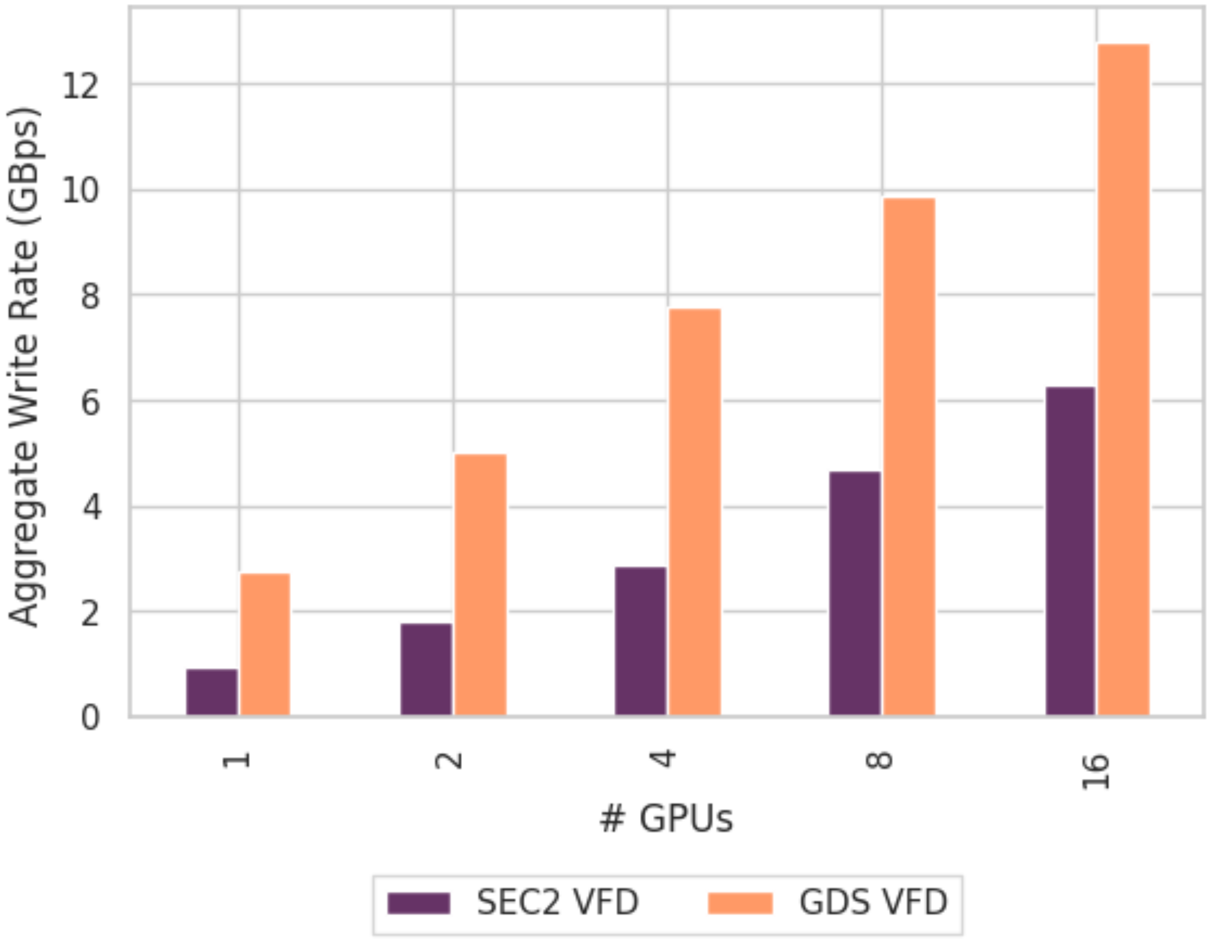
- Using more threads increases write rates dramatically (almost 2x speed for using 8 threads instead of 4 threads)
- Varying blocksize did not change much
- Default behavior of SEC2 (no threading)

Multi-Threaded Read, Single GPU, Lustre File System



- SEC2 read rates are best in most cases
- More threads did not offer an improvement in read rate
- Read ahead was left on for this experiment

Multi-Process Writes, Multiple GPU, Lustre File System



- GDS VFD clear advantage over SEC2 VFD for a distributed file system

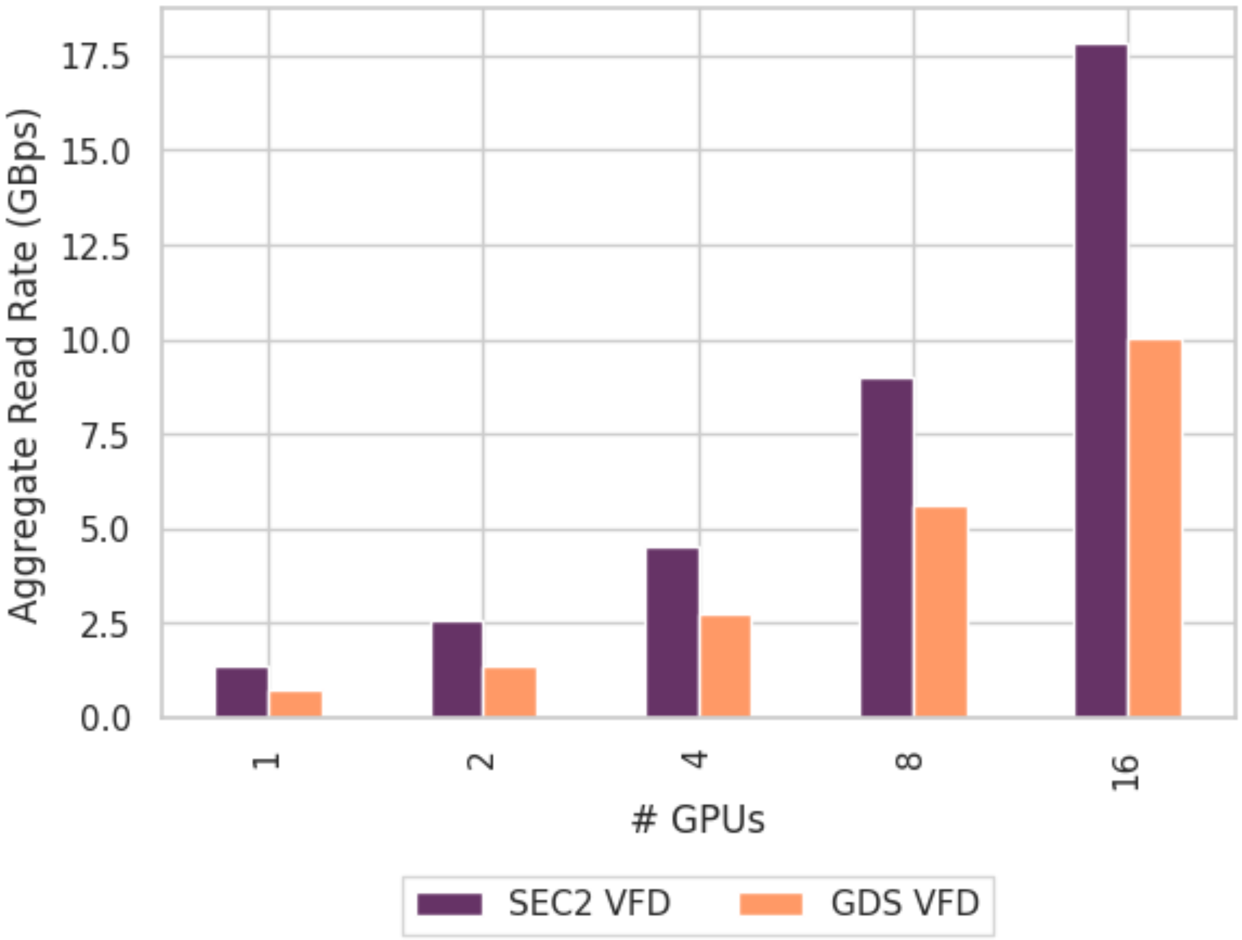
GDS VFD Knobs

- 4 threads (OSTs)
- 1MB blocksize (strip size)

Multi-Process Writes

- Single GPU per MPI Rank
- Single HDF5 file per MPI Rank
- File size: 1GB

Multi-Process Reads, Multiple GPU, Lustre File System



- SEC2 VFD dominates over GDS VFD (read ahead was left enabled)

GDS VFD Knobs

- 4 threads (OSTs)
- 1MB blocksize (strip size)

Multi-Process Reads

- Single GPU per MPI Rank
- Single HDF5 file per MPI Rank
- File size: 1GB

Conclusions

- HDF5 GDS VFD improves the write rates over SEC2 VFD
- HDF5 SEC2 VFD seems to offer higher read rates over GDS VFD mainly because of optimizations at other layers (read ahead)

Future Work

- GDS for Parallel HDF5 – MPIIO VFD
  - MPI-IO developers are working on this
- HDF5 GDS VFD tuning knobs for Distributed File Systems
- Avoiding the overhead
  - Track data buffer locations
  - Track data buffer reuse
  - Async IO

Thank you

EXASCALE COMPUTING PROJECT

National Energy Research Scientific Computing Center

- Contact:

John Ravi jjravi@lbl.gov
Quincey Koziol koziol@lbl.gov
Suren Byna sbyna@lbl.gov