Integration of Burst Buffer in High-level Parallel IO Library for Exa-scale Computing Era

SC 2018 PDSW workshop

Kaiyuan Hou, Reda Al-Bahrani, Esteban Rangel, Ankit Agrawal, Robert Latham, Robert Ross, Alok Choudhary, and Wei-keng Liao

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Overview

- Background & Motivation
- Our idea – aggregation on burst buffer
  - Benefit
  - Challenges
- Summary of results

I/O in The Exa-scale Era

- Huge data size
  - >10PB system memory
  - Data generated by application are in similar magnitude
- I/O speed cannot catch up the increase of data size
  - Parallel File System (PFS) architecture is not scalable
- Burst buffer introduced into I/O hierarchy
  - Made of new hardware such as SSDs, Non-volatile RAM …etc.
  - Tries to bridge the performance gap between computing and I/O
- The role and potential of burst buffer hasn't been fully explored
  - How can burst buffer help on improvement I/O performance

I/O Aggregation Using the Burst Buffer

- PFSs are made of rotating hard disks
  - High capacity, low speed
  - Usually used as main storage on super computer
  - Sequential access is fast while random access is slow
  - Handling large data is more efficient than handling small data
- Burst buffers are made of SSDs or NVMs
  - Higher speed, lower capacity
- I/O aggregation on burst buffer
  - Gather write requests on the burst buffer
  - Reorder requests into sequential
  - Combine all requests into one large request

Related Work

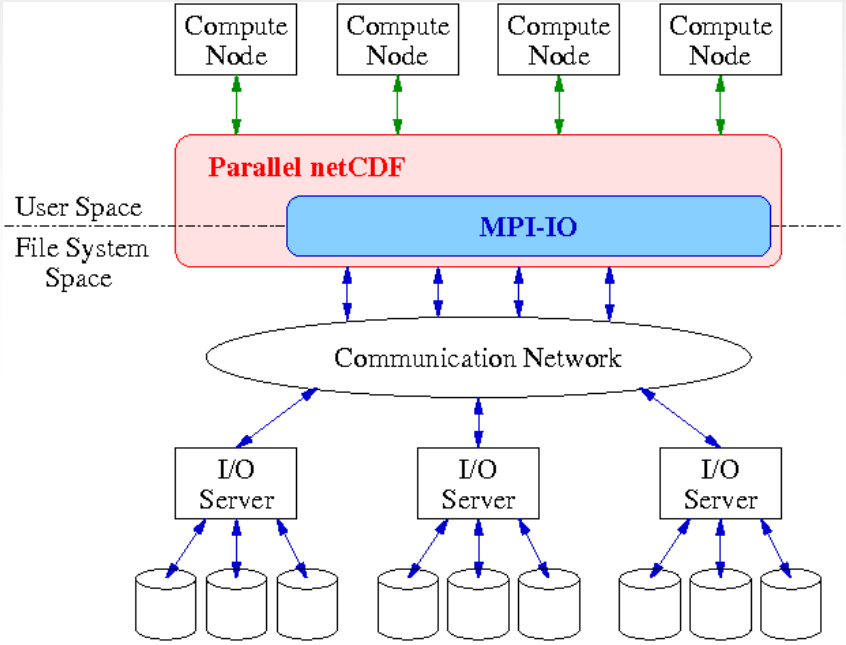
- LogFS [1]
  - I/O aggregation library using low-level offset and length data representation
    - Simpler implementation
    - Does not preserve the structure of the data
  - Log-based data structure for recording write operations
- Data Elevator [2]
  - A user level library to move buffered files on the burst buffer to PFS
  - File is written to the burst buffer as is and copied to the PFS later
    - Does not alter I/O pattern on the burst buffer
  - Work only on shared burst buffer
  - Faster than moving the file using system functions on large scale
    - When number of nodes larger than number of burst buffer servers

[1] D. Kimpe, R. Ross, S. Vandewalle and S. Poedts, "Transparent log-based data storage in MPI-IO applications," in Proceedings of the 14th European conference on Recent Advances in Parallel Virtual Machine and Message Passing Interface , Paris, 2007.

[2] Dong, Bin, et al. "Data Elevator: Low-Contention Data Movement in Hierarchical Storage System." *High Performance Computing (HiPC), 2016 IEEE 23rd International Conference on*. IEEE, 2016.

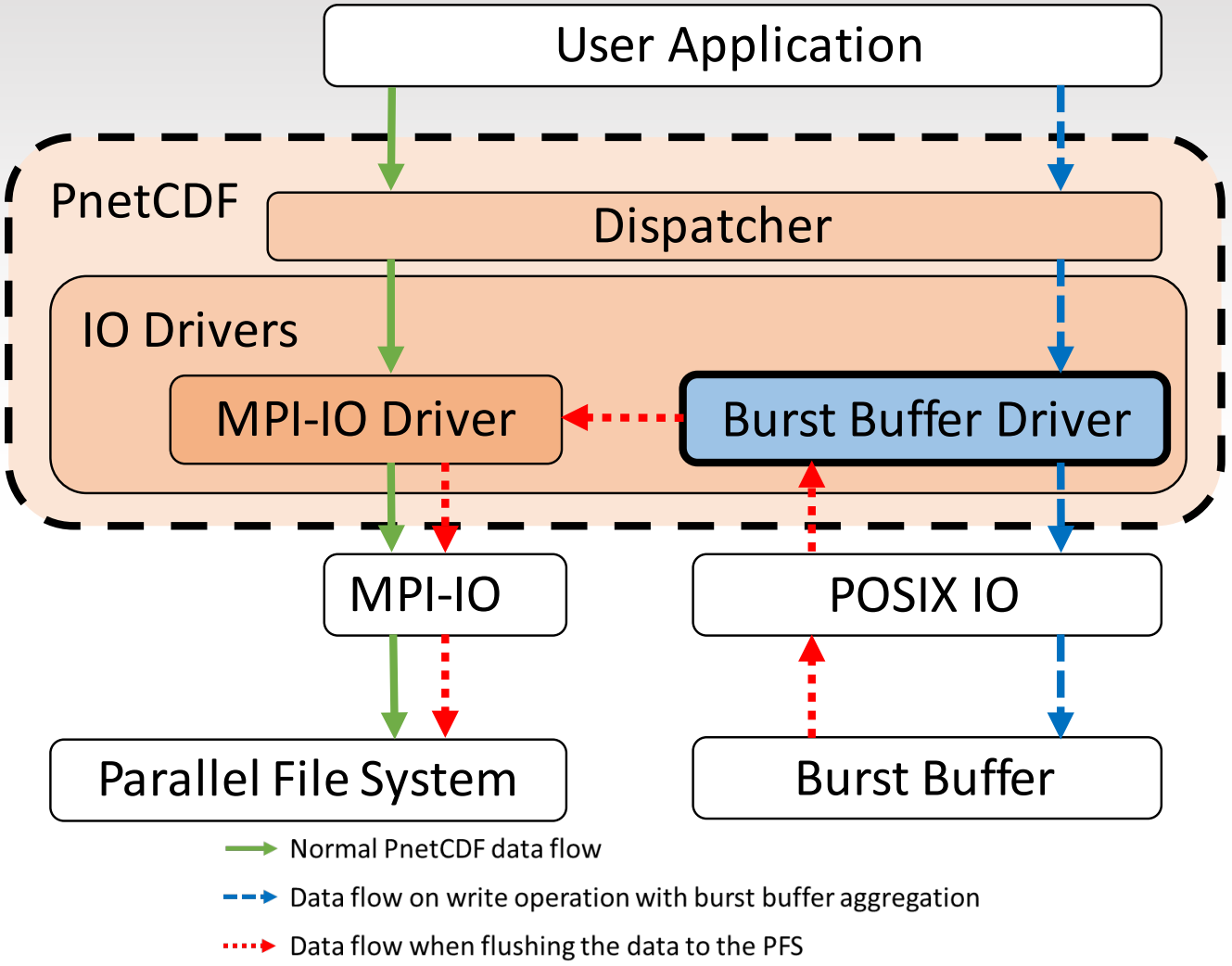
About PnetCDF

- High-level I/O library
  - Built on top of MPI-IO
  - Abstract data description
- Enable parallel access to NetCDF formatted file
- Consists of I/O modules called drivers that deal with lower level libraries
- https://github.com/Parallel-NetCDF/PnetCDF

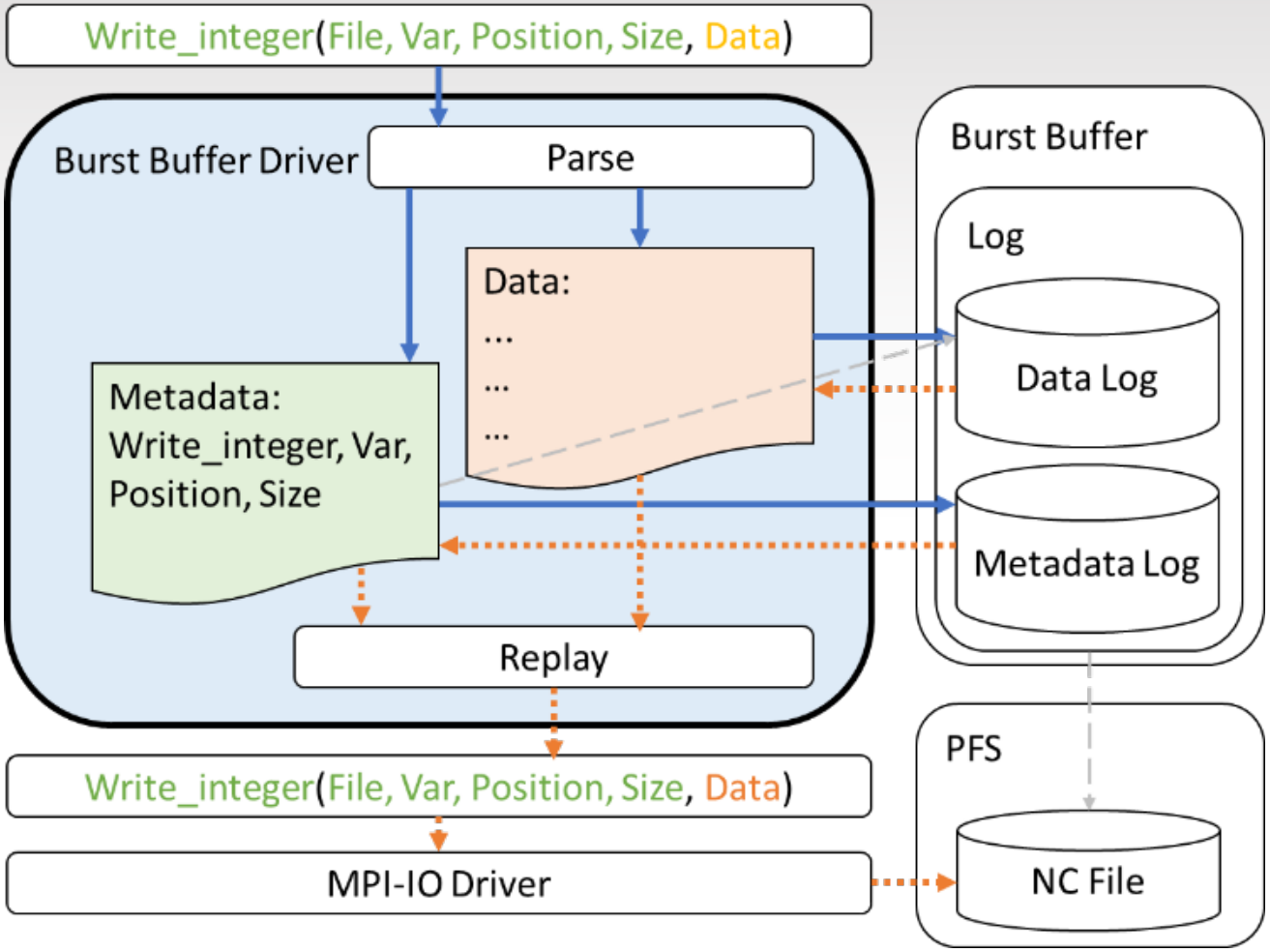


Picture courtesy from:
Li, Jianwei et al. "Parallel netCDF: A High-Performance Scientific I/O Interface." *ACM/IEEE SC 2003 Conference (SC'03)* (2003): 39-39.

I/O Aggregation in PnetCDF



Recording Write Requests

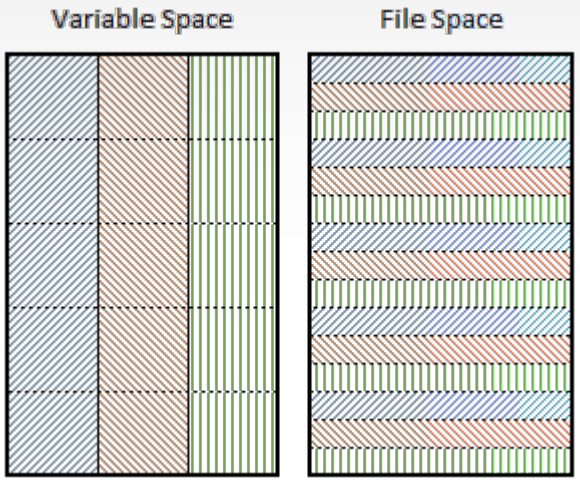


Compared to Lower-level Approach

- Retain the structure of original data
  - Most scientific data are sub-array of high-dimensional arrays
  - Performance optimization
  - Can be used to support other operations such as in-situ analysis
- Lower memory footprint
  - One high-level request can translate to multiple offsets and lengths
- More complex operations to record
  - Not as simple as offset and length
- Must follow the constraint of lower-level library
  - Less freedom to manipulate raw data

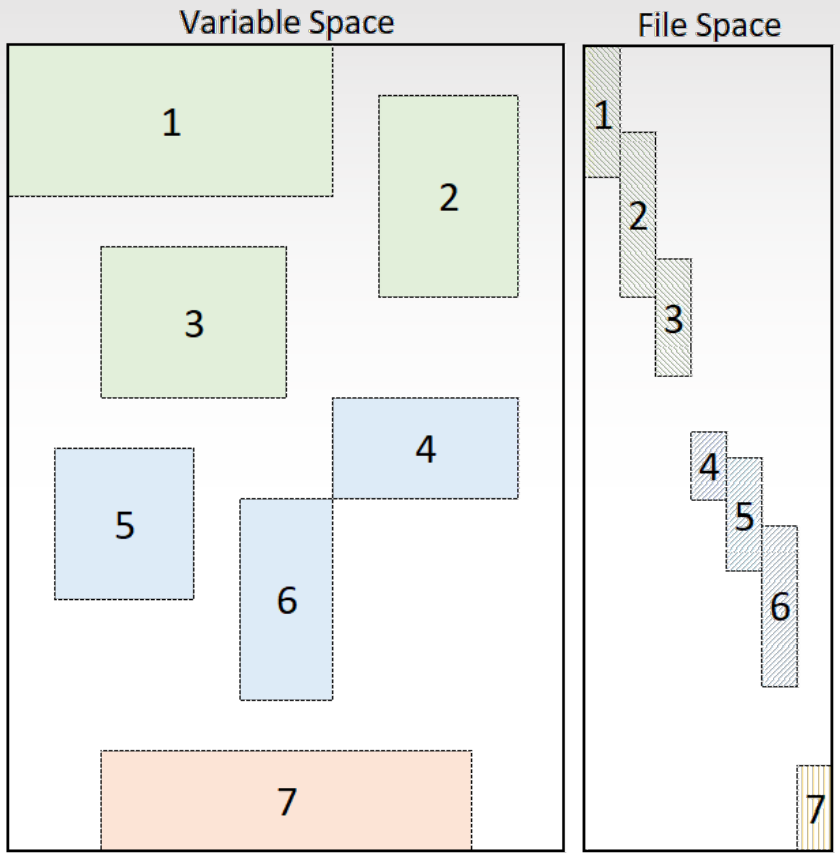
Generating Aggregated Request

- Limitation of MPI-IO
  - Flattened offset of a MPI write call must be monotonically non-decreasing
- Can not simply stacking high-level requests together
  - May violate the requirement
- Offsets must be sorted in order
  - Performance issue on large data



2-stage Reordering Strategy

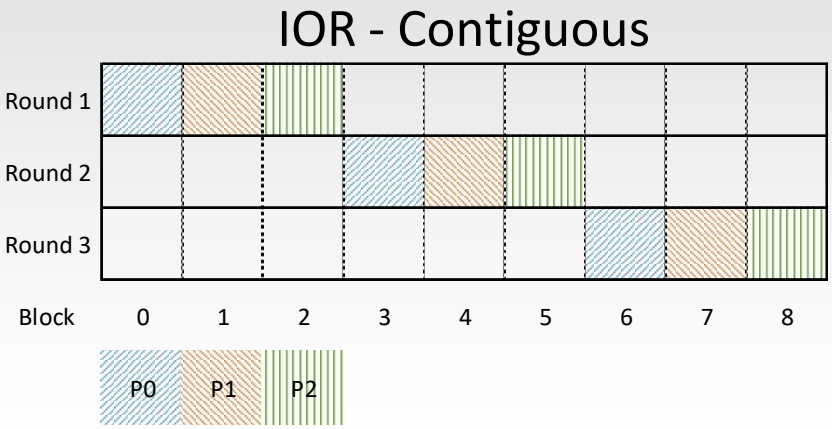
- Group the requests
  - Requests from different group will never interleave each other
  - Requests within a group interleaves each other
- Perform sorting on groups
  - Without broken up request to offsets
- Perform sorting within group
  - Break up requests

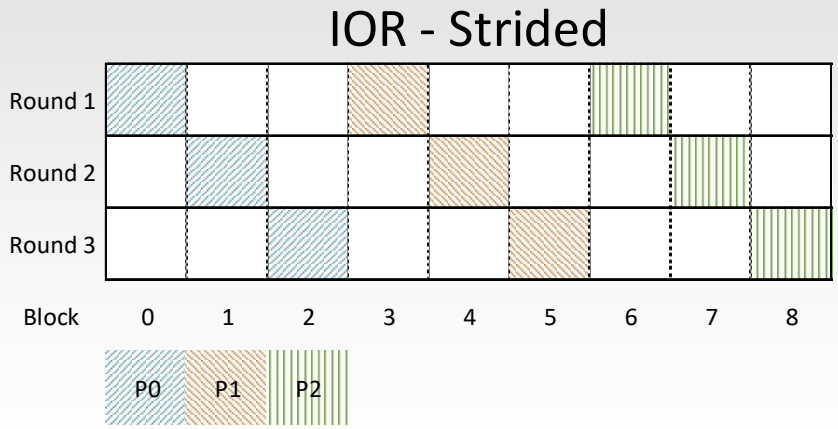


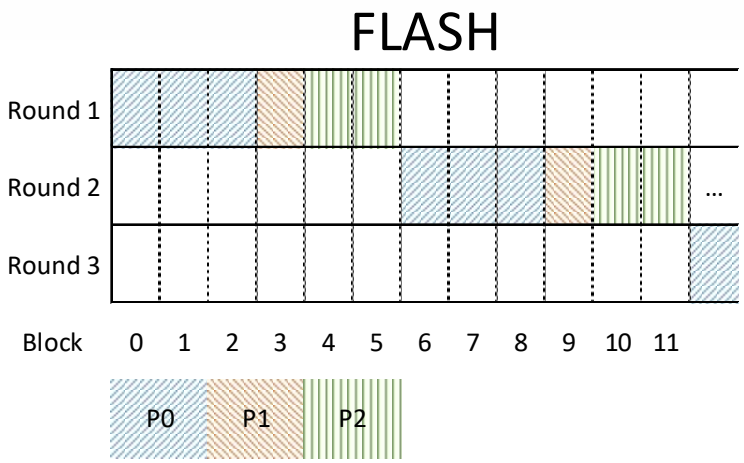
Experiment

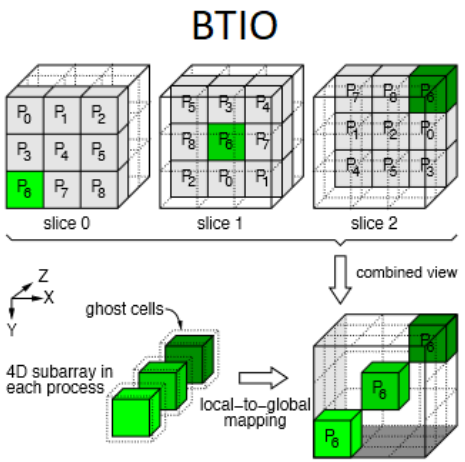
- Cori at NERSC
  - Cray DataWarp – shared burst buffer
- Theta at ALCF
  - Local burst buffer made of SSD
- Comparing with other approach
  - PnetCDF collective I/O without aggregation
  - Data elevator
  - Cray DataWarp staging out functions
  - LogFS
- Comparing different log to process mapping

Benchmarks



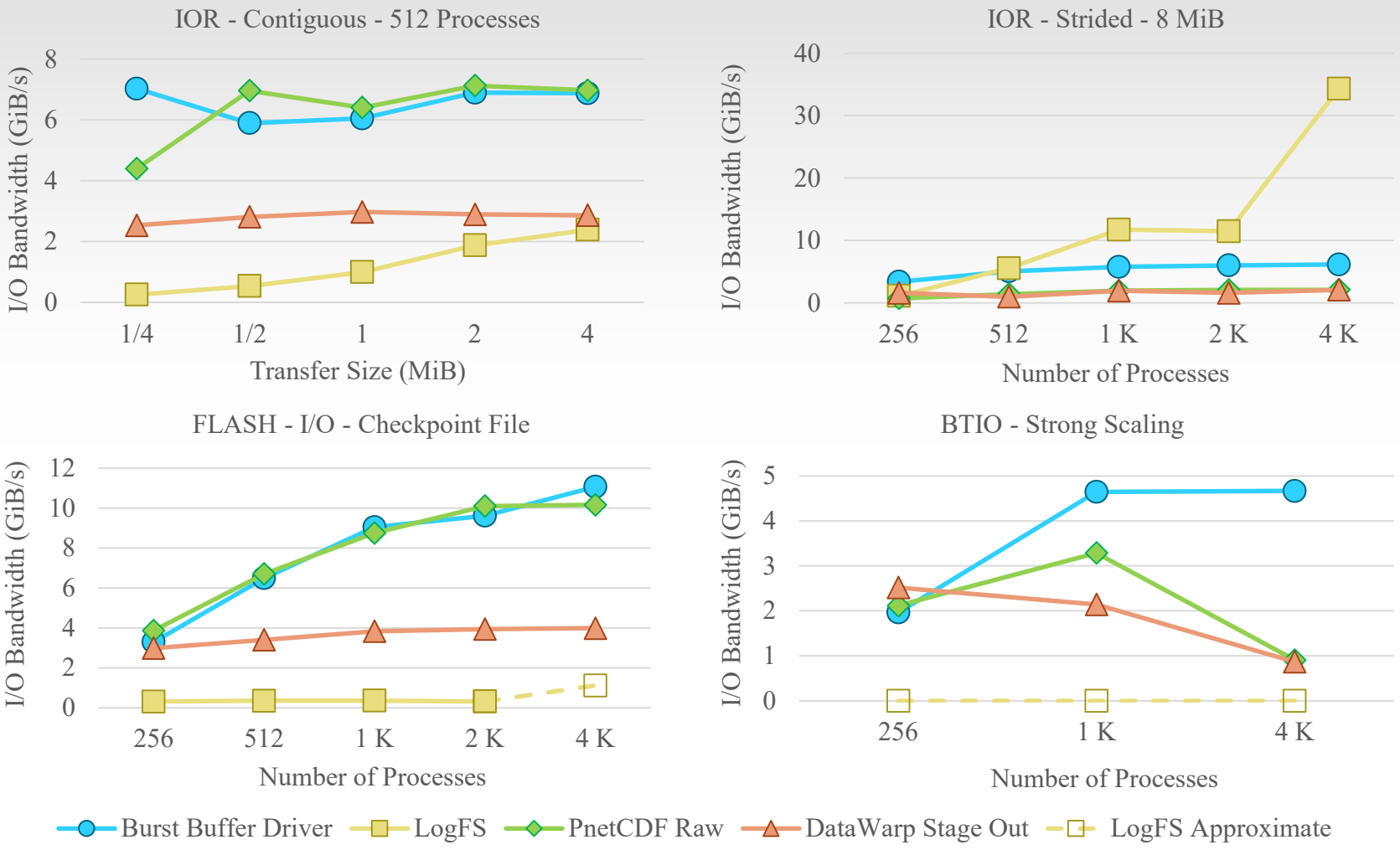




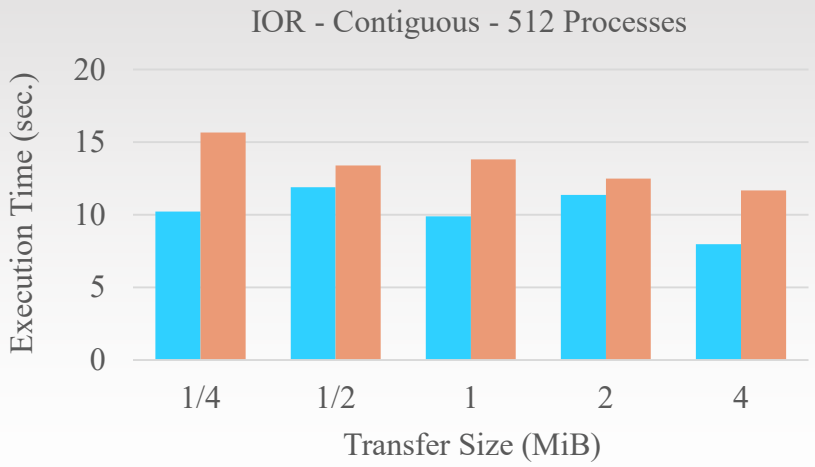


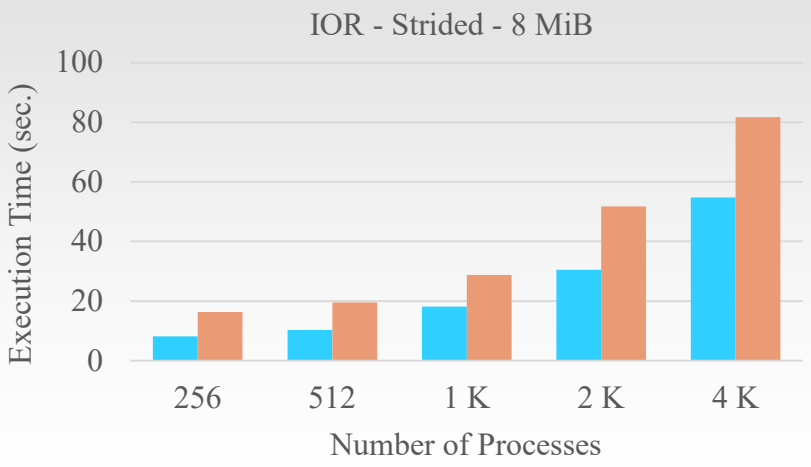
Picture courtesy from:
Liao, Wei-keng, et al. "Using MPI file caching to improve parallel write performance for large-scale scientific applications." *Supercomputing, 2007. SC'07. Proceedings of the 2007 ACM/IEEE Conference on*. IEEE, 2007.

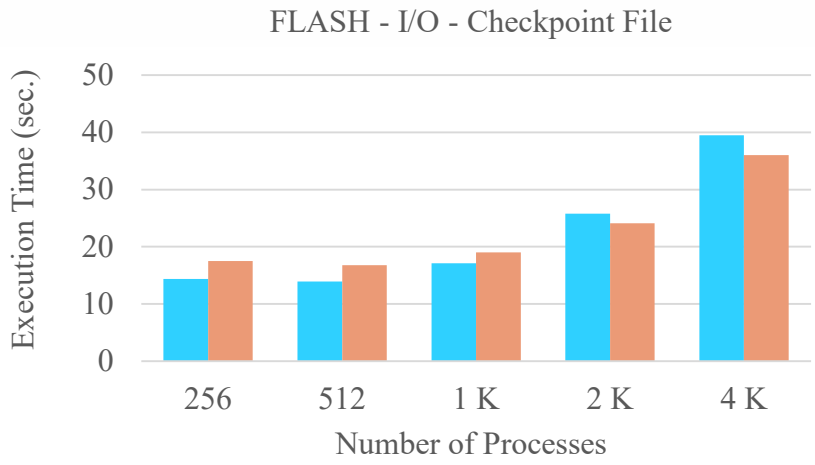
Cori – Shared Burst Buffer

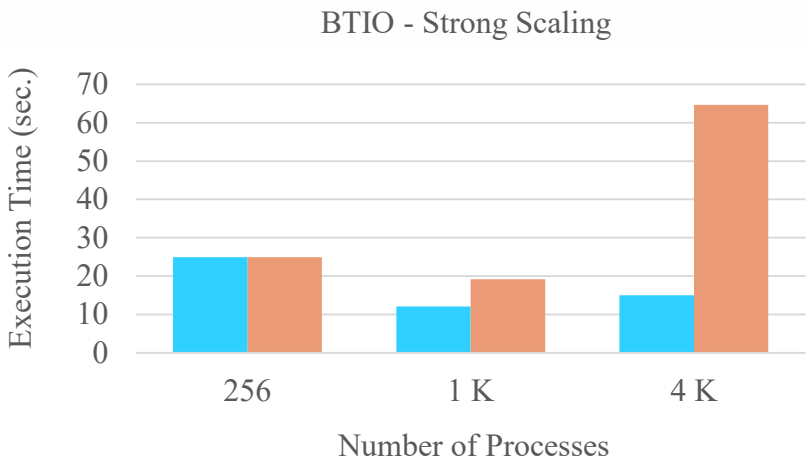


Cori – Shared Burst Buffer



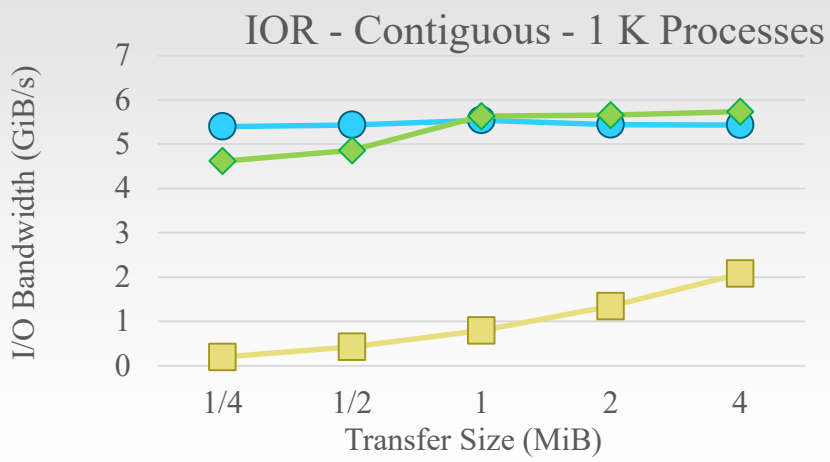


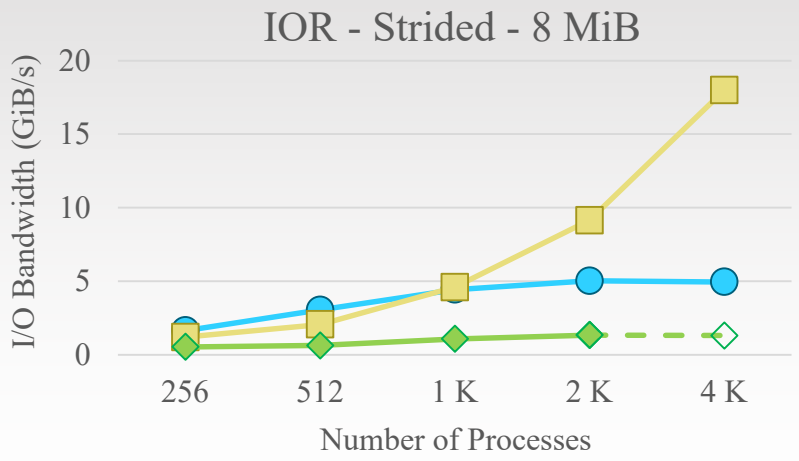


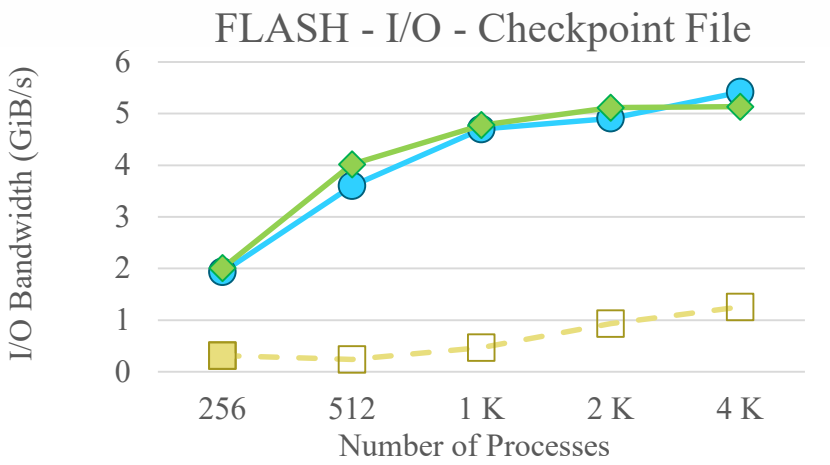


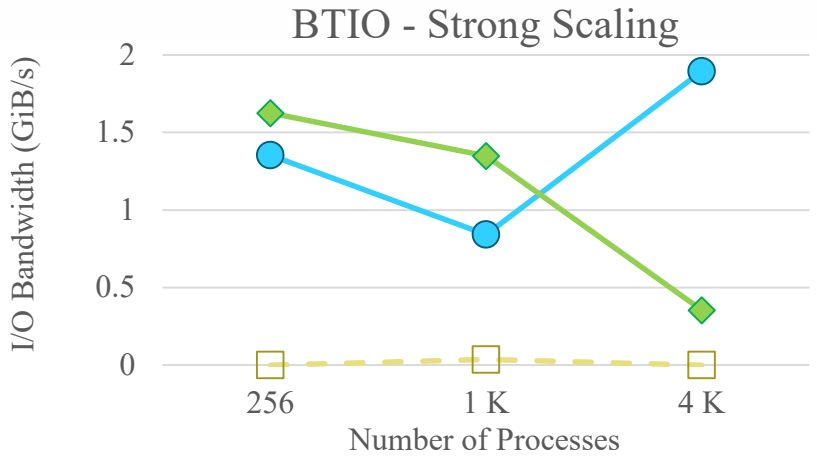
Burst Buffer Driver Data Elevator

Theta – Local Burst Buffer



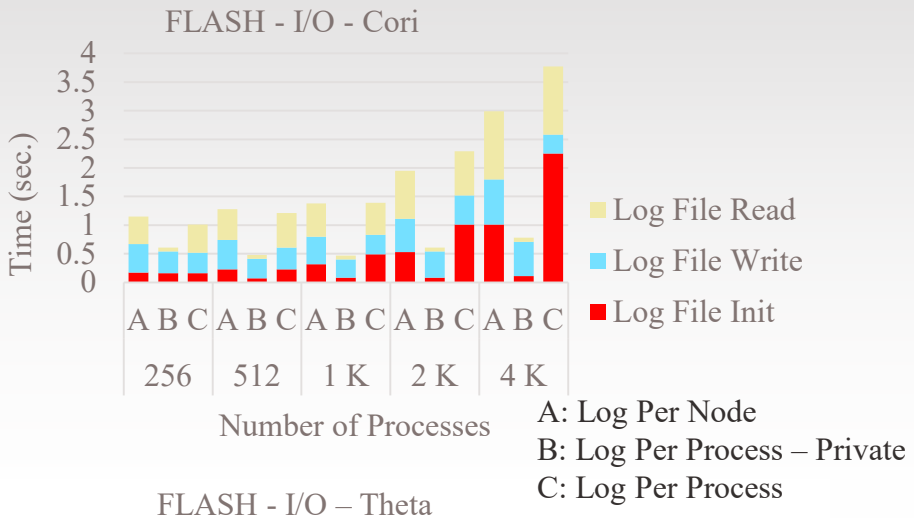


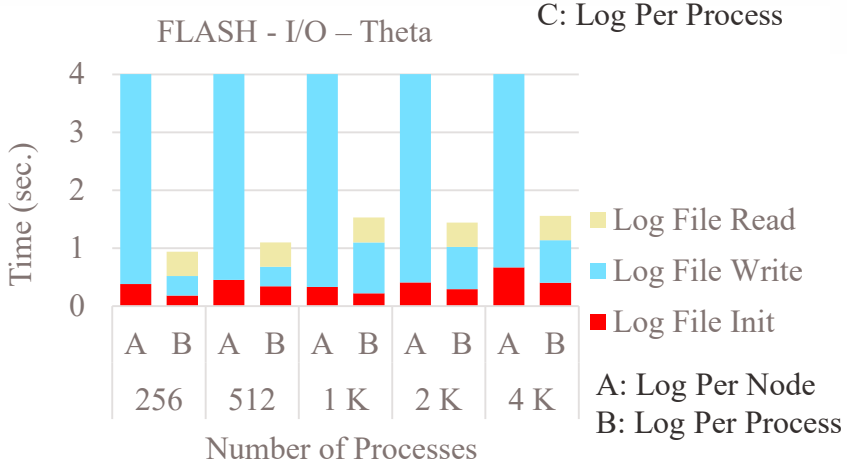




Impact of log to process mapping

- Use log per node on shared burst buffer
  - Metadata Server bottleneck when creating large number of files
- Use log per process on local burst buffer
  - Reduce file sharing overhead
- Use local burst buffer if available
  - Configure DataWarp to private mode





Conclusion and Future work

- Burst buffer opens up new opportunities for I/O aggregation
- Aggregation in a high-level I/O library is effective to improve performance
  - The concept can be applied to other high-level I/O libraries
    - HDF5, NetCDF-4 … etc.
- Performance improvement
  - Overlap burst buffer and PFS I/O
    - Reading from burst buffer and writing to PFS can be pipelined
  - Support reading from the log without flushing
    - Reduce number of flush operation

Thank You

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

Northwestern | McCORMICK SCHOOL OF ENGINEERING