Northwestern

A Case Study on Parallel HDF5 Dataset Concatenation for High-Energy Physics Data Analysis

***Sunwoo Lee**, *Kai-yuan Hou, *Kewei Wang, #Saba Sehrish, #Marc Paterno, #James Kowalkowski, §Quincey Koizol, $Robert B. Ross, *Ankit Agrawal, *Alok Choudhary, and *Wei-keng Liao

*: Northwestern University

#: Fermilab

§: Lawrence Berkeley National Laboratory

$: Argonne National Laboratory

Introduction to A Case Study on HEP Data Aggregation

Metadata I/O

Raw data I/O

End-to-End Performance Evaluation

High-Energy Physics Workflow and Data Storage

- In High-Energy Physics (HEP), the quantities of experimental data generated from large-scale instruments are often immense
- Traditional HEP workflows are designed for grid-oriented environment: many independent processes produce ***a large number of files of moderate sizes***
- To better facilitate data management, transfer, and analysis on large-scale HPC platforms, it is advantageous to aggregate data into ***a smaller number of larger files***

A Case Study

- Considering the fast growth in size of HEP data, the data aggregation becomes the bottleneck of data analysis workflow

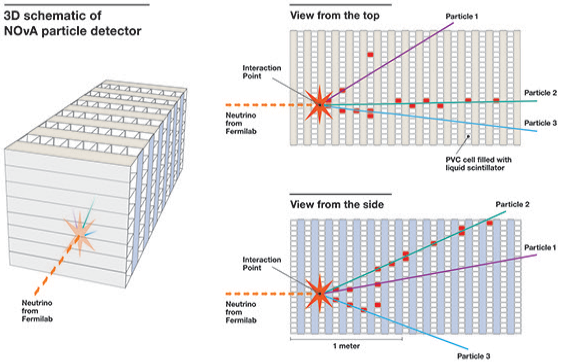
- In this work, we consider data aggregation of a large-scale HEP data
  - NOvA experiment data (hundreds to thousands HDF5 files)
  - Raw data: 1TB ~ 17TB
  - Compressed files: 35GB ~ 213 GB

- Explore a subset of HDF5 features used to implement NOvA data aggregation, and present our evaluation and analysis of their performance impacts

NOvA Experiment Data

- NuMI Off-axis ν_e Appearance (NOvA) experiment
  - Designed to study neutrino oscillations using the particle collision event data recorded by two detectors

- Near Detector (ND) data and Far Detector (FD) data
  - ND is at the Fermilab in Batavia, IL, and FD is in Ash River, MN
  - Both detectors observe neutrinos from a beam generated at Fermilab
  - NOvA produces many HDF5 files because of the grid-oriented processing environment



https://phys.org/news/2014-02-nova-long-distance-neutrinos.html

Tabular Structure of NOvA Experiment Data

- ***Run***: a period of data collection that represents a stable period of detector operation (a few hours ~ 24 hours)
- ***Subrun***: a subdivision of a run period that limits the output file size
- ***Spill (event)***: one 10 μs pulse of neutrinos generated at the Fermilab every 1.3 sec
- ***Slice*** (***subevent***): a fixed period of detector activity discovered within a spill
- Each file contains all spills in one ***subrun***

Table 1
NOvA data table organization with one entry per slice.

Run	Subrun	Event	Sub-event	distallpngtop	... 35 more ...
433	61	6124	35	nan	
433	61	6124	36	-0.7401	
433	61	6124	37	nan	
433	61	6125	1	nan	
433	61	6125	2	423.633	
433	61	6125	3	-2.8498	

NOvA Experiment Data Statistics

	ND files	FD files
Number of files	165	6,400
Number of groups per file	999	701
Number of 1-D datasets	15,965	12,925
Number of 2-D datasets	8	6
Number of empty datasets	13,396	9,374
Raw data size before compression	***1001.1 GB***	***17 TB***
Raw data size after compression	***23.5 GB***	***101.9 GB***
Metadata size	***11.7 GB***	***104.1GB***
Total file size	35.2 GB	212.5

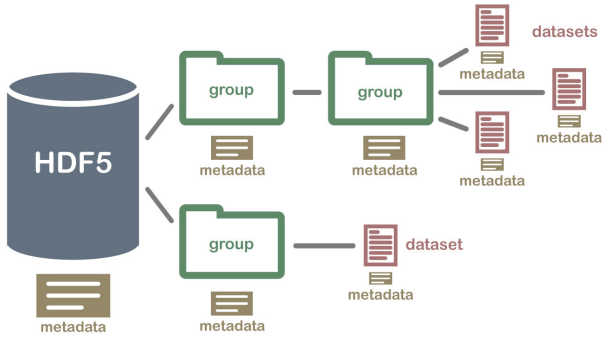
1. High compression ratio
2. A large amount of metadata

Hierarchical Data Format (HDF5)

- HDF5 is a popular I/O library in scientific communities, that enables users to store data in a portable, self-describing file format
  - Many open-source and commercial software packages for data visualization and analysis use HDF5
  - HDF5 supports parallel I/O for data compression (from 1.10.3)

- An HDF5 file consists of ‘group’s and ‘dataset’s
  - One file can have multiple groups
  - Each group can have multiple groups
  - Each group can have multiple datasets
  - Each HDF5 object has its own metadata

- We use HDF5 1.10.5 in this case study



Hierarchical Data Formats - What is HDF5? | NSF NEON | Open Data to Understand our Ecosystems (neonscience.org)

Parallel Data Aggregation

- Given F HDF5 files, all individual datasets are concatenated across the F files and stored into a single large output file
  - All files have the same groups and datasets of various sizes

- Parallel dataset concatenation workflow
  1. Evenly distribute F input files to P processes
  2. Collect and aggregate dimension sizes of D datasets from all input files
  3. Create a new shared output file
  4. Create D datasets of aggregated sizes in the output file
  5. For each dataset, read the dataset from all assigned input files to a memory buffer by appending one after another and then write the concatenated buffer to the output file

Experimental Settings

- All experiments are conducted on Cori XC40 supercomputer at NERSC
  - Used Haswell nodes to take advantage of the larger memory space and I/O speed than KNL nodes
  - Each node has 128 GiB DDR4 memory space, two sockets of Intel Xeon E5-2698v3 CPUs

- Parallel File System of Cori
  - Both the input files and the output file are stored on a disk-based Lustre parallel file system
  - File stripe size is 1MiB and the file stripe count is 128

Introduction to A Case Study on HEP Data Aggregation

Metadata I/O

Raw data I/O

End-to-End Performance Evaluation

Reading Metadata From Input Files (1/2)

- To concatenate each dataset, the data type and array sizes should be read from all the input files

- Collecting the metadata from many NOvA files can be expensive
  - HDF5 allows metadata and raw data to be stored separately in almost any where in the file
  - Metadata collection: small and non-contiguous accesses to many files
  - Hundreds to thousands of files each of which has 13K~16K datasets

Reading Metadata From Input Files (2/2)

- Two possible options for metadata reading
  - On-the-fly I/O: many small and non-contiguous I/O
  - In-Memory I/O: read the entire files in a memory buffer at once (memory footprint ↑)

Metadata collection time for 165 ND files. Up to four processes run on each node.

# of processes	3	6	11	21	42	83	165
# of nodes	1	2	3	6	11	21	42
On-the-fly I/O	4099 sec	2188 sec	1122 sec	663 sec	304 sec	131 sec	64 sec
In-memory I/O	***61.9 sec***	***26.8 sec***	***16.3 sec***	***8.9 sec***	***4.9 sec***	***3.6 sec***	***1.5 sec***

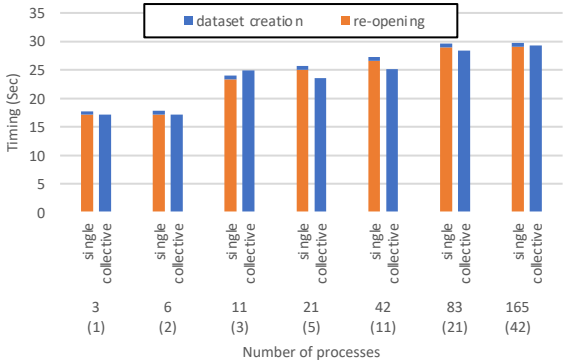
Metadata collection time for 6,400 FD files.

# of processes	100	200	400	800	1600
# of nodes	25	50	100	200	400
On-the-fly I/O	356 sec	180 sec	93 sec	55 sec	37 sec
In-memory I/O	***42.1 sec***	***21.3 sec***	***13.9 sec***	***8.7 sec***	***5.7 sec***

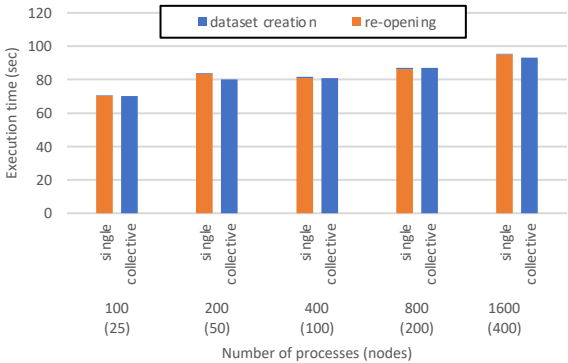
Writing Metadata To Output File (New Dataset Creation)

- Once the metadata is all collected, the same groups and datasets are created in the output file

- Two possible design options
  - One process opens the output file in POSIX mode and create all the datasets
  - All the processes open the output file in MPI I/O mode and collectively create all the datasets



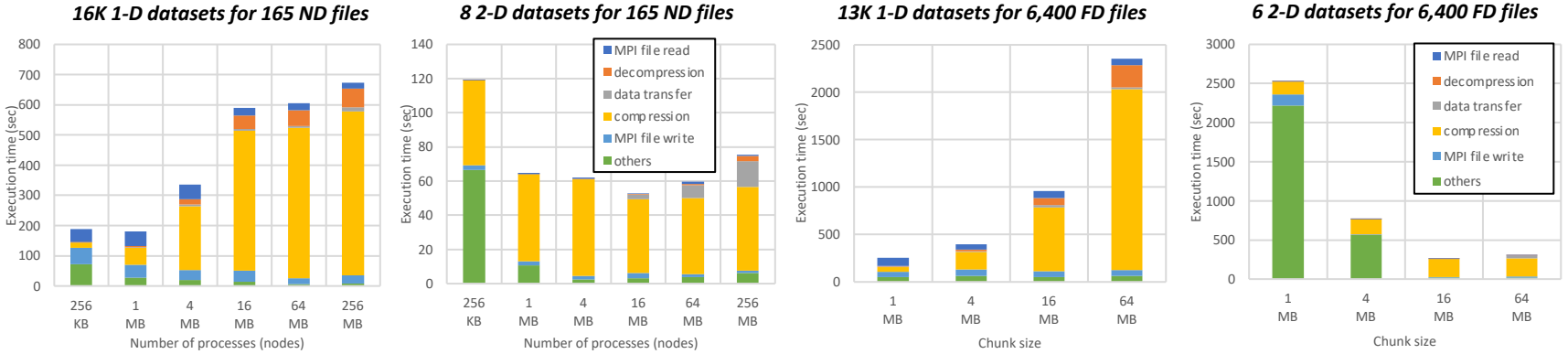
Timing breakdown for 165 ND files



Timing breakdown for 6,400 FD files

Chunking and Compression Strategy

- HDF5 requires users to use ‘chunked’ layout to enable data compression
- The chunk size affects not only I/O cost but also (de)compression and communication cost
  - Each chunk is owned by a unique process who has the largest portion of the chunk
  - When writing, all the processes collect their own chunks via inter-process communications, and then compress and write them into the output file



Metadata Caching

- Given 13K~16K datasets, caching metadata and later flushing in bigger and aggregated write requests appears to be a good strategy

- HDF5 allows users to adjust the metadata cache size (Default: 2MB and automatic increase based on hit-ratio)
  - Set the size to 128MB per process (maximum)
  - Turn off the automatic adjustment

Performance comparison (sec) between default setting and our setting for 165 ND files

Steps	Default	Our setting	Performance gain
Metadata collection	1.5	1.5	-
Dataset creation	30.7	29.3	4.5%
1-D dsets Read	15.5	15.9	-
1-D dsets Write	255.2	167.2	34.5%
2-D dsets Read	17.2	17.3	-
2-D dsets Write	54.1	48.2	10.9%
Overall	374.3	279.4	25.3%

Introduction to A Case Study on HEP Data Aggregation

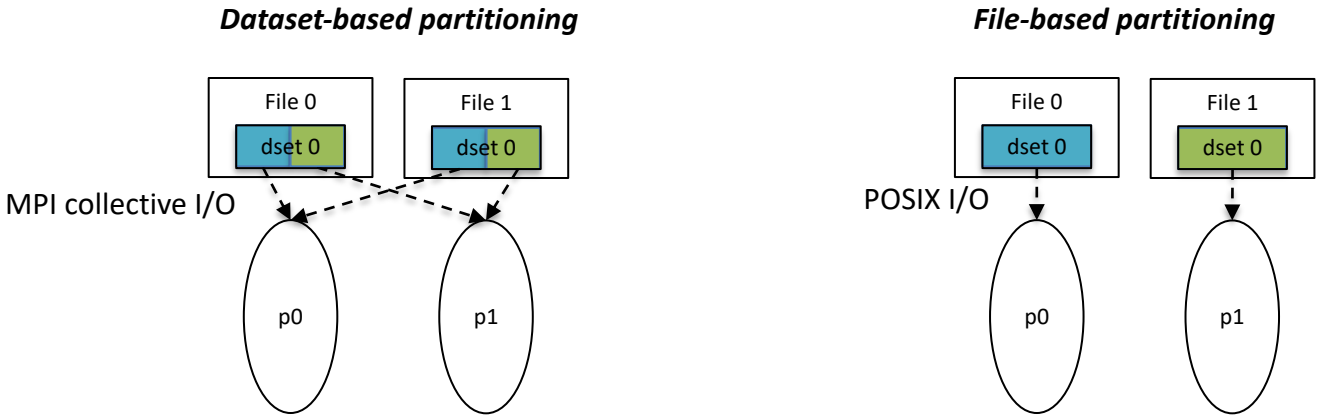
Metadata I/O

Raw data I/O

End-to-End Performance Evaluation

Parallel Read (1/2)

- Once all the datasets are created in the output file, each dataset is read from all the input files into a memory buffer
- Two design options for parallel read: ***dataset-based*** partitioning and ***file-based*** partitioning

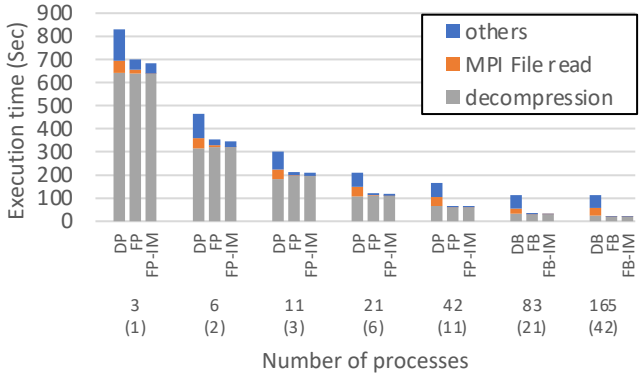


Parallel Read (2/2)

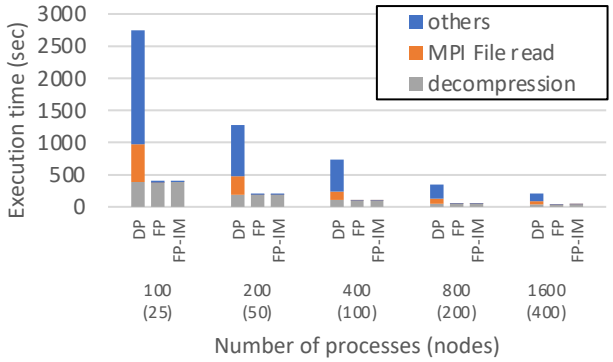
DP: Dataset-based Partitioning
FP: File-based Partitioning
FP-IM: File-based Partitioning with In-Memory I/O

- File-based partitioning better fits to NOvA experiment data
  - Enables each process to independently use POSIX I/O
  - The number of read operations is proportionally reduced as more processes run (Dataset-based partition always performs the same number of reads)

Timing breakdown of 2-D dataset read from 165 ND files



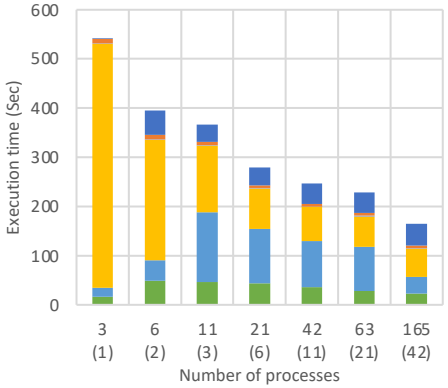
Timing breakdown of 2-D dataset read from 6,400 FD files



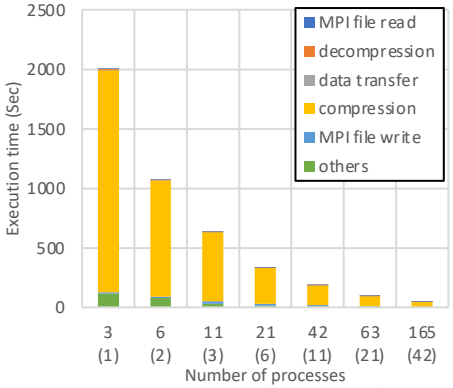
Parallel Write

- Finally, each concatenated dataset is written into a shared output file
- Regardless of the read strategy, parallel write is always MPI collective I/O (HDF5 allows only one dataset to be written at a time)

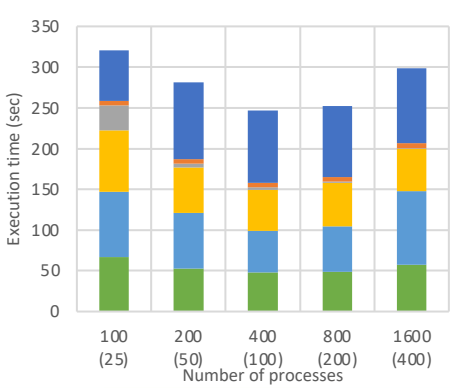
Timing breakdown of 1-D dataset write (165 ND files)



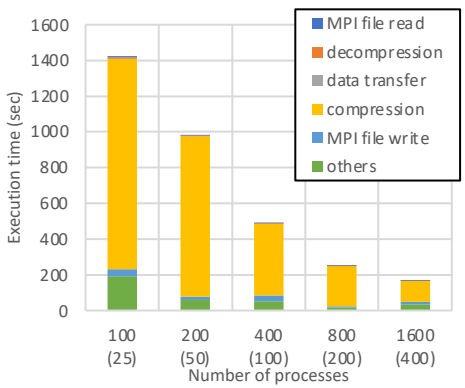
Timing breakdown of 2-D dataset write (165 ND files)



Timing breakdown of 1-D dataset write (6,400 FD files)



Timing breakdown of 2-D dataset write (6,400 FD files)



Introduction to A Case Study on HEP Data Aggregation

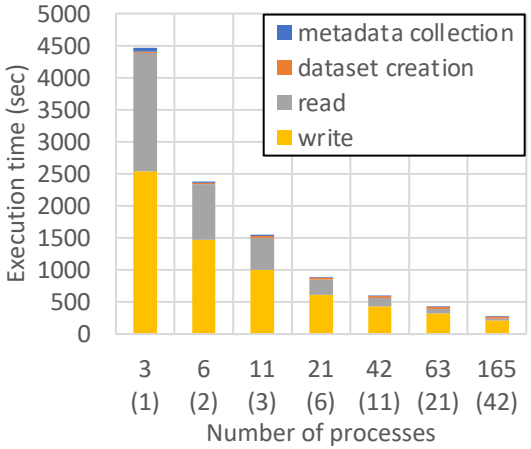
Metadata I/O

Raw data I/O

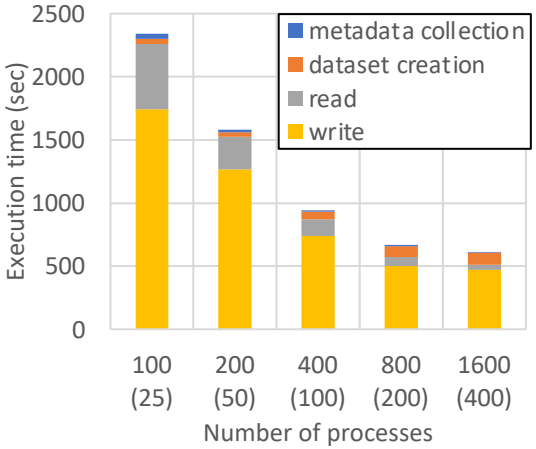
End-to-End Performance Evaluation

End-to-End Performance Evaluation

Concatenation time for 165 ND files



Concatenation time for 6,400 FD files



Data	ND files	FD files
# of processes (nodes)	165 (42)	1600 (400)
Timing (sec)	***279.4***	***611.8***
Data size before compression	1 TB	16.9 TB
Output file size	15.1 GB	77.9 GB
Metadata size	57.7 MB	93.8 MB
Raw data size	15.0 GB	77.4 GB

Summary

- Concatenating many small files into one large file is the first step of HEP analysis programs
- Our study explore a variety of HDF5 features and tune them to achieve scalable parallel I/O performance
- The lessons learned from our case study can provide guidance in selecting the appropriate HDF5 settings for scientific applications that exhibit similar I/O characteristics
- Currently under review (2nd round) by Elsevier Parallel Computing

Acknowledgment

- This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program, under the ``RAPIDS Institute'' and grant ``HEP Data Analytics on HPC'', No. 1013935.

- This work is also supported in part by the DOE awards DE-SC0014330 and DE-SC0019358.This research was also 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 through the ExaHDF5project under Contract No. DE-AC02-05CH11231.

- 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 Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Q & A