# Parallel Data Object Creation: Scalable Metadata Management in Parallel I/O Library

PDSW 2025

Presenter: Youjia Li
Nov 17, 2025

**Youjia Li, Ankit Agrawal, Alok Choudhary, Wei-keng Liao**
*Department of Electrical and Computer Engineering, Northwestern University*
**Robert Latham, Robert Ross**
*Mathematics and Computer Science, Argonne National Laboratory*

Argonne NATIONAL LABORATORY

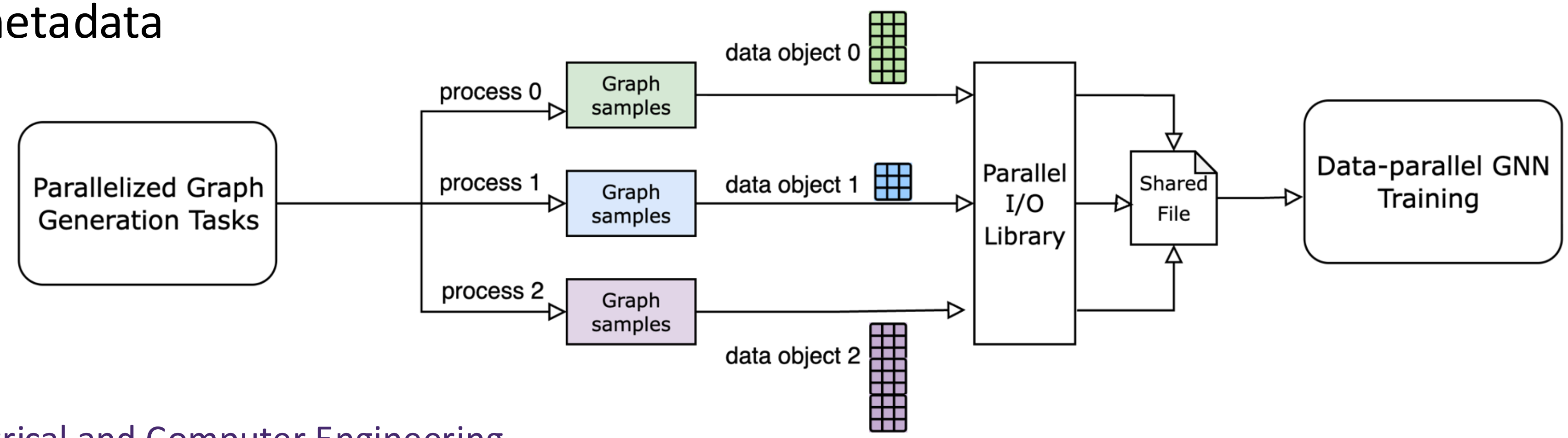
Northwestern | Electrical and Computer ENGINEERING

# Motivation and Background

- Data Object Creation in Parallel I/O Library
  - Data objects: primary units of storage that refers to the named containers that store actual data
    - In HDF5: *Dataset*, *Group*
    - In netCDF: *Variable*, *Dimension*
  - Metadata: the data describing the data objects. Example: name, data type, and annotation.
    - With I/O libraries, creating data objects requires metadata from the user application.
    - In typical use cases, metadata is relatively small in volume.
  - Data Object Creation
    - Parallel HDF5
      - Collective operation with identical metadata across processes
      - HDF5 for DAOS file system version mentions independent object creations (H5daos_set_all_ind_metadata_ops)
    - PnetCDF
      - Requires collective object creation with identical metadata across all processes.
      - Create new data objects in **define mode** and write data to those objects in **data mode**. The **end-define stage** marks the transition, during which the root process writes metadata.

# Motivation and Background

- Case Study: *Neutrino particle collision simulation from the Exa.TrkX project*
  - Sensor collect data within fixed time intervals representing detected particle activity.
  - Volume of data objects and their metadata is proportional to the number of sensors and the duration of the experiment.
  - GNN-based trajectory reconstruction task
    - Training sample contains multiple variable-sized arrays (nature of graph data)
    - The GNN model training task reads a single input file containing all the graph samples
    - The upstream graph generation task runs in parallel and thus must synchronize metadata



# Metadata Management Challenge

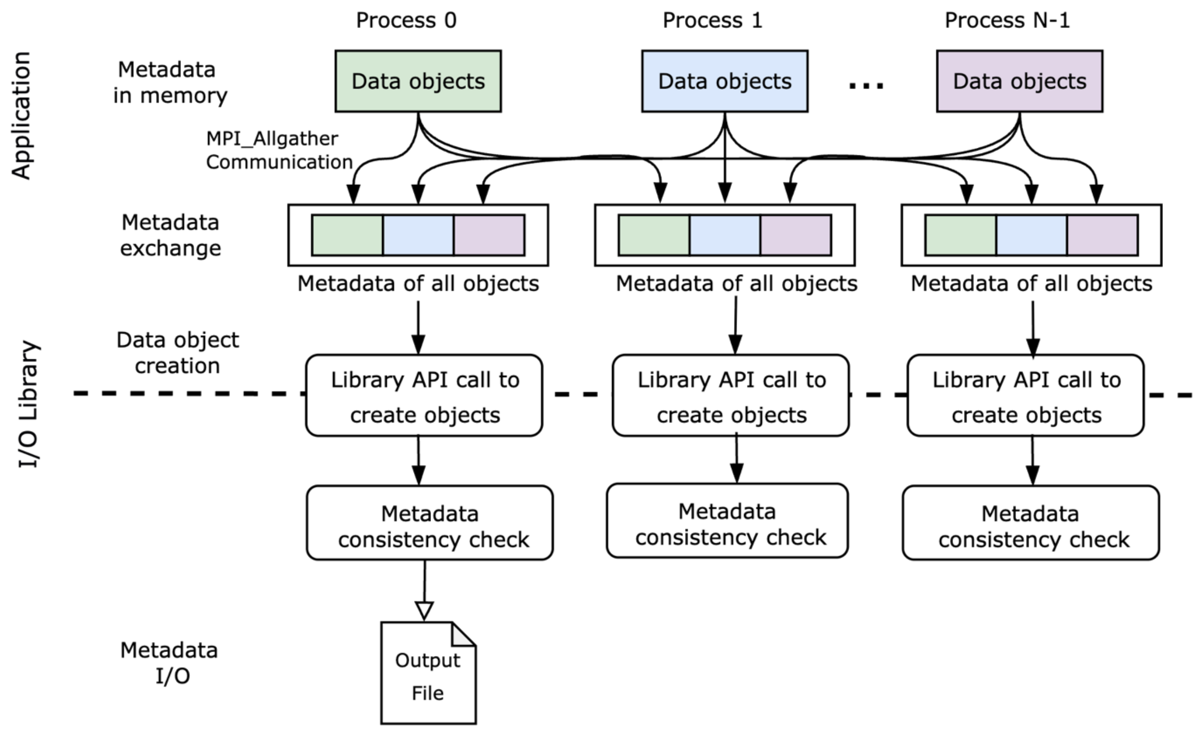
- **Data Management Challenge:** Data object creation is not scalable in modern parallel I/O libraries as the library requires collective operation.
  - Existing I/O libraries are not designed for handling a large number of unique data objects
  - Variable-sized, irregular scientific data has no predictable dimensions. In parallel mode, one process is unaware of the metadata associated with data objects created by other processes.
  - Contention on metadata consistency check
    - **Metadata consistency check**: to ensure that the metadata of data objects to be created are the same among the processes and each data object is uniquely defined
    - PnetCDF library uses a hash table for quick lookup for object names in consistency check
    - HDF5 library uses B-trees in symbol table to manage this

# Design and Implementation

- **Proposed Solution:** parallel data object creation while achieving a scalable performance through developing a specialized variant of I/O library and novel file header format
  - Enable independent data object creation in the library API
  - The I/O library should distinguish between shared and non-shared data objects during consistency checks.
  - Reduce metadata consistency check overheads
- Main approaches:
  - Application-level baseline approach
  - Library-level baseline approach (independent data object creation)
  - New header file approach (scalable metadata management)

# Application-level Baseline Approach

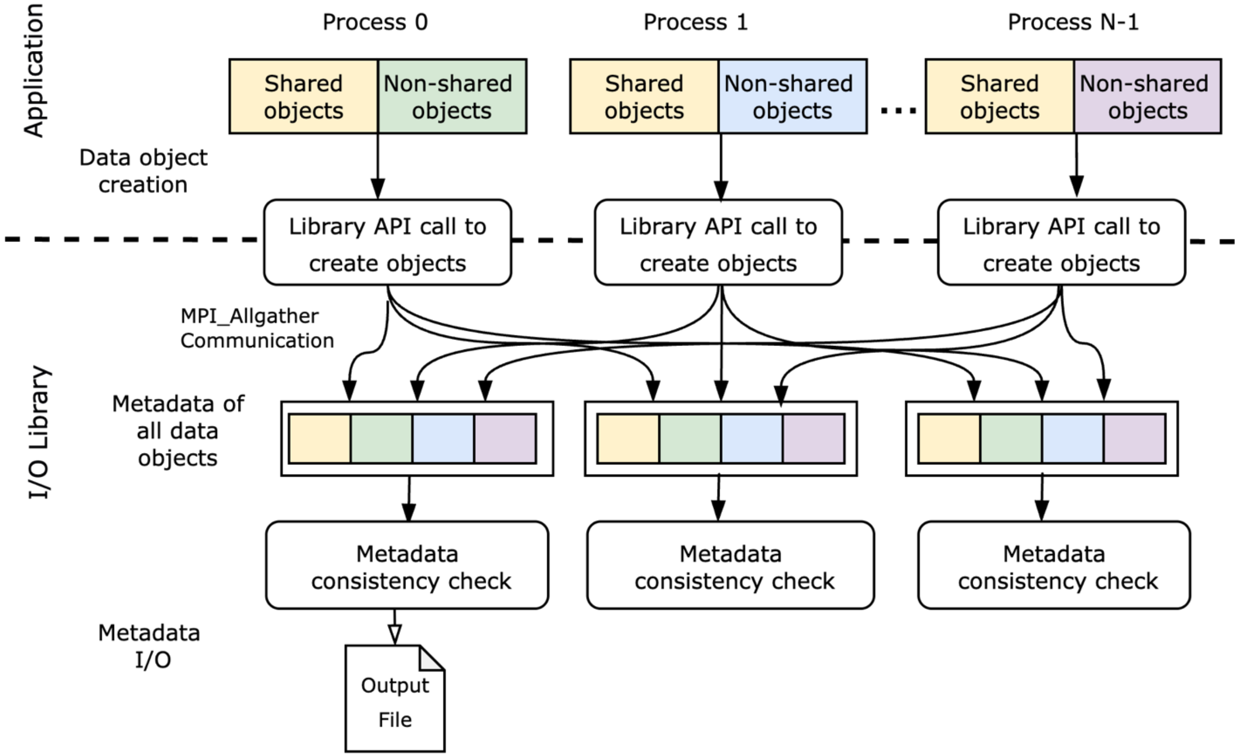
- Application-level Baseline Approach
  - To meet parallel I/O library requirements for collective data object creation, applications need explicitly synchronize metadata first.
    - Metadata is first serialized into a single send buffer on each process
    - Makes an MPI_Allgather communication call to exchange metadata
    - Makes the high-level I/O library calls to create  created data objects




Metadata flow in the application-level baseline approach

# Library-level Baseline Approach

- Library-level Baseline Approach
  - Offloads the metadata synchronization from application to library
  - Enables independent data object creation
    - Shared data objects created collectively
    - Non-shared objects are created independently.
    - The I/O library automatically detects shared/non-shared objects
  - Baseline consistency check cost:




Metadata flow in the library-level baseline approach

$$O(n) = n \cdot \frac{n}{2k}$$

With uniform hashing, inserting n objects into a table of size k gives n/k objects per slot, so average cost ≈ n/2k comparisons per insertion

# New Header Format Approach

- New File Header Format Approach
  - Prior approaches require all processes to hold a full copy of metadata
  - A new file header format that can reduce communication and consistency check

  - Header reorganized into 2 sections:
    - **Index table** stores references to individual metadata blocks
    - **List of metadata blocks** contains the actual metadata for data objects, where each block is can be created by one process independently

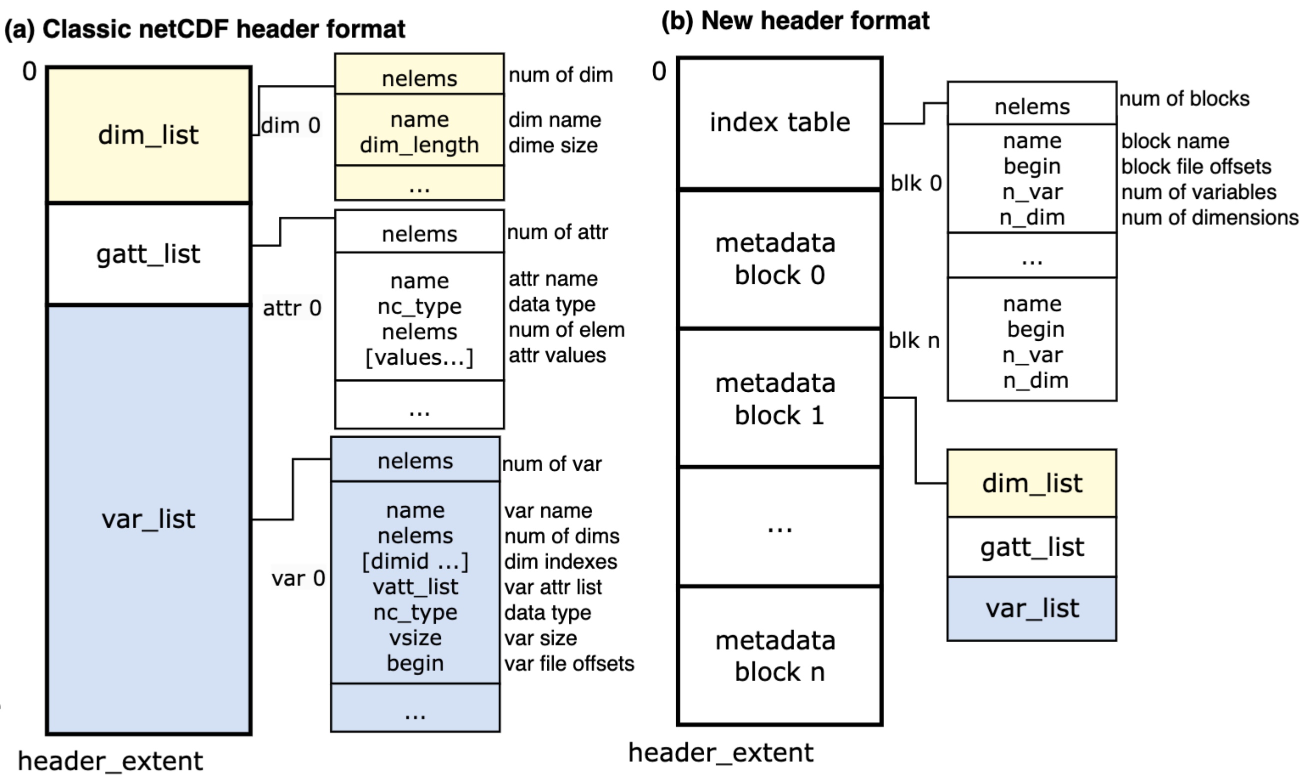
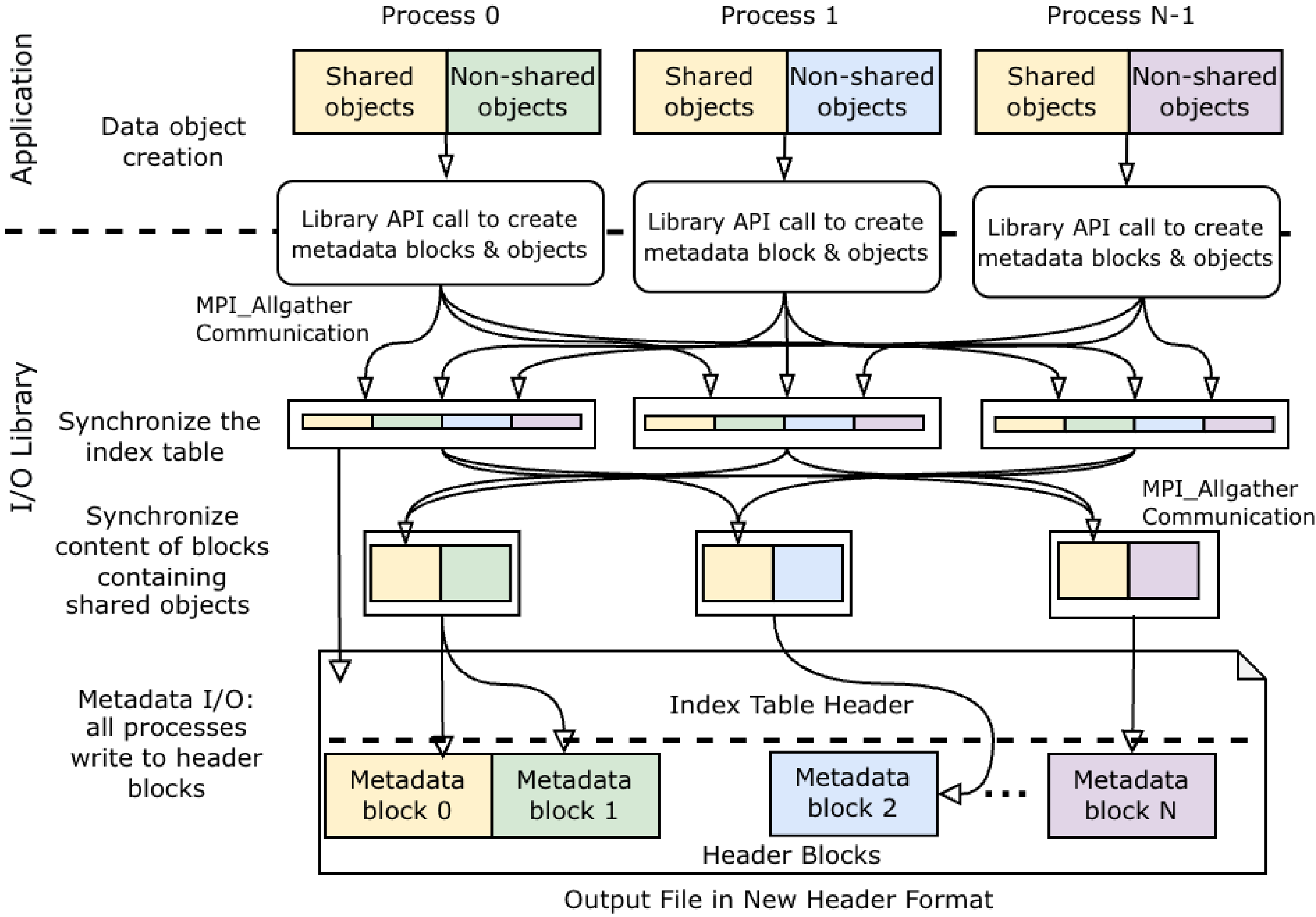



Figure (a) :the original classic netCDF file header
(b) the proposed new header format

# New Header Format Approach

- New File Format Approach
  - Identifies each object using a combination of metadata block and data object name
  - Shared and non-shared objects are identified by metadata block
  - Metadata consistency check is applied in a hierarchical manner
    - Data object name conflicts locally checked within each block
    - Block name are checked at index table synchronization



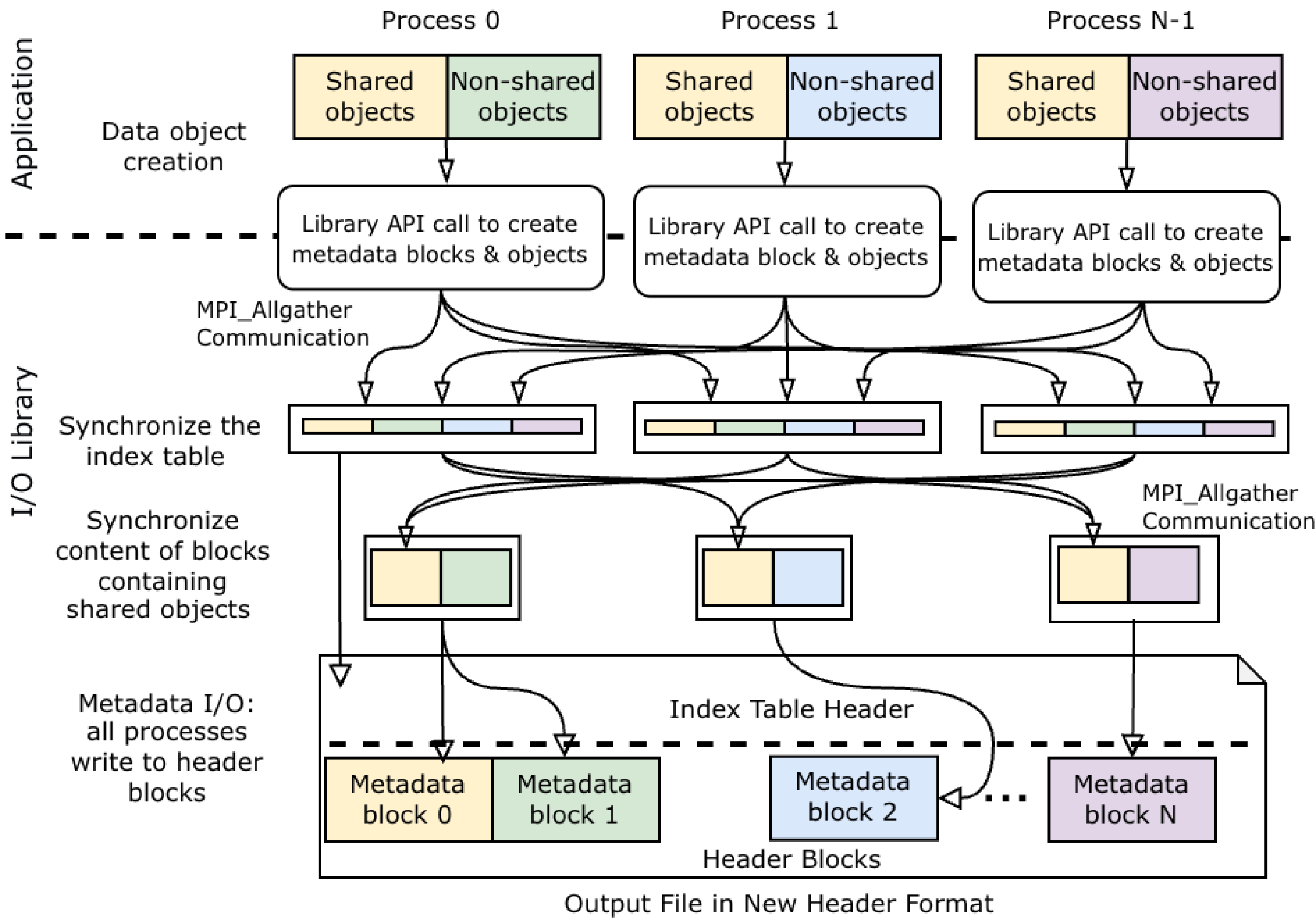

Metadata flow in the new file format approach

# New Header Format Approach

- New File Format Approach
  - Metadata flow:
    - Step 1: Only metadata block names and sizes (index table) are synchronized by MPI communication
    - Step 2: Synchronized shared metadata blocks (if any)
    - Step 3: Root process writes index table. All processes  parallelly writes their metadata blocks to header sections
  - Computation cost for consistency check

$$O\left(\frac{n}{p} \cdot \frac{n}{2kp}\right) + O\left(p \cdot \frac{p}{2k}\right) \approx O\left(\frac{n^2}{2kp^2}\right)$$

When p << n, a speedup by a factor of $p^2$ compared to baseline cost




Metadata flow in the new file format approach

# Performance Evaluation

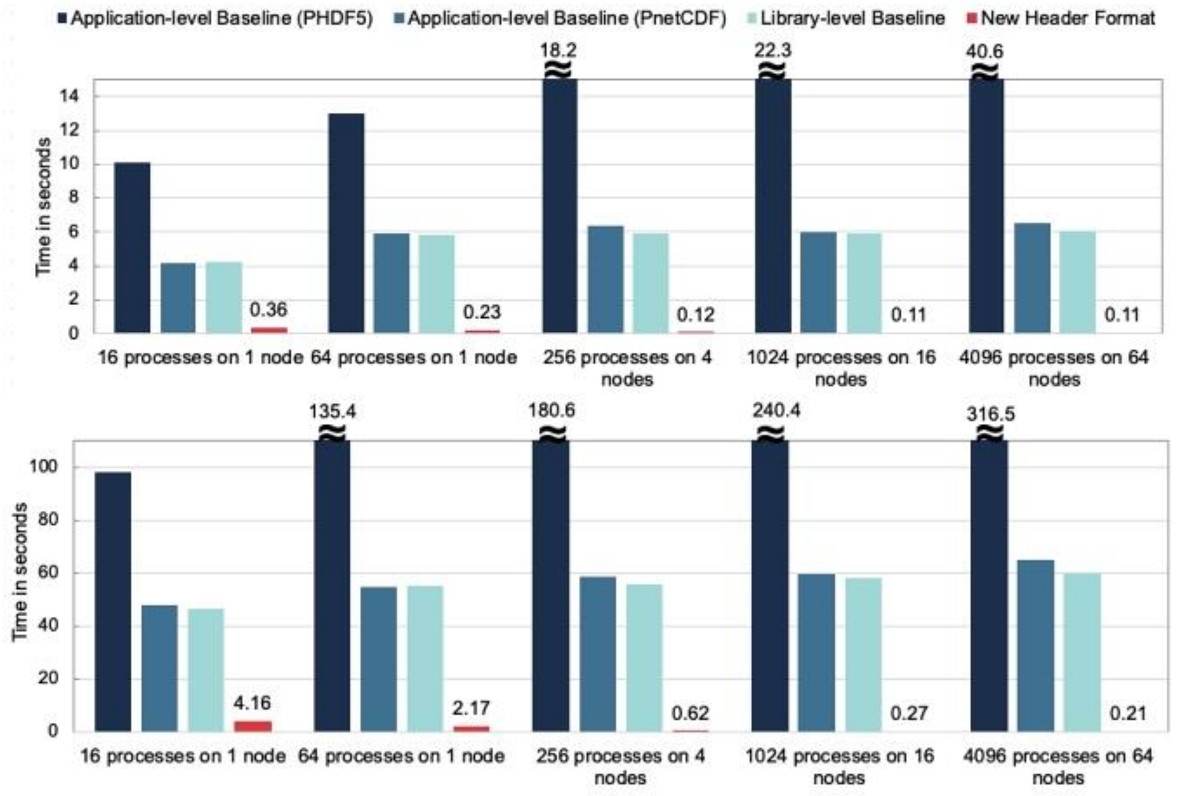
- Experimental Settings
  - Datasets from Neutrino particle collision simulation: $dataset\_568k$, $dataset_5$m
    - In HDF5: 568,480 dataset objects bucketed in 35,530 groups
    - In netCDF: 568,480 variables and 852715 dimensions
  - Strong scaling case experiment (e.g. $dataset_5$m below)

| No. of processes | Total | 4 | 16 | 64 | 256 | 1024 | 4096 |
|---|---|---|---|---|---|---|---|
| No. of netCDF Variables assigned per process | 5684800 | 1421200 | 355300 | 88825 | 22206 | 5552 | 1388 |
| No. of netCDF Dimensions assigned per process | 8527150 | 2131788 | 532947 | 133237 | 33309 | 8327 | 2082 |
| Metadata amount assigned (MB) | 802.20 | 200.55 | 50.14 | 12.53 | 3.13 | 0.78 | 0.20 |

  - Max MPI processes: 4096 (across 64 HPC nodes)
  - System: Perlmutter at NERSC
  - File system: Lustre with 248 OSTs

# Data Object Creation and Metadata Write Test

- Timings collected for writing metadata




End-to-end metadata write time when using $dataset$_568k (top) and $dataset$_1G (bottom)

- Application-level and library-level baseline approach do not scale at all as the number of processes increases.
- Consistency check is the major bottleneck
- HDF5 has periodical metadata cache synchronization. If following parameters are appropriately tuned and configured, data object creation time can be effectively reduced
  - dirty_bytes_threshold
  - istore_k
  - metadata_write_strategy
  - metadata_block_size

# Data Object Creation and Metadata Write Test

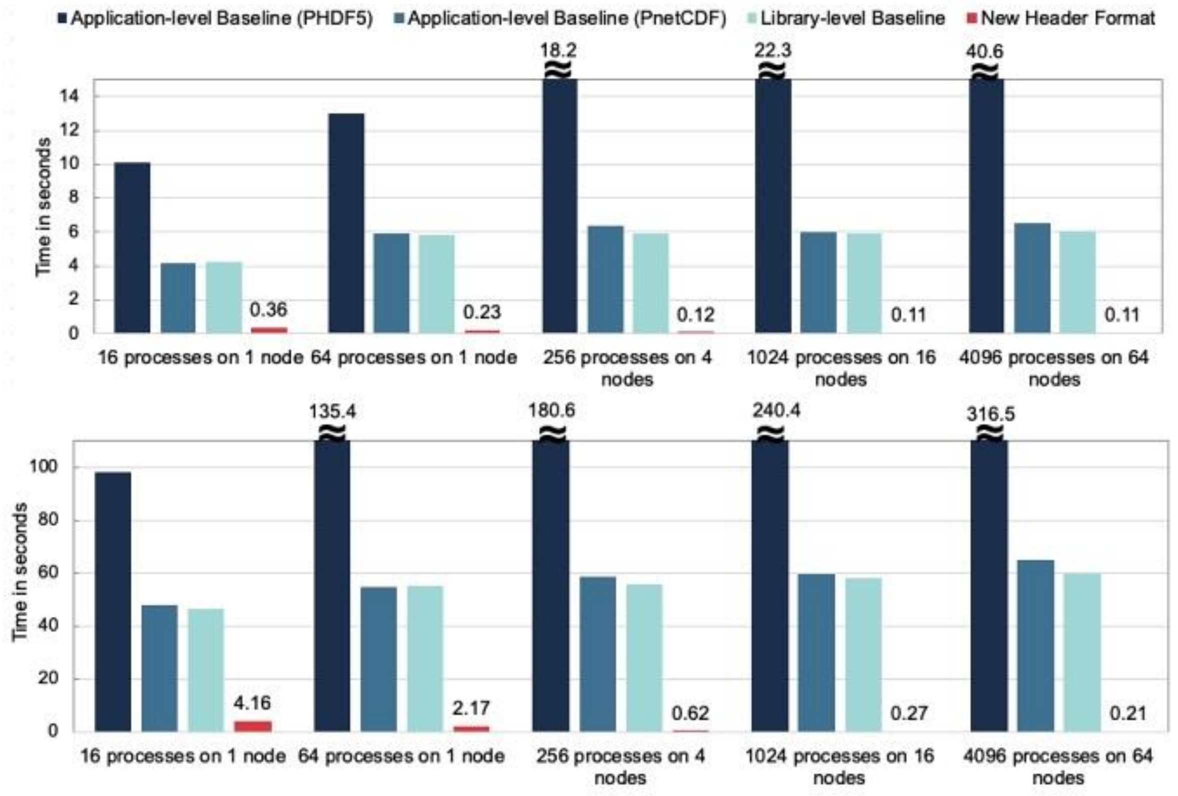
- Timings breakdowns for writing metadata

| Approaches | Phases | *dataset_568k* | | | *dataset_5m* | | |
|---|---|---|---|---|---|---|---|
| | | 16 processes on 1 node | 256 processes on 4 nodes | 4096 processes on 64 nodes | 16 processes on 1 node | 256 processes on 4 nodes | 4096 processes on 64 nodes |
| Application-level Baseline (PHDF5) | Metadata Exchange | 0.1368 | 0.1853 | 0.3819 | 1.3526 | 1.6456 | 2.0878 |
| | Metadata Consistency Check | 7.5985 | 15.1938 | 37.4385 | 73.2926 | 152.0087 | 288.3926 |
| | Others | 2.3453 | 2.8399 | 2.8183 | 23.5353 | 26.9642 | 26.0051 |
| Application-level Baseline (PnetCDF) | Metadata Exchange | 0.1143 | 0.2333 | 0.2607 | 2.1035 | 4.3414 | 5.4854 |
| | Metadata Consistency Check | 3.4065 | 4.6854 | 5.0956 | 39.3717 | 44.4244 | 48.1038 |
| | Others | 0.6429 | 1.0069 | 1.1562 | 6.4511 | 9.8935 | 11.7557 |
| Library-level Baseline (PnetCDF) | Metadata Exchange | 0.1360 | 0.2519 | 0.2519 | 1.5054 | 2.5563 | 2.5563 |
| | Metadata Consistency Check | 3.3178 | 4.4949 | 4.5481 | 36.7110 | 41.3237 | 44.4320 |
| | Others | 0.7826 | 1.1282 | 1.2749 | 8.4328 | 12.2421 | 13.1527 |
| New Header Format (PnetCDF) | Metadata Exchange | 0.0001 | 0.0005 | 0.0019 | 0.0001 | 0.0006 | 0.0068 |
| | Metadata Consistency Check | 0.2138 | 0.0153 | 0.0016 | 2.4409 | 0.0069 | 0.0130 |
| | Others | 0.1493 | 0.0998 | 0.1027 | 1.7215 | 0.6156 | 0.1931 |

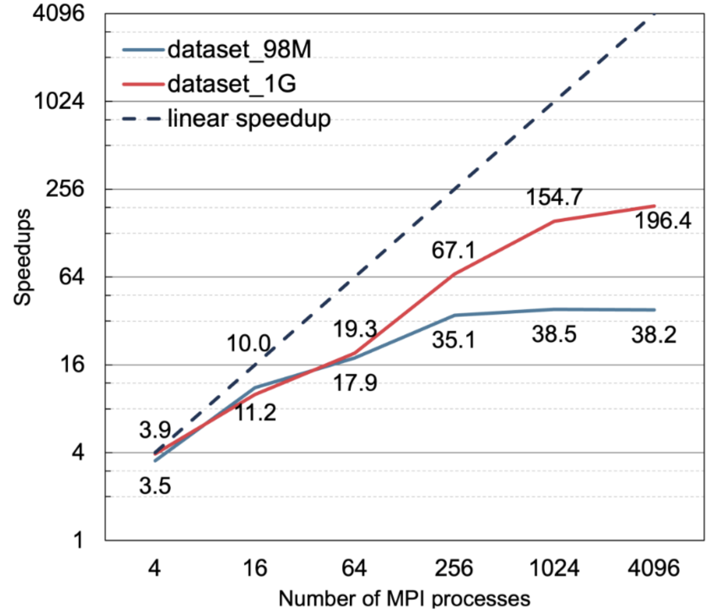
Timing breakdowns of metadata write time when using *dataset*_568k (top) and *dataset*_1G (bottom)

# Data Object Creation and Metadata Write Test

- Speedups of new header format approach




End-to-end metadata write time when using $dataset$_568k (top) and $dataset$_1G (bottom)

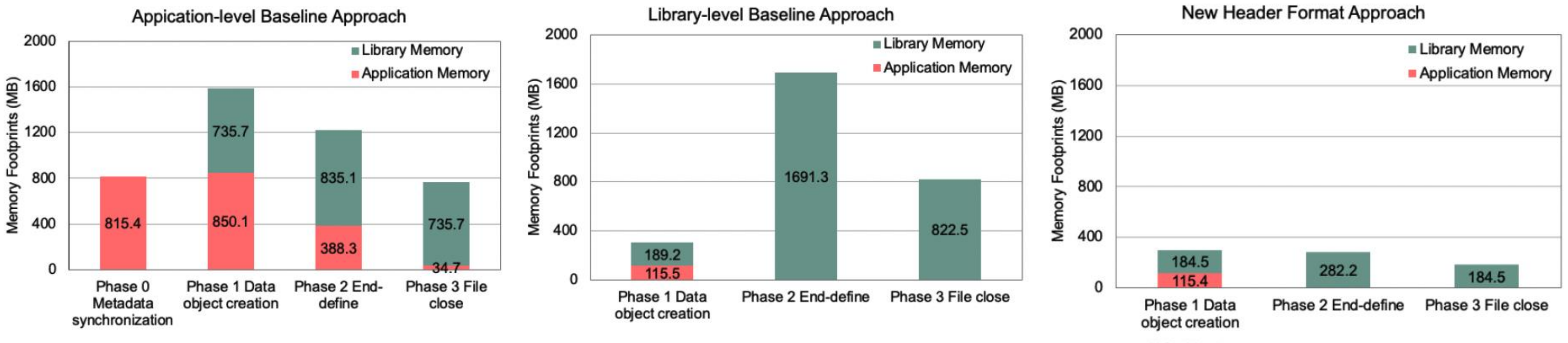



Speedups of the new header format approach

- A significant improvements for the new header format approach over other approaches
- Achieves a better scalability. Speedup of 196 is observed when running 4096 processes on the large dataset

# Data Object Creation and Metadata Write Test

- Memory Footprint
  - Memory consumption reaches its high watermark after metadata are synchronized and duplicated in the created data objects.
  - New file format approach consumes 25% of the memory used by baseline




Memory footprints for all proposed approached using $dataset\_98M$ running 4 MPI processes on 1 node. The values are the sum of memory footprints of all processes.

# Metadata Read Test

- Metadata Read Test
  - Weak-scaling experimental setting where each process reads the complete set of metadata
    - With the new file format approach, application can also selectively read metadata blocks
    - Input test file: 512 metadata blocks
  - The new header format shows read performance comparable to the classic format

| | *dataset_568k* | | | *dataset_5m* | | |
|---|---|---|---|---|---|---|
| No. of processes | 64 procs on 1 node | 256 procs on 4 nodes | 512 procs on 8 nodes | 64 procs on 1 node | 256 procs on 4 nodes | 512 procs on 8 nodes |
| Classic netCDF format | 1.38 | 1.58 | 1.60 | 15.25 | 16.36 | 16.57 |
| New header format | 1.81 | 2.02 | 2.28 | 15.38 | 15.45 | 15.80 |

Timings of reading the entire file header using the proposed new header format and classic format for two datasets.

# Conclusion

- Summary and Future Works
  - Data object creation can become a major bottleneck in large-scale workflows when each process generates unique data objects.
  - The new header format achieves scalable performance by enabling parallel metadata writes for non-shared data objects.
  - Future works may include optimizing library memory management and the refinement of new header format.

# Thank you!

# Appendix

- *BP5 Two level metadata aggregation and reduction reduced memory impact of collecting metadata and therefore is more scalable in terms of numbers of variables and writers than BP4*