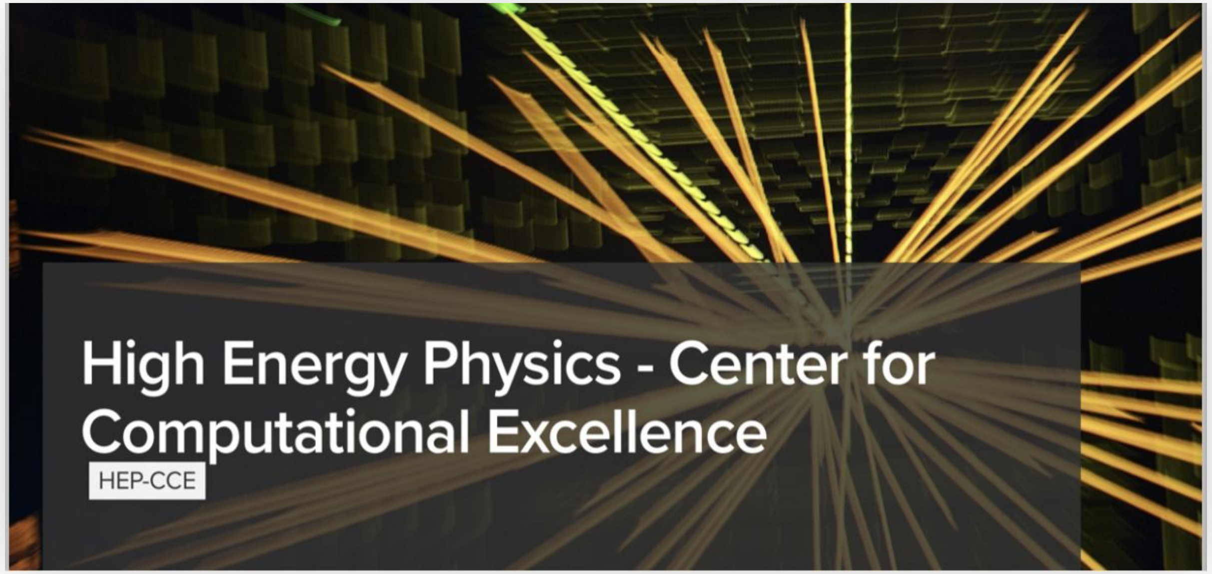
HEP-CCE & Future of HEP Computing

YSSS Symposium

Amit Bashyal
Argonne National Laboratory

December 5, 2023



HEP-CCE : A Brief Introduction

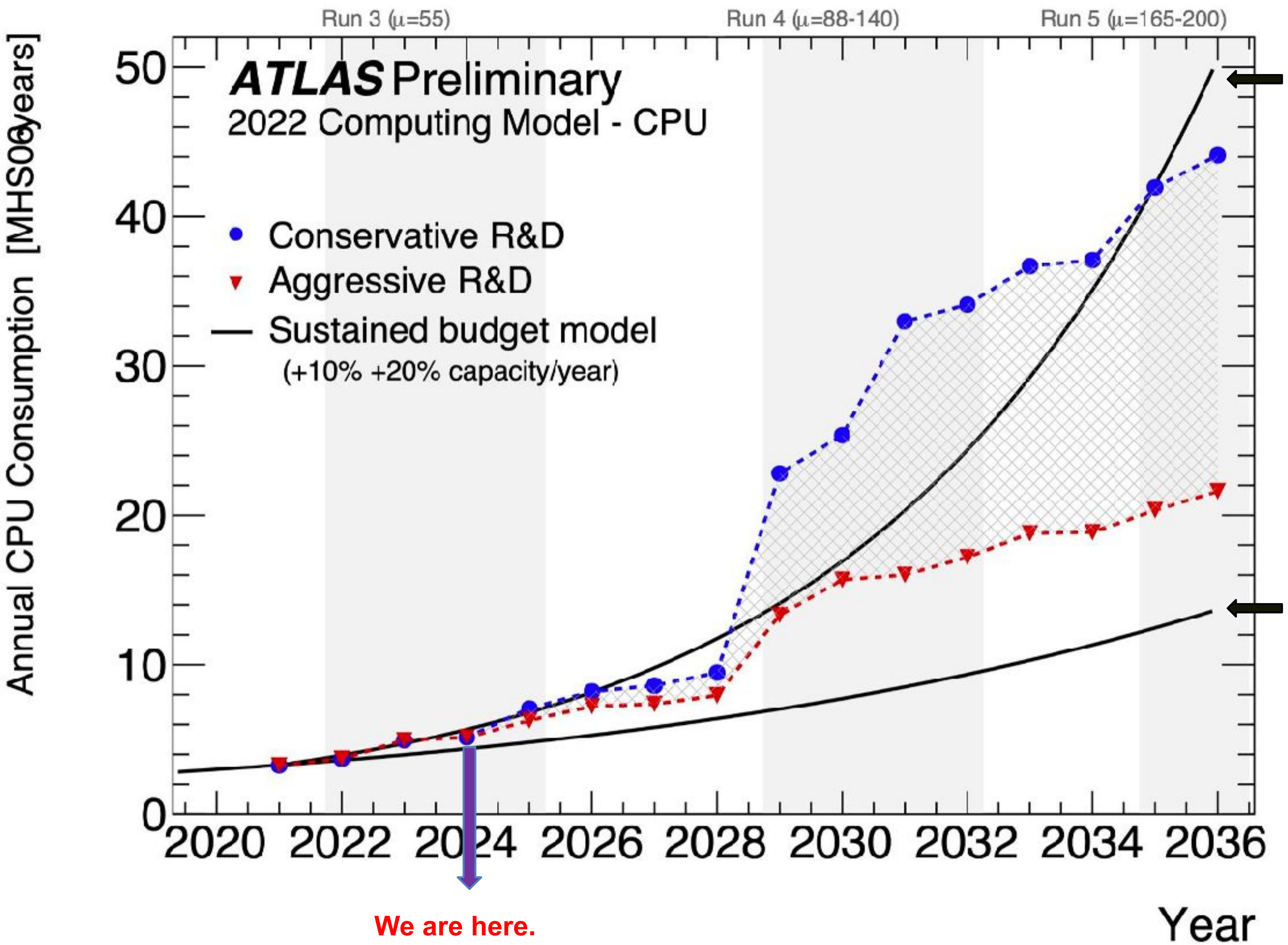
HEP-CCE (Center for Computational Excellence)

- 2020-2023 Pilot Project
- 6 Experiments (Energy, Intensity and Cosmic Frontiers) and HPC Experts*
- 4 US National Labs

Efforts:

- **IOS: Input/Output and Storage Studies on the HPC***
- PPS: Portable Parallel Strategy
- EG: Event Generators
- CW: Complex Workflows

ATLAS: Computational Needs in the Future



Improvement in computing capacity with 20% increase in R&D budget annually

Conservative: Existing person power level maintained at current level with current level of expertise.

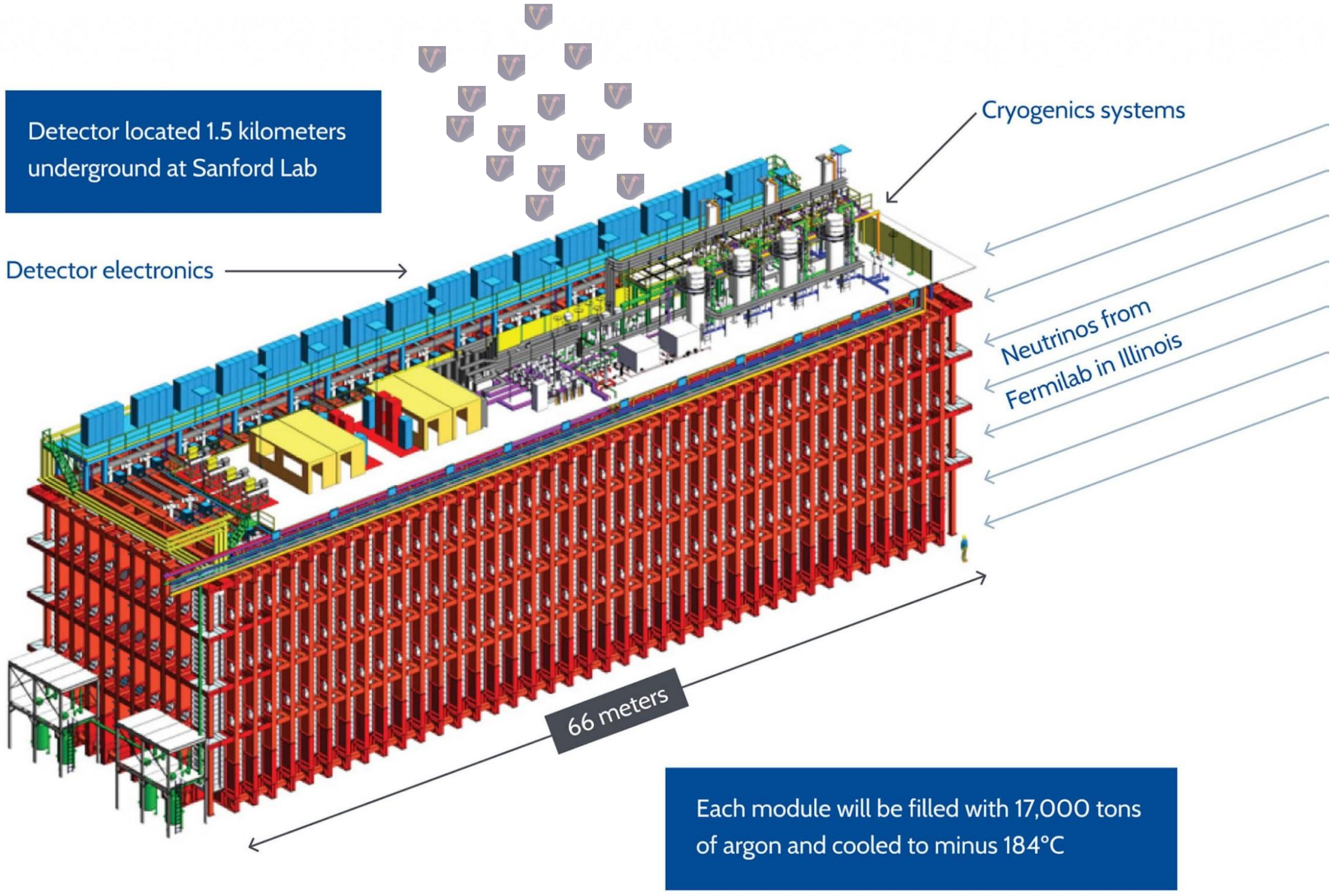
Aggressive: Increase in expertise

- Increase the person power
- Increase the expertise of existing person power

Improvement in computing capacity with 10% increase in R&D budget annually

DUNE: Challenges and Opportunities

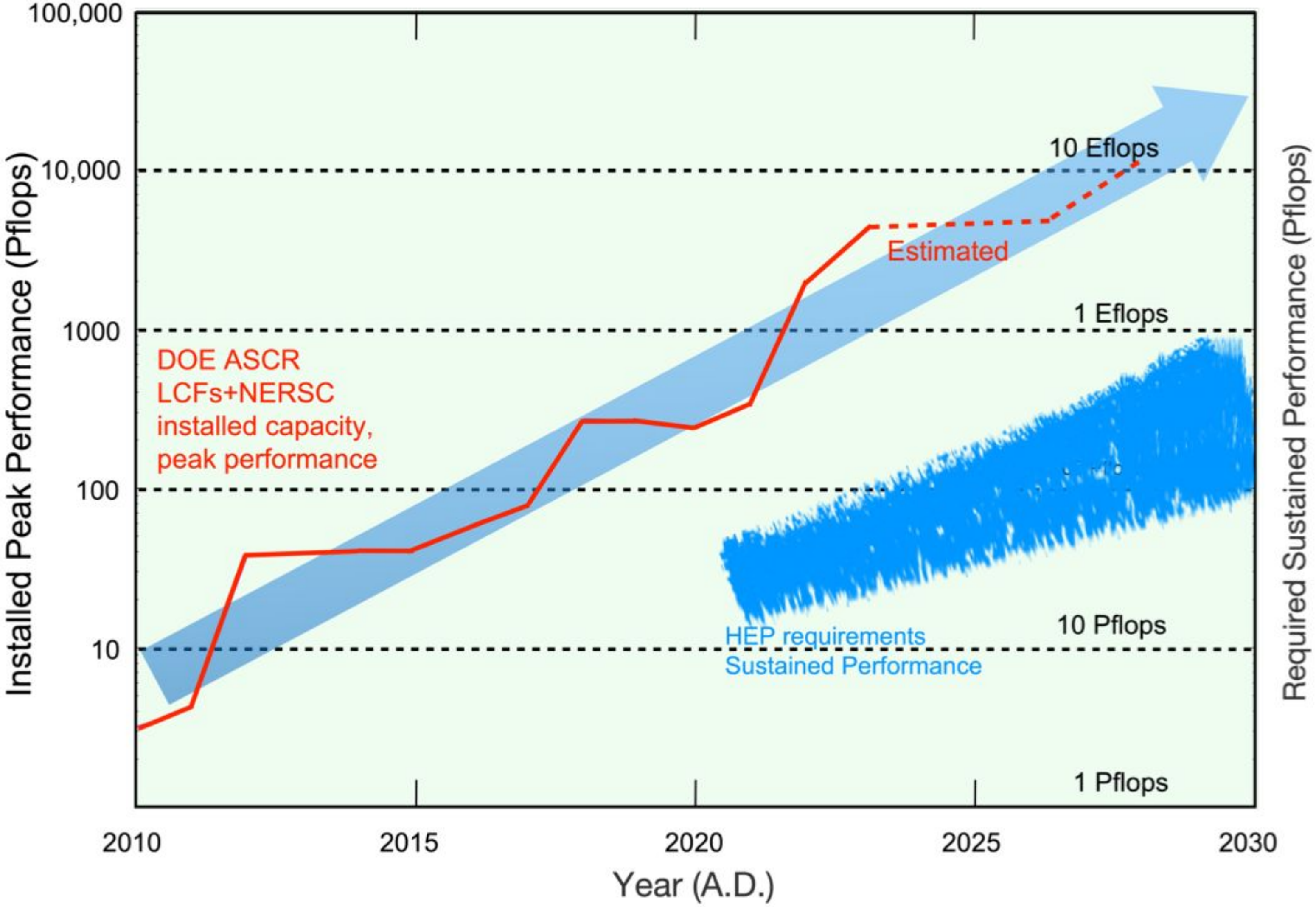
Super Nova induced event
$\approx 100\ TBs$ in ~100 seconds



Beam induced event readout size $\approx 6\ GB$

Argonne NATIONAL LABORATORY

Computing Resources for Future HEP Experiments



SOLUTION:
HPCs can fulfill the computing needs through the era of HL-LHC (Run 4) and DUNE.

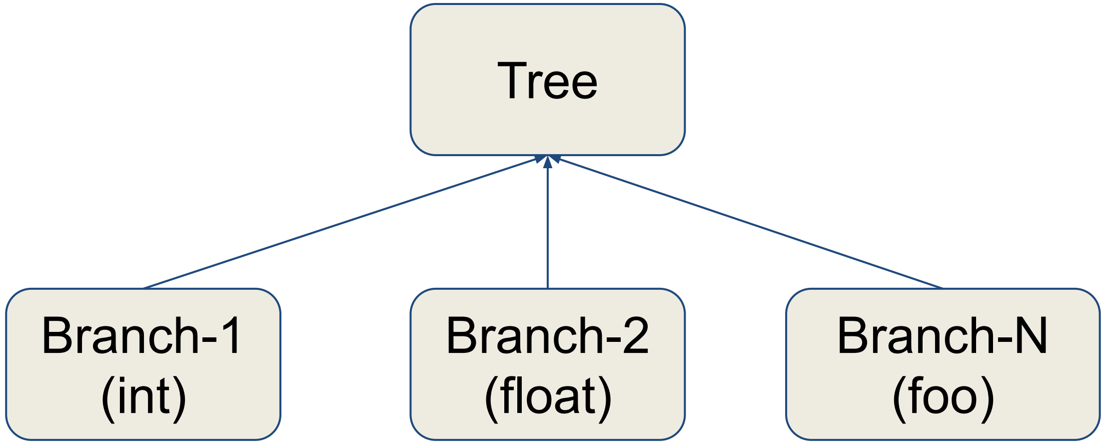
BUT

HEP software must be refactored appropriately to utilize HPC resources.

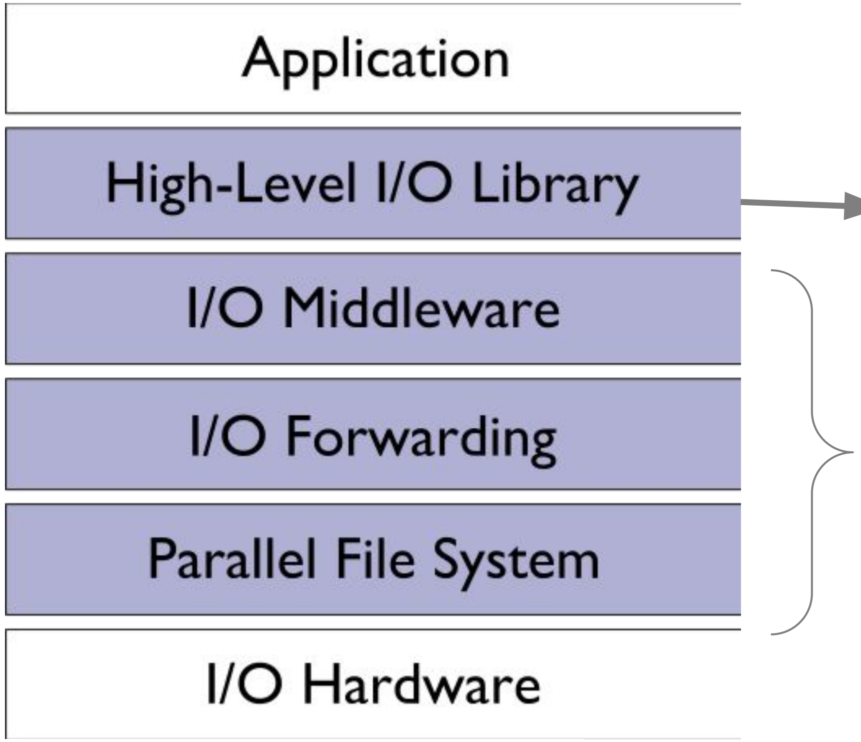
Argonne NATIONAL LABORATORY

HEP Data and ROOT Data Model

- **ROOT** has been workhorse of HEP experiments
  - Data processing, storage and analysis
- HEP **data models are complex**
  - Using C++ language features: pointers, inheritance, polymorphism
- Use ROOT to read and write data into **ROOT::TTree**
  - TTrees store **data of any types** (TBranch)
  - Use of **internal libraries, metadata handlings and functionalities** for efficient and scalable I/O

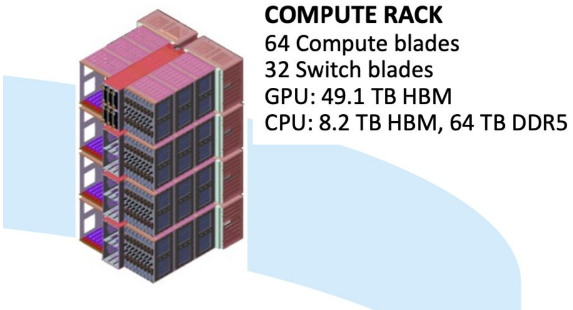


HPC Storage Systems



- **High Level libraries** (like HDF5) can hook into the HPC I/O stack.
- Allows us to take advantages of optimizations such as collective I/O (*collective I/O example in back up slides**)

HPCs have an established I/O software stack used to support parallel file system



Compute rack of Aurora

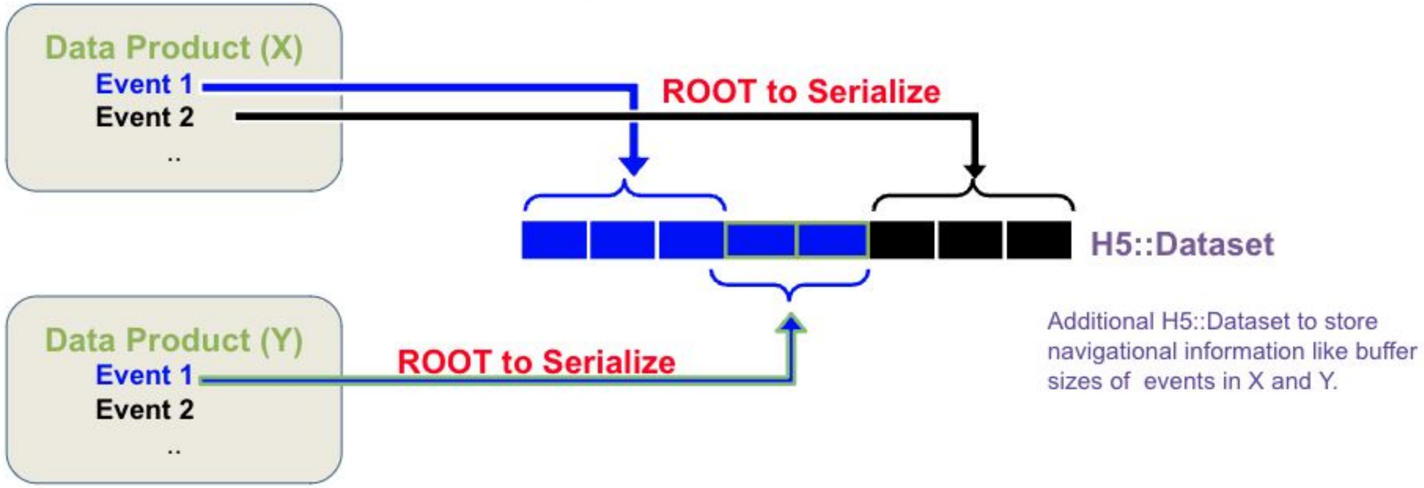
HPCs use **customized hardwares like GPUs or TPUs** designed for specific tasks like scientific simulations, AI/ML etc.

- Don't support complex data models that are usually supported by ROOT

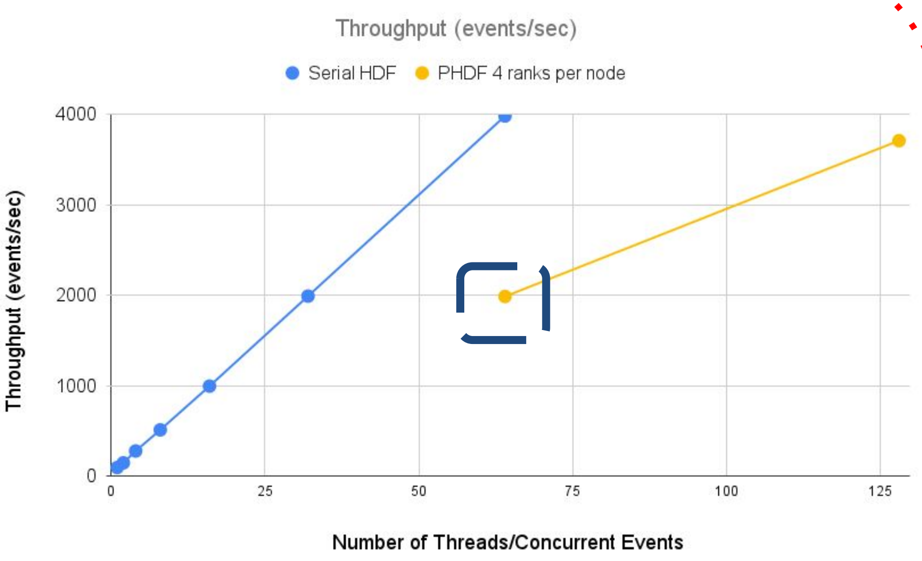
Argonne NATIONAL LABORATORY

HEP-CCE (IOS) Phase I : HPC Friendly Storage for HEP

- Develop an experiment agnostic framework to store HEP data in HPC friendly storage format (like HDF5) and perform I/O.
  - Snowmass White paper on the need of HPC friendly storage format (Link) and presented in snowmass CompF4 Topical Group meeting (Link)
  - Toy Framework to study and develop I/O routines to write HEP data into HDF5 and other formats (Link)
    - Developed algorithm to utilize HDF5 collective I/O (an optimized parallel I/O routine) to write HEP data in HDF5 format (Link)
    - I/O scaling tests done in the CORI @ NERSC
    - Results presented in CHEP 2023 (Proceeding submitted)



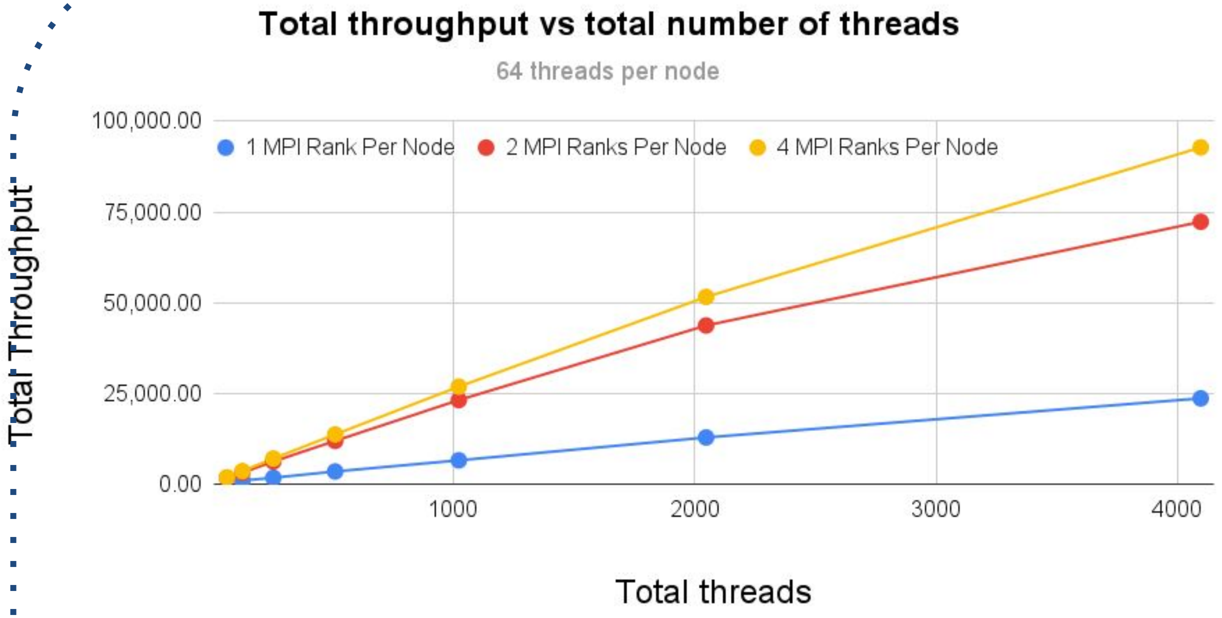
Parallel I/O with HDF5



Test done in a single node
Batch size of 100 events

Throughput = (Number of Events processed)/ (Application Run time)

For Parallel I/O:
4 parallel processes
Threads per Rank: #Threads/4

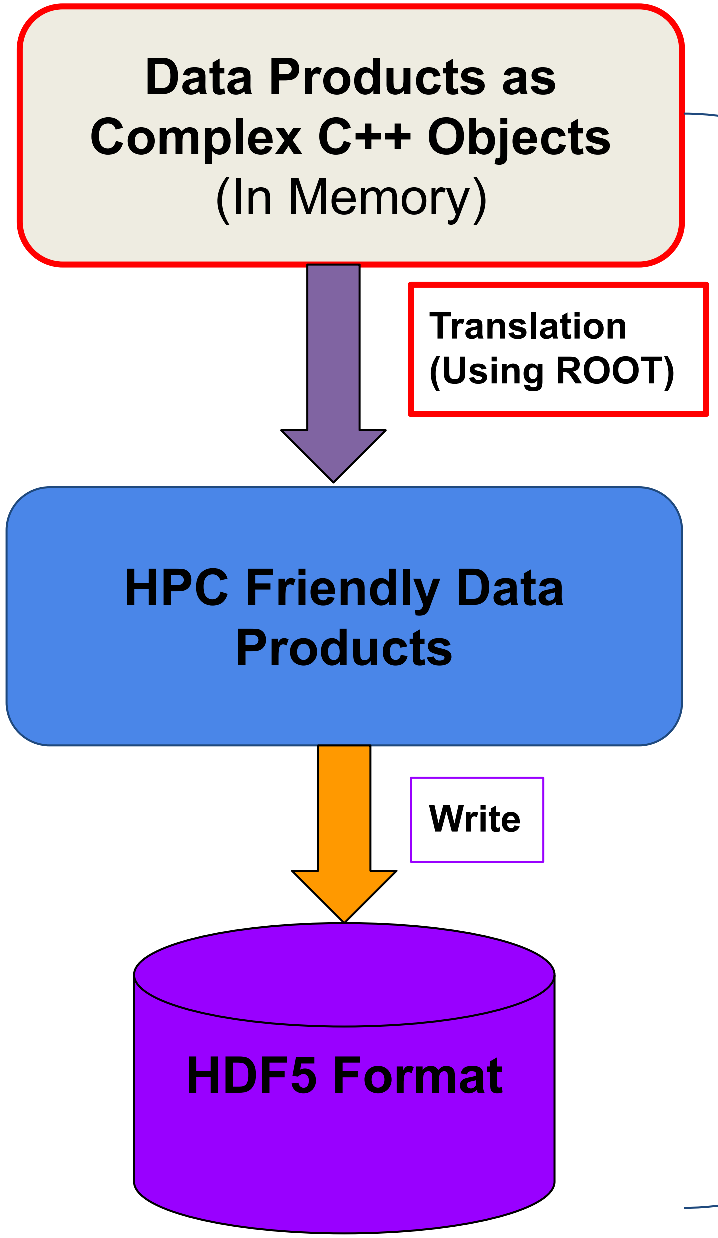


- **Total Throughput**: (Throughput per rank) X(MPI Ranks)
- Test with **64 threads per node**.

I/O Calls	Fraction of Total I/O Time
MPI calls (external to HDF5)	14%
Write data into HDF5 file	32%
Other (including serialization)	54%

Argonne NATIONAL LABORATORY

HPC Friendly Storage System



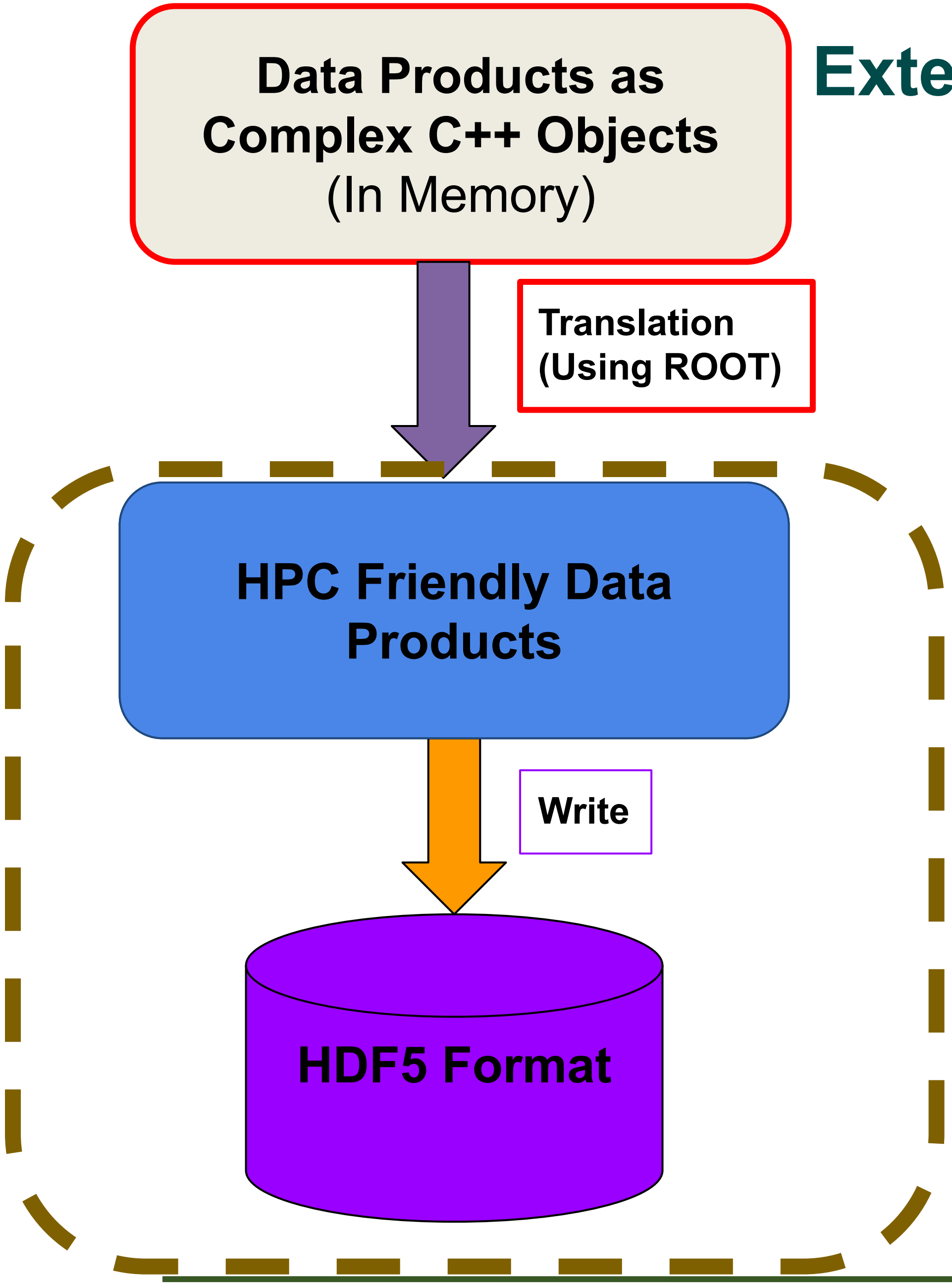
- Use ROOT to serialize HEP data products to make it HPC friendly.
- Collective writing of data into HDF5 file
- **HPC friendly storage but not data**
    - Data needs to be GPU friendly

HEP data needs to serialize/deserialize using ROOT.

Complex objects cannot be offloaded directly into the GPUs.

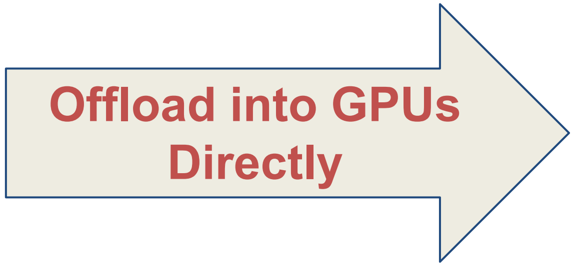
Argonne NATIONAL LABORATORY

Extension to Direct GPU Offloading

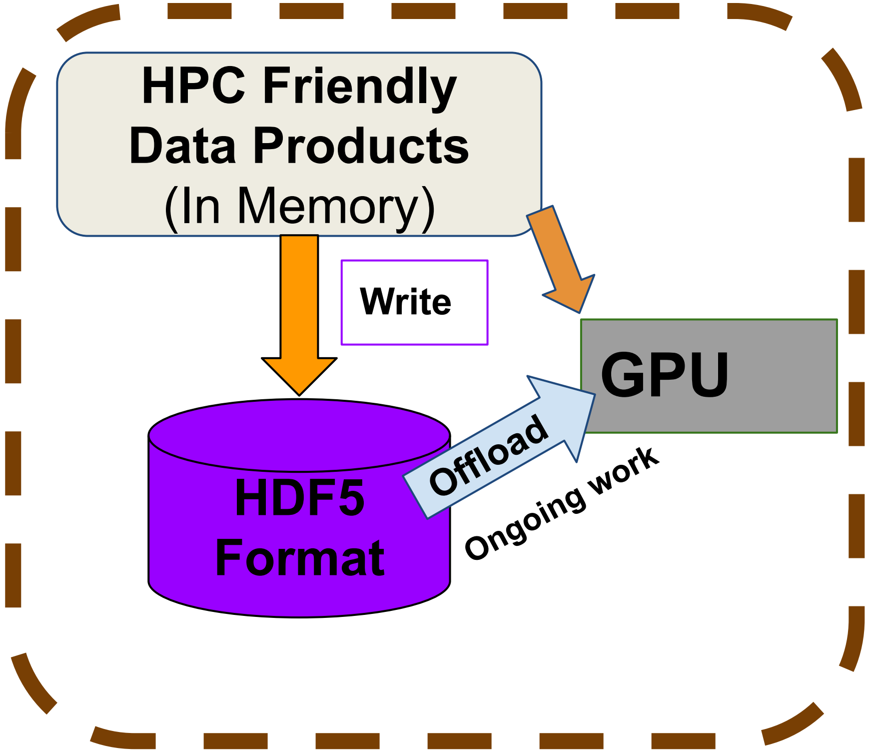


HEP data needs to serialize/deserialize using ROOT.

Complex objects cannot be offloaded directly into the GPUs.



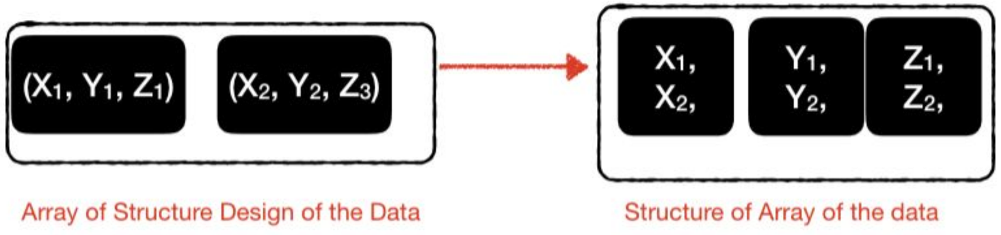
Design Data Model that is HPC (GPU) friendly



GPU Friendly HEP Data

- Conducted survey among HEP experiments (ATLAS, DUNE, CMS etc)
  - Understand the efforts made by experiments to make their data HPC friendly
  - **Common Challenges**
    - HEP data models are object oriented with complex data models optimized for traditional computing workflow
    - Design based on experimental needs and computational technology at that time
  - **Common Solutions**
    - Utilization of Arrays, nested arrays, (Ao)SoA
    - Experiments apply these common solutions according to their use case and experimental needs
  - Survey results as one of the deliverables of first iteration of HEP-CCE and basis for second iteration of HEP-CCE effort

Black boxes represent contiguous memory layout



Outlook in Second Iteration of HEP-CCE (Ongoing and Planned Works)

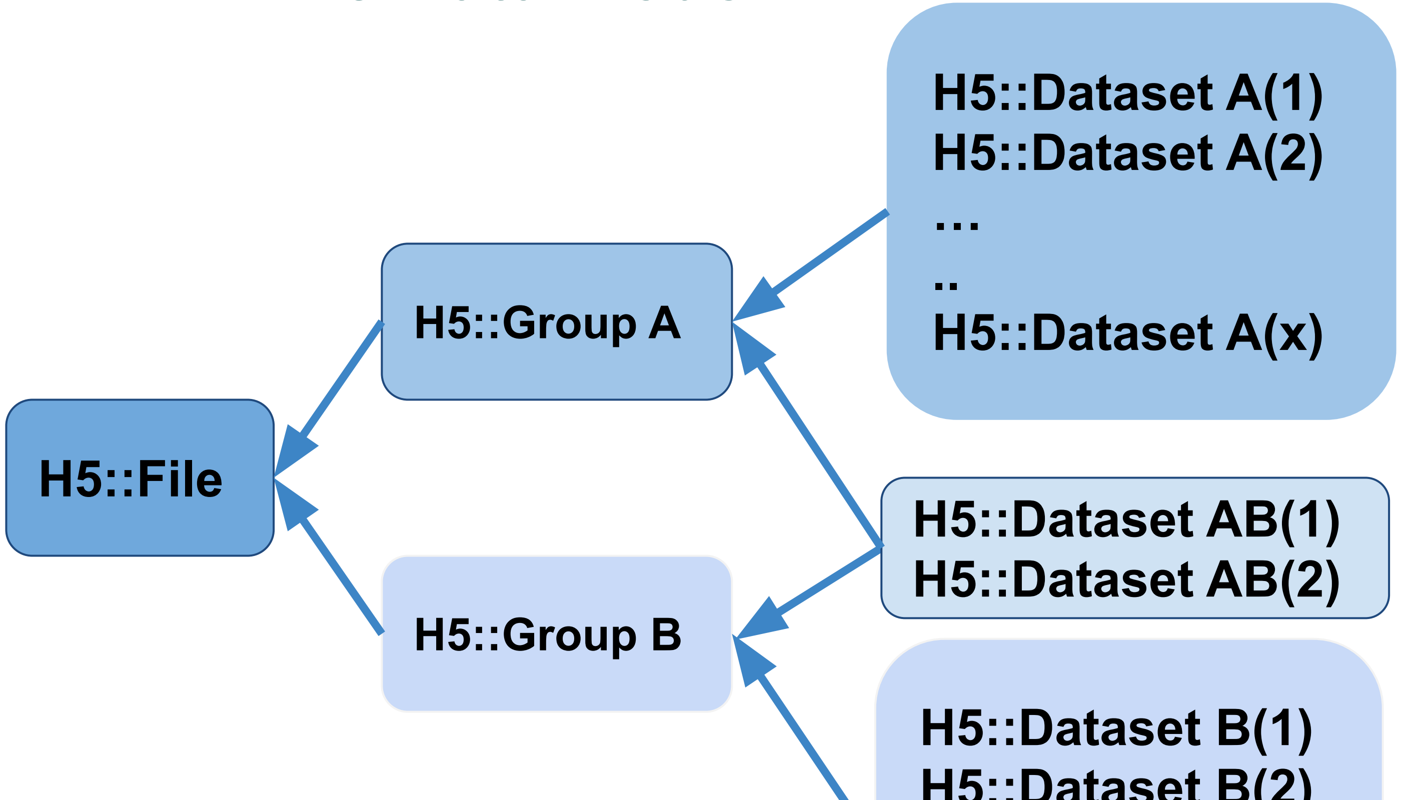
- Development of GPU Friendly data model (experiment agnostic) with the framework that mimics I/O in both host and device (Link)
  - Structure of Arrays data model based
  - Initial tests with ProtoDUNE Trigger Data model and CAF Data
- RNTuple will replace TTree as the primary I/O and storage system in ROOT
  - Limited support for data models
  - Ideal to synchronize the GPU Friendly data model effort with RNTuple
  - Development of data models that can be offloaded in GPUs and persisted in both RNTuple and HDF5 (or other HPC Friendly storage system) (Link)
  - Optimize Data storage with tuned I/O patterns better suited for HPC platforms

*There are many things happening in many fronts in HEP-CCE. This talk highlights the works that I led or where I played significant role.

THANK YOU!

BACK UP

HDF5 Data Model



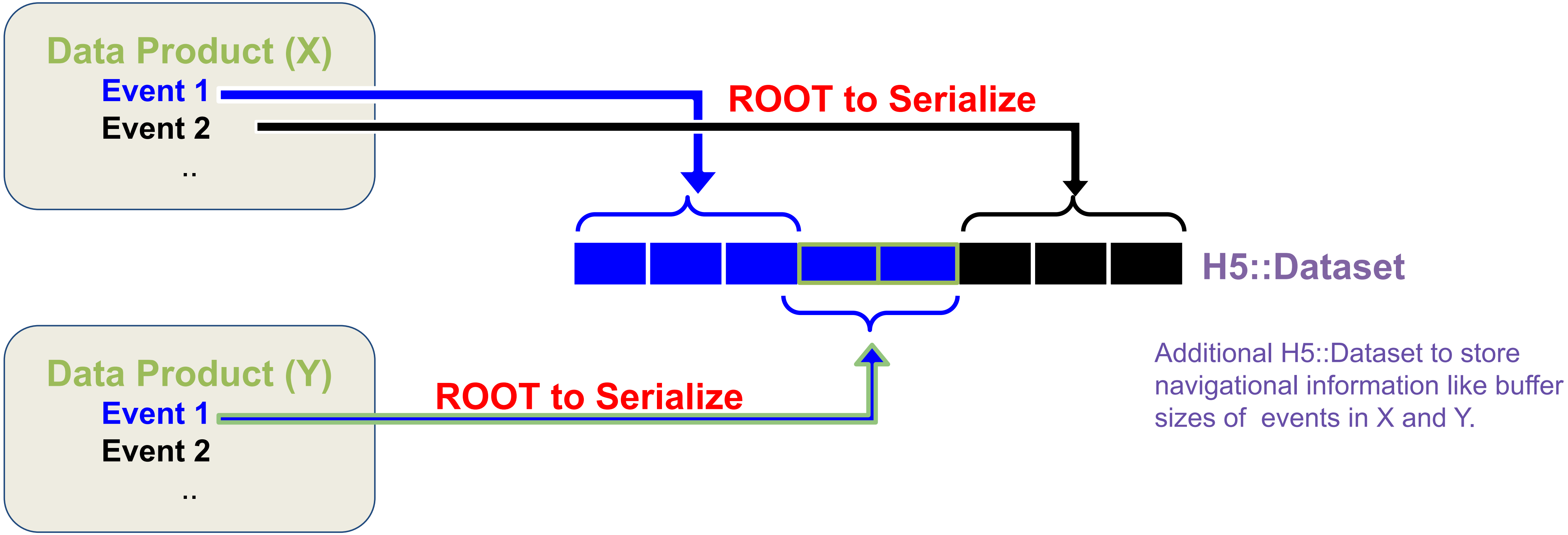
File stores the data.
Groups are used to organize data objects.

- Data written in **Datasets**.
- **Datasets** can be:
  - **Grouped** together to organize data objects
  - **Shared** among groups
- Store H5::Attributes for metadata

- **MPI libraries** implemented to perform **parallel I/O** on the HDF5::Datasets

HDF5 File needs to be opened with the MPI Flag to enable the parallel I/O.

HDF5 as Data Storage Format

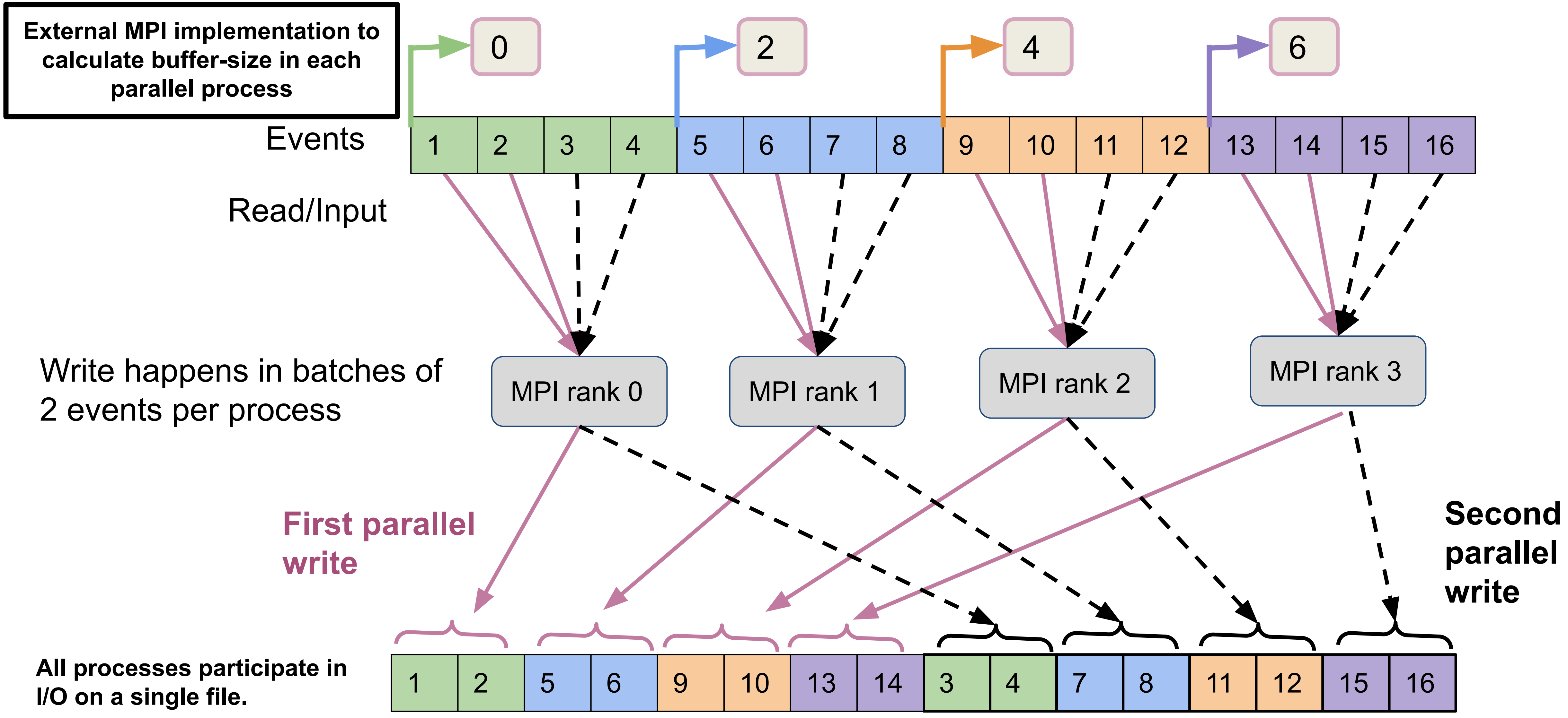


Data Products are experiment specific C++ objects usually written in ROOT format.

Use **ROOT** as common tool to serialize C++ objects into byte stream array buffers

HDF5 Datasets store serialized data products with mapping optimized for parallel I/O. Mapping is independent of experiments.

Parallel (Collective) I/O using HDF5



Argonne NATIONAL LABORATORY

POSIX (TOP) and STDIO OVER-VIEW (BOTTOM)

RNTuple

files accessed	8
bytes read	3.19 KiB
bytes written	0 Bytes
I/O performance estimate	7.72 MiB/s (average)

Overview

files accessed	8
bytes read	8 Bytes
bytes written	50.66 MiB
I/O performance estimate	77.29 MiB/s (average)

TTree

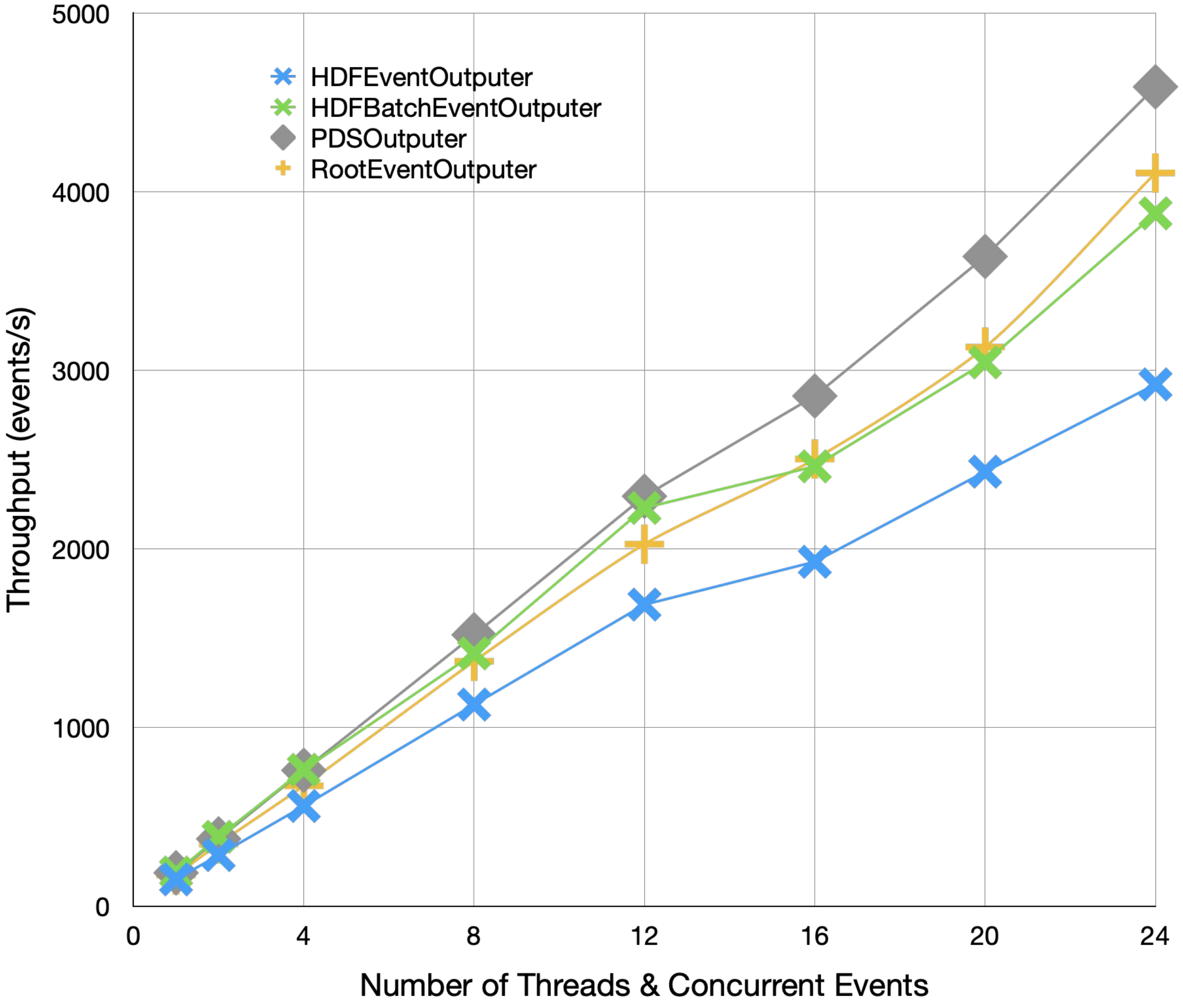
Overview

files accessed	9
bytes read	3.19 KiB
bytes written	50.46 MiB
I/O performance estimate	1437.81 MiB/s (average)

Overview

files accessed	6
bytes read	8 Bytes
bytes written	0 Bytes
I/O performance estimate	0.04 MiB/s (average)

I/O Performance Comparison



I/O performance of the toy framework is shown in various output modes including ROOT.

Study was done in CORI Machine.