# Results from the High Energy Physics Center for Computational Excellence HEP-CCE/IOS and Future Plans

Amit Bashyal, Peter van Gemmeren, Rui Wang (ANL) **for HEP-CCE**

***HSF HCAF***
June 14, 2023

# Input/Output and Storage: Activities

## Measuring performance of ROOT I/O in HEP workflows on HPC systems

- Darshan a scalable HPC I/O characterization tool has been enhanced (including fork safety) and used to monitor HEP production workflows.

## Investigate HDF5 as intermediate event storage for HPC processing

- Relying on ROOT to serialize complex Event Data Model used in Simulation/Reconstruction workflows
- Implementing Collective Writing to avoid potential merge step
- Mimicking framework for understanding scalability and performance of HEP output methods
  - Experiment agnostic tool allows scaling I/O beyond what is currently accessible by production and has uncovered/fixed bottlenecks in ROOT and frameworks.

## HPC friendly Data Model

- Together with PPS team started investigating efforts to make data model more suitable for offloading to accelerator and storage on HPC.

# I/O and Storage in the HPCs

- Differences in **HTC** (High Throughput Computing) and **HPC** (High Performance Computing) resources → Cannot directly move HEP computing workflow into HPCs
- HEP-CCE I/O and Storage studies the HEP general computing framework in the HPCs.
  - **Storage**: Writing data in storage format supporting parallel I/O
  - **I/O**: Performing parallel I/O on HEP data with minimal changes on existing computing workflow
  - **Optimization**: Tuning of parallel libraries to optimize the performance
  - **Data Mapping**: I/O performance based on various ways data is written in HPC friendly format

- Test-framework development
  - **Experiment agnostic**: Should work for common HEP data models
  - **Parallel I/O** of the HEP data using MPI (Message Parsing Interface) and HDF5 libraries
  - **Multi-threading** using TBB libraries
  - **HDF5 and MPI parameters** tuning to optimize I/O and storage

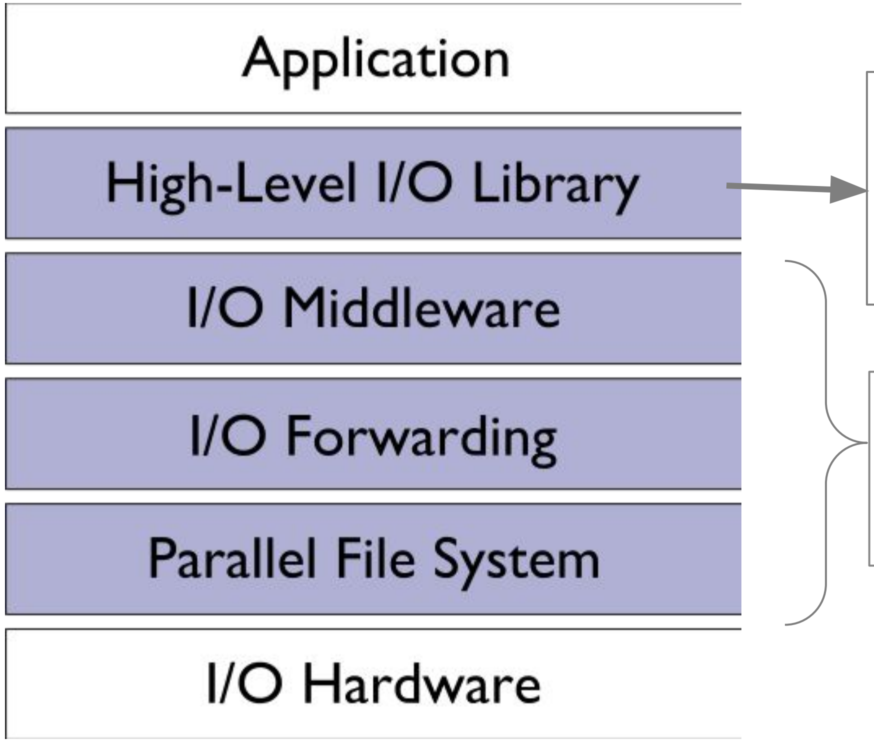


U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
Brookhaven National Laboratory
Fermilab
BERKELEY LAB Bringing Science Solutions to the World

# HEP Requirements for the HPC Storage Systems



HPCs use **parallel file systems** for data-storage and access.

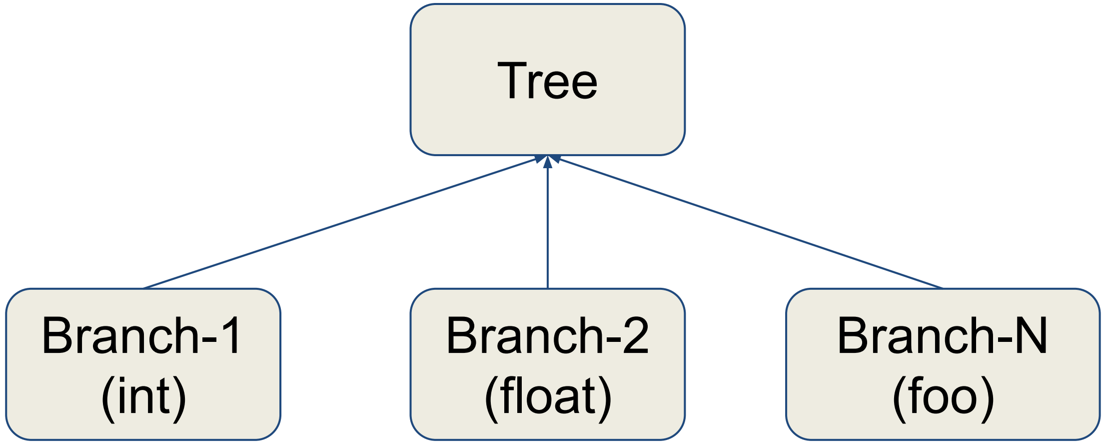



- High Level libraries can hook into the HPC I/O stack.
- Allows us to take advantages of optimizations such as collective I/O

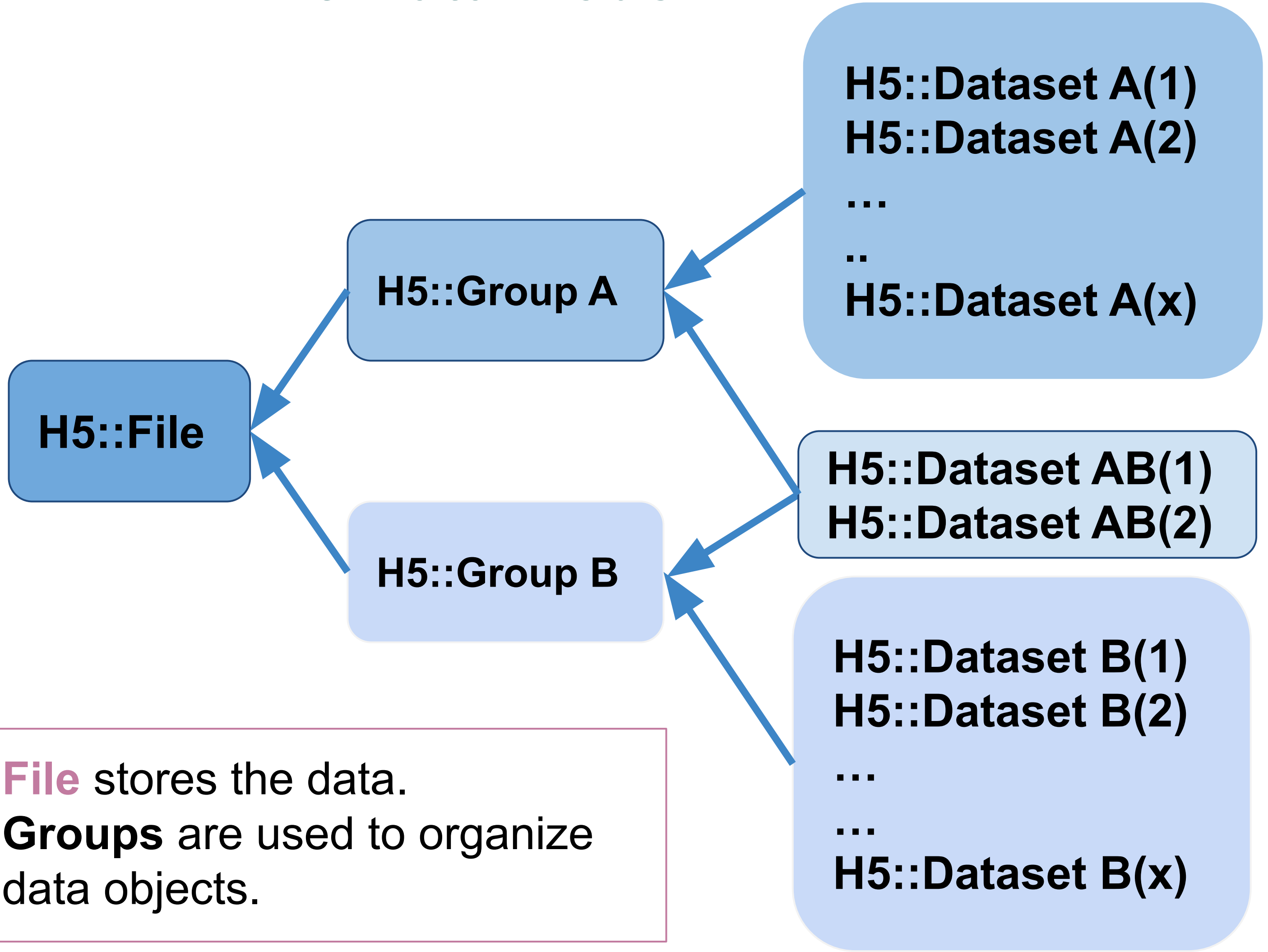
HPCs have an established I/O software stack used to support parallel file system

# 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




# HDF5 Data Model



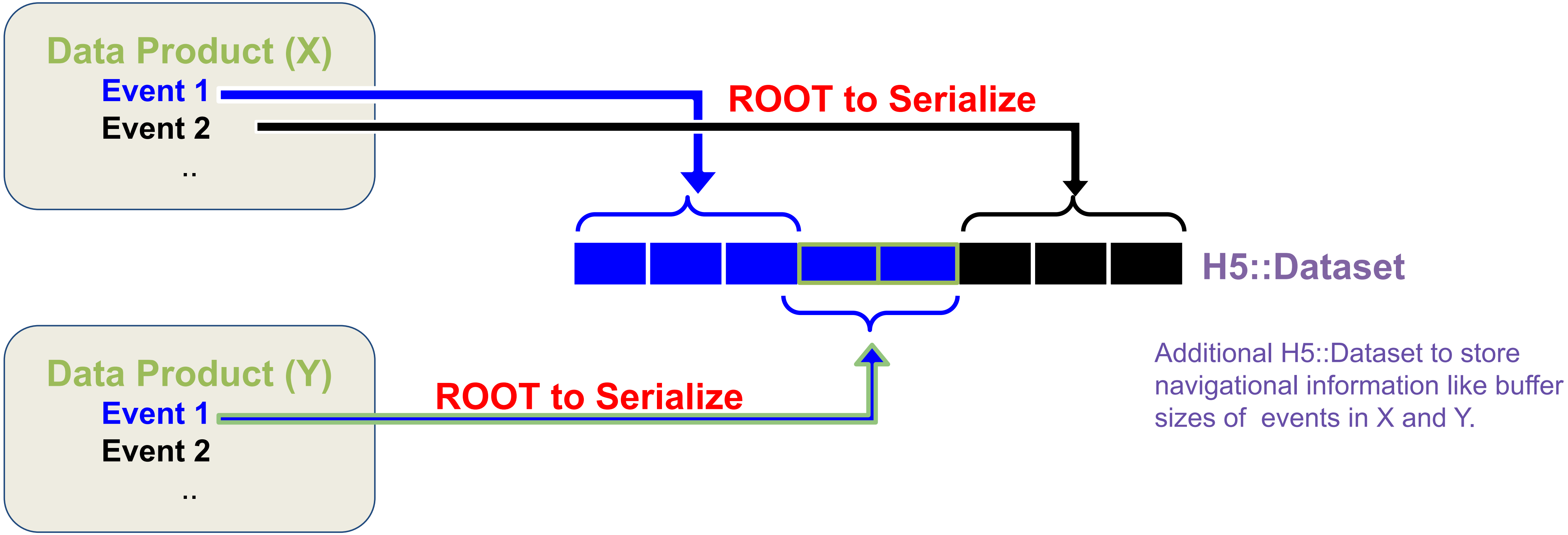

- Data written in **Datasets**.
- **Datasets** can be:
  - **Grouped** together to organize data objects
  - **Shared** among groups
- Store H5::Attributes for metadata

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

- **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

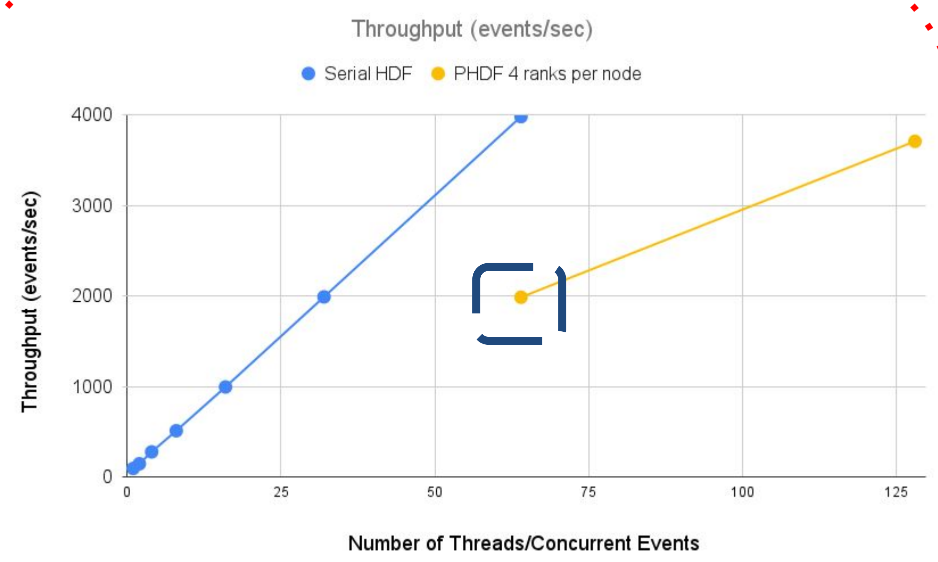



**Data Products** are experiment specific C++ objects usually written in ROOT format.

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.

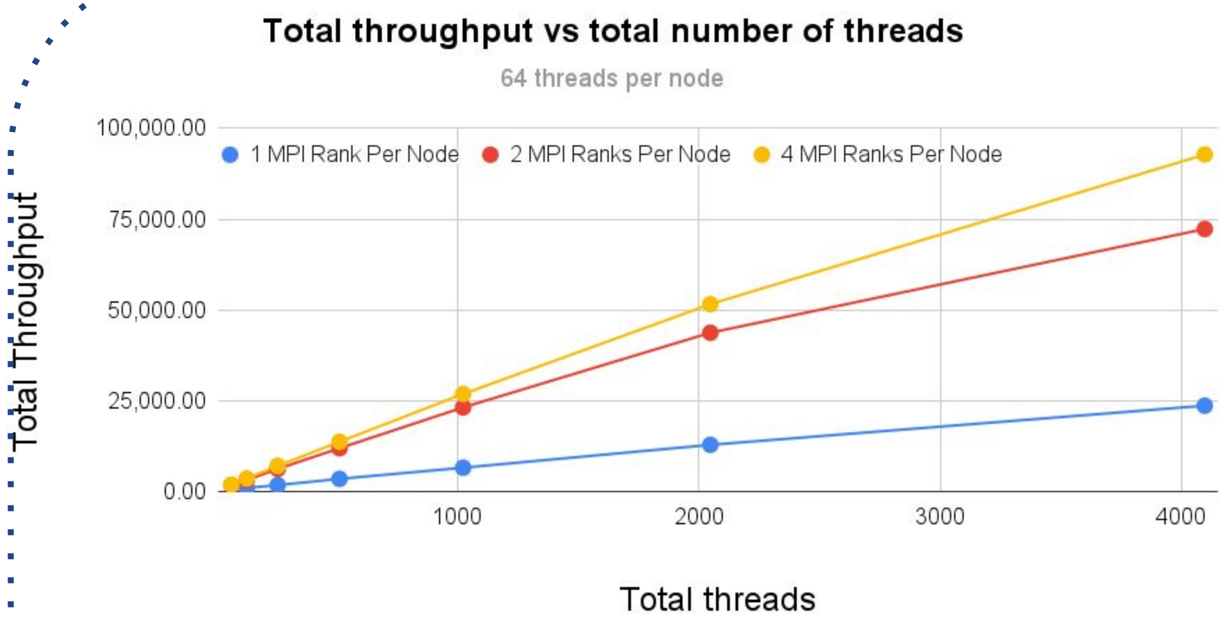
# Parallel I/O with HDF5




**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% |



U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
Brookhaven National Laboratory
Fermilab
BERKELEY LAB Bringing Science Solutions to the World

# Darshan

- Darshan is a lightweight I/O characterization tool that captures concise views and entire traces (DXT) of applications' I/O behavior

- *Widely available* – Deployed (and commonly enabled by default) at many HPC facilities
  - LCFs, NERSC, etc. and CVMFS

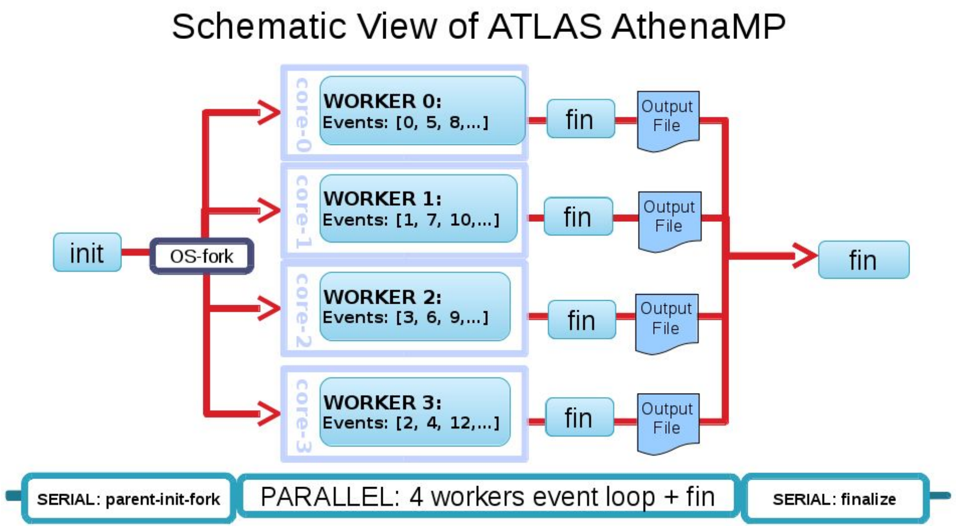
- Has become a popular tool for HPC users to better understand their I/O workloads
  - *Easy to use* – no code changes required
  - *Modular* – straightforward to add new instrumentation sources

https://www.mcs.anl.gov/research/projects/darshan/

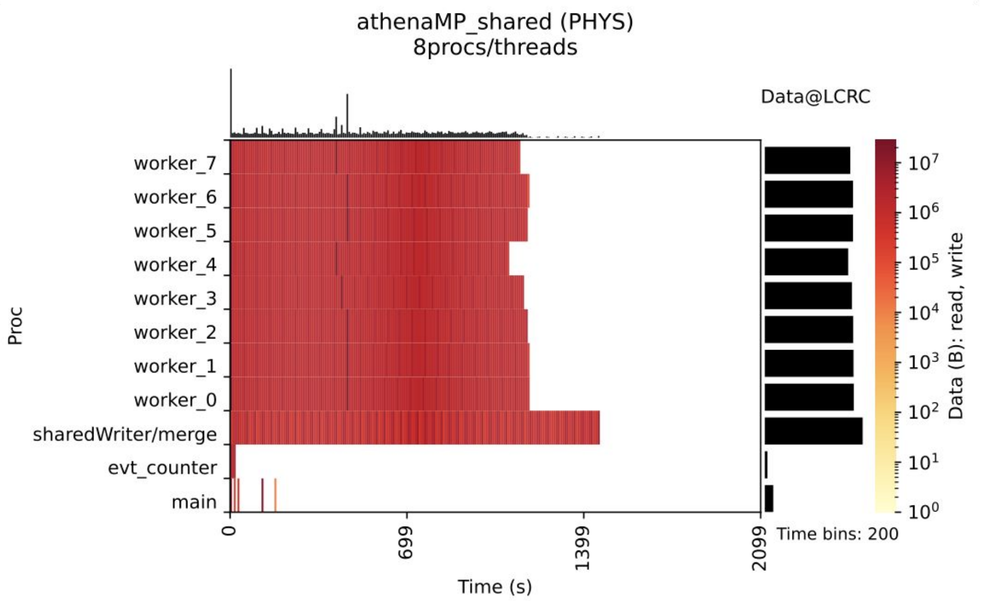
# Darshan enhancements for HEP use case

- ❖ Originally designed specifically for message passing interface (MPI) applications, but recently we have modified Darshan to also work in non-MPI contexts
  - ➢ HEP workflows are traditionally not been based on MPI
  - ➢ In recent Darshan versions (3.2+), any dynamically-linked executable can be instrumented

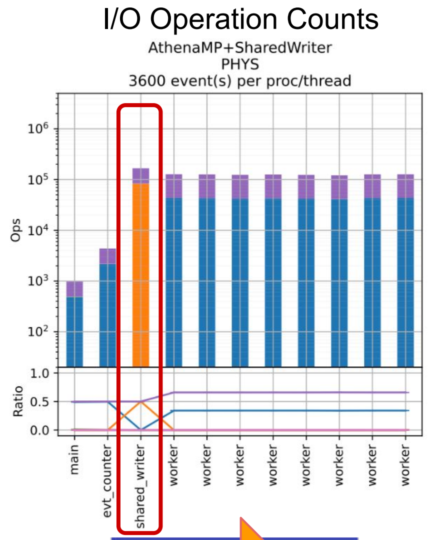
- ❖ Ability to instrument the forked processes
  - ➢ AthenaMP (multi-process offline software of ATLAS) creates parallel workers which are forked from the main process

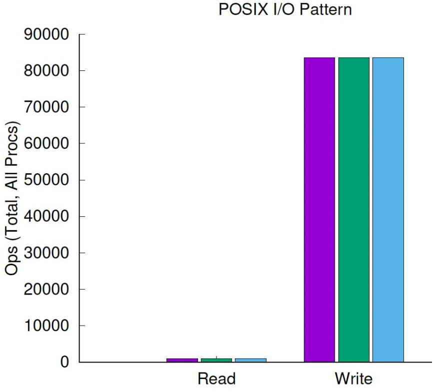



https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults



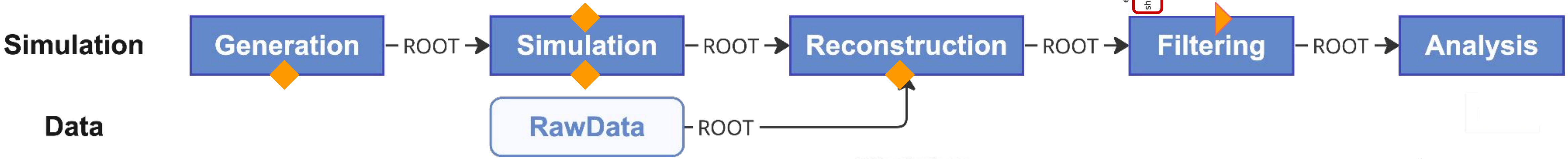
# Case study: I/O operations



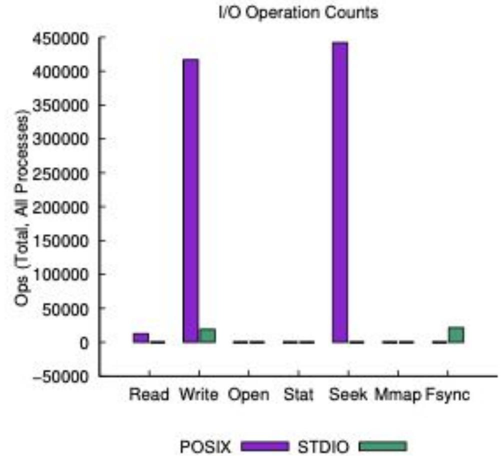




Broadwell on LCRC@ANL
GPFS
SDCC@BNL
GPFS

- **Equal number of writes/seeks**
  - Generation & Simulation & Reconstruction & SharedWriter process in Filtering stage at ATLAS (marked)




Haswell on Cori @Nersc
SSD + Lustre
**100 events, 16 threads**

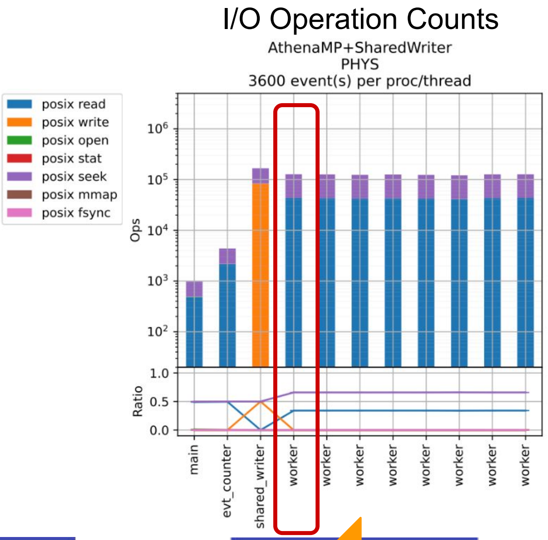


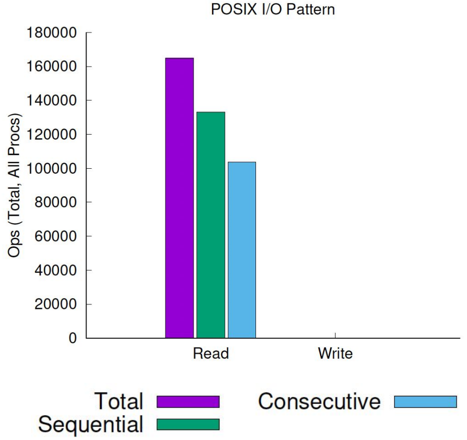

- **Equal sequential & consecutive I/O**
  - Sequential – next access came somewhere after the last one in the file
  - Consecutive – next access starts with the byte immediately following the last access

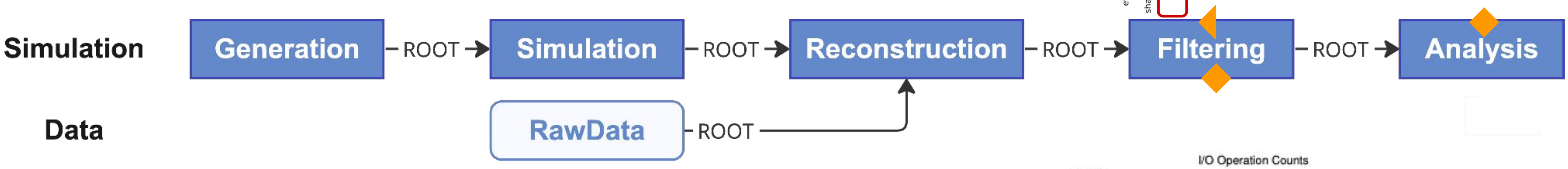
# Case study: I/O operations

Broadwell on LCRC@ANL
GPFS
SDCC@BNL
GPFS

- **Seeks > reads**
  - Filtering stage (worker process at ATLAS)
  - Analysis stage

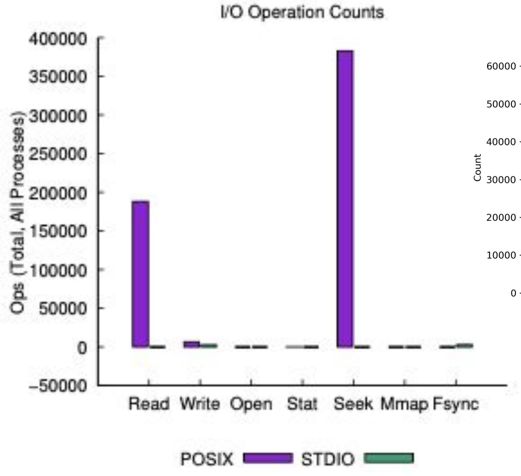


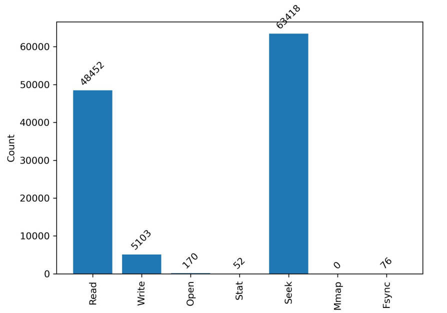




Haswell on Cori @Nersc
SSD + Lustre
**100 events, 16 threads**

- **Sequential > consecutive I/O**
  - Sequential – next access came somewhere after the last one in the file
  - Consecutive – next access starts with the byte immediately following the last access

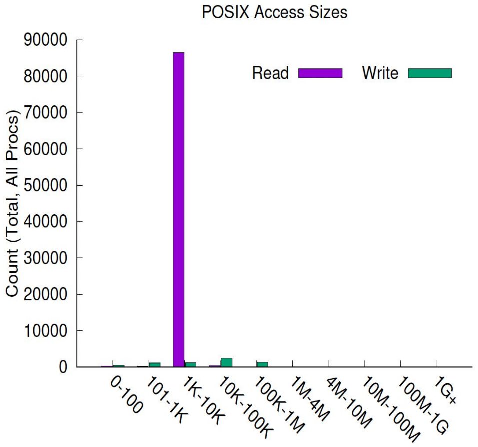


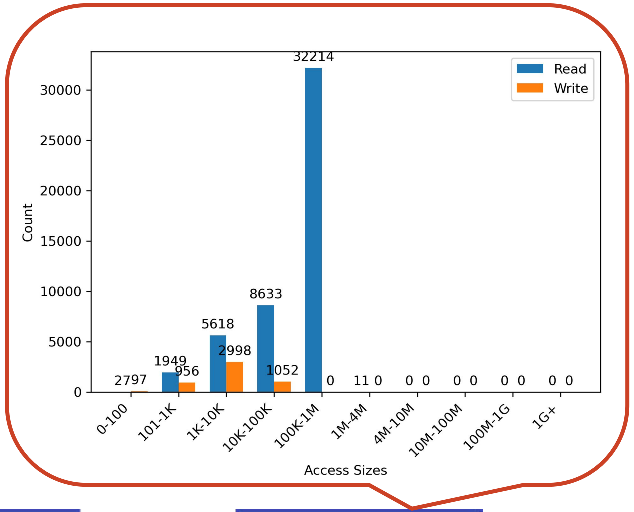



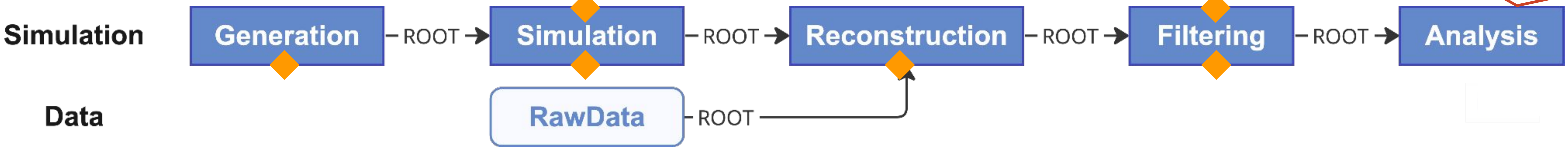
# Case study: Access size

ATLAS EXPERIMENT

Broadwell on LCRC@ANL
GPFS
SDCC@BNL
GPFS

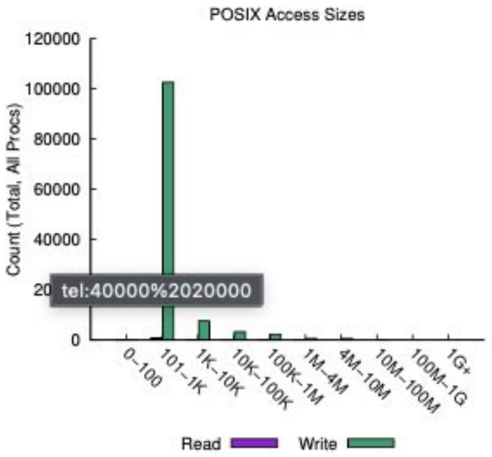



CMS

Haswell on Cori @Nersc
SSD + Lustre
**100 events, 16 threads**



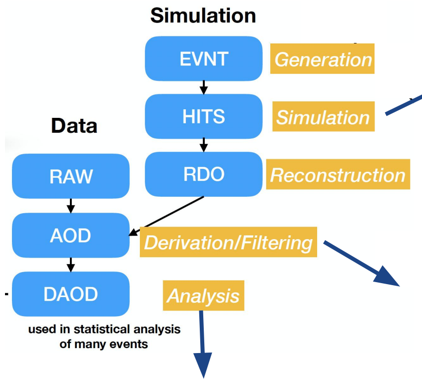

**Small reads/writes at O(1KB)**

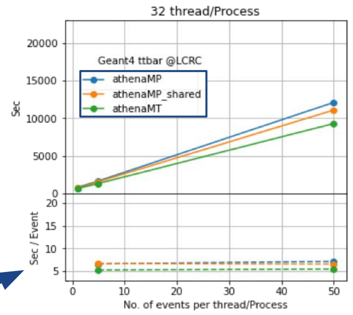
- All stages (marked) except ATLAS Analysis which is at O(100KB)
- Related to ROOT TTreeCache vector I/O support on certain FSes
- Potential bottleneck
- ROOT has a data sieving concept (overread) that might be taken advantage of

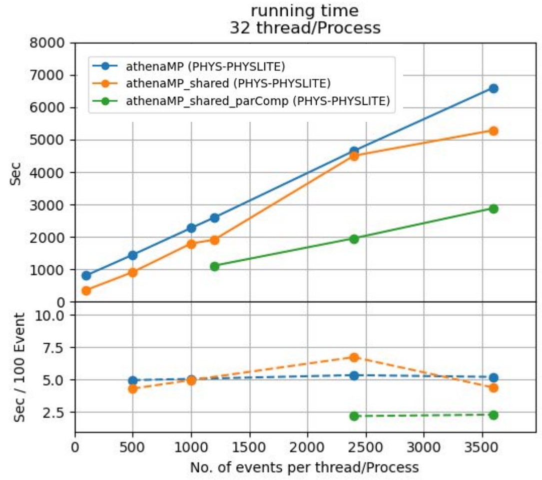
# I/O and Storage: Recommendations

Work of the HEP-CCE/IOS team has resulted in

- Worthwhile insight to I/O behavior of HEP workflows
  - Including on HPC and for scales beyond current production.
- Fixes/enhancements to common software and experiments frameworks
  - Darshan included fork-safety and better filtering for I/O.
  - ROOT serialization bottleneck was fixed.
  - Patch to resolve the Athena library issue on DSO loading hooks which cause PyRoot crash when running with Darshan
- Prototype development of new functionality in collaboration with experiments:
  - ATLAS developed functionality to store their production data in HDF5




```
xAH_run.py --files InputFileList.txt
--inputList --nevents=0 --config
../sources/sh4b/config/minimal_commonCP.
py --daodPhys --submitDir
./sh4b-InputFileList-test --inputTag
*DAOD* --isMC --nevents=405000 direct
```

| | |
|---|---|
| Job ID | 19765 |
| # Processes | 1 |
| Run time (s) | 1318.6418 |

Darshan Monitoring of different ATLAS workflow steps



U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
Brookhaven National Laboratory
Fermilab
BERKELEY LAB Bringing Science Solutions to the World

# PPS and IOS: Next Steps

Finalize and publish results and recommendations

Meet with stakeholder experiments to present conclusions

- General meetings, seminars and focused workshops

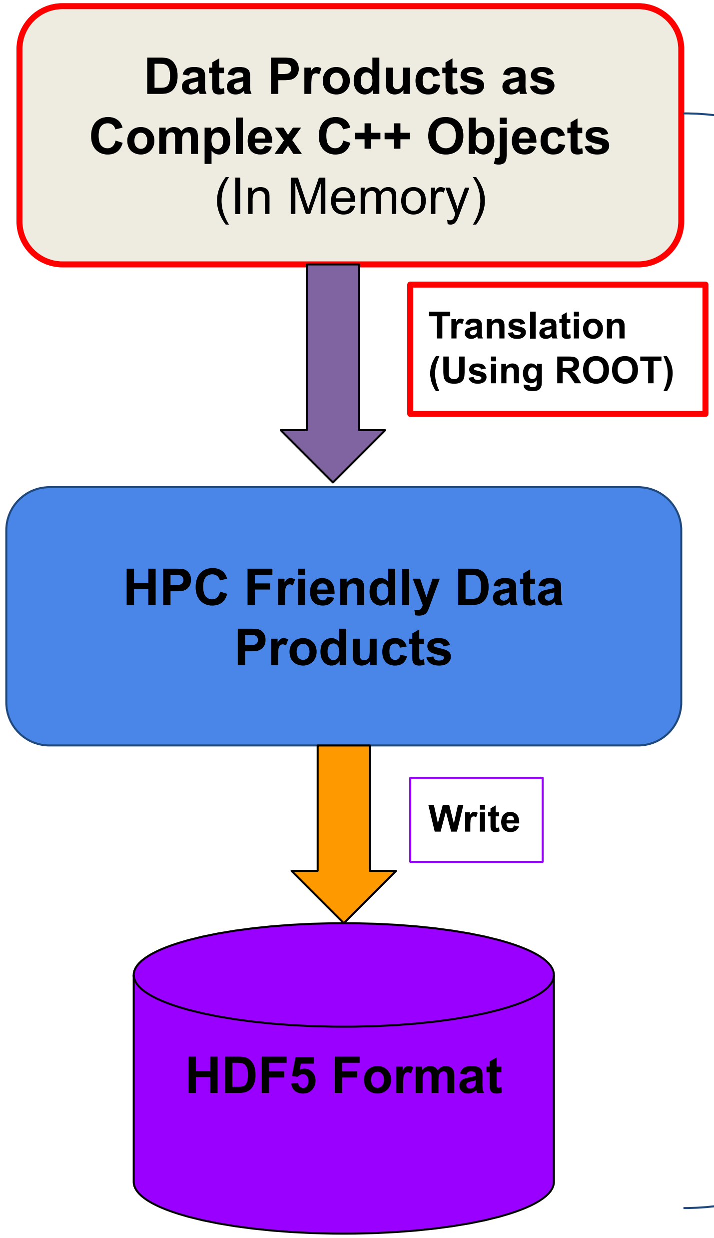
Present to HEP community via larger forums and external partners

- HSF
- WLCG
- IRIS-HEP
- OpenLab

Outreach to other experiments to transfer knowledge and experiences

# Continuing HEP-CCE IOS

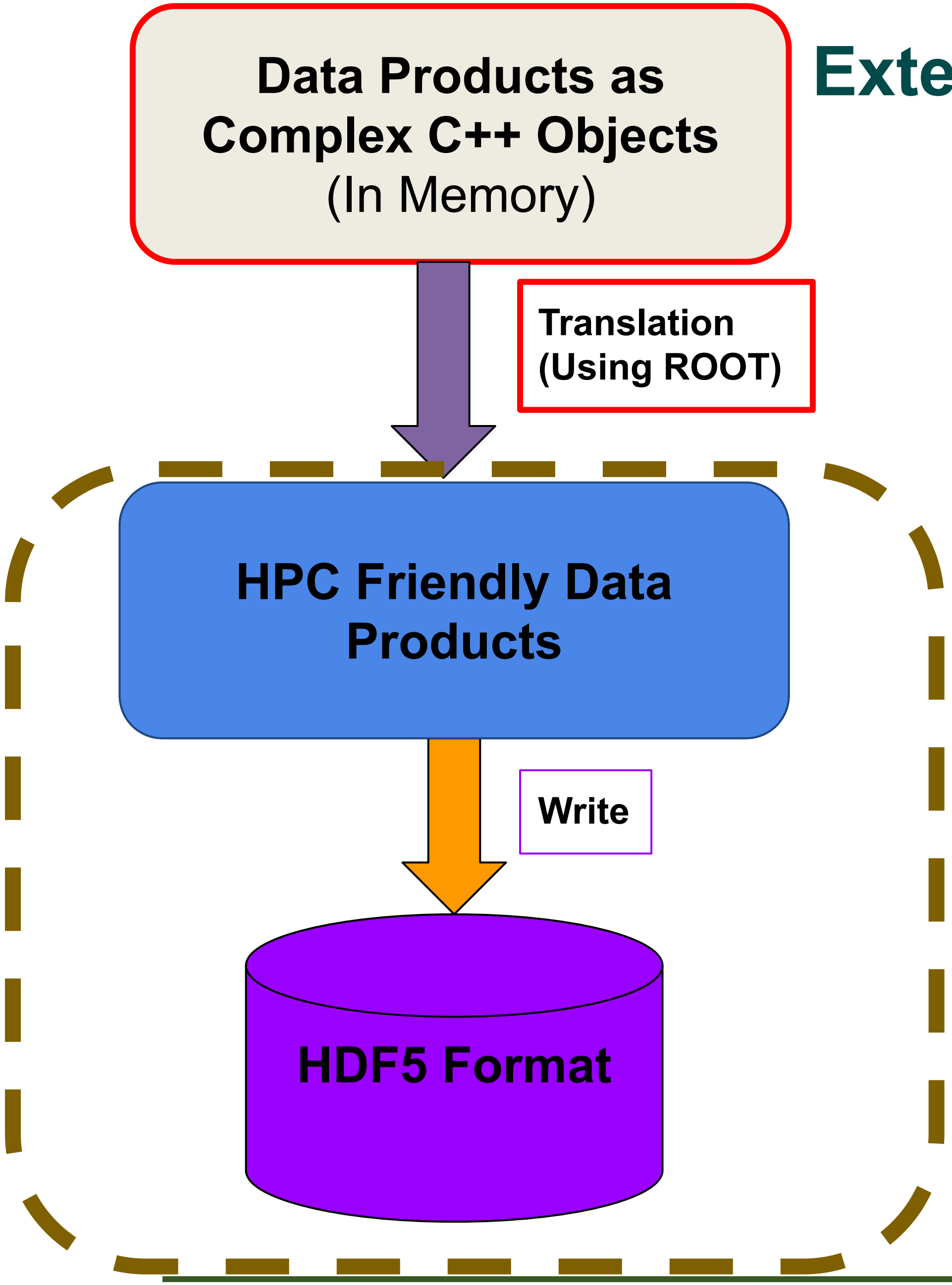
# CCE2: Extending the Test Framework




- Use ROOT to serialize HEP data products to make it HPC friendly.
- Collective writing of data into HDF5 file

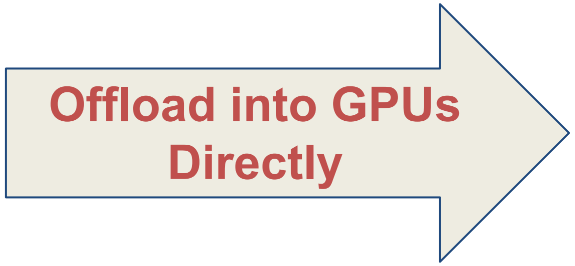
- HPCs **rely on both CPU and GPUs** to achieve high computational capability.
- Fully utilizing HPC resources requires to use GPU resources as well.
- Serialized data cannot be offloaded into the GPUs directly.
- **Using GPUs might need different data organization**.

# Extension to Direct GPU Offloading

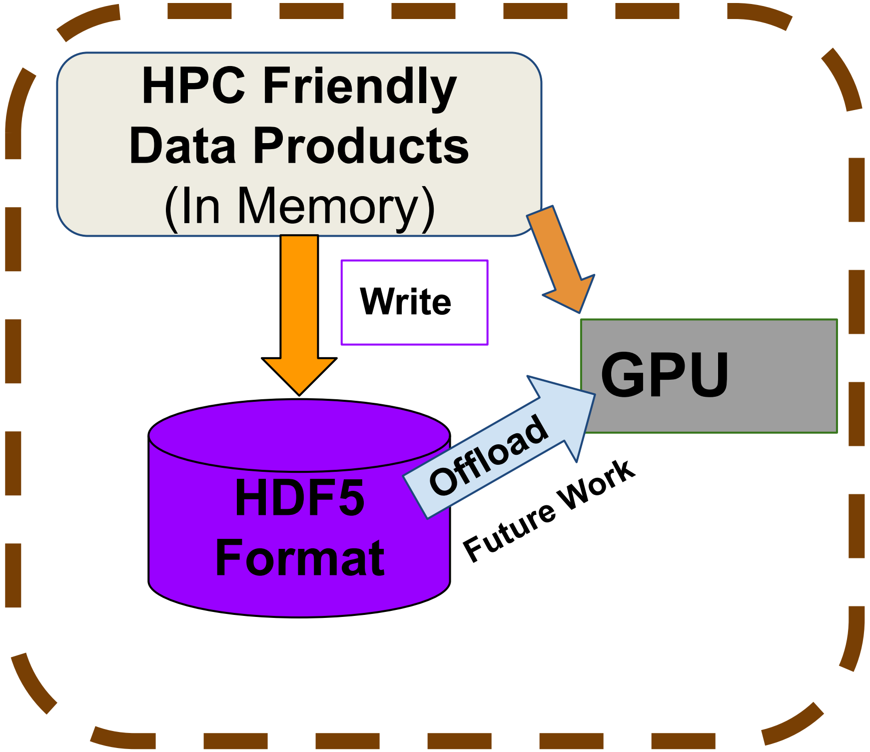



HEP data needs to serialize/deserialize using ROOT.

Complex objects cannot be offloaded directly into the GPUs.




Design Data Model that is HPC (GPU) friendly (next slide)



# HPC Friendly Data Model

- Initiation of investigation of HPC Friendly data models for the HEP experiments
  - Modern HPCs rely on heterogeneous resources (often CPU+GPU) for compute acceleration
  - HEP data models: Heavily Object oriented → Not GPU (and thus HPC) friendly
  - Development of toy framework to offload simple data structures onto GPU devices
  - Future expansion of this effort based on a separate survey carried out by HEP-CCE on experiments' efforts to make their data model HPC friendly
    - Speakers from ATLAS, CMS, DUNE, NOvA and EDM4HEP developers invited to share their experience on developing HPC friendly data models
    - Survey results recorded [here]

# HPC Friendly Data model Survey Results

| Experiment | ATLAS | CMS | DUNE | EDM4hep | NOvA* |
|---|---|---|---|---|---|
| **General Overview** | | | | | |
| **Speaker** | Scott Snyder | Matti Kortelainen | Mike Kirby | Benedikt Hegner | Marc Paterno |
| **Talk Link (Indico)** | https://indico.fnal.gov/event/57595/contributions/256583/attachments/162731/215145/2023-01-10-edm.pdf | https://indico.fnal.gov/event/55536/ | https://indico.fnal.gov/event/58260/ | https://indico.fnal.gov/e/55542 | https://indico.fnal.gov/event/58962/contributions/262454/attachments/165673/220182/DataOrganizationForParallelProcessing.pdf |
| **Github Link:** | https://gitlab.cern.ch/akraszna/asyncgaudi.git | | – | https://github.com/key4hep/EDM4hep | https://github.com/art-framework-suite/hep-hpc |
| **Languages** | C++/CUDA | C++/CUDA/ALPAKA | C++/CUDA | C++/python | python (PandAna*) |

Screenshot of the part of the table of summary from the talks given by speaker. Complete summary is here.

## Findings

- Experiments' directions of effort are based on their current data model
- Experiments want to retain as much of current data model as possible
- Acknowledge some of the transformations (AoS →SoA) are necessary.

## CCE2: Plans based on Survey

- Implement the generalized approach by experiments in the test framework
- Implement tasks (simplified and generalized) that can be offloaded into GPUs
- Design existing framework that can also be used as a training tool

# Next steps for Darshan

**Instrumentation of Intel DAOS I/O libraries**

- Upcoming exascale system at Argonne, Aurora, will feature a new-to-HPC object-based storage system
- Appealing performance characteristics for I/O middleware (e.g., HDF5 and ROOT) that can effectively leverage storage model
- File-based module complete, native object-based module underway

**Darshan analysis tools for workflows**

- Refactor PyDarshan code to more easily allow aggregation and visualization of Darshan data across multiple logs
  - Multiple logs generated by the steps of an HEP workflow
  - Students from various programs during the summer

# Plan – Darshan for HEP

### Workflow I/O characterization

- Capture MPI and HDF I/O
- GPU workflows Benchmarking
- Darshan with container (SULI project)
- Monitor analysis workflows to better understand optimal storage parameter for data products

### Workflow I/O monitoring

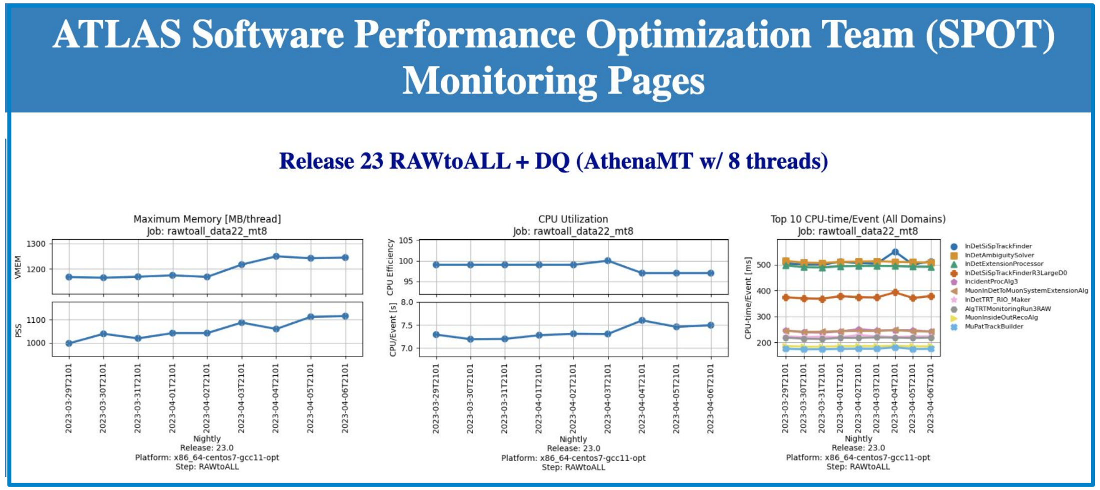
- Software performance monitoring between releases
  - Monitoring the performance of the software, including the transient and persistent event data models
    - Insight on forked processes in time & detailed data access of specific file(s)
  - Guiding the evolution of the software and EDM in order to optimize performance in its multiple aspects: technical performance, resource usage needs and usability for analysis
- Workflow monitoring
  - Integrated task monitoring
  - Input, output & condition data


**Adhithya Vijayakumar**

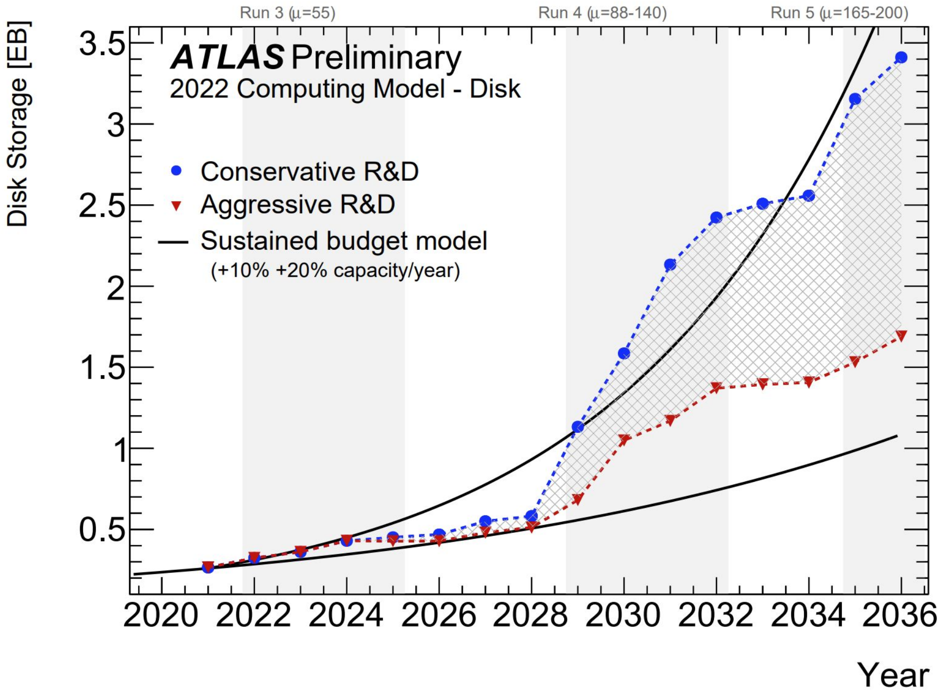
Texas A&M University

Physics


https://atlaspmb.web.cern.ch/atlaspmb/



# Plans for CCE-2: Storage Reduction



# HEP-CCE/IOS: STORAGE AND COMPRESSION

## Future Priorities

- The current cycle of HEP-CCE has been mainly focused on making HEP applications make [efficient] use of High Performance Computer
- This addresses the crucial need for **CPU cycles** expected for HEP experiment at the HL-LHC, DUNE and beyond.
- HEP, however, faces similar challenges for **disk and tape storage**, which also need to be addressed
  - Additional compute cycles may help, but won't solve this issue

# HEP-CCE/IOS: STORAGE AND COMPRESSION

## RNTuple, very brief, relevant experts are in the room.

- **RNTuple** part of ROOT7 has been implemented in ATLAS/CMS for most derived production analysis products
  - Promises significant storage savings and I/O speed up
  - Limited Data Model support vs. TTree
    - Streamlined design of RNTuple will require leaner approach than TTree
- HEP-CCE Role:
- Adjustments to complex Simulation/Reconstruction Data Models
  - Development of techniques to hide complexity from persistence
    - Synergetic to HPC friendly data model work
  - Performance Testing and Optimization, e.g. using Darshan monitoring and I/O mimicking
- Consolidate requests for additional functionality to ROOT

# INTELLIGENT LOSSY COMPRESSION

## Computing Science

- Most experiment HEP data is stored in a compressed format using **standard loss-less compression** algorithms
- More advanced/intelligent, but often lossy compression algorithms are less common
  - Exception: CMS Nano-AOD, soon? ATLAS PHYSLITE
- Overview of "intelligent" data compression:
  - Oct 18, 2022: **Speakers**: Prof. Anand Rangarajan , Prof. Sanjay Ranka: Hybrid Learning Techniques for Scientific Data Reduction with Performance Guarantees
  - Nov 29, 2022: **Speakers**: Dr Franck Cappello (ANL), Dr Sheng Di (ANL): Compression of Scientific Data with SZ
  - Mar 21, 2023: **Speaker**: John Wu (LBNL): Statistical Similarity for Data Compression

# Thank you

This work was supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics, High Energy Physics Center for Computational Excellence (HEP-CCE). This research used resources at Argonne Leadership Computing Facility, NERSC and BNL Scientific Data and Computing Center.

# Backup

# CCE2: Include Storage Challenge Strategies



- The current HEP-CCE cycle focuses on solving the HEP processing challenge by moving HEP workflows to HPC
  - Addresses different Storage architectures by enabling workflows to store intermediate data via HPC friendly backend, such as HDF5.
    - Including collective I/O
  - Provides and enhances tools, such as Darshan, that can monitor HEP workflow I/O on HPC, helping to identify and mitigate bottlenecks
  - Developed experiment agnostic I/O mimicking framework, allowing I/O scaling tests beyond the reach of current HEP workflows
- For the next cycle we propose to include the upcoming storage challenge for HEP
  - ROOT RNTuple is scheduled to replace TTree (used by many HEP experiments) and promises significant storage savings.
    - Experiments will face (common) challenges adapting their data models (especially for Simulation/Reconstruction)
    - RNTuple uses more modern architecture
  - Intelligent, but lossy compression techniques promise large storage reductions, but…

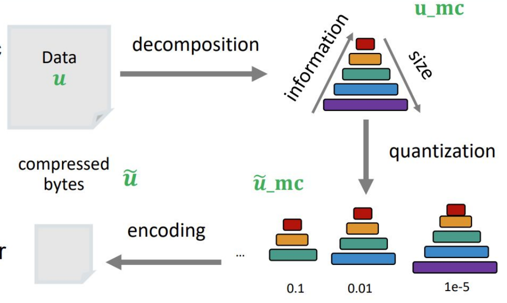
# HYBRID LEARNING TECHNIQUES FOR SCIENTIFIC DATA REDUCTION

Prof. Anand Rangarajan , Prof. Sanjay Ranka: **GitHub - CODARcode/MGARD: MGARD: MultiGrid Adaptive Reduction of Data**

- Compression of scientific applications differ from video and image compression
  - Guarantees on Quantities of Interest (QoI): Scientists are principally interested in QoI that are derived from raw data. The ability to quantify these with realistic bounds is essential.
- Compression Ratio of ~30-40 for fusion code data

MGARD – Multigrid Adaptive Recursive Decomposition

- MGARD is a transform-based compressor
  - Decompose the original data u into u_mc by recursively performing $L^2$ projection and multilinear interpolation on the multilevel grids.
  - Quantize u_mc to ũ_mc keeping error tolerance
  - Encode ũ_mc to ũ to reduce the number of bits



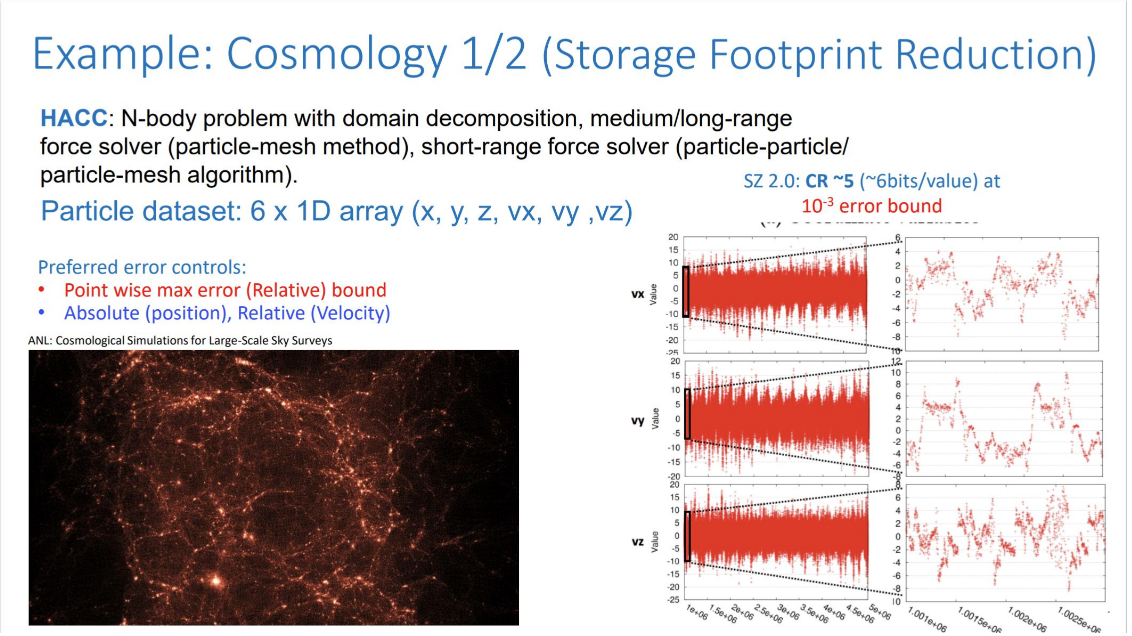

- PD Guarantees: Method controls the compression error for a variety of s-norms. The relation between $||u\_mc - \tilde{u}\_mc||_s$ and $||u - \tilde{u}||_s$ is mathematically preserved
- QoI Guarantees: Provide error management on linearly derived QoI.
- **Compression ratios at PD error and QoI error within NRMSE $10^{-3}$ for XGC: 30-40. Compression is smaller for lower QoI error bounds.**

- Used for XGC Fusion Code that can produce 4.3 PB/day.

# COMPRESSION OF SCIENTIFIC DATA WITH SZ

Dr Franck Cappello (ANL), Dr Sheng Di (ANL): **SZ Lossy Compression | SZ Lossy Compressor (szcompressor.org)**

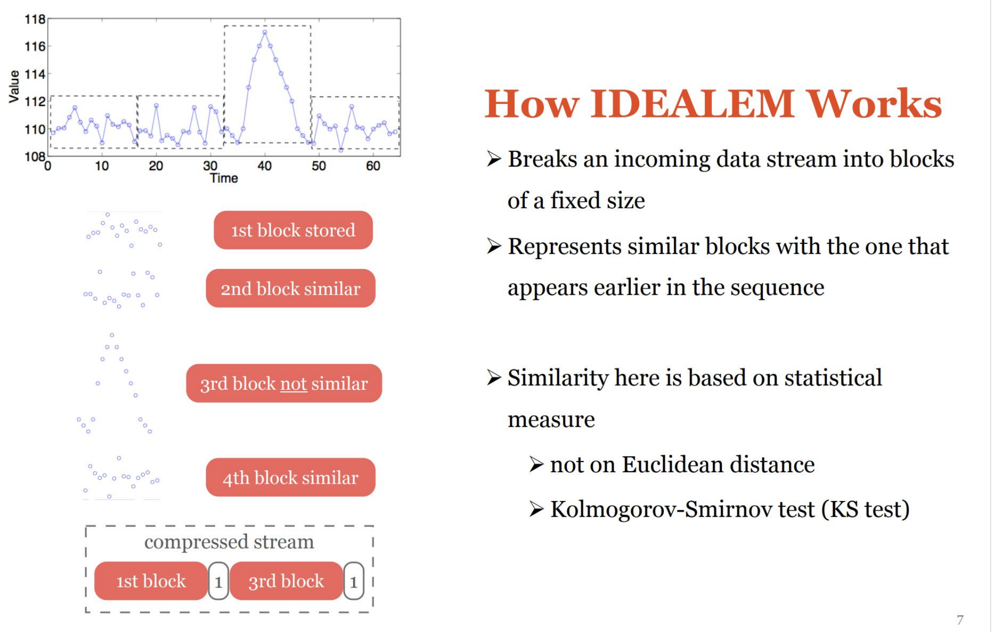
- Consist in reducing scientific data volume by leveraging correlations and reducing precision
- Goal: keep the same science
  - Requires error bounds on observables
- Compression Ratio of ~5-100 for scientific data



# STATISTICAL SIMILARITY FOR DATA COMPRESSION

John Wu (LBNL): **IDEALEM at LBNL (lbl.gov)**

- Motivated by reading out many (1000s) of micro-Phaser Measurement Unit over time
  - Monitoring device is capable of sample dozens of measures many thousands of times a second
- That's for the power grid, don't ask me how, but does not sound so unsimilar to some detectors.
- Compression Ratio of ~100-200 in PMU example!




- One drawback/challenge: KS test is computational expensive.