High Energy Physics Center for Computational Excellence

Optimizing Data Storage for Next-Generation HEP Experiments

IOS Project · INDICO-FNAL (Indico)

Amit Bashyal, Rob Ross, Peter van Gemmeren
for HEP-CCE

DOE HEP CCE Review
December 19 2023

Optimizing Data Storage for next-generation HEP experiments: Summary

Building on the work done by IOS:

- IOS Project · INDICOx-FNAL (Indico)
- Meetings on Wednesdays 11AM-noon, everyone is welcome
- Sign-up: https://docs.google.com/spreadsheets/d/1PCgm2KpII8rS5yprlzc9V8muoFxgCHULQ4vnyX2QgbU/edit?usp=sharing
- Considered expression of interest, not funding.
- Will continue work on Darshan, HDF5 and Data Model

Tracking and aiding the evolution of ROOT I/O, in particular RNTuple

ROOT: TTree to RNTuple migration

- After successfully having served HEP for decades, ROOT's **TTree** container is being replaced by RNTuple:
  - HEP has collected well over **1 ExaByte** of data stored in TTree
  - ROOT will maintain TTree readability 'forever'
  - However, there will be no performance updates to TTree
- **RNTuple** is one of the upcoming **ROOT 7** features, targeting **HL-HLC production deployment**.
  - Comes with modern C++ interfaces
    - unique_ptr, templates, better error handling, std::string
  - Promises better performance than TTree I/O
    - including **significant storage reduction**
  - Already available in ROOT 6.24
    - in 'Experimental' namespace

CAF, RNTuple and GPU Friendly Data Model

Amit Bashyal
Argonne National Laboratory

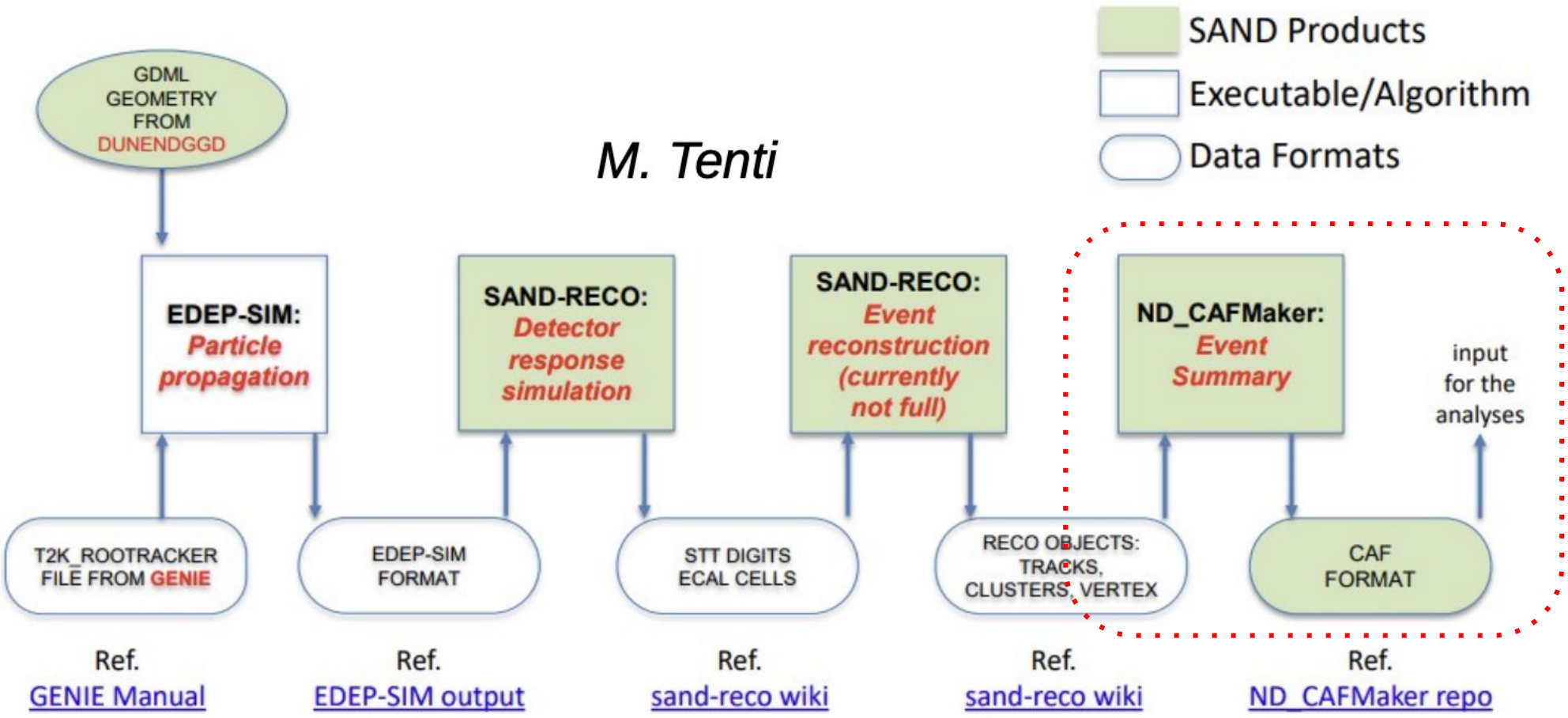
CAF and CAF Ana

- Designed by NOvA to use simpler than art object for the oscillation analysis
  - Original Paper (Link)
  - Later optimized for HDF5 by Saba Sehrish, Jim Kowalkoski and Marc Paterno
  - Talk Given by Marc in CCE Survey: Link
- Challenge:
  - Fine Granularity of the NOvA detectors → Too large *art* Files with intricate structure
  - Complications to analyse data with ease and speed
- CAF
  - Object oriented with multiple level of hierarchies and segmentation
  - Data organized in columnar table format
  - Discard Hit by Hit Information
    - Reconstructed variables from hit by hit information
  - Stored as flat ROOT TTree originally
    - True for NOvA, DUNE
  - Overall Data Organization is similar for all Oscillation Experiments
    - DUNE's CAF Format is incomplete stored as flat TTree

Why CAF Data Format

- Analysis level data model that is used by multiple oscillation experiments
- Relatively simpler
  - Serves as a good initiation with the deliverable to multiple future looking experiments including DUNE
- CAF and CAF analysis framework is stand alone
  - Makes it easier to port the CAF features into testing framework
- CAF Framework is simpler
  - While doing the RNTuple persisting studies, allows us to find out relevant aspects of the framework that needs to be adjusted to use and utilize the RNTuple I/O capabilities.

DUNE Near Detector Event Reconstruction Chain



Screenshot taken from here.

CAF Data Format

- Each *StandardRecord* is a collection of objects that reconstructs an “event record” of a neutrino interaction candidate
  - An event is “essentially” collection of “slices” (correlated “activities” within an event record).
  - Neutrino Event candidate will contain
    - Vertex, tracks, cluster activities
    - Different reconstruction algorithm reconstructing these
      - Different neutrino hypothesis

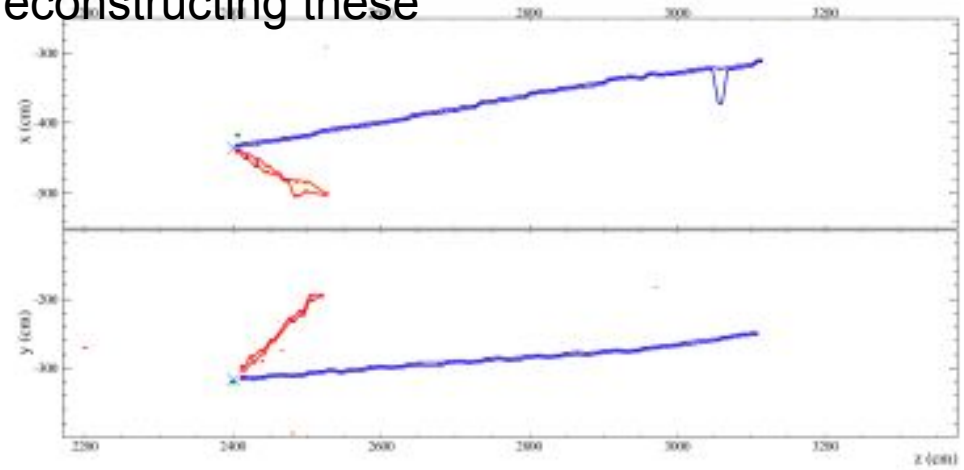


Figure 2. Reconstructed “slice” in NOνA. The x-z and y-z views are displayed separately. Two tracks are shown, one red and one blue.

Standard Record and SR Objects

- Each Standard Object contains a “SRHeader” object which contains metadata information:
  - Run
  - Subrun
  - Event
  - Sub-Event
- CAF data in TTree is flattened.
- CAF Analysis Framework does the structured I/O*
  - Not just the persistency but something that mimics some CAF framework functionalities
    - Provide as example
    - Use for performance tests
- CAF data format in HDF5
  - Reorganized in HDF5 to utilize parallel I/O
  - Reorganized into HDF::Groups, HDF::Subgroups, HDF::Datasets
  - Metadata information in each group.

CAFAna

- Analysis framework to analyze CAF Fromat data
  - TTree like **eventloop is hidden** from the users.
  - Analysis tools like **reconstructing** higher level variables and selection requirements (**cuts**) are handled by user defined lambda functions which are also CAFAna type objects (overloaded boolean/arithmetic objects)
  - Loop is done over StandarRecord objects (**SpectrumLoader**)
    - User can select single or multiple files

```
const Cut kCrudeMuonSel({"trk.nkalman", "trk.kalman.len"},
                        [](const caf::StandardRecord* sr)
                        {
                          if(sr->trk.nkalman == 0) return false;
                          return sr->trk.kalman[0].len > 200;
                        });
```

```
const Var kTrackLen({"trk.kalman.len", "trk.nkalman"},
                        [](const caf::StandardRecord* sr)
                        {
                          if(sr->trk.nkalman == 0) return 0.0;
                          return sr->trk.kalman[0].len;
                        });
```

```
SpectrumLoader loader("reconstructed_events.root");
```

Spectrum builds a histogram out of Binning, SpectrumLoader, Var and Cut.

```
Spectrum len("Track length (cm)", bins, loader,
                                        kTrackLen, kCrudeMuonSel);
```

Visualizing CAF as data table (NOvA)

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	

Table 2
NOvA data table organization with one entry per vertex.

run	subrun	event	sub-event	vtxid	npng3d	... 6 more ...
433	61	6124	35	0	0	
433	61	6124	36	0	1	
433	61	6124	36	1	1	
433	61	6124	36	2	5	
433	61	6125	1	0	1	
433	61	6125	3	0	0	

Figure: Tabular representation of the data in NOvA. Each complex variable (SRvertex object in this example (Table 2)) is written in separate tables with the relevant attributes and metadata. Since the indexing or the metadata (the first 4 columns in the tables above)are part of each separate table, it is stored redundantly in the memory to allow faster I/O and is highly compressed in the disk.

Each Table corresponds to a SR type object.

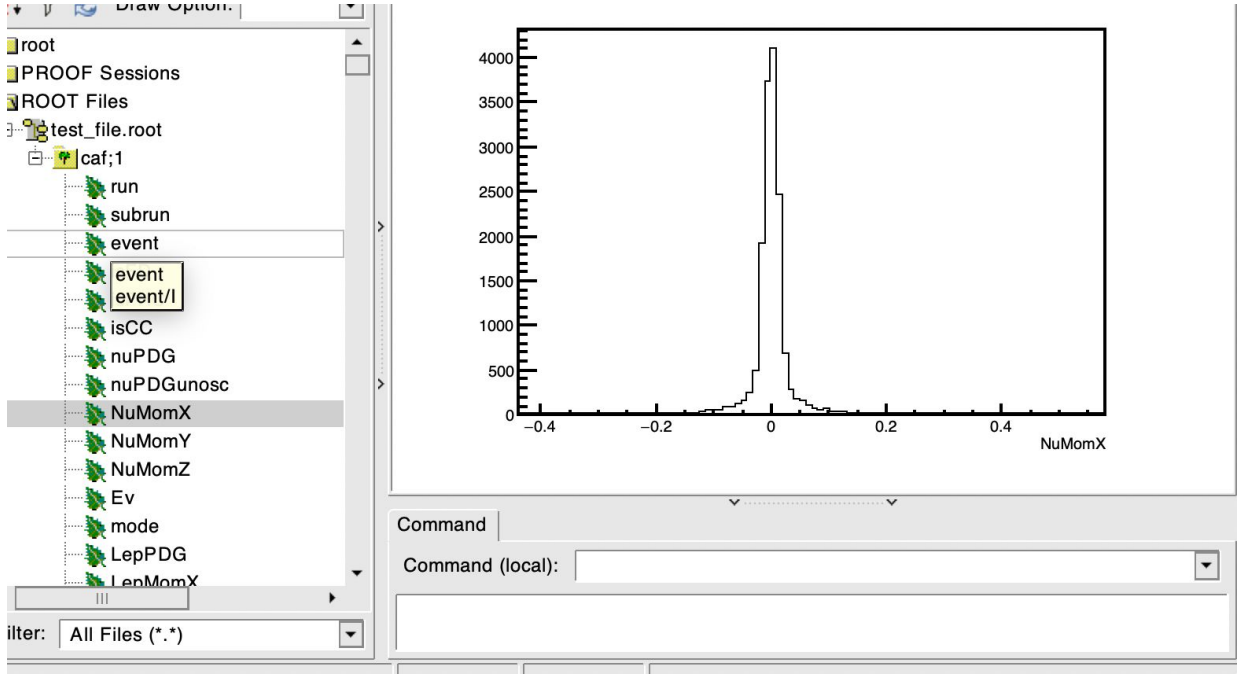
- Each table contain information (first 4 columns) to connect them to other members of Standard Record object.

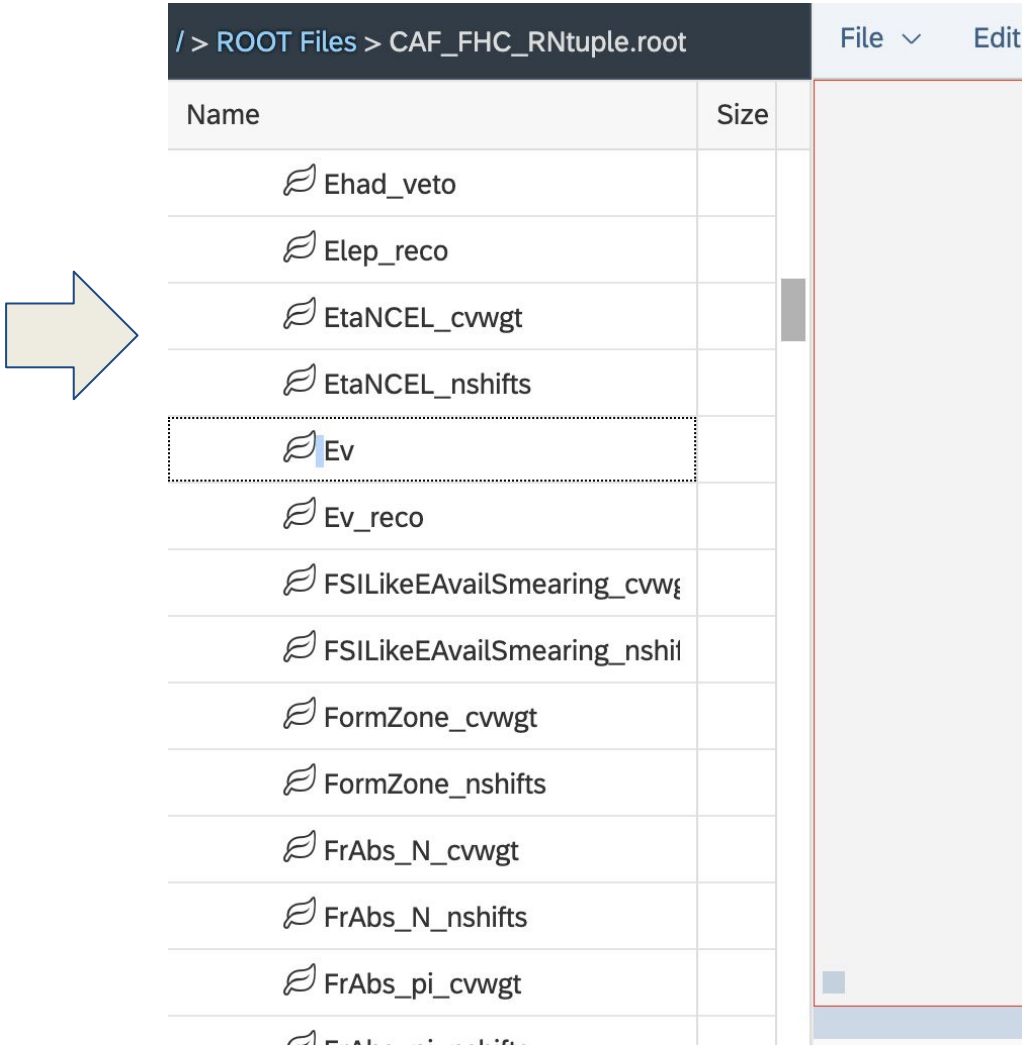
Persistency of CAF Data Format in RNTuple

- (1) Store as AoSoA (and Initial Thoughts for future)
  - Design with GPU friendly data model in mind
  - Each Complex (sub)SRObjects will be stored as a separate SoA within the parent SoA
  - *StandardRecord* class →RNTuple::Model
  - Members of *StandardRecord* class →Attributes of *StandardRecord Model*
  - Users can access needed members of *StandardRecord* Model only (?)
  - Is it better than flat CAF in TTree ?
    - Memory to load members/submembers of *StandardRecord* model
    - How is indexing happening in second/third level of hierarchy?
    - I/O performance
  - Persistence of memory-aligned data (?)

- (2) Store as flat TTree in RNTuple
  - We (Meghna) is writing a python API for (flat) TTree to RNTuple converter.
  - Duplicate current TTree persisted CAF model
  - Compare performance and storage size with (1)

Persistency of CAF Flat TTree into RNTuple





- Persistency of Flat Tree into RNTuple is straight forward.
  - ROOT has infrastructure
- However at least some CAF like tools to show example support needed.
- Further optimizations?

Persistency in RNTuple (Example)

- Persistency of Simple AoSoA in RNTuple is straightforward
  - *Links point to work in progress codes (will evolve over time)
  - Use Macros to Generate User Defined AoSoA. (Link)
  - Generate Dictionary. (Link)
    - Does not require to define AoSoA attributes in the dictionary.
  - Read and Write in RNTuple. (Link)

```
Generate_Arrays(msoa,
                Addvector(uint32_t, a);
                Addvector(uint32_t, b);

                );

Generate_Arrays(asoa,
                Addvector(uint32_t, aa);
                AddSoA(msoa, subsoa);

            );
```

User creates an *asoa* with a pre-compiler macro

```
*********************************** NTUPLE ***********************************
* N-Tuple : NTuple                                                           *
* Entries : 4                                                                *
******************************************************************************
* Field 1               : MyASoA (soa::ASoA::asoa)                           *
*   Field 1.1           : aa (std::vector<std::uint32_t>)                    *
*     Field 1.1.1       : _0 (std::uint32_t)                                 *
*   Field 1.2           : subsoa (soa::ASoA::msoa)                           *
*     Field 1.2.1       : a (std::vector<std::uint32_t>)                     *
*       Field 1.2.1.1   : _0 (std::uint32_t)                                 *
*     Field 1.2.2       : b (std::vector<std::uint32_t>)                     *
*       Field 1.2.2.1   : _0 (std::uint32_t)                                 *
******************************************************************************
{
  "MyASoA": {
    "aa": [5, 3, 8, 5, 6, 8, 3, 4, 2, 6, 7, 8, 9, 0, 7, 1, 5, 6, 0, 1, 2, 1, 7, 2,
8, 4, 7, 0, 9, 5, 1, 3, 6, 8, 6, 2, 1, 0, 4, 8, 4, 4, 3, 8, 1, 9, 0, 2, 1, 4, 1, 5,
    "subsoa": {
      "a": [7, 8, 8, 6, 2, 3, 7, 0, 5, 5, 9, 4, 5, 8, 9, 2, 4, 1, 1, 6, 4, 3, 5, 7,
 6, 7, 6, 4, 7, 3, 9, 9, 1, 0, 3, 0, 6, 3, 7, 1, 8, 4, 7, 3, 1, 4, 4, 6, 3, 2, 0, 3
      "b": [3, 5, 2, 0, 1, 4, 9, 6, 3, 2, 5, 0, 5, 5, 1, 9, 5, 9, 2, 9, 4, 1, 2, 3,
 4, 2, 1, 6, 7, 8, 0, 0, 7, 6, 4, 7, 2, 0, 9, 3, 7, 9, 7, 0, 9, 0, 4, 4, 6, 3, 0, 1
    }
  }
}
```

Easily Persisted.

Ongoing Work →SoA with data alignment

- Test class here (Incomplete)
- Memory Ownership and Data Layout (CMS like)
  - **Layout class**
    - Generate arrays (columns) using macros
    - Run time define size & alignment has to be compile time.
    - No memory ownership but should inform memory information
  - **View Class** (not done yet but have some idea)
    - Based on Layout (not Derived from)
    - Pointers to SoA columns (arrays)
    - Pointers should be able to copied from host to device
      - Memory alignment in layout is still preserved
  - Both View and Class types should be generated from a single macro

Long Term Prospective

- Part of CCE-2 GPU (HPC) Friendly data model and RNTuple effort
- Based on suggestions from CCE2 Survey
  - Transformation of AoS →SoA type data model
- Aim of This work:
  - Take CMS like approach to transform Data models to GPU Friendly (Ao)SoA
  - Use simple data models like CAF (and DUNETriggerData) to test implementation
  - Document progress/challenges as work evolves
  - Use Darshan to profile I/O during the tests
- Final Product
  - Experiment Agnostic Framework to generate GPU Friendly SoA objects
  - Persist in HDF5 and RNTuple
  - Example codes to read/write sample data models and do simple performance tests
  - Collaborate with Experiments for outreach
- Metric of Success
  - Aims and Final Products are delivered.
  - Experiments adopt/recognize the findings of this work according to their requirements
  - Final Framework becomes a deliverable of the CCE-2

Scope Within CCE-2 (Based on CCE Report)

- Applying Lesson Learned to HEP experiments (Section 3.1)

- Continue work on HDF5 and other storage technologies for HEP data without ROOT serialization; while ProtoDUNE and DUNE are the primary targets, some data products of the collider experiments do not rely on very complex data models. In particular, the structure of analysis products is much simpler and could potentially be stored in HDF5 with direct mapping. The lessons learned and tools developed for storing HEP data in HDF5 will also make testing additional alternative backends, such as Parquet, for storing HEP data easier.

  The DUNE example is a specific instance of how HEP-CCE can aid experiments in directly offloading data processing to accelerators found on HPC systems. The work on HPC-friendly data models has the following goals:

  - Current experiment approaches will be generalized, simplified, and implemented in an experiment-agnostic way. The experiment examples and implementations will allow users to get hands-on experience in transforming different HEP experiment data attributes into GPU-friendly ones.
  - Identify algorithms that can benefit from the use of compute accelerators, such as the testbed algorithms used by PPS. Extract their data model of the input and output data to make them usable within the test framework. This will allow users to understand and optimize the tasks without having to learn the inner workings of the experiment frameworks themselves.
  - The test framework should be simple, flexible, and modular enough to use as a training tool. Still it should mimic the essential characteristics of HEP computational workloads.

Scope within CCE-2 (Based on CCE Report)

- Study of RNTuple persistency will aid the evolution of ROOT I/O

3.5.1 Tracking and aiding the evolution of ROOT I/O

With over one ExaByte of data stored, a user base of thousands, and a solid project structure, ROOT is expected to remain the primary I/O technology for the next generation of HEP experiments. To meet the performance and storage requirements of the next decade, the ROOT team plans to offer, besides the traditional `ROOT::TTree` I/O subsystem, a new, simpler, and more efficient I/O subsystem called `ROOT::RNTuple`. While not as flexible as `TTree`, early studies showed that `RNTuple` can provide significant overall resource savings. `RNTuple`'s limited support for object-oriented data models is synergistic to CCE work on HPC-friendly data models. Simple data models will ensure that HEP data can 1) be stored in either HPC native storage formats such as HDF5 or as a `RNTuple`, allowing sizable storage savings, and 2) efficiently offloaded to compute accelerators like GPUs. CCE is in a unique position to leverage knowledge from the

24

Plans and Milestones

- Assuming 0.5 - 1.0 FTE (to be discussed at the AHM)
- 1 year
  - Finish simple Layout and View type class with memory alignment that can be persisted in RNTuple
  - Development of (RNTuple) persistable AoSoA with simple data alignments
  - Common data model class that can be derived to persist in HDF5 or RNTuple format
  - Integrate with the GPU testing/offloading framework
- ~ 2 years
  - Layout and View type with Persistable AoSoA in RNTuple
  - Integrate with GPU testing/offloading framework
  - Add example codes similar to CAFAna type, DUNETriggerRecord functionalities type
- ~ 5 years
  - Investigate other possible data model candidates
  - Final Framework to generate data models and test in HPCs (CPU+GPU)
  - ….
- Documentation of the work and outreach to experiments to refine/optimize strategies as they go.

Reduced Precision and Intelligent Domain-specific Compression Algorithms

Reduced Precision Storage

- Most experiment HEP data is stored in a compressed format using standard lossless compression algorithms
    - Lossy compression algorithms are less common
- To reduce storage requirements further, experiments and ROOT are investigating means of reduced-precision storage as much of the data is derived from measurements with inherent uncertainties
    - For derived data, not RAW
    - Currently used in CMS nano-AOD
    - Under study for ATLAS PHYSLITE data
- Potential storage savings ~20-30%

Intelligent Domain-specific Compression

- IOS team has surveyed different tools developed by computer scientists and used elsewhere in science that could find application for some HEP domains:
    - Hybrid Learning Techniques for Scientific Data Reduction with MGARD
    - Compression of Scientific Data with SZ
    - Statistical Similarity for Data Compression with IDEALEM

Reduced Precision and Intelligent Domain-specific Compression Algorithms

Challenges

- As data is one of the most important assets of HEP experiments, any reduction of stored content is rightfully scrutinized, but at the same point storing values with precision far greater than their measurement wastes storage and compute resources that could better be used elsewhere

HEP-CCE Role

- HEP-CCE will carry out research and testing of experiment use-cases in a coordinated fashion
- Intelligent/lossy compression with ROOT resident data may require infrastructure enhancements that would benefit from common developments
- Tools, metrics, and methods for validation and safeguarding of lossy compressed data to not reduce its physics potential can be developed within HEP-CCE

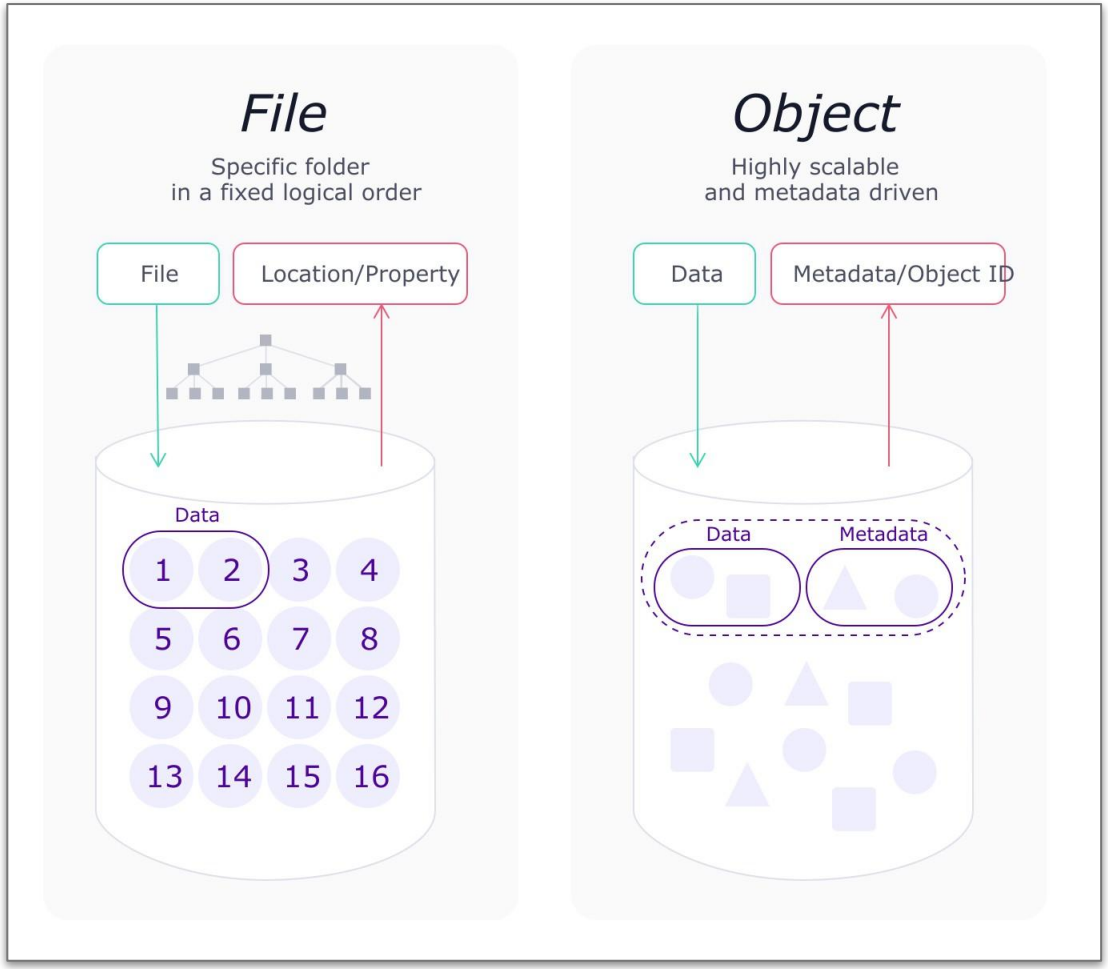
RNTUPLE, OBJECTS, AND DAOS

ROB ROSS
MATHEMATICS AND COMPUTER SCIENCE DIVISION
ARGONNE NATIONAL LABORATORY
RROSS@MCS.ANL.GOV

Argonne NATIONAL LABORATORY

OBJECT STORAGE: WHAT IS IT?

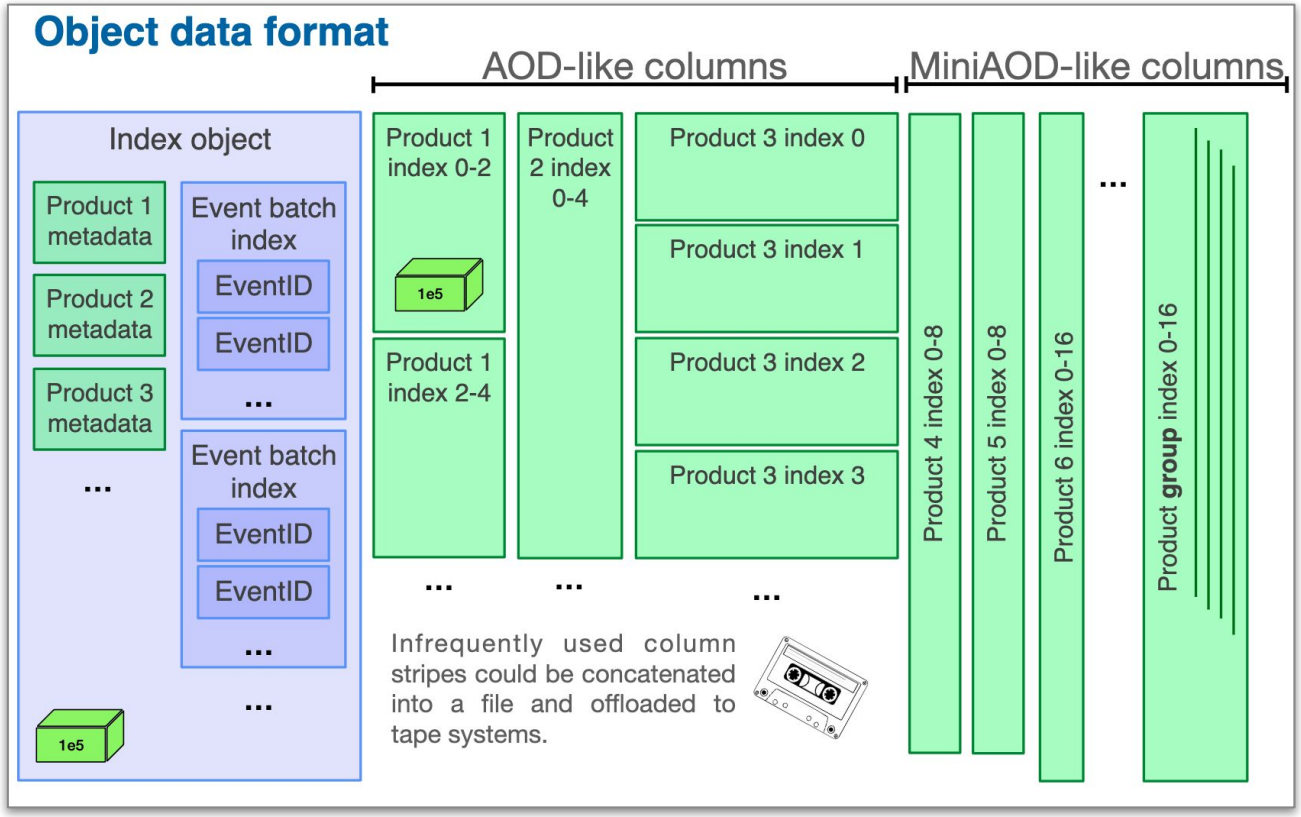
- "Object storage" is a term used to describe a collection of storage technologies that focus on the ability to store, reference, and access large collections of unstructured data
- Unlike a file system, how you find things is generally handled separately (e.g., via DB)
- There are lots of flavors of object storage being used today in different contexts:
  - Cloud storage (e.g., S3): large, immutable objects, HTTP access
  - Distributed filesystems (e.g., RADOS): large, mutable and byte-addressable objects with file system access (i.e., Ceph)
  - HPC storage (e.g., DAOS): semi-structured, mutable, byte-addressable objects with key/value access



https://www.scaleway.com/en/blog/understanding-the-different-types-of-storage/

STORING ROOT DATA IN OBJECTS

- Numerous potential advantages for using in HEP:
  - Allow fine-grained versioning, avoiding replication of unchanged objects
  - Facilitate user-driven data augmentation, to subset of events
  - These methods of referencing save storage space
- FNAL is looking at Ceph for CMS datasets in particular (right)
- BNL also investigating Ceph



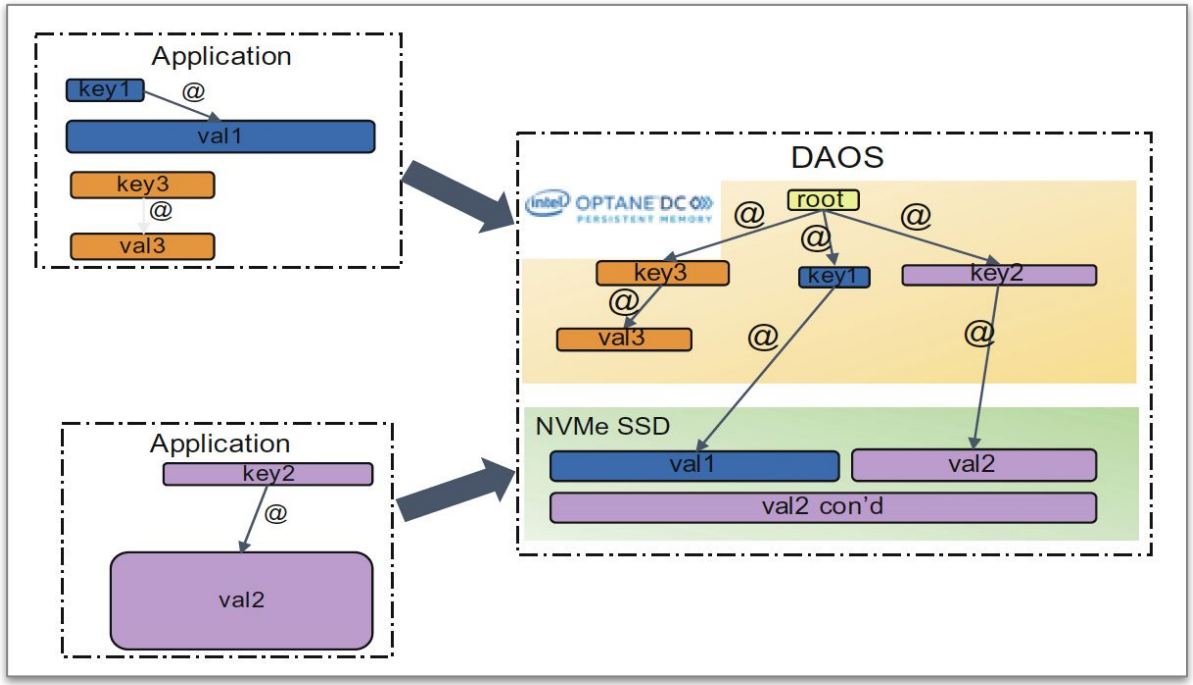
The FNAL team is investigating mapping CMS datasets into Ceph objects. The approach is not specific to Ceph, although different mappings might be more advantageous on specific underlying technologies.

Bo Jayatilaka, Christopher Jones, Nicholas Smith, “Using CEPH Object store with ROOT serialization in CMS”, December 2022.
https://indico.fnal.gov/event/57189/contributions/254706/attachments/162368/214598/ncsmith-uscms-objectstoresQ4.pdf

DAOS AND RNTUPLE

DISTRIBUTED ASYNCHRONOUS OBJECT STORAGE (DAOS)

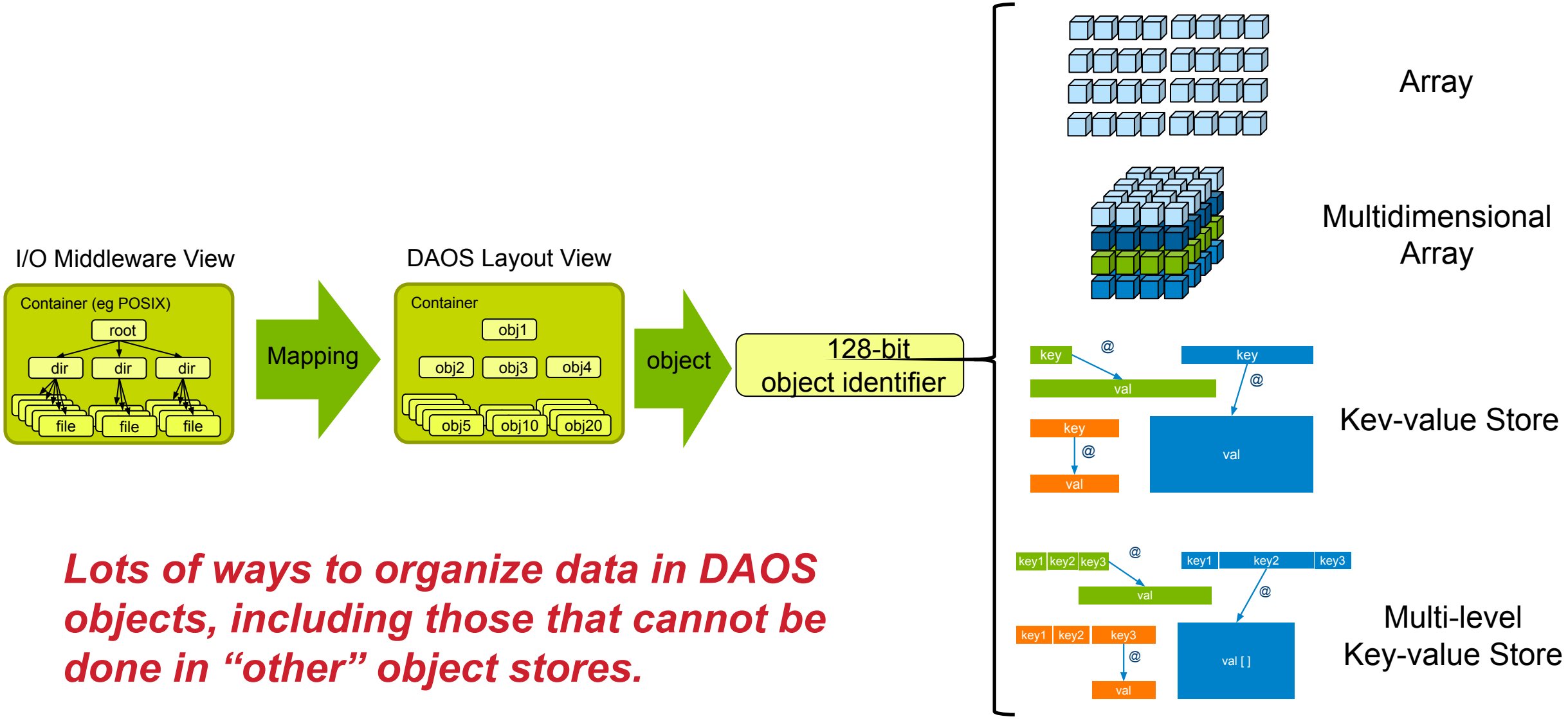
- DAOS is an object storage service developed for use on persistent memory technologies as a very high performance online storage layer
- Data model includes both key:value objects and array objects
- Array objects can be used to streamline storage of large multidimensional arrays with record addressability
- Access can be via POSIX or directly via custom API
  - Custom API, array objects, striping all provide opportunities for optimization beyond a “standard” object store



Example of keys and references employed in a DAOS volume. Array objects preserve record addressability that is incredibly valuable in many HPC contexts (e.g., HDF5 arrays).

Zhen Liang, Johann Lombardi, Mohamad Chaarawi, Michael Hennecke,”DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory,” June 2020.
https://doi.org/10.1007/978-3-030-48842-0_3

DAOS DATA MODEL: OBJECTS

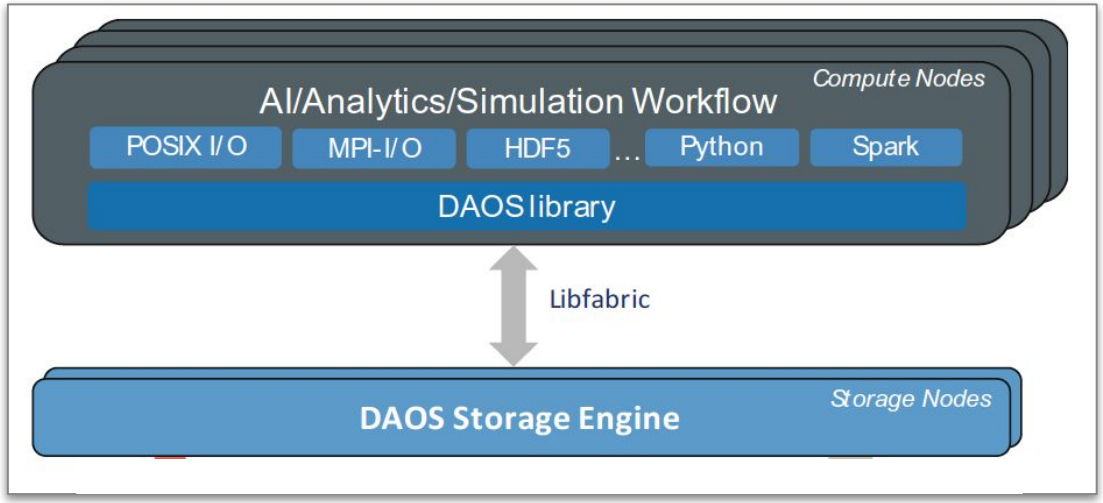


Lots of ways to organize data in DAOS objects, including those that cannot be done in "other" object stores.

Slide from Intel DAOS

OBJECT STORES, DAOS, AND RNTUPLE

- HEP-CCE will study RNTuple DAOS implementation using Darshan
- Darshan already provides initial support for characterizing DAOS storage access
- IOS has successfully used Darshan for current HEP workflows using ROOT
- Aligns with, and will benefit from, other activities to understand and tune DAOS use by team members
- We will also look at how to incorporate TTPerfStats and RNTuple performance data into analysis (generally beneficial)



A variety of user APIs have already been developed for using DAOS from applications, including POSIX, HDF5, and ROOT.

Zhen Liang, Johann Lombardi, Mohamad Chaarawi, Michael Hennecke,"DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory," June 2020. https://doi.org/10.1007/978-3-030-48842-0_3

SUMMARIZING OBJECT WORK

OPPORTUNITIES AND PLANS WITH OBJECTS

- Benchmarking on DOE platforms
  - Talk with Florine de Geus (CERN) on where to get these?
  - Also Analysis Description Language (ADL) benchmarks?
  - Coordinate on scaling, test file and DAOS (DFS and object) configurations
- Darshan and ROOT (and RNTuple) (and DAOS)
  - Capture, persist, analyze TTPerfStats and RNTuple performance data
  - Will need io_uring support for RNTuple file
  - May lead to performance optimization suggestions…
- Other object storage technologies
  - FNAL, BNL continue looking at Ceph
  - Coordinate, share experiences, build shared view of technologies and capabilities, socialize methods to best exploit objects

Optimized Data Delivery to HPC systems

Data Delivery and Xrootd

- HEP grid computing relies on availability of distributed data access solutions
    - Deliver data from distributed data stores to computing elements
- Xrootd is a high-performance, scalable, fault-tolerant distributed data access solution deeply integrated into the storage solutions of many HEP experiments
    - Major component of storage systems at CERN, Fermilab, and other labs

- In collaboration with the **SLAC team** responsible for Xrootd development
- CCE Phase 2 will evaluate opportunities to optimize the (streaming) delivery of data to object stores and parallel file systems running on HPC systems

Xrootd/streaming for HDF5

- Using our findings on HDF5 data formats and organization, the research proposed here will focus on the development and scheduling required for multi-threaded and serial writes to HDF5 from within the DUNE analysis framework I/O layer
- Additional I/O R&D will focus on optimized streaming delivery of HDF5 data via the xrootd or alternative protocols and tuning the DUNE data model to improve compatibility with HPC data storage systems

Student Engagement

Yanli Lyu (Summer intern from NMSU)

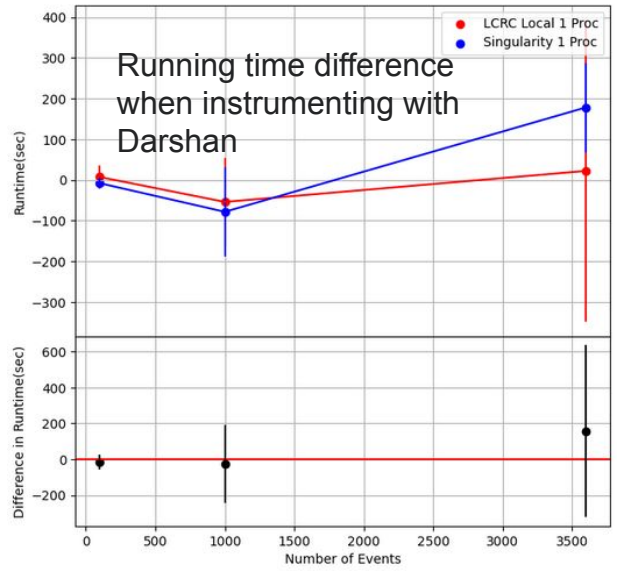
- Refining PyDarshan log analysis framework for workflow analysis

Yanli Lyu
New Mexico State University

Adhithya Vijayakumar
Texas A&M University
Physics

Adhithya Vijayakumar (2023 summer SULI from Texas A&M)

- I/O monitoring for portable HPC applications (report)
  - Darshan with container



Nehemyah Green (Summer student from IIT)

- Working on Ceph, learn Darshan with Ken

Optimizing Data Storage for next-generation HEP experiments: Summary

Building on the work done by IOS:

- IOS Project · INDICO-FNAL (Indico)
- Meetings on Wednesdays 11AM-noon
- Sign-up: https://docs.google.com/spreadsheets/d/1PCgm2KpII8rS5yprlzc9V8muoFxgCHULQ4vnyX2QgbU/edit?usp=sharing
- Will continue work on Darshan, HDF5 and Data Model

Adding new focus on Optimizing Data Storage for next-generation HEP experiments

- This work will leverage our experiences were possible
- Take advantage of all the group's membership and expertise
- Act as forum for HEP experiments

Improvements will have significant impact to resource needs:

- Storage requirements for HL-LHC and DUNE are very sizeable and much larger than previous experiments.
- New approaches can lead to large savings and it may be worthwhile to invest in previously neglected techniques.
- New technologies allow for possibilities that weren't available in LHC Run 1-3