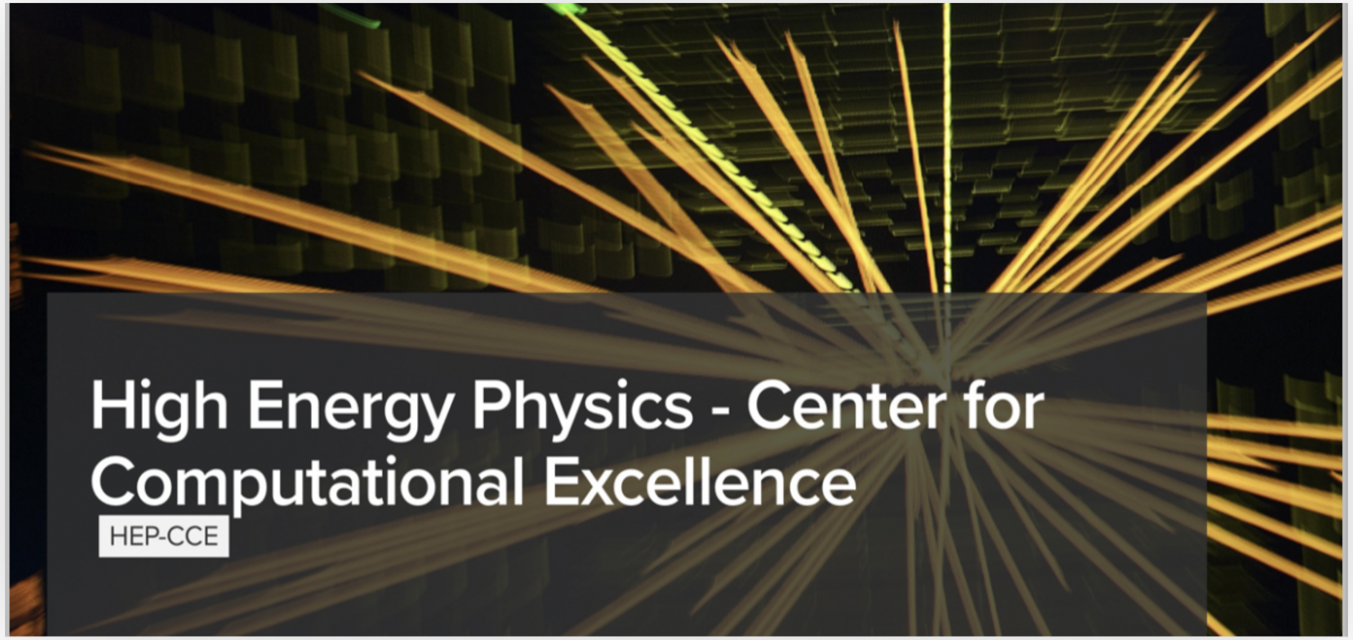
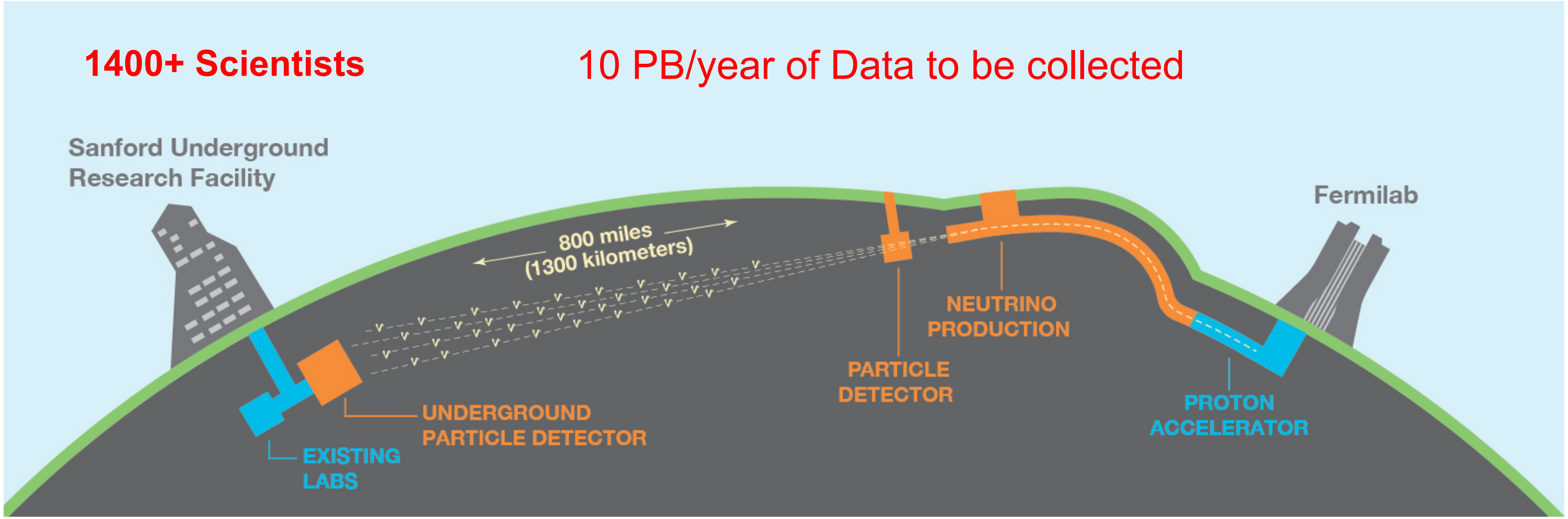
# DUNE Analysis Data Format in RNTuple

Amit Bashyal, Peter Van Gemmeren
On Behalf of HEP-CCE2/SOP
Argonne National Laboratory

July 22, 2024



# Deep Underground Neutrino Experiment (DUNE)



**1400+ Scientists**

10 PB/year of Data to be collected

Sanford Underground Research Facility
800 miles (1300 kilometers)
Fermilab
NEUTRINO PRODUCTION
PARTICLE DETECTOR
UNDERGROUND PARTICLE DETECTOR
PROTON ACCELERATOR
EXISTING LABS

- Far Detector (in SD)
- 40,000-ton x 4 Liquid Argon Detector
- Will measure neutrino oscillation
- (Beam) event:  ~6 GB/event
- Event Rate: ~mHz

- Near Detector (in IL)
- Multiple heterogenous detectors
- Will measure neutrinos at source
- (Beam) event: ~5 MB/trigger
- Event Rate: ~Hz

# Common Analysis Format Data Model

- Oscillation Experiments like DUNE require measuring the neutrino events in near and far detectors to extract neutrino oscillation parameters.

Raw Data

Hit Level Data

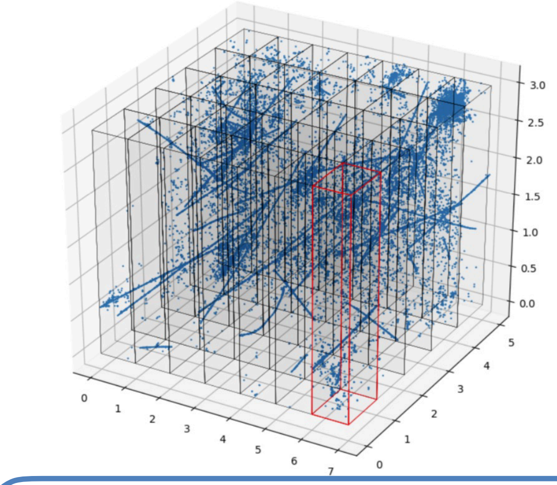
Higher Level Data

- Raw data collected in both near and far detector.

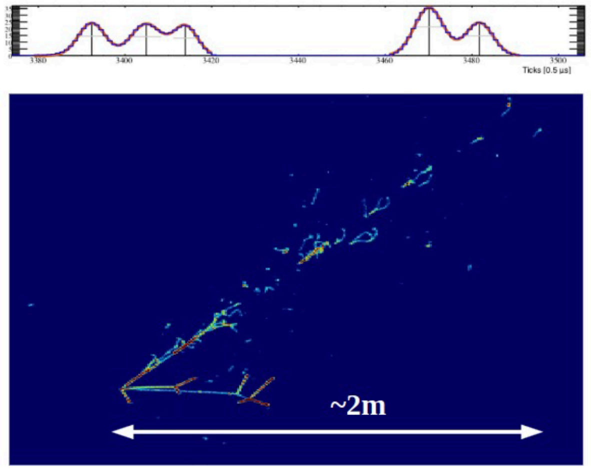
- Detailed information with intricate structure
- Problem for analyzing data with ease and speed

- Neutrino events recorded as tracks, showers and other physics variables reconstructed in both detectors

CAF records the summary of neutrino events as collection of tracks, showers and other physical attributes of particles reconstructed in near and far detectors.

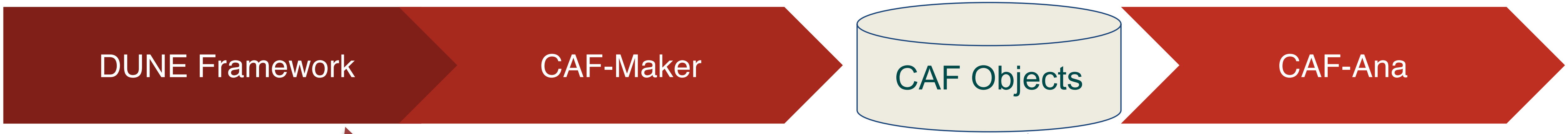


Representative ND neutrino event in DUNE



Representative FD neutrino event in DUNE

# CAF in DUNE Reconstruction Chain



DUNE Framework

CAF-Maker

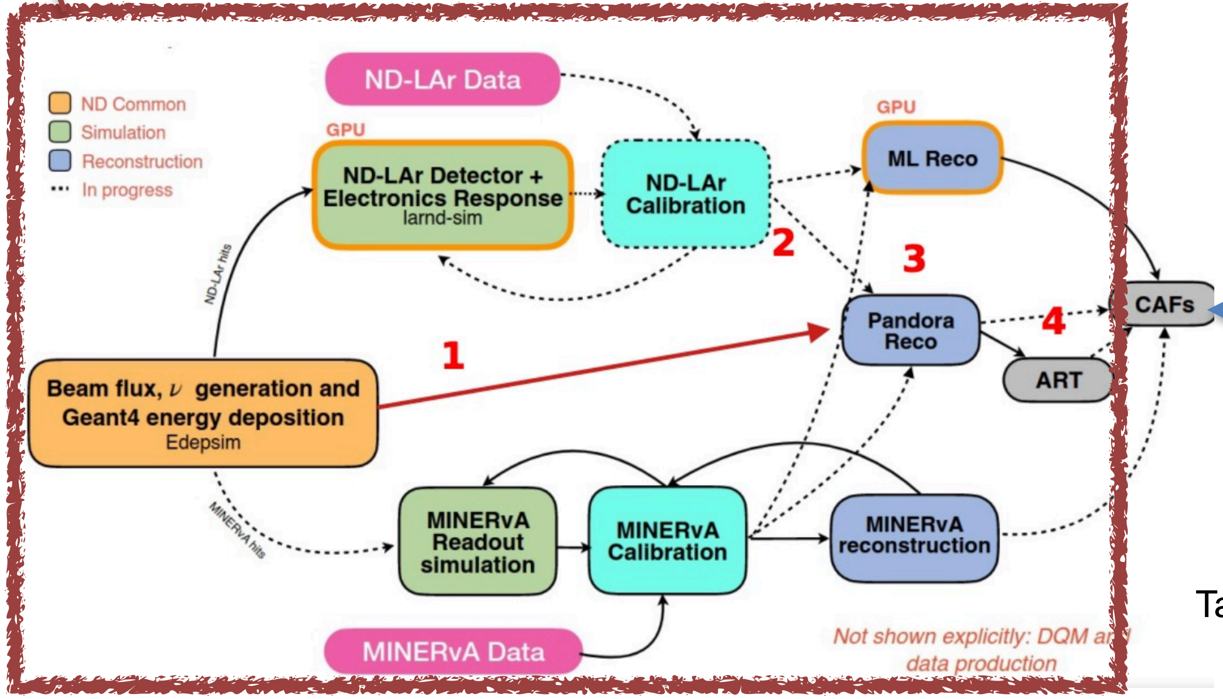
CAF Objects

CAF-Ana

- DUNE Framework (currently in design) will produce the reconstructed objects.

- Produces CAF objects

Users use prebuilt tools for analysis using CAF Objects



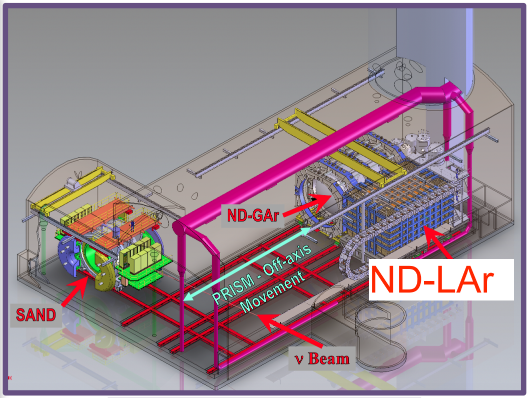
CAF-Maker produces CAF objects as end product

Reconstruction chain of 2x2 Demonstrator for DUNE

Taken from link

# CAF Data Model in DUNE

- StandardRecord: Top-level CAF Object
  - Metadata about the detectors
  - Beam Configuration
  - Generator Level Information
  - **Reconstructed variables in ND**
    - **ND-LAr**
      - Std::vector<**Reconstructed Interaction**> method1
      - Std::vector<**Reconstructed Interaction**> method2
    - ND-GAr
    - ND-TMS
    - ND-SAND
  - Reconstructed variables in FD
  - Common Reconstructed variables

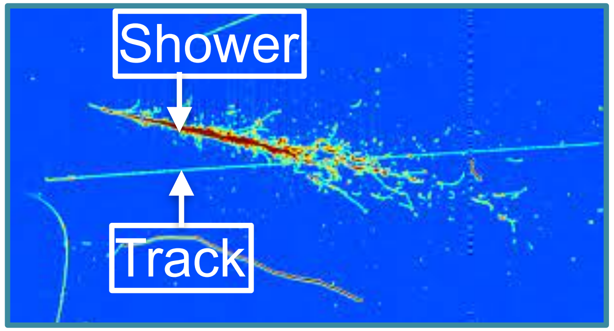


Left: DUNE ND Complex
CAF Object corresponding to Near Detector records interactions in all detectors located in the Near Detector complex.

- **Reconstructed interactions** are recorded as "**Track**" and "**Shower**" objects.
- "**Track**" and "**Shower**" objects record physical attributes like position, direction, energy deposited etc as 3D vectors, stl:::vectors and C++ simple types.

- StandardRecord (SR): Summary of neutrino event
  - Metadata, beam configuration related data , generator level data and reconstructed data
  - Records higher level reconstructed variables instead of hit-level information
- Hierarchical data model

- Member of SR that records interactions from the detectors in ND complex.
- Example: **ND-LAr** records interactions in the liquid argon detector of ND complex



A reconstructed interaction in a liquid argon detector. Taken from link

# RNTuple: Storage backend of Upcoming HEP experiments

- By 2029 RNTuple will Replace TTree as the primary I/O sub-system of ROOT
  - TTree has evolved to address HEP complex data needs
    - Over 1 Exabyte of data stored in TTree format
  - However TTree evolution predates recent overhauls in C++, modern programming paradigms and evolving computational landscape
  - TTree will be available only as legacy support (no new developments)

- RNTuple will be the new storage backend in next ROOT release (ROOT 7)
  - State of the art HEP community supported storage and I/O subsystem
  - Address the storage and I/O requirements of upcoming HEP experiments
  - Use of modern C++ standards
    - Adoption of smart pointers, better error handling mechanisms and modern C++ libraries

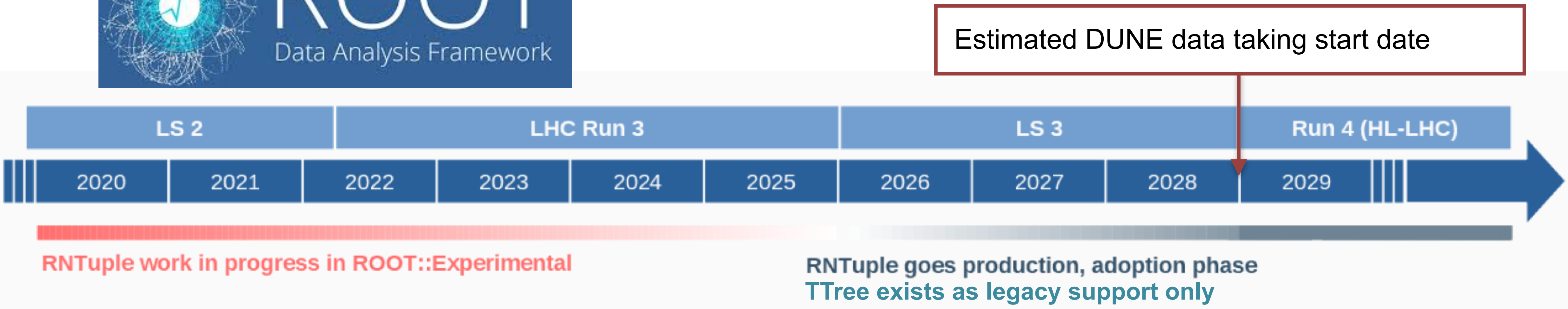
- **Upcoming HEP experiments like DUNE** will have to adopt RNTuple to stay state-of-art ROOT ecosystem

# RNTuple and the DUNE Experiment

- DUNE will start taking data by the end of this decade.
  - RNTuple will replace TTree as the primary storage backend

- DUNE Data Processing Framework (currently in design) will adopt RNTuple API
- **This talk: Adoption of RNTuple in DUNE’s analysis level data (CAF)**

ROOT
Data Analysis Framework

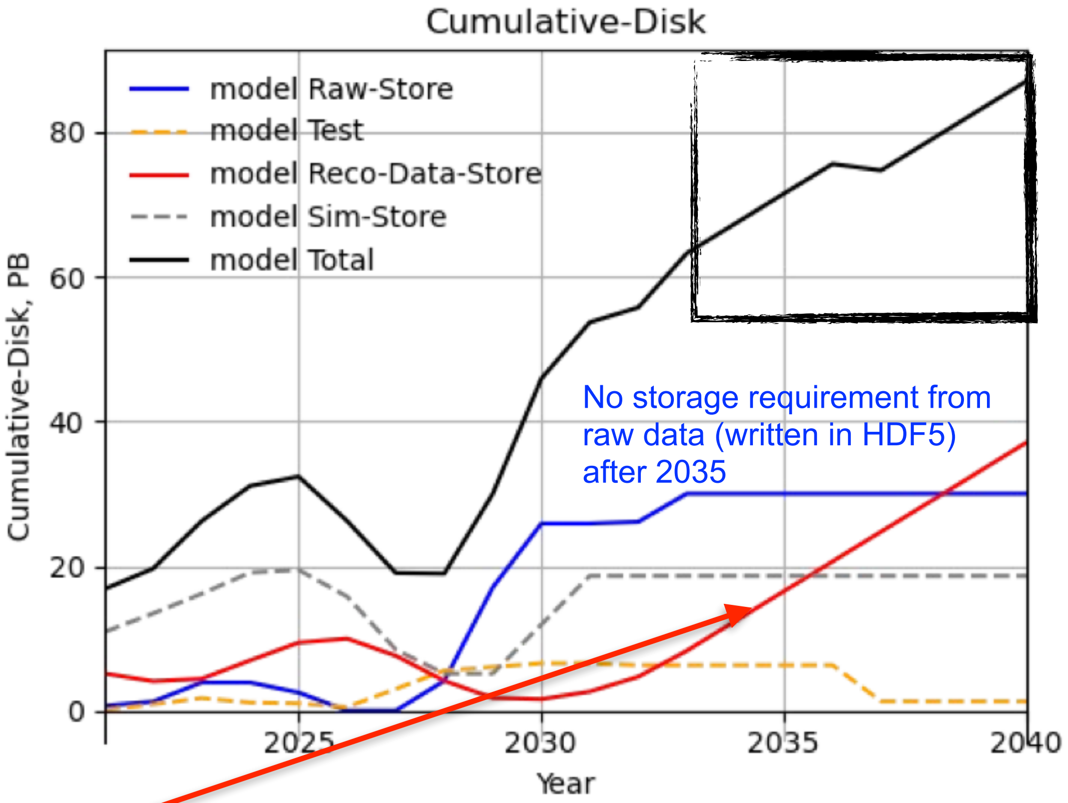

Estimated DUNE data taking start date

RNTuple work in progress in ROOT::Experimental

RNTuple goes production, adoption phase
TTree exists as legacy support only

# CAF Data Model in DUNE and RNTuple

- Currently (Proto)DUNE supports persistency of CAF objects in TTree in two ways:
  - Directly as CAF Objects
  - CAF Objects as Simple Flat Types
- DUNE must adopt RNTuple to write CAF objects
  - Storage Requirement study with RNTuple backend
  - Note that CMS and ATLAS report ~30-60% storage saving with RNTuple as storage backend
- **This work explores persistency of CAF Objects in RNTuple.**



No storage requirement from raw data (written in HDF5) after 2035

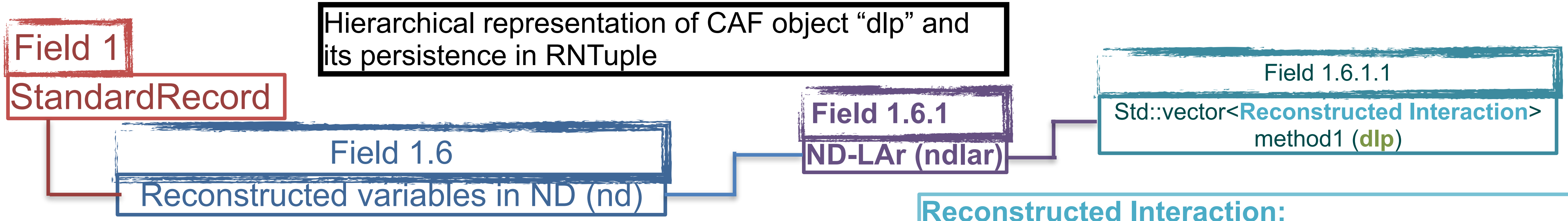
After 2030s, increase in storage requirement for DUNE from non-raw data (including CAF) that will be written in RNTuple

Projected storage requirements for DUNE over time. Taken from LBNC 2023 (Link). Projections do not consider RNTuple adoption.

# Persisting CAF Objects in RNTuple

- **StandardRecord Objects** are written as ROOT::Experimental::RFields (analogous to ROOT::TBranch)
- Each **SR object has a unique field number** with SR object members persisted as “sub-fields”, “sub-sub-fields”, …..
- Below is an example of how a persisted CAF Object is represented by RNTuple:

Hierarchical representation of CAF object “dlp” and its persistence in RNTuple



Field 1
StandardRecord
Field 1.6
Reconstructed variables in ND (nd)
Field 1.6.1
ND-LAr (ndlar)
Field 1.6.1.1
Std::vector<Reconstructed Interaction> method1 (dlp)

**Reconstructed Interaction:**

- track Field 1.6.1.1.1.1
  - Start (3D vector) Field 1.6.1.1.1.1.1
    - X Field 1.6.1.1.1.1.1.1
    - Y Field 1.6.1.1.1.1.1.2
    - Z Field 1.6.1.1.1.1.1.3
  - End (3D vector {x,y,z}) Field 1.6.1.1.1.1.2
    - ….
  - …..
- ….

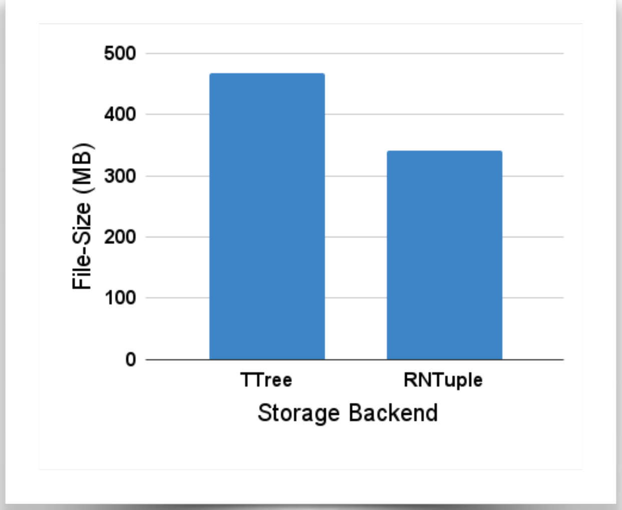


Figure: File size with 50,000 StandardRecord objects persisted in TTree and RNTuple.

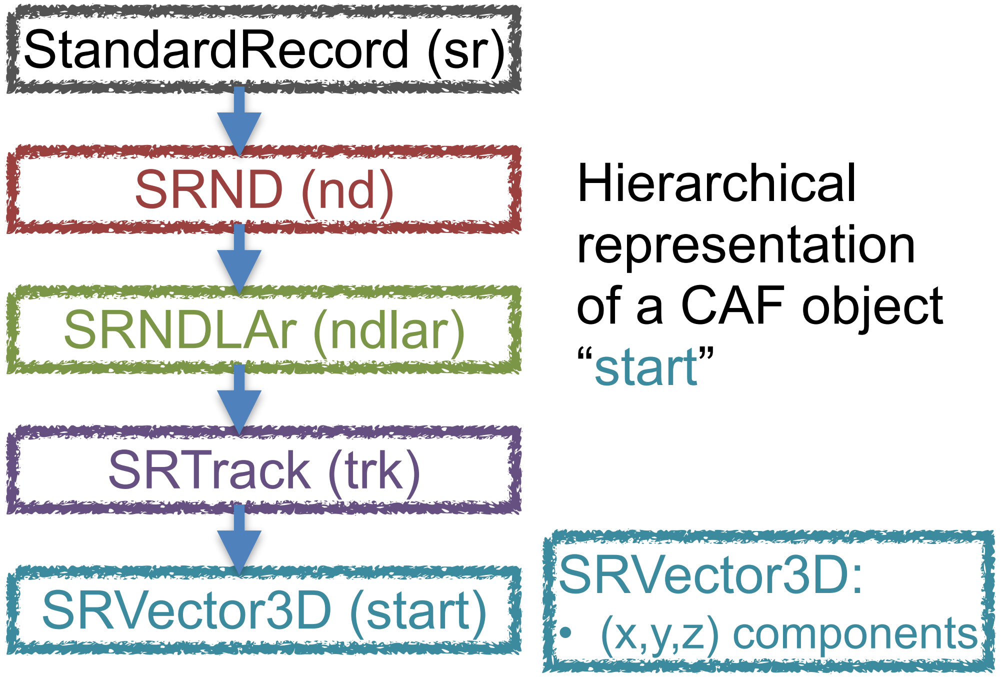
**~30% less storage requirement in RNTuple compared to TTree**

# Persisting CAF Objects as Flat Types in RNTuple

- Flattened CAF Objects are simple C++ type objects or arrays
- Redesigning I/O tools (Proxy reader/writer) needs heavy lifting
  - Heavy dependence upon TTree and related APIs, pointers, raw arrays (T[N])

**Implemented Proxy writer with RNTuple support**

- TTree APIs replaced by RNTupleModel APIs
- Discard raw arrays in favor of std::arrays or stl::vectors
- Modification of functions that used raw pointers to write data
- Naming scheme adjustment to support RNTuple
- Each SR Object is decomposed into over 1300 Fields of flattened CAF object members (arrays and non-arrays)



StandardRecord (sr)

SRND (nd)

SRNDLAr (ndlar)

SRTrack (trk)

SRVector3D (start)

Hierarchical representation of a CAF object “start”

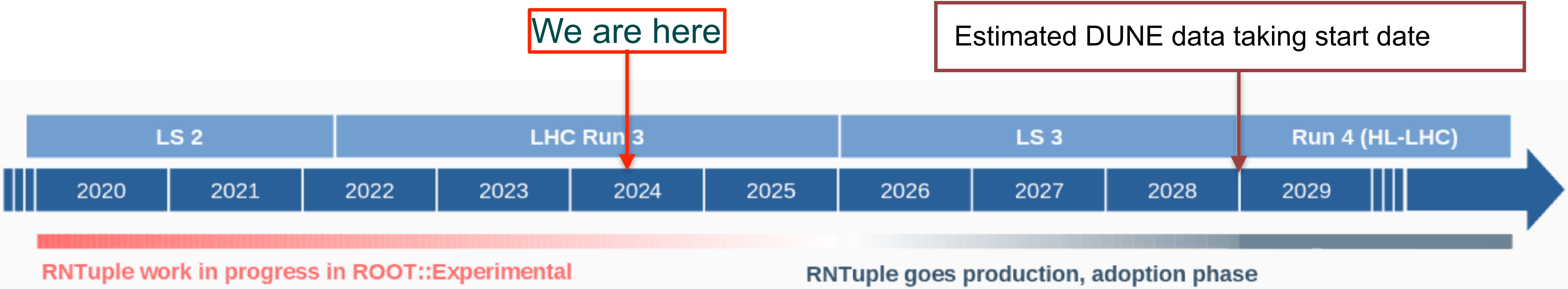
SRVector3D:
- (x,y,z) components

CAF object “start” flattens as:

- Field N: sr_nd_ndlar_trk_start_x
- Field N+1: sr_nd_ndlar_trk_start_y
- Field N+2: sr_nd_ndlar_trk_start_z

# Conclusions and Outlook

- **Below is the current situation from DUNE’s perspective**



We are here

Estimated DUNE data taking start date

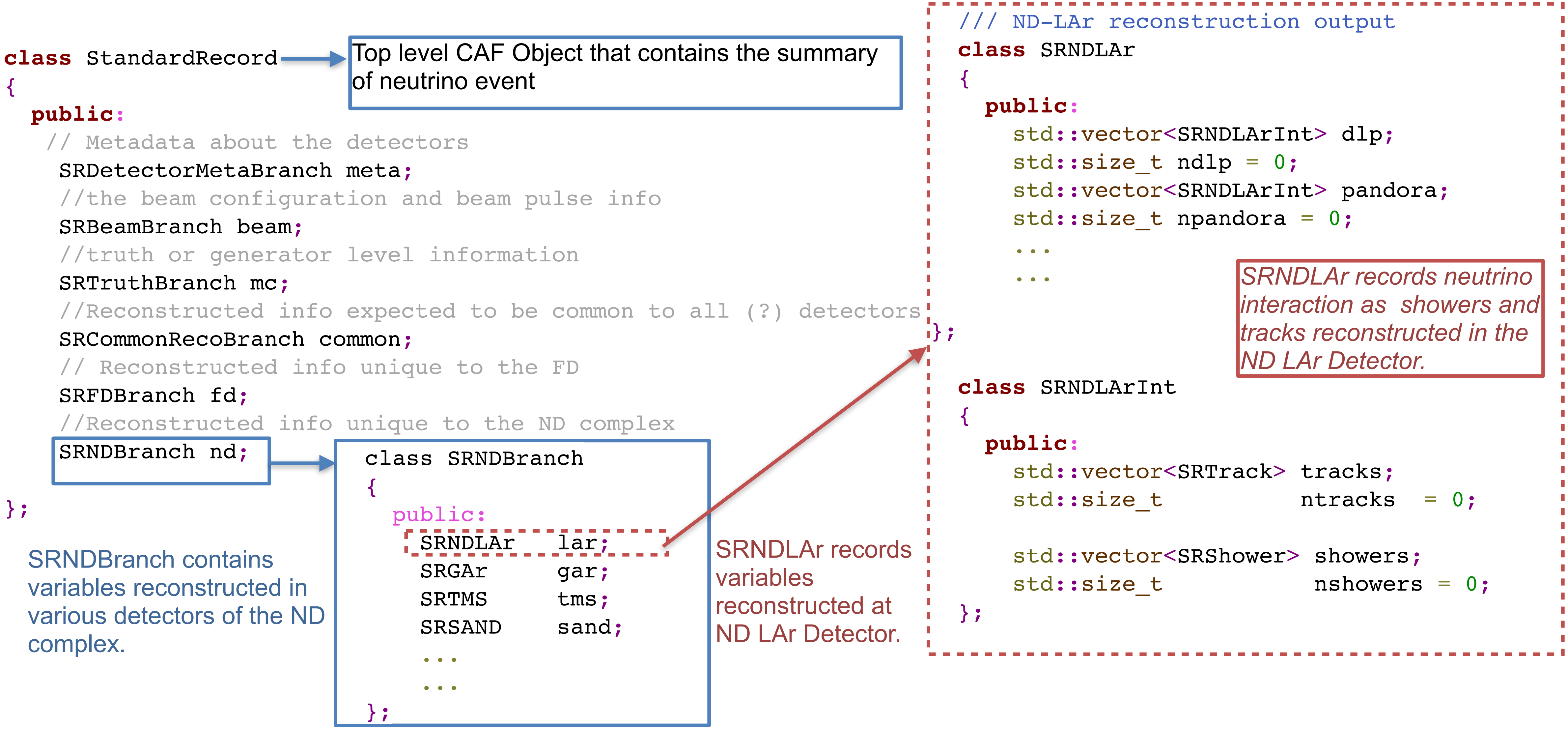
LS 2
LHC Run 3
LS 3
Run 4 (HL-LHC)
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
RNTuple work in progress in ROOT::Experimental
RNTuple goes production, adoption phase

- DUNE’s Data processing Framework is currently in development which has to fully adopt RNTuple
- This work shows that RNTuple can support DUNE’s analysis level data (CAF)
  - Supports persistence and I/O of both CAF objects and flattened CAF objects with Proxy Writers

- **Deliverable as part of CCE-2 Effort for the DUNE collaboration**
  - Collaborate with DUNE stakeholders (CAF developers) for testing and integration into core framework

# Thank You!

# CAF Data Model in DUNE



```
class StandardRecord
{
  public:
   // Metadata about the detectors
    SRDetectorMetaBranch meta;
    //the beam configuration and beam pulse info
    SRBeamBranch beam;
    //truth or generator level information
    SRTruthBranch mc;
    //Reconstructed info expected to be common to all (?) detectors
    SRCommonRecoBranch common;
    // Reconstructed info unique to the FD
    SRFDBranch fd;
    //Reconstructed info unique to the ND complex
    SRNDBranch nd;

};
```

Top level CAF Object that contains the summary of neutrino event

```
class SRNDBranch
{
  public:
    SRNDLAr    lar;
    SRGAr      gar;
    SRTMS      tms;
    SRSAND     sand;
    ...
    ...
};
```

SRNDBranch contains variables reconstructed in various detectors of the ND complex.

SRNDLAr records variables reconstructed at ND LAr Detector.

```
/// ND-LAr reconstruction output
class SRNDLAr
{
  public:
    std::vector<SRNDLArInt> dlp;
    std::size_t ndlp = 0;
    std::vector<SRNDLArInt> pandora;
    std::size_t npandora = 0;
    ...
    ...
};

class SRNDLArInt
{
  public:
    std::vector<SRTrack> tracks;
    std::size_t          ntracks   = 0;

    std::vector<SRShower> showers;
    std::size_t           nshowers = 0;
};
```

*SRNDLAr records neutrino interaction as showers and tracks reconstructed in the ND LAr Detector.*