IRIS-HEP Fellowship Presentation:

Adding RNTuple to the Analysis Grand Challenge

Fellow: Giedrius Juškevičius

Mentor: David Lange

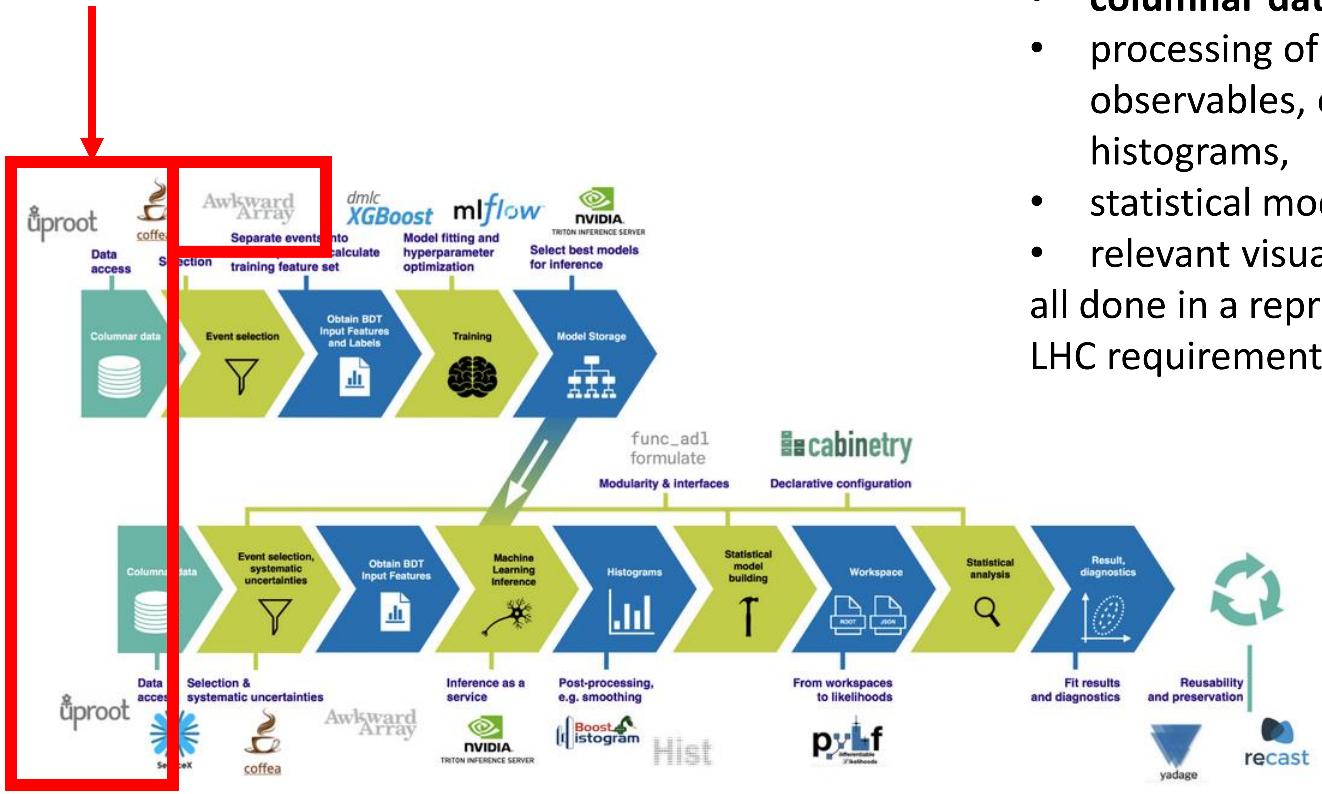
2024-09-18

The bigger picture

- The High-Luminosity Large Hadron Collider (HL-LHC) project aims to crank up the performance of the LHC in order to increase the potential for discoveries after 2029.
- The objective: to increase the integrated luminosity by a factor of 10 beyond the LHC's design value.
- The higher the luminosity, the more data the experiments can gather to allow them to observe rare processes.
- However, analysis workflows commonly used at the LHC experiments do not scale to the requirements of the HL-LHC.
- To address this challenge, the IRIS-HEP software institute started the "Analysis Grand Challenge" (AGC) project **to optimize the data pipeline**.

AGC pipeline and scope of the task

Scope of my task,
which includes using these Python tools:
uproot, Awkward Array, coffea



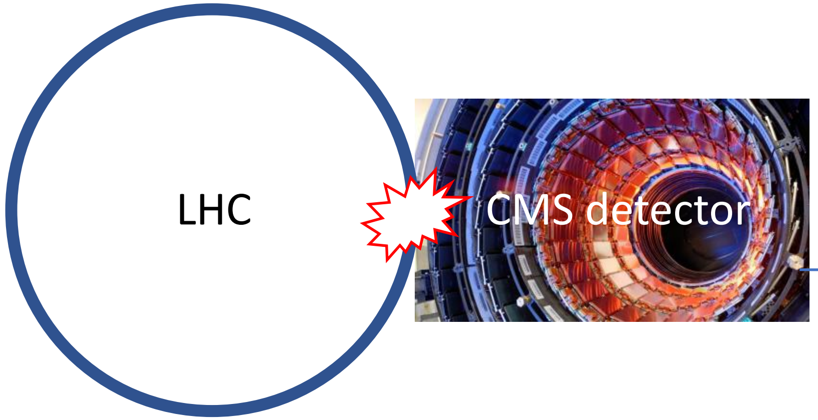
The Analysis Grand Challenge (AGC) is about performing the last steps in an analysis pipeline at scale to test workflows envisioned for the HL-LHC. This includes:

- **columnar data extraction from large datasets,**
- processing of that data (event filtering, construction of observables, evaluation of systematic uncertainties) into histograms,
- statistical model construction and statistical inference,
- relevant visualizations for these steps,

all done in a reproducible & preservable way that can scale to HL-LHC requirements.

Source: https://agc.readthedocs.io/en/latest/#

Columnar data storage



Abstraction level

Coffea (Python) — Basic tools and wrappers for enabling not-too-alien syntax when running columnar Collider HEP analysis.

Uproot (Python) — Uproot can access data in ROOT files without a dedicated C++ ROOT compiler.

Columnar data storage (ROOT data format)* — A data format developed by CERN, designed to efficiently store large amounts of data and support common operations on it.

For 25+ years, columnar data was structured in **TTree** format.
RNTuple is a redesigned I/O subsystem aiming to:

- Reduce disk and CPU usage for the same data content;
- Be supported by modern hardware (built for multi-threading and async I/O);
- [other more]…

Criteria of my current task:

- **uproot** and **coffea** (was not able to continue) workflows should support RNTuple data files without errors;
- After comparing TTree and converted RNTuple files, data should not be corrupted;
- RNTuple workflow performance should be compared to TTree;
- Found issues should be documented for the team.

* ROOT: C++ interpreter and data format developed by CERN - both of them have the same name.

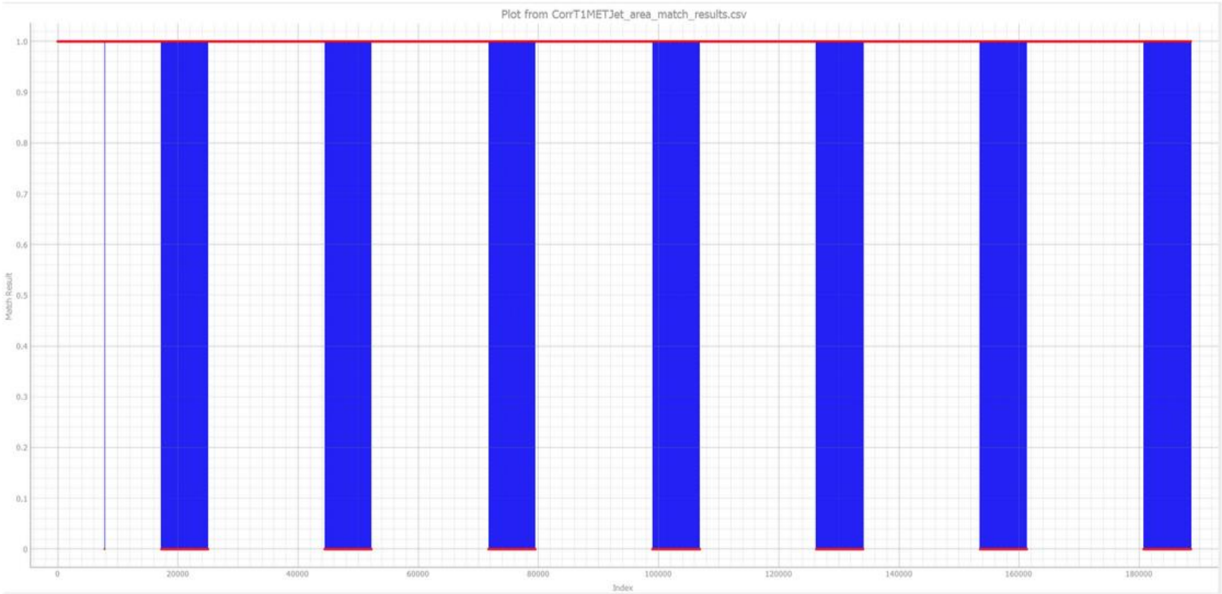
Technical steps

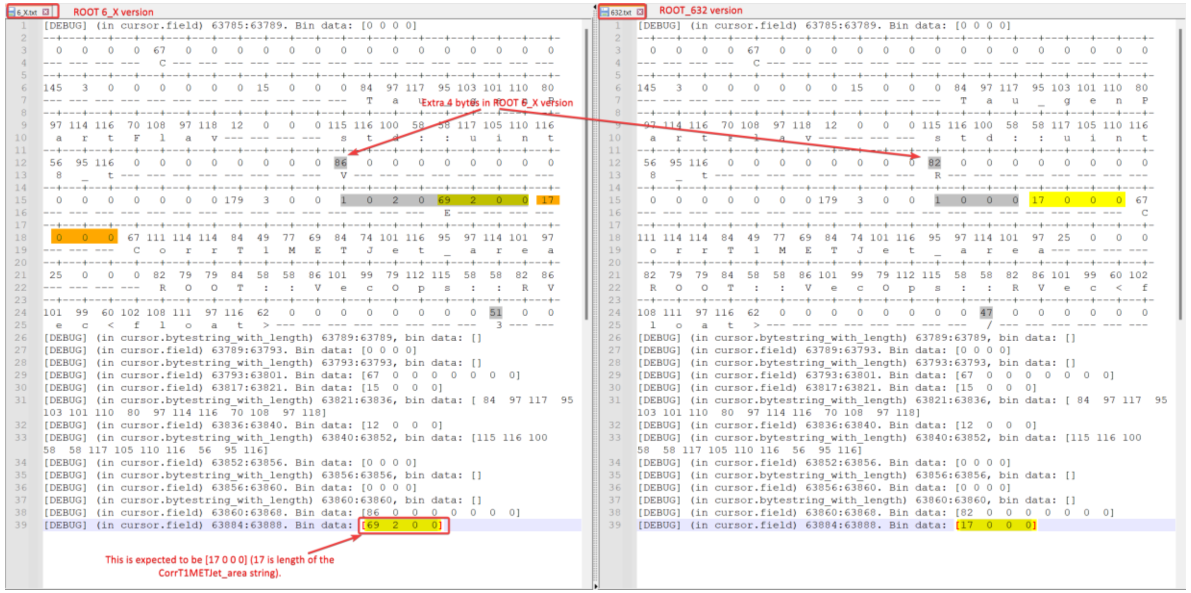
- Started with already existing Jupyter notebook workflow in Github, which works with TTree without issues; (Github link: agc-coffea-2024.ipynb)
- Used RNTuple files on already existing TTree analysis workflow;
- Setup loading RNTuple files while using two approaches: **uproot** and **coffea**;
- Eventually, the coffea approach was blocked because Dask is not yet supported by uproot’s RNTuple interface.
- Created our own RNTuple files with ROOT instead of using preexisting ones;
- Main focus was put on uproot approach; multiple problems related to RNTuple were found; time was spent debugging, documenting and solving these problems in uproot repository;
- After solving critical bugs, multiple tests were created to compare the operation performance between TTree and RNTuple files when using uproot.

Tools used

- Using external computing resources: **Coffea Casa server**
- Easily managing and sharing code results: **Jupyter Notebooks**
- Loading and analyzing ROOT data: **Coffea, Awkward array, Uproot**
- Visualizing data for easier debugging**: matplotlib, PyQT5**
- Creating RNTuple files from TTree: **LXPLUS** server, **ROOT**
- Managing different versions of code and contributing: **git, Github**
- Sharing code snippets more easily: **Github gists**

Error analysis examples





When visualizing data match results for TTree and RNTuple, it became clear that bug is related to clusters (there are 7 clusters in the example above).

After RNTuple file reading failure, failing and non-failing files where compared byte by byte. This helped to understand the cause of error.

Issues found in uproot’s RNTuple object

- While using Coffea, RNTuple object was missing some keys() arguments (solved)
- Dask is not supported by RNTuple interface yet (**RNTuple blocker**)
- RNTuple object has no attribute 'num_entries’ (solved in draft PR from the past)
- Data integrity was lost when loading arrays from multiple arrays (solved)
- Uproot file loading fails when using RNTuple file created with recent master ROOT (issue created)

Performance test results (uproot scope)

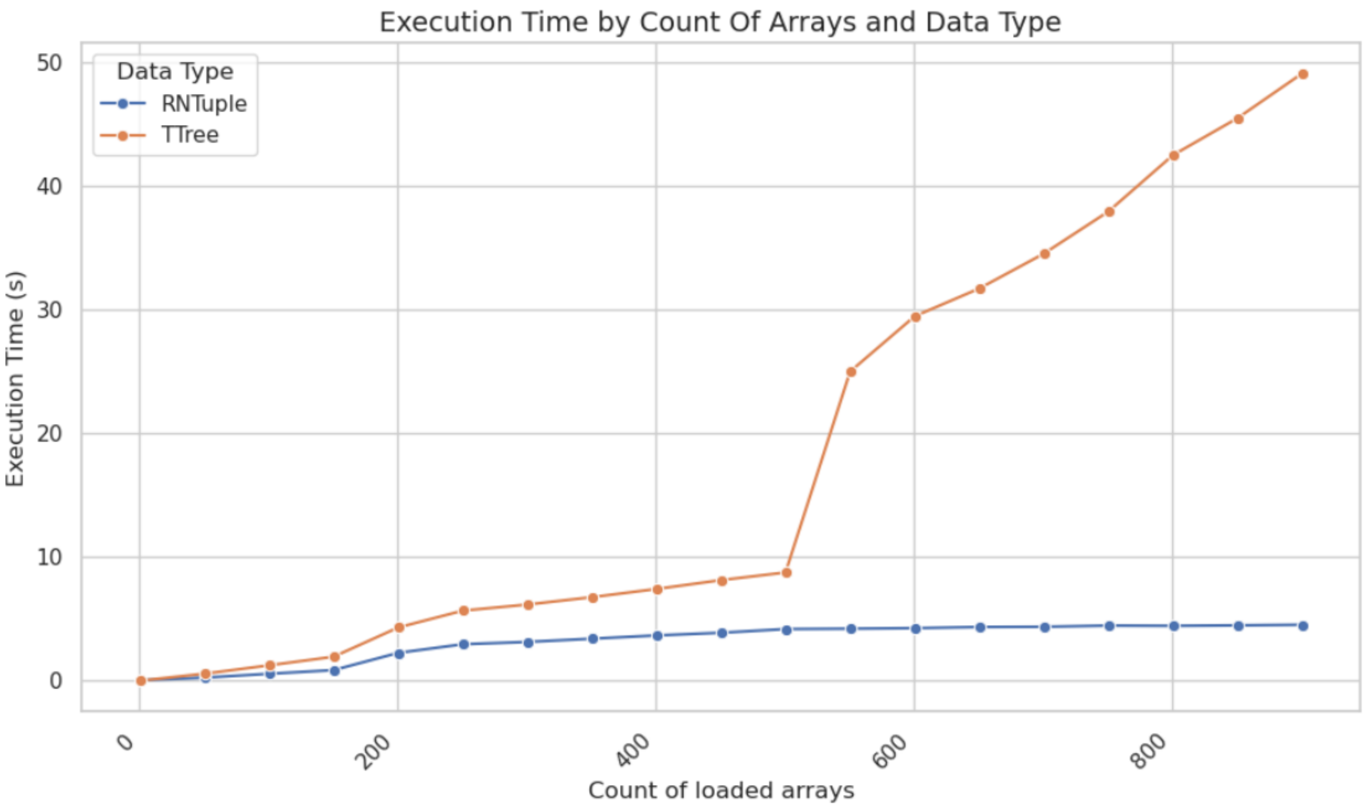
- Notebook link: [.../calver-coffea-agc-demo/.../ttree_and_rntuple_comparison.ipynb]

Table 1. Execution duration comparison for various operations (in seconds)

Operation	RNTuple	TTree
load 24 arrays while using filter name	0.902	1.720
load all arrays	4.629	46.538
load all arrays while using filter name	4.476	52.715
load one array while using filter name	0.157	0.250
load arrays for each key	54.289	20.028
load file	0.001	0.303
Total	64.454	121.553

```
def load_arrays_for_each_key(events):
    for key in events.keys():
        events.arrays(filter_name=[key])[key]
```

(Not recommended operation, because currently it is very slow.)



Potential next steps

- RNTuple's data integrity must be ensured with each new ROOT release;
- The support of Dask and coffea is required to distribute computations across computer clusters when using RNTuple;
- Performance testing should be done in the Dask/coffea scope when using RNTuple;
- The use cases of physics analysis should be examined to create additional performance tests within the scope of uproot.

Questions

References

- https://cmsexperiment.web.cern.ch/detector/identifying-tracks
- https://github.com/iris-hep/analysis-grand-challenge/blob/main/analyses/cms-open-data-ttbar/ttbar_analysis_pipeline.ipynb
- https://indico.cern.ch/event/1168602/contributions/4907387/attachments/2456778/4210978/Uproot.pdf
- https://coffeateam.github.io/coffea/