RNTuple Infrastructure needed for production workflows in ATLAS

Alaettin Serhan Mete

Argonne National Laboratory

with inputs from

Peter Van Gemmeren *(ANL)*, **Marcin Nowak** *(BNL)*, **Maciej Szymański** *(ANL)*

RNTuple Format and Feature Assessment
6-7 November 2023

Introduction

- **Full production needs a lot more than reading/writing `RNTuples`**
- **The shopping list includes (but is not limited to):**
  - Custom indexing (a la `TTree::BuildIndex`)
  - Fast Merging `RNTuples`
  - Having various utilities/tools to peek into, compare, validate, … `RNTuples`
  - Having the ability to optimize parameters for various use-cases
  - Having support for relational `RNTuples` (a.k.a. *friendship*)
- **As in any such big migration, new hurdles will be uncovered**
  - Therefore, having a close collaboration along the way is extremely crucial
- **This talk will highlight some of these essential topics**
  - Focus is given to those that need input/work beyond ATLAS/Athena
  - It would, nonetheless, be beneficial to share work/expertise across experiments if applicable

Custom Indexing (a la `TTree::BuildIndex`)

- **A while ago we switched to building/using custom indices**
  - Each tree has an additional branch, called `index_ref`, holding a unique id for each event
  - For each object the value of this index is stored in a token (along w/ the branch info. etc.)
  - When reading an object, we use this information through `TTree`'s custom index support
    - `TTree::BuildIndex`, `TTree::GetEntryNumberWithIndex`, etc.
- **This functionality is essential for a number of use cases**
  - Fast merging worker outputs in AthenaMP jobs with `SharedWriter` (more on this later on)
  - Cross-referencing event sample augmentation trees (more on this later on)
- **The same functionality is also needed for `RNTuple`**
  - In the current prototype we *essentially* use simple row indices
    - We have an associated field for `index_ref` but this is internally mapped to a plain index

Fast Merging (via `TFileMerger`)

- **We use fast-merging primarily in the DAOD production jobs**
  - These jobs execute in multi-process Athena (`AthenaMP`) w/ `SharedWriter`
    - Each worker produces its own in-memory output (via `TMemFile`)
    - This allows us to parallelize CPU intensive I/O operations such as compression
    - Then `SharedWriter` fast-merges the worker outputs into a single file
      - Following `parallelMergeServer`/`parallelMergeClient` approach of ROOT
- **This relies on ROOT's ability to fast merge `TTree`/`RNTuple`**
  - For now we cannot utilize this version of `SharedWriter` for `RNTuple` samples
  - `SharedWriter` also supports another, so-called, *legacy* mode that can handle `RNTuples`
- **We have an ongoing effort w/ the core ROOT I/O team on this**
  - A functional `RNTupleMerger` prototype is being iterated at root/pull/13858
  - It will definitely need a number of follow-up PRs to fully iron things out (`hadd` etc.)

Miscellaneous Topics

- **We have a number of tools built around the `TTree` infrastructure**
  - Peeking into in-file meta data (see Maciej's talk) to configure the job based on the input file
  - Checking if two files are content-wise identical as far as event/meta data is concerned
  - Validating the output file at the end of the job to ensure there is no corruption
  - Summarizing the file content in terms of in-memory/disk-space sizes per container
- **Some are inherently ATLAS specific but some can be shared**
  - A good candidate is comparison of files, which is a crucial functionality for data processing
  - Another good candidate is validating against data corruption etc.
- **Are there any plans within ROOT and/or can such an effort be coordinated cross-experiment?**
  - Such an effort can be coordinated elsewhere, e.g., HEP Software Foundation etc.

Miscellaneous Topics (cont'd)

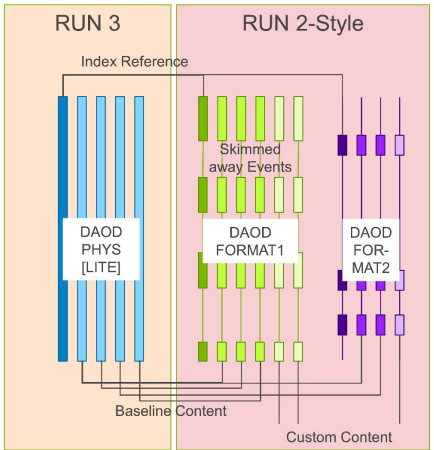
- **Support for various modes of operations: Athena(MP/MT)**
  - Athena workflows support serial, multi-process, and multi-threaded modes
  - We also utilize ROOT's IMT (both ROOT and Athena uses TBB, hence share thread pools)
- **Recently discovered AthenaMT + ROOT IMT + `RNTuple` don't get along very well**
  - Originally noticed in multi-threaded reconstruction jobs producing AODs w/ `RNTuple`
  - Traced back to the default buffered writing (likely compression, but to be followed-up)
  - Worked around (for now) with `RNTupleWriteOptions::SetUseBufferedWrite( false )`
- **Different modes/workflows also have different conditions**
  - For example, how we read input data: Linear vs semi-linear vs random event access etc.
  - Detailed studies w/ `TTree` were done that resulted in, e.g., root/pull/1065 from 2017
  - Needless to say, similar studies need to be performed for `RNTuple`, too
- **Uncovering/debugging/fixing such issues rely on early testing**
  - We shouldn't overlook the importance of such subjects

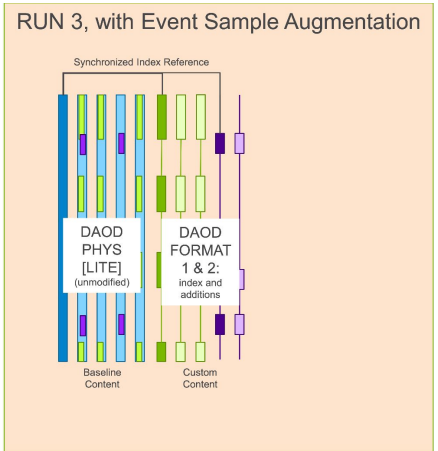
Custom Optimizations for Different Use-cases

- **Needless to say, not all data products are the same**
  - For example, event size can vary a lot between up/down-stream data products
- **Over the years we optimized these for the `TTree`**
  - A prime example of this is the event clustering that can be tuned w/ `TTree::AutoFlush`
  - Such optimizations have implications on CPU/memory/disk-space usage, quite complex!
- **How these would translate into `RNTuple`?**
  - What would be the knobs we can turn etc.
  - We should have ample time to test the pre-production infrastructure well ahead of time
- **I think this is another synergetic topic across all clients**
  - Sharing expertise together w/ the guidance of the core ROOT I/O is very important

Event Sample Augmentation (a.k.a. *friendship*)

- **In Run-3 we had a paradigm shift in the analysis model**
  - Left Run-2's many (skimmed) DAODs in favor or a single-ish (unskimmed) DAOD(_PHYS)
  - What happens if (many) analyses need (even a handful of) additional information in this format?
    - Option 1: Expand the common format w/ the new variables
      - Wasteful because not all events need all variables unconditionally
    - Option 2: Copy the common variables + new variables in a new format (Run-2 Style)
      - Wasteful because one replicates the (possibly large) common data unnecessarily
    - Option 3: Augment the main unskimmed data w/ skimmed variables (Event Sample Aug.)





See Peter's CHEP talk

Event Sample Augmentation (a.k.a. *friendship*)

- **Prototype shows promising results in terms of disk-space**
  - Augment unskimmed PHYS content w/ Long-Lived Particles (LLP) content
    - Increase the event size by 40%, for skim of 40% of the events
    - **Run-2 Style Approach:** 140% extra data for 40% of events → **56% more storage**
    - **Event Sample Augmentation:** 40% extra data for 40% of events → **16% more storage**
- **`TTree` based implementation**
  - Use a custom branch as the main index (via `TTree::BuildIndex`)
    - Synchronized across all trees and used to cross-reference events across trees
  - Athena has its own navigational structure to handle reading/writing
  - However, physics analyses software rely on the `TTree` friendship concept
- **A similar functionality in `RNTuple` would be the ideal scenario**
  - `RPageSourceFriends` is perhaps the way forward with this…

Conclusions & Outlook

- **Adopting RNTuple in production needs a lot of work!**
  - The effort goes well beyond just data model support (beyond ATLAS, too)
- **We need a number of core functionality on the ROOT side**
  - We highlighted a few of these in this talk, e.g., fast merging RNTuples, friend RNTuples etc.
- **A number of tools are probably needed by all experiments**
  - For example, diff-ing RNTuples for ensuring binary identicality of event/meta data etc.
  - Collaborating w/ other customers on these can be beneficial to all
- **Core Athena support ≠ Everything is done**
  - Physics analyses need to adopt the new technology as well, which needs time/effort
- **Our overall goal/plan is to:**
  - **Address all open issues on the core Athena side by the end of Run-3, i.e., next ~1.5 years**
  - Use much of the Long Shutdown afterwards to test/optimize/deploy in production scenarios

Thank you for your attention!