# CMS feedback on `RNTupleReader` and `RNTupleWriter`

Chris Jones, Matti Kortelainen, Dan Riley

HEP-CCE SOP meeting

3 April 2024

# High level

- Separate APIs for reading (`RNTupleReader`) and writing (`RNTupleWriter`) matches what the CMSSW framework does already
  - But we would not be surprised if some CMS users would expect read-write API
    - But strictly speaking that is outside of the scope of the review
- CMS wants the “application ownership” model for the data that interacts with RNTuple

# RNTupleParallelWriter

- architecture.md under Ownership model: “The parallel writer assumes exclusive access to the underlying file during the entire lifetime of the writer”
  - It seems we could not use the parallel writer, because we need to have several writers associated to the file in order to store our different top-level RNTuple objects (“Event”, “LuminosityBlock”, “Run”)
    - We believe having an asynchronous API for the TFile access would help with this problem

# RNTupleParallelWriter (2)

- RNTupleParalleWriter documentation also says there is no guaranteed entry order between different RNTupleFillContext objects. The only guarantee is between to consecutive Fill()s to RNTupleFillContext, the entry number will have gone up (but not necessarily with consecutive numbers).
  - Makes our own metadata impossible to track
  - Made us realize a missing requirement from our side: **We need to be able to associate a Run/LuminosityBlock/Event ID to the entry ID in a given TTree/RNTuple that holds the Run/LuminosityBlock/Event**
  - Presently (with TTree) we assume the order we call Fill() is the order in which entry numbers for the Run/LuminosityBlock/Event are assigned
    - RNTupleParallelWriter explicitly breaks this assumption
    - It is unclear if we could reorganize CMSSW code such that it would be compatible with RNTupleParallelWriter
  - If RNTupleParalleWriter would tell us, in some way, the final entry index associated to each Fill() call, we could easily continue our own metadata tracking

# `RNTupleParallelWriter` (3)

- Can `RNTupleParallelWriter::CreateFillContext()` be called during the data processing?
  - This idea might help to logically group Events from the same LuminosityBlock, so that the data of these Events would be written to the file close to each other
- We want to explore further parallelization strategies that would be compatible with RNTuple file format specification

# RNTupleReader

- architecture.md has “When reading data, RNTuple uses the `RRawFile` class unless a `TFile` is explicitly enforced (e.g. for reading from a `TMemFile`). The `RRawFile` owns its own file descriptor and does not interfere with `TFile` objects concurrently reading the file.”
  - How does one “explicitly enforce” reads to be done via `TFile`?
  - Can we register our own RRawFile handlers per-protocol basis?
    - Presently CMS interposes its own handlers (`TFileAdaptor`) to tweak the I/O parameters for the characteristics of different storage systems, and tune xrootd settings. E.g. :
      - Xrootd: retries, multisource, pre-fetching etc
      - Site-specific optimizations via site configuration
    - We need RNTuple to not bypass the customizations we do for reading
    - We also need to figure out ourselves what are the exact behaviors of TFileAdaptor we want, and then figure out what would be the best way to achieve those behaviors
      - Ideally at least some of the knobs would be included in upstream ROOT/xrootd

# RClusterPool

- CMS has done lots of optimization on its I/O, and we want to be sure that we will be able to keep our system at least as optimal as presently
  - From the documentation alone it is not clear if the present `RClusterPool` would do the job
    - E.g. with `TTreeCache` we needed ways to tell how we configure it

# Documentation

- We would like to have a better description of the behavior of RNTuple reading and writing when implicit multithreading is enabled
- `RNTupleReader` documentation has several “must” statements. It would be good to document what happens if the condition is violated (undefined behavior, exception, error message …)

# Lesser details

- architecture.md on `RNTupleModel` has “Addition of fields invalidates previously created entries.” Does “entries” mean REntry objects? Or also entries already written to the disk?
- `RNTupleView<T, bool>` should really be two different classes for the two values of `bool`
  - When `bool` is false, some of the member functions are not supposed to be called
  - Would be better handled by having two different classes: `RNTupleView` and `RNTupleUserOwnedView`.
  - We have already communicated this feedback to ROOT team