First Feedback Summary

RNTuple Format and Feature Assessment
2023-11-07

ROOT
Data Analysis Framework
https://root.cern

Feedback and Questions

- Features foreseen for deprecation
  - There is a path towards removal of dynamic polymorphism and network pointers, which includes
    - Support of std::variant in current production ROOT I/O
    - Addressing TTree schema evolution issues for a gradual EDM transition
    - HEPMC, TH1*: no unsplit storage
  - Keep std::map and std::set in RNTuple
  - TTree::Scan to be replaced by RDataFrame, TTree::Show available in RNTuple
  - TTree::SetAlias used only to rename branches
    - Equivalent functionality exists with RNTuple "projected fields"
  - No obvious blockers regarding other feature deprecations
    (TRef, in-memory trees, recursive data structures, page-level selection of compression algorithm)

Feedback and Questions

- Type support in RNTuple
  - Full functionality of ROOT low-precision floats in RNTuple
    - Work in progress on better ways of controlling nested use of low precision floats (e.g., in collections)
  - No low-precision ints foreseen (should be handled well by compression)
    - May change if there is a need for packed ints in memory (?)
  - Need for std::unordered_map, std::unordered_set, multi-dimensional C-style arrays
  - Support for std::span, std::mdspan
    - Nothing prevents us from adding support but we would wait for a concrete need, i.e. currently not foreseen for a first production version
  - I/O for Python dictionaries: is there a concrete need?
    - std::map can be instantiated from a Python dict in cppyy

Feedback and Questions

- Reading and writing
  - TTreeCache fully replaced by RClusterPool (async cluster preloading, turned on by default)
  - Bulk read & write API: need for an option to put the framework in charge of the memory allocation
  - Need indexed (friend) event iteration for the first production release
  - Plans for concurrent writes
    - "Mild scalability": one entry per thread, filling (= serialization) protected by a mutex
      - Low/no memory overhead
    - "High scalability": one cluster per thread
      - Serialization + compression lock-free, thread-driven (= parallel) writing with very brief serialization
      - Requires order of cluster size extra memory per thread (<100MB)
      - Can evolve into more sophisticated version where several threads aggregate writing, foreseen as a feature after the first production release
  - Plans for direct I/O to GPU memory: absolutely, but likely not for the first production release

Feedback and Questions

- Requests for the existing production I/O system
  - Desirable to generally propagate standard int types down to the ROOT I/O layer
  - Desired improvement for enums: load dictionary without auto-parsing header
  - Passing arbitrary data to the read rules
- Common ground for additional tooling (e.g., data diffing, validation)
  - Maybe a good opportunity for external contributions?
- Schema evolution will work in the same way than for TTree
  - Note that we are likely starting "fresh" with data stored in RNTuple,
    which emphasizes good testing since we don't have the real-world validation of year of class schema history
- TTree will always be part of ROOT
  - Focus of attention gradually moves to RNTuple

Feedback and Questions

- Meta-data: there is a sense of a common problem core shared by all experiments, but that core is not yet clearly identified
  - Proposal: dedicated meta-data functionality in RNTuple after the first production version
  - Should use next year to narrow down the problem
    - Possible subject/center of next RNTuple workshop.
- I/O for SoA
  - Bulk reading of entire columns (ALICE case)
    - Benefit from new interface to communicate the read buffer location to reduce memcpy
  - Reading of classes into a user-provided layout (CMS case)
    - Requires a new interface to communicate the in-memory layout between application and ROOT

Work Scheduling Notes

- High-priority items to unblock other activities
  - Fast merging to start larger-scale ATLAS workflow tests
  - Chains to start AGC tests
  - Remaining type support for CMS MiniAODs
    - Modulo dynamic polymorphism but with required bits for gradual transition
  - API adjustments in preparation of the interface review