High Energy Physics
Center for Computational Excellence
Optimizing Data Storage for Next-Generation HEP Experiments

Peter van Gemmeren
for HEP-CCE

ROOT RNTuple API Review
September 11 2024

HEP-CCE Review findings and ROOT team responses

1. Need mechanism to **customize page size** for particular fields:
   - ATLAS sees that few/certain data objects are too large to achieve good compression with standard page size (cases where TTree optimized baskets to be >~ 1 MB) and storage sizes go up significantly.
   - If not addressed this can offset a large fraction of storage savings from RNTuple or even lead to an overall increase. Therefore, we would like the capability to overwrite/increase the default page size for some fields.

ROOT will address this (and point 4) as [ntuple] Adaptive page sizes by jblomer · Pull Request #16311 · root-project/root · GitHub
*and believe that this will make manual page size tuning for specific fields unnecessary (**to be validated**).*

2. RClusterPool: Based on past experience with TTree I/O, the experiments desire **configurability** similar to TTreeCache.

ROOT will address this as [ntuple] Better customization options for RClusterPool (read-ahead) · Issue #16325 · root-project/root · GitHub

3. I**ndexing capability** similar to TTree to associate RNTuple to another instance with sparse entries (not required for current workflows, do have framework work-around).

ROOT will address this with new ***`RNTupleProcessor`*** *as* [v632][hist] THnSparse::Scale iterate only over non-filled bins by guitargeek · Pull Request #15166 · root-project/root · GitHub
initial version

4. The experiments need an ability to **tune the memory** usage in RNTupleWriter

Should be addressed via page sizes (point 1)

5. CMS foresees to not to be able to use **RNTupleParallelWriter** as long as it has the restriction of having only one Writer per file (CMS presently stores several TTree objects in one file, and foresees to do the same with RNTuple)

*ROOT believes that this point is **already addressed** by the new method `FillNoCommit()`.*

- *This allows the framework to control when exactly the RNTupleParallelWriter uses the TFile and enables the framework to, e.g., lock such accesses.*
- *PR #15239 provides more details. A planned improvement of the `FillNoCommit()` API is tracked in issue #16241.*

6. RNTupleParallelWriter: for every writer's Fill() call, CMS will need to know what entry number that Fill() call corresponds to in the RNTuple.
   - It is sufficient to know that mapping periodically when the data is being written (whereas waiting until the file close time is known to cause high memory usage).

ROOT will address this as [ntuple] Better control of cluster ordering for parallel writes · Issue #16326 · root-project/root · GitHub

some ideas, but may need further discussion

7. When a user owns the memory to be filled from storage in any **other way than via std::shared_ptr**,
   - the user has to pass a dummy std::shared_ptr<void> to RNTupleReader::GetView<void>(), after which the user has to call the RNTuple::BindRawPtr() to set the actual memory address. CMS would prefer a more direct interface that would avoid the dummy std::shared_ptr<void>.
8. Having to have **two separate loops** to call RNTupleModel::AddField() and RNTupleModel::GetToken() feels suboptimal
   - AddField() requires the model to be unfrozen, and GetToken() requires the model to be frozen.
9. REntry::RFieldToken not having a default constructor is somewhat inconvenient, even if it can be worked around with std::optional for use cases such as member data.

*ROOT will improve the API along the described lines. Possibly without detailed tracking in individual issues. But has questions **to be discussed September 25th***