ROOT Developer Retreat: ROOT I/O

13 December 2020

ROOT I/O Person Power in 2021

- Philippe [50 %]
- Oksana [50 %]
- Jakob [50 %]
- Javier [80%, funded by the EP R&D programme]
- Max [3 FTM, funded by IRIS-HEP]

- David [HPC benchmarks, openlab]
- Few student months from GSoC (only 2 months / student this year!), GSoD
- Plus contributions from Vincenzo, Danilo, CERN storage team, and others

There is interest from beyond the ROOT core team in I/O.
We should try to leverage on that by suggesting proper “satellite tasks”.

ROOT I/O Areas of Work

1. **Core I/O & TTree**: maintenance, user support, bug fixes, support for critical new features
2. **RNTuple**: first exploitation and adoption, performance engineering, schema evolution
3. **Cross-cutting issues**: compression, error handling, benchmarks

Task classification

Difficulty:	starter project	→	core team	→ R&D
Urgency:	nice to have	→	important	→ essential (target <= v6.28)
Progress:	drawing board	→	well underway	→ almost done

Core Business

1. **User support and bug fixes (forum, bug tracker, etc.)**
   Difficulty: core team Urgency: **essential**

2. **Bug fixes**
   Difficulty: core team Urgency: **essential**

3. **Thread-safety and performance improvements**: including writing into TBufferFile
   Difficulty: core team Urgency: **essential** Progress: planned **~2 FTME**

Core Business

4. **TBufferFile larger than 1GB**
   Difficulty: core team Urgency: **essential** Progress: started **~2 FTME**

5. ***Schema Evolution Improvement***
   Difficulty: core team Urgency: **essential** Progress: started **~2 FTME**

6. **Advance C++ type support:** shared_ptr, optional, variant, nested std::array (partially uncovered)
   Difficulty: core team Urgency: important Progress: planned ~4.5 FTME

TTree: Integration

1. **RDataFrame Bulk I/O DataSource** (planned to be addressed more fundamentally in RDF)
   Difficulty: core team Urgency: important Progress: planned ~2 FTME

2. **Direct path TTree → Bulk I/O → Awkward arrays**: provides a “fast” connection between ROOT I/O and the Python world
   Difficulty: core team Urgency: nice to have Progress: planned ~1 FTME

Core I/O: Stretch Goals

1. ***Improve performance of TBufferFile**: remove virtual function calls*
   Difficulty: advanced *Urgency: important* *Progress: planned* *~1 FTME*

2. ***I/O of interpreted classes**: avoid having to spell out all used class template instances (somewhat blocker for ROOT7 histograms)*
   Difficulty: advanced *Urgency: important* *Progress: planned* *~1 FTME*

3. ***I/O of interpreted collections**: allow streaming of all interpreted classes*
   For experiment relying heavily on class template, including potentially ROOT v7
   Difficulty: advanced *Urgency: important* *Progress: planned* *~2 FTME*

4. ***Double32_t improvements**, customization of vector<Double32_t>, similar feature for integer*
   Difficulty: advanced *Urgency: nice to have* *Progress: planned* *~1+2 FTME*

RNTuple Core Business

1. **Optimize footer format for large data sets (files >10G)**
   Difficulty: core team Urgency: **essential** Progress: drawing board **~2 FTME**

2. **File format backwards and forward compatibility:** includes format specification
   Difficulty: core team Urgency: **essential** Progress: started **~3 FTME**

3. **Finalization of the initial I/O type system**: type casting rules, schema evolution, test of complex classes
   Difficulty: core team Urgency: important Progress: drawing board ~5 FTME

4. **Attribute API:** storage of namespace'd key-value pairs, such as "root.timestamp", "cms.uuid"
   Difficulty: starter project Urgency: important Progress: drawing board ~1.5 FTME

RNTuple Usability and Performance I/II

1. **Friends and chains:** virtual storage backends
   Difficulty: core team Urgency: **essential** Progress: almost done **~1 FTME**

2. **TTree to RNTuple converter:** disk-to-disk conversion as in hadd
   Difficulty: starter project Urgency: **essential** Progress: drawing board **~1 FTME**

3. **Fast merging, hadd support**
   Difficulty: core team Urgency: **essential** Progress: almost done **~1 FTME**

4. **RBrowser integration**
   Difficulty: starter project Urgency: important Progress: well underway ~1 FTME

RNTuple Usability and Performance II/II

5. **User-facing bulk API:** retrieve spans of entries
   Difficulty: core team Urgency: important Progress: drawing board ~1 FTME

6. **RDF optimization**: use of new RVec, align event ranges to clusters, use of bulk API
   Difficulty: core team Urgency: important Progress: drawing board ~2 FTME

7. **DAOS (object store) backend including "data mover"**
   Difficulty: core team Urgency: important Progress: well underway ~2.5 FTME

8. **Buffered writes:** cluster layout optimization, multi-threaded compression
   Difficulty: core team Urgency: important Progress: drawing board ~2.5 FTME

RNTuple First Exploitation I/II

1. **CMSSW nanoAOD generation in RNTuple format** **[Max]**
   Difficulty: starter project Urgency: **essential** Progress: drawing board ~3 FTME

2. **Comparison with HDF5 on HPC**
   Difficulty: starter project Urgency: **essential** Progress: drawing board ~3 FTME

3. **RNTupleLite:** C API for basic read support without libCore dependency **[Oksana]**
   Difficulty: core team Urgency: important Progress: well underway ~2 FTME

4. **RNTuple PODIO backend**
   Difficulty: starter project Urgency: important Progress: drawing board ~2 FTME

RNTuple First Exploitation II/II

5. **High-speed pipe into ML tools: should this be done through RDF?**
   Difficulty: core team Urgency: important Progress: drawing board ~4 FTM

6. **Data layout and handling on GPUs** (Nvidia DirectStorage, Alpaka)
   Difficulty: R&D Urgency: nice to have Progress: drawing board ~6 FTME

RNTuple Stretch Goals

1. **Fine-grained multi-threading**: concurrent reading on shared cluster buffers
   Difficulty: R&D Urgency: important Progress: drawing board ~4 FTME

2. **Direct data exchange with Apache Arrow**
   Difficulty: core team Urgency: nice to have Progress: drawing board ~3 FTME

3. **Backend for other object stores (e.g. S3)**
   Difficulty: R&D Urgency: nice to have Progress: drawing board ~4 FTME

Error Handling

1. **Validate crash recovery in RNTuple**
   Difficulty: starter project Urgency: important Progress: drawing board ~2 FTME

2. **RNTuple error injection testing**
   Difficulty: starter project Urgency: important Progress: drawing board ~2 FTME

ROOT Compression Library/Engine

1. **Review compression settings interface overhaul**
   Difficulty: advanced Urgency: important Progress: planned ~1 FTME

2. **Create ROOTZip library (based on RZip object library) to make easier compression settings and debugging**
   Difficulty: advanced Urgency: important Progress: planned ~2 FTME

ROOT Lossless Compression Algorithms

1. **Update zlib-cloudflare with zlib-ng (it has already zlib-cloudflare patches upstreamed)**
   Difficulty: core team Urgency: important Progress: planned ~2 FTME

2. **Investigate ZSTD byte-stream compression** and test it for RNTuple and TTree *(e.g. BYTE_STREAM_SPLIT encoding from Parquet, which improves a compression ratio and compression speed for certain types of floating-point data where the upper-most bytes of a values do not change much)*
   Difficulty: starter project Urgency: nice to have Progress: drawing board ~3 FTME

3. **Investigation how compression/decompression speed of LZMA could be improved (e.g. Fast-LZMA2, SSE4.2/AVX2)**
   Difficulty: core team Urgency: important Progress: planned ~0.5 FTME

4. **Investigate existing and experiment with a new compression schemas, such as “heuristic mixed compression” for RNTuple**
   Difficulty: starter project Urgency: important Progress: drawing board ~3 FTME

ROOT Lossy Algorithms (Floating Point)

1. **Investigation of lossy compression through ZFP in ROOT (it was already tested in CMSSW for NanoAODs, but never for RNTuple in ROOT)**
   Difficulty: starter project Urgency: important Progress: planned ~3 FTME

2. **Incorporate lossy compression engine** (Accelogic)
   Difficulty: core team Urgency: important Progress: planned ~ 1 FTME

Investigation of ROOT I/O Performance

1. **Implementation/review of performance metrics for better I/O benchmarking (e.g. TreePerfStats or the newer root-readspeed by Enrico)**
   Difficulty: core team Urgency: important Progress: planned ~ 5 FTME

2. **Performance continuous testing for ROOT I/O critical parts**: especially for RNTuple (e.g. rootbench)
   Difficulty: core team Urgency: important Progress: planned ~ 2 FTME

3. **Multi-client / shared storage performance behavior** in collaboration with XRootD and EOS teams
   Difficulty: core team Urgency: important Progress: planned ~ 1 FTME

Discussion

Final Remarks

- It’s time to get adoption and feedback from experiments for RNTuple

- Oksana: implement/test a new functionality for RNTuple at the first priority, after TTree
- Oksana: think and show examples how to interconnect different ROOT parts with RNTuple: example of modern ML pipeline or modern analysis pipeline and etc.
    - It could help to collect the new requirements for IO and prioritize tasks