2018 I/O POW

Ph. Canal, B. Bockelman, D. Piparo, Jakob B., J. Pivarski, Z. Zhang, O. Shadura

How can we communicate the results of this work item?

- Announcement in ROOT I/O Workshops, ROOT Planning meeting and conferences (CHEP, ACAT, ROOT User's Workshop)

- Release notes, Post on the forum, tutorials

- We ought to be (even more) proactive in also presenting these results in experiment software weeks and similar fora.

a) LZ4 becoming the default compression algorithm
Oksana, .25 FTME

Why: Faster for small size penalty
Who: mostly individual analysis that will see immediately ‘speed-up’
Now?: Start of release cycle, need new ‘tagged’ realease of v5.34, v6.10 and v6.08.

b) Parallel unzipping
Zhe, 1 FTME

Why: Allow to use spare cores. Alternative to IMT GetEntry.
Who: Individual analysis in particular those now yet use TDF or multi-thread explicitly.
Now?: Almost completed.

c) Support shared_ptr.
Danilo, 1.0 FTME

Why: clarify object ownership, essential component of v7
Who: all users including ROOT v7 developers.
Now?: Blocker for v7 and some experiment code improvement.

d) Improvement to thread-safety and performance, including making the ROOT's RecursiveRemove feature completely thread safe and performant.
Philippe, 2 FTME

Why: Necessary for correct (and speedy) functioning of ROOT multi-thread features (TDF)
Who: All users of ROOT in multi-thread environment
Now?: Blocker for wide spread adoption of ROOT multi-thread features (TDF)

e) Parallel Buffer/Tree merger: Task oriented, performance.
Guilherme, 3 FTME

Why: Improve performance of multi-thread writing of TTrees, improve coordination with TBB frameworks.
Who: Framework developers and users, TDF users, etc.
Now?: About to be adopted by CMS already seeing (new) bottleneck in some cases.

f) Bulk I/O: put in place general framework, add first application/usage
Brian, 1 FTME
Note also Bulk I/O <-> TDF in the Parallel Analysis Plan.

Why: Much faster than regular I/O
Who: TDF, Framekwork developers.
Now?: Already known to be faster and already mock-up (uproot) in the wild.

g) TTree -> Bulk I/O -> NumpyArray
Jim, 1 FTME

Why: Essential component of ‘fast’ connection between ROOT I/O and Python World.
Who: Users of python tools
Now?: Growing demand for use of python tools (and consequent ‘translation layers’)

h) Update to OptimizeBasket (one basket per).
Brian, .5 FTME

Why: Improve run-time and compression of TTree Clusters.
Who: Potentially all users of TTree.
Now?: Weaknesses of OptimizeBasket unaddressed for years.

i) Skip offset array in TBasket when it is redundant, including for TLeafElement and unsplit STL collection. Extent to as many case as possible. [Reduce file size]
Brian, .75 FTME

Why: Reduce data size of ROOT files with no data loss.
Who: All users, , in particular for CMS new data formats
Now?: Introduce ways to allow ’nicely’ forward-incompatible change in TBasket.

j) Investigate Factoring out compression-dictionary from TBasket to TBranch, using Google Zstd there is a potential of being as fast as lz4 and have lzma level compression. [Potential to gain space ***and*** time and in large set of CMS use cases]
Oksana, 2 FTME

Why: Potential to gain significant space and time.
Who: Everybody
Now?: Availability of Zstd and other new compression library/ies

k) Improve performance of TTreeReader/Proxy
Axel .5 FTME

Why: Bottleneck in TDF.
Who: All users of TDF
Now?: Makes TDF looks not as good as it could.

l) I/O of interpreted classes
Lukas, 1 FTME

Why: Avoid having to spell out all used class template instances, including internal ones.
Who: Experiment relying heavily on class template, including ROOT v7
Now?: Somewhat blocker for v7 (histograms)

TTree/TFile for v7: initial prototypes

- Why is it needed:
  - provide a thread-safe I/O
  - offer modern C++ iterator/cursor interfaces to ROOT data sets
  - offer a well-defined bulk I/O interface
  - allow for arbitrarily nested split collections
  - Perhaps: open the door to store time series data in ROOT (ALICE)
- Who can use it:
  - other ROOT code and ROOT users
- Why now:
  - I/O capabilities and interfaces are the foundation of other tasks, e.g. parallel histogram filling
  - New column-wise storage formats/libraries are rising (e.g. Parquet, Arrow), ROOT should not fall behind in terms of performance/features.
- Time: Initial prototype: 2.5 months

n) Support for std::variant
Axel, 1.5 FTME

Why: Modern version of C++ union which is actually streamable.
Who: ROOT v7 and ‘modern’ experiment frameworks.
Now?: Blocker for ROOT v7.

o) Improve performance of TBuffer[File] skipping virtual functions calls.
Philippe 1 FTME

Why: Improve run-time when reading/writing ROOT files.
Who: All users of ROOT I/O
Now?: Started separating Binary and Text StreamerInfo Actions.

p) Investigate support for collection of std::array.
Danilo, .5 FTME (then more for implementation)

Why: Complete support for std::array
Who: Data model needing collection of fixed size array
Now?: Just added support for std::array outside of collections.

q) I/O of interpreted collection. ***** stretch goal***
Unassigned, 2 FTME

Why: Allow streaming of ALL interpreted classes.
Who: Experiment relying heavily on class template, including potentially ROOT v7
Now?: Interpreted I/O partially supported (see earlier), v7.

r) Double32_t improvements, customization of vector<Double32_t>, similar feature for integer. ***** stretch goal***
Unassigned, 1 FTME (Double) 2 FTME (Integer)

Why: Potential gain in file size
Who: Users interested in trading off (unneeded) precision for space.
Now?:

s) Explore cost (or lack thereof) of byte-swap [and copy] vs memory-copy.
Philippe/Brian 1.5 FTME

Why: Figure out whether changing the on-file format will bring significant performance improvement or not (lately we are leaning toward no).
Who: Potentially all users but even more or so those leveraging Block I/O
Now?: Bulk I/O is about to be put in production and would benefit most.

t) Support for std::optional
Axel, 1.5 FTME

Why: Modern version of using a vector of 0 or 1 element.
Who: ROOT v7 and ‘modern’ experiment frameworks.
Now?: part of C++17