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

Faster for small size penalty
Mostly individual analysis that will see immediately ‘speed-up’
In the next few days: need new ‘tagged’ release of v5.34, v6.10 and v6.08.

b) Parallel unzipping, Zhe, 1 FTME

Allow to use spare cores. Alternative to IMT GetEntry.
For individual analysis in particular those now yet use TDF or multi-thread explicitly.
Completed.

c) Support shared_ptr, Danilo, 1.0 FTME

Clarify object ownership, essential component of v7
For all users including ROOT v7 developers.
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

Necessary for correct (and speedy) functioning of ROOT multi-thread features (TDF)
For all users of ROOT in multi-thread environment
Blocker for wide spread adoption of ROOT multi-thread features (TDF)

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

Improve performance of multi-thread writing of TTrees, improve coordination with TBB frameworks.
For framework developers and users, TDF users, etc.
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.

Much faster than regular I/O

For TDF, Framekwork developers.

Already known to be faster and already mock-up (uproot) in the wild.

g) TTree -> Bulk I/O -> NumpyArray

Jim, 1 FTME

Essential component of 'fast' connection between ROOT I/O and Python World.

For users of python tools

Growing demand for use of python tools (and consequent 'translation layers')

h) Update to OptimizeBasket (one basket per).

Brian, .5 FTME

Improve run-time and compression of TTree Clusters.

For potentially all users of TTree.

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

Reduce data size of ROOT files with no data loss.

For all users, , in particular for CMS new data formats

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

Potential to gain significant space and time.
For everybody
Availability of Zstd and other new compression library/ies

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

Bottleneck in TDF.
For all users of TDF
Makes TDF looks not as good as it could.

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

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

Modern version of C++ union which is actually streamable.
For ROOT v7 and 'modern' experiment frameworks.
Blocker for many ROOT v7 features

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

Improve run-time when reading/writing ROOT files.
For all users of ROOT I/O
Started separating Binary and Text StreamerInfo Actions.

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

Complete support for std::array
For data model needing collection of fixed size array
Just added support for std::array outside of collections.

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

Avoid having to spell out all used class template instances, including internal ones.
For experiment relying heavily on class template, including ROOT v7
Somewhat blocker for v7 (histograms)

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

Allow streaming of ALL interpreted classes.
For experiment relying heavily on class template, including potentially ROOT v7
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)

Potential gain in file size
For users interested in trading off (unneeded) precision for space.

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

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

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

Modern version of using a vector of 0 or 1 element.
For ROOT v7 and 'modern' experiment frameworks.
part of C++17