What's In Store For ROOT I/O

Philippe Canal

July 21, 2011

Overview

- TBaskets Management
- I/O Customization
- Multi Threads / Multi Processes
- Optimizations
- TTreeCache
- Other New Features
- Current Priorities

TBaskets Management

- Reimplementation of OptimizeBaskets (*weeks - focused*)
  - Current algorithm designed and test to minimize the number of baskets over the whole file ***without*** clustering.
  - With clustering this algorithm is no longer optimal (occupancy rate of many of the baskets is 'low')
  - Goals:
    - Minimize the number of baskets per cluster
    - Maximize basket occupancy
    - Stay within requested memory budget
    - Clarify interface of the automatic basket sizes allocation (compressed vs uncompressed size)

⚠ Has to be run/tested on a very large set of layouts.

TBaskets Management

- Explore using compression ‘windows’ (*weeks - focused*)
  - Reduce decompression cost in case of partial read by being able to decompress a single entry from a basket.
  - Reduce memory use
  - *or* increase compression factor.
- Reduce memory copy (*weeks*)
  - Could use the TTreeCache memory directly to do the uncompressing.
  - When using the WriteCache, could write directly into the cache.

I/O Customization

- Fix support for base classes renaming when used in a split TTree (*weeks*)
- Implement better dependency tracking and placement (*days*)
  - In particular add better support for pre and post rules.
- Nested Objects (*several weeks*) 👉
- Raw Reading rules (*days - focused*)
  - For direct interaction with the TBuffer

I/O Customization

- Optimize custom I/O rule usage in TStreamerInfo::ReadBuffer (*days - focused*)
- Add automatic support for reading STL<A> into a STL<B> when an A can be read into a B (*days*)
- Write Rules (*weeks - focused*)
- Just-in-time compilation of rules (*days - focused*)

Multi Threads / Processes

- Parallel Prefetching
  - Available in v5.30
  - Useful for remote reading
  - Needs more testing
- Parallel Tree Merging
  - v5.30 has new TMemFile class
  - Need to be tied with a (socket) connection and automatized (weeks)

Multi Threads

- Ability to read multiple TBranch data in parallel (*weeks*)
  - Top level branches can be uncompressed and un-streamed independently.
- Thread safety of TStreamerInfo creations
  - This is in addition to the TClass and interpreters threading issues.
  - Will be fixed by finishing the I/O engine re-engineering

Optimization

- Finish optimization of the TStreamerInfo::ReadBuffer (*weeks*)
  - Stalled at the implementation for base classes (last large feature)
    - needs to properly handle the relationship between the streamerInfos, in particular in case of reload
  - Improve STL performance by finishing to remove all virtuality use within CollectionProxy (The *virtual* interface around Collections).
- Implement the same optimizations in the object writing code (*several weeks - focused*)
- Continue optimization of TBranch::GetEntry (*days*)

Optimization

- Explore changing the on-file byte format to little endian (*days*)
  - For ROOT 6
- Improve algorithm to detect in TTree when to use MapObject or not (*days - focused*)
- Explore using memory pools for objects allocated by TTree (*weeks*)

TTreeCache

- Allow customization of the TTreeCache fill algorithm to support a wider range of use cases (*days - focused*)
  - Investigate adaptive algorithm to handle more cases (reading branches for the first time outside the learning period) (*weeks*)
- Resolve the issue of the startup time (*days - focused*)
  - During the learning phase, we currently revert to individual reads.

TTreeCache

- Find a solution to leverage the os prefetcher (*weeks*?)
  - i.e. be able to (always) go faster than the case with read ordered baskets.
- Allow more than one TTreeCache per file (automatically) (*days*)
- Update fast-merging to leverage the TTreeCache (*days*)

New Features

- Record typedef information in ROOT files (*days*)
- Upgrade SetAddress and SetBranchAddress (*days - focused*)
  - Support being passed an object (rather than a pointer)
  - Automatic detection of when SetMakeClass is needed.
- New interface to facilitate reading TTree data from compiled code (*weeks - Axel*)
  - Keeps memory ownership with the TTree (realloc!) and Typesafe

```
TTreeReader tr("T");
TTreeReaderValuePtr< MyParticle > p(tr, "p");
while (tr.GetNextEntry()) {
  printf("Particle momentum: %g\n", p->GetP());
}
```

- Extend TClonesArray interface (for faster writing) (*days*)

Bottlenecks

- Currently the main issues are:
  - Lack of concurrent writes to a file
    - Expected large increase in the user of PROOF or PROOF-like solution.
  - CPU required for compressing and streaming
  - Pure I/O latency seems mostly negligible compared to CPU used.

Current Priorities (v5/32)

- Bug Fixes / Support
- Parallel File Merge
- TClonesArray extensions (*short*)
- Continue optimization of the TStreamerInfo::ReadBuffer

Backup slides

Outstanding Deficiencies

- Problem with Cloning a TTree pointing at an 'evolved' StreamerInfo ...
- Missing support in MakeProxy for
  - Split vector of pointers
  - Array of objects.
- See also https://savannah.cern.ch/projects/savroot

Cling Based Improvements

- Reimplementation of TTreeFormula as compiled code.
- Just in time compilation of rules (in particular the ones extracted from a ROOT File).
- Investigate JIT-ing the streaming functions.