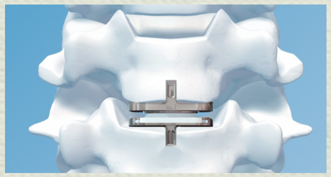
What's In Store For ROOT I/O

Philippe Canal

March 22nd, 2012



What's new in ROOT I/O

- Automatic support for more than one *TTreeCache* per file (Thanks to Peter)
  - *TTree::SetCacheSize(Long64_t)* no longer override/delete existing cache
  - Each cache is independent
    - So the worst case scenario is the rare case of two large *TTree* that are strongly intertwined in the file
  - *TFile::SetCacheRead* has been extended to take a second (optional) argument which is a pointer to the cached *TTree*
  - Existing workarounds are still supported (but no longer necessary)

```
TTree *tree1, *tree2;

input.GetObject("tree1",tree1);
tree1->SetCacheSize(300*1024);

input.GetObject("tree2",tree2);
tree2->SetCacheSize(400*2048);

tree1->GetEntry(entry1);
tree2->GetEntry(entry2);
```

What’s new in ROOT I/O

- Add support for emulated class inheriting from abstract base classes
  - Used both in simple schema evolution (concrete class being removed) and in executing schema evolution rules
  - Leverage infrastructure developed for the schema evolution rules.
- Enable streaming of *TSelectors*, prerequisite to implement processing by selector object in PROOF
- Add option 'par' in *TFile::MakeProject* to pack in a PAR file the generated code

```
TFile *f = TFile::Open("http://root.cern.ch/files/data/event_1.root")
f->MakeProject("packages/myevent.par", "*", "par");
```

What's new in ROOT I/O

- Bug fixes
  - Improved asynchronous prefetcher (case of *TChain* and *TArchiveFile*) (Thanks Elvin)
  - Update *TTree::SetEstimate*(-1) to be equivalent to *SetEstimate(GetEntries()+1)*
  - Fix *TTree::GetEntry* for the legacy fast merging technique used by CDF!
  - In *TBuffer::Expand* avoid shrinking the buffer so much that there isn't space to hold the name of the branch
    - Branch names have now grown well past 100 characters in some cases!
- *Coverity* induced cleanups

Ideas still bubbling up

- Disclaimers: those ideas might or might not come to fruition ☺
- Write only once files (***Hadoop***)
  - At the possible expense of file size, write the directory information at the end rather than the beginning
  - Lose the ability to detect truncated files ; not forward compatible
- In *TBasket* compress each entry individually (for *large* basket)
  - Also ***copy*** the compression dictionary from one basket to the next
  - Allow for fast sparse reads.   Copy allows for better compression
- Find a way to avoid storing the byte count and version number for deep hierarchy
  - Idea is to record a 'flavor' once per buffer/(IO operation) per top level class and to associate a fully unrolled sequence of actions to this flavor

TMultiFile

- A file containing a set of ***ROOT*** files, either
  - A “meta” file containing a list of URI’s
  - A concatenation of a number of ***ROOT*** files
- ***xrdcp*** and ***PROOF*** will understand the meta file format and will on copy stream out a single concatenated file
- Typical functions like *TMultiFile::Get*() will return a merged object or a *TChain*

TMultiFile

- Simple meta file format

```
#root <name> [<title>]
root://host/volume1/file1.root
root://host/volume2/file2.root
root://host/volume3/file3.root
```

- Simple concatenation

```
cat file1.root file2.root file3.root > multifile.root
```

- Advantages:
  - Allows files to be written in parallel but managed as a single file
  - Robust
- Disadvantages
  - Objects (except TTree's) need to be merged when accessed
  - Some space wasted by having N versions of the same objects

Meanwhile in Build/Core

- Hash values
  - *TString::Hash* now uses ***MurmurHash3_x64_128***
  - For pointers now uses a simple bitwise xor (Time critical for I/O!)
  - new *TString::MD5* function should be used for persistent hash values
- ***ROOT*** error handlers installation can be disabled in a *rootrc* file:

```
Root.ErrorHandlers:        1
```

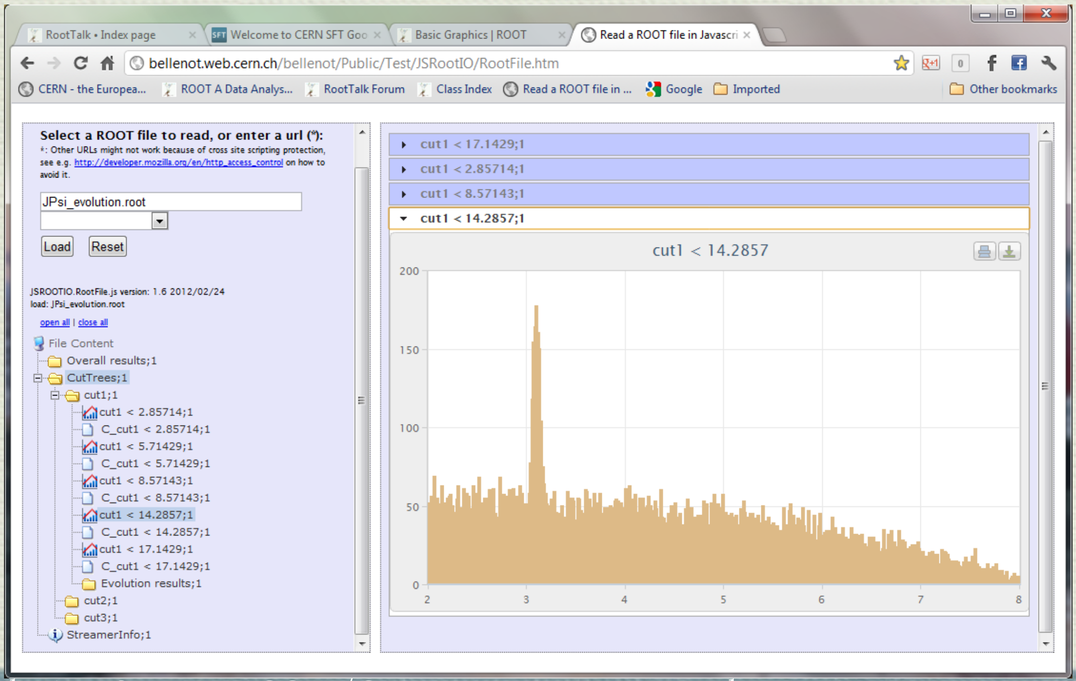
- Explicitly linking of dependent library is now the default on both ***MacOS*** and ***Linux*** (and ***Windows***)

Meanwhile in Proof

- Add support for selector-by-object processing in ***PROOF***
  - Continue to increase symmetry between local and Proof analysis
- Introduced *TProofPerfAnalysis*
  - A set of tools to analyze the performance tree
  - See http://root.cern.ch/drupal/content/analysing-performance-tree
- Provides improved benchmarking suite

Meanwhile …

- *Cling* continue to progress towards release
  - Started ***rootcling*** which will use cling and precompiled header modules for dictionary generation.
  - Good progress toward ***cling/ROOT*** integration
- Browsers and ***HTML5***
  - Provide ***ROOT*** file access (***ROOT-IO.js***) and graphics directly in the browser



Select a ROOT file to read, or enter a url (*):
*: Other URLs might not work because of cross site scripting protection, see e.g. http://developer.mozilla.org/en/http_access_control on how to avoid it.

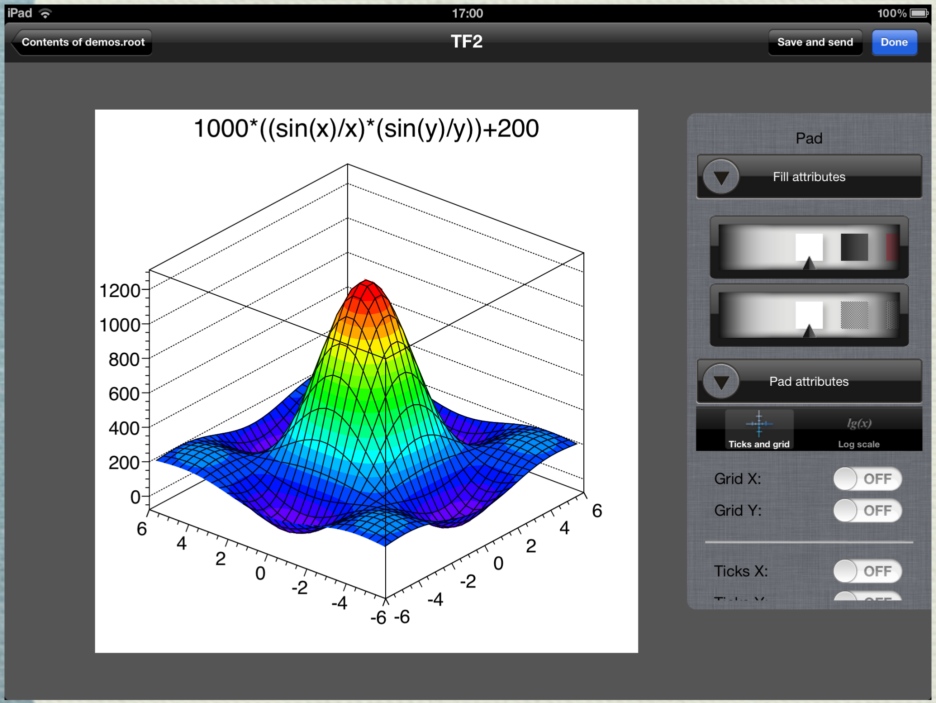
JPsi_evolution.root

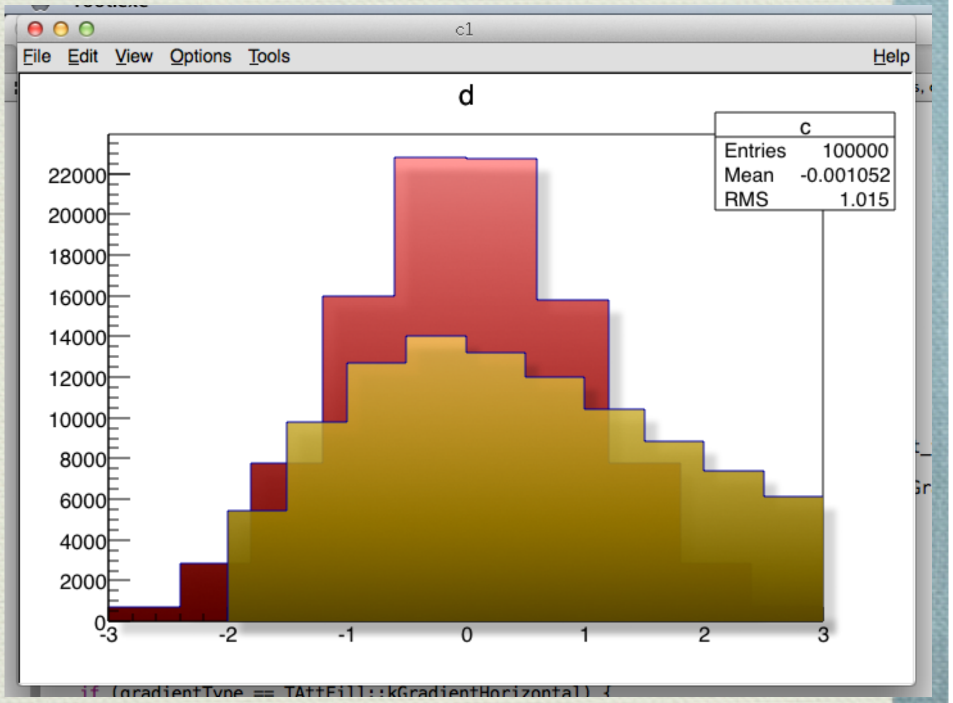
JSROOTIO.RootFile.js version: 1.6 2012/02/24
load: JPsi_evolution.root

```
File Content
├ Overall results;1
├ CutTrees;1
│ ├ cut1;1
│ │ ├ cut1 < 2.85714;1
│ │ ├ C_cut1 < 2.85714;1
│ │ ├ cut1 < 5.71429;1
│ │ ├ C_cut1 < 5.71429;1
│ │ ├ cut1 < 8.57143;1
│ │ ├ C_cut1 < 8.57143;1
│ │ ├ cut1 < 14.2857;1
│ │ ├ C_cut1 < 14.2857;1
│ │ ├ cut1 < 17.1429;1
│ │ ├ C_cut1 < 17.1429;1
│ │ └ Evolution results;1
│ ├ cut2;1
│ └ cut3;1
└ StreamerInfo;1
```

Meanwhile …

- Improvements in Graphics
  - *IOS* and native *Mac OS X*
- Continued development in Stats, Math





Release Schedule

- ***ROOT*** Release v5-34-00
  - Version v5-34-rc1 will be released May 02, 2012
  - Version v5-34-rc2 will be released May 16, 2012
  - Version v5-34-00 will be released May 30, 2012
    - New features will be back ported to v5.34/xx branch on request as long as needed
- ***ROOT*** Release v5-35-02 (***Cling*** based dictionary)
  - Mid July 2012
- ***ROOT*** Release v6-00 (***Cling*** is the only supported interpreter)
  - November 2012
- ***ROOT*** Release v6-02
  - May 2013

Current Priorities

- **Bug Fixes / Support**
- Parallel merging daemon (v5.34)
- ***Cling*** (v6.00)
- Post v6.00
  - *I/O Customization: Nested Objects (several weeks)*
  - Explore changing the on-file byte format to little endian (days)
  - Explore other small change in file format to reduce size (days)
  - Update fast-merging to leverage the *TTreeCache* (days)
  - Upgrade *SetAddress* and *SetBranchAddress* (days - focused)
  - Continue optimization of *TBranch::GetEntry* (days)

v5.34 Contributions

- Done
  - Testing of parallel prefetching – Brian
  - Allow more than one TTreeCache per file (automatically) – Peter
  - Upgrade write I/O infrastructure to support sequence of I/O actions - Chris
- In progress
  - Fast Merging sort by cluster and branches – Brian – Almost ready.
  - Reimplementation of OptimizeBaskets – Brian
    - Need to be implemented (i.e. start being tested) in next couple of weeks or be delayed to next release.
  - Test environment - Ilija/Wahid
    - Ready for input from other experiments/developers

v5.34 Contributions

- No longer being pursued
  - I/O Customization: Write Rules – Chris
- Upcoming
  - Update TTreePerfStats to support multiple cache per file – Peter
  - Resolve the issue of the TTreeCache startup time – Peter

Backup slides

Cling Miltestones

1. April 15: EOF / buffer handling (Vassil)
2. June 30: declaration queries by name (Vassil, Paul)
3. June 30: rootcling (Philippe, Axel) [1,2]
4. July 31: TCling's reflection interfaces (Paul; Vassil for the cling side) [1,2]
5. July 31: TCling's interpreter interfaces and ".class", ".typedef",... (Axel) [1]
6. July 31: TCling's TROOT interface, autoloading and avoiding double library load (Philippe) [3,4]
7. August 31: PyROOT moved to cling (Wim) [4]
8. a. September 30: remove CINT, Cintex, Reflex [6,7]
   b. September 30: validate TClass, I/O (Philippe, Paul) [5,6]
   c. September 30: validate name translation (Axel, Vassil) [4]

Parallel Merge

- New class TMemFile
  - A completely in memory version of TFile.
- New class TParallelMergingFile
  - A TMemFile that on a call to Write will
    - Upload its current content to a parallelMergerServer
    - Reset the TTree objects to facilitate the new merge.

```
TFile::Open("mergedClient.root?pmerge=localhost:1095","RECREATE");
```

- New daemon parallelMergeServer
  - Receive input from local or remote client and merger into request file (which can be local or remote).
  - Fast merge TTree.   Re-merge all histogram at regular interval.

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
  - v5.32 has client and server.
  - Needs more testing

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.
- 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());
}
```

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 un/compressing and streaming
  - Pure I/O latency seems mostly negligible compared to CPU used.

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.