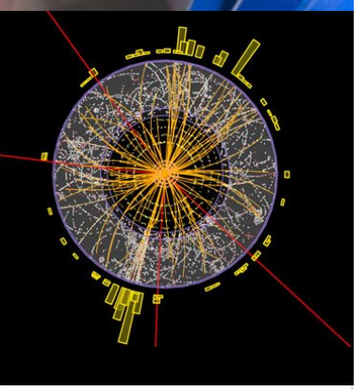
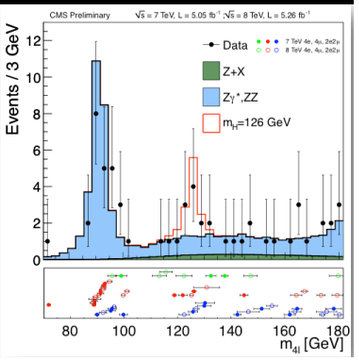
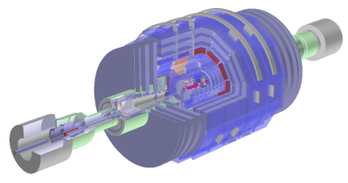


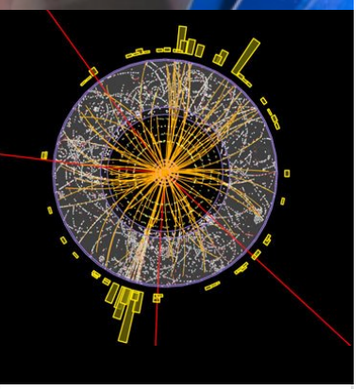


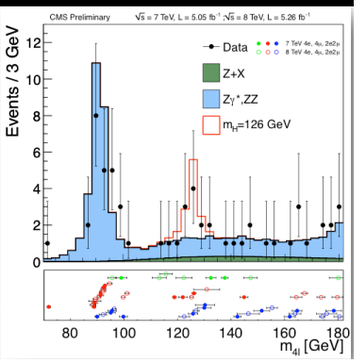


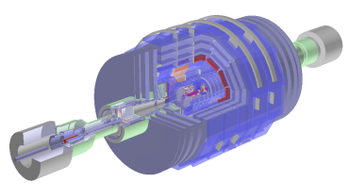
ROOT I/O Workshop, December 2013

Philippe CANAL
root.cern.ch

Schedule







09:00- 09:15	Intro and ***I/O*** Update
09:15 - 09:35	***CMS***
09:35 - 09:55	***ATLAS*** Core Software
9:55 - 10:10	Coffee/Tea break
10:10 - 10:30	***ATLAS*** Distribute Comp.
10:30 - 10:50	***LZ4*** Compression
11:00 - 11:30	Discussion

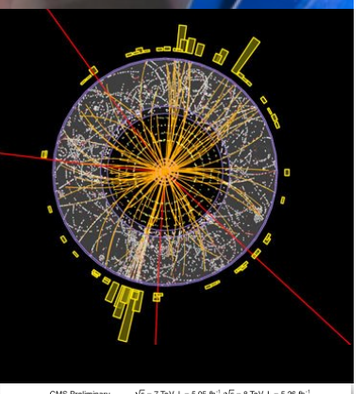
Release Schedule

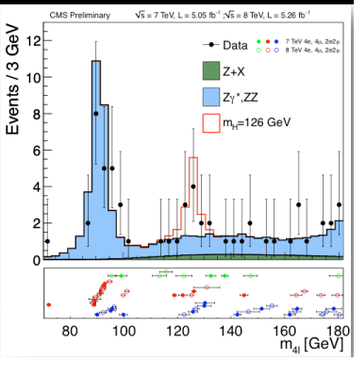
ROOT Release v6-00

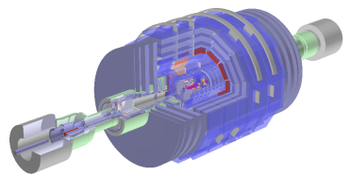
- Beta 2, January 29, 2014
- Beta 3, March 26, 2014
- Production, May 28, 2014

ROOT Release v5-34-00 patches

- v5-34/13 last week
- as needed …



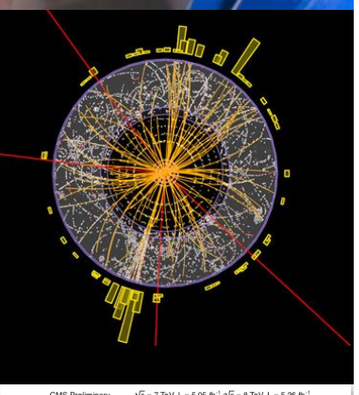


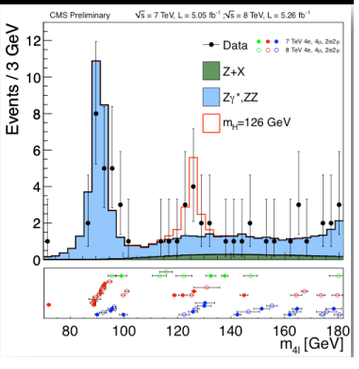


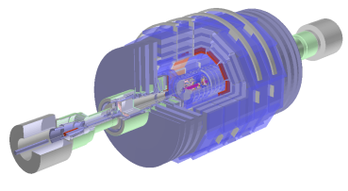
Update on ***ROOT I/O***

Philippe Canal
Fermilab

Overview







- Recent Updates
- ***I/O*** and v6
- ***TTreeCache***
- Priorities

Recent Updates

- Bug fixes and ***Coverity*** induced updates and a few thread safety and tear down order improvements
- Parallel prefetching:
  - bug fixes, performance improvements,
  - Still one possible outstanding instabilities issues (irreproducible by hand)
- ***TFileMerger***
  - Fix the handling on non-mergeable object.

Recent Updates

- New ***S3*** support class.
- Full support conversion to/from any ***STL*** collection.
- Improved performance of reading a branch with an ***std::list<int>*** by 25%.
- Repaired support for ***std::bitset***.
- Added Error message when missing dictionary for ***STL*** collection.
- Added the concept of implicit rules to (centrally) support automatic translation (eg for ***STL*** collection)
- Added support for custom collection which are not templated

TTreeReader

- Implemented and available
- Considering upgrading ***MakeClass***/***Selector*** based on it

```
#include "TFile.h"
#include "TH1F.h"
#include "TTreeReader.h"
#include "TTreeReaderValue.h"

void TreeReaderSimple() {
   TH1F *myHist = new TH1F("h1","ntuple",100,-4,4);

   TFile *myFile = TFile::Open("hsimple.root");
   TTreeReader myReader("ntuple", myFile);

   TTreeReaderValue<Float_t> myPx(myReader, "px");
   TTreeReaderValue<Float_t> myPy(myReader, "py");

   while (myReader.Next()) {
      myHist->Fill(*myPx + *myPy);
   }

   myHist->Draw();
}
```

ROOT I/O and v6

- A few things left open:

  - Update to ***CheckSum***
  - Type with template arguments that are enums
    - For example ***std::shared_ptr***
  - I/O customization *renaming rules* issues
    - Necessary to provide full backward compatibility
    - See JIRA: 5035,3211,3670,3708,5264
  - Support for ***I/O*** for private classes
  - Full Backward and Forward compatibility testing

I/O CheckSum

- ***CheckSum*** are being changed
  - Will switch from using typedef to using normalized name in ***ROOT 6***.
  - Eliminate false match ; leverage ***CompareContent*** to avoid false mismatch.
    - Current ***CheckSum*** does not detect if a typedef is used and changed from float to int for example.
  - Typedef no longer usable to enforce a platform independent checksum value.
- Need to decide whether this is the time to change the policy on the use of (or lack thereof) ***std::*** in normalized names.
- Thinking of integrating support (i.e. opaque typedefing) standard typedef ***int32_t***, ***int64_t***

Normalized Name

- Fully qualify name
  - Except for not mentioning std::
- All typedef removed except for
  - ***std::string***
  - ***Double32_t***, ***Float16_t***, ***Long64_t*** (later ***int32_t***, etc. ?)
    - typedef defined in ***std*** and points to a compiler implementation details (i.e. defined in ***__gnu_cxx*** and name starting with_)
- Replace ***basic_string<char>*** with ***string***
- Default template parameter expanded except for
  - ***STL*** container
  - ***shared_ptr*** (and later all std classes) *[Will be done next week]*
- "New" issues: template parameter that are ***enum*** constant.

Additional Selection Mechanism

Replacing ***Reflex::selection*** with:

```
// user header
template <class T, class U = int> class C {
private:
  C<T, float>* fX; // example for a "dependent" dictionary
};

// selection header, to be exposed to genreflex
namespace ROOT {
  namespace Selection {

    template <class T, class U = int> class C:
      public HideLastDefaultTemplateArguments<1> {
        Dict<kSelected + kTransient> fX;
    };
  }
}
```

TTreeCache?

- Added ***TTreeCache*::*LearnPrefill*** (not default ….)
- Still need to:
  - evaluate/install the new ***OptimizeBasket*** proposals
  - Start using it in ***TTreeCloner***.
  - Allow alternative algorithm
  - Tests, tests and tests
  - Switch on by default

TTreeCache

- New Plan!

- Add missing global enable/disable API
  - Contribution welcome
- Turn on by default
- Install the new ***OptimizeBasket*** proposals
- Tests.

- Parallel prefetching
  - Also need to be added to the global enable/disable API
  - Needs to be further tested.

Effort

- My effort spread over ***ROOT I/O***, ***Cling*** and ***Geant/V***P
  - Split 50/50 between ***ROOT*** (But has been focused on ***Cling***) and Geant
- Extra effort required to make any real progress
  - Effort from ***ATLAS***
  - ***ROOT*** Team effort (Danilo and I) should increase after v6 release
- See June presentation for plan (for now on hold)
- Summer Students and other external contribution
  - ***TTreeReader*** delivered
  - Runtime generation of ***CollectionProxy*** Started

Priorities Recapitulations

- Fix blocking issues / User Support
- Long outstanding issues
  - Yes ☺ I mean ***TTreeCache*** and ***OptimizeBasket***.
- Multi-threading, Multi-processing
  - Requires v6 (hence push for cling).
- File Format upgrades
  - Cost of repeated [deep] hierarchies
  - Write I/O customization Rules
- Performance Improvements
  - Including vectorization of I/O (***TTree::Draw***)
- Interface Simplification
  - ***SetBranchAddress***, ***TTree::Draw***,

- Backup Slides

What's in a name

- Implemented normalization routines that
  - Adds full qualification
  - Adds default template parameter except for ***STL*** containers
  - Keeps opaque typedefs
- Extra care to preserve user typed spelling and be as close as possible to the "***ROOT I/O*** name"
- ***However*** some names ***must*** change
  - ***Outer::Tplt<Inner>*** -> ***Outer::Tplt<Outer::Inner>***
  - Adding missing default template arguments
- Risk/Consequences alleviated by
  - Renaming I/O customization rules
  - Automatic matching of different spelling
  - Added flexibility in checksum matching cross-checks

Priorities Recapitulations – Nov Rel.

- Fix blocking issues / User Support

***Red items** only possible with extra effort.* !

- Required for ROOT 6 beta release
  - Renaming rules - 2w – ***July*** (5035,3211,3670,3708,5264)
  - Genreflex – ***August*** (see cling)
- Multi Processing
  - *First new revision on histogram parallel merge - 3w –* ***September*** (5071)
  - *Parallel merge daemon* – 2w – ***October*** (5070)
- File Format upgrades
  - *Write only once files (Hadoop)* – 1w - ***September*** (5075)
  - *Switch from big endian to little endian* – 1w - ***October*** (5073)

Priorities Recapitulations – Nov Rel.

- Performance
  - TTreeCache and TTreeCloner – 1w – ***August*** (5078)
  - *Testing plan for OptimizeBasket,TTreeCache* – 2w- ***September*** (5080)
- New Features
  - TTreeFormula and long long ***[Atlas]*** (5084, 5085)
  - TTreePerfStat and multiple TTree ***[Atlas]*** (5079)
- Nice to have
  - TTreeReader ***[External Contribution]*** (5165)
  - Runtime generation of CollectionProxy ***[Summer student]*** (5164)

Priorities Recapitulations – May Rel.

- Fix blocking issues / User Support
- More documentations and fix more outstanding issues.
  - See detailed list …
- Multi-processing
  - Refine parallel merging based on user experience
  - Start upgrading to support multi-threading/tasking
- File Format upgrades
  - Cost of repeated [deep] hierarchies
  - Write I/O customization Rules
- Performance Improvements
  - OptimizeBasket
- Interface Simplification
  - SetBranchAddress, TTree::Draw, etc.

TTreeReader

```
class h1analysisTreeReader : public TSelector {
public:
  TTreeReader                    myTreeReader;//!
  TTreeReaderValue<Float_t>      fPtds_d; //!
  TTreeReaderValue<Float_t>      fEtads_d; //!
  TTreeReaderValue<Float_t>      fDm_d; //!
  TTreeReaderValue<Int_t>        fIk; //!
  ...
```

```
// entry is the entry number in the current Tree
// Selection function to select D* and D0.
myTreeReader.SetLocalEntry(entry);
if (!useList) {
  // Return as soon as a bad entry is detected
  if (TMath::Abs(*fMd0_d-1.8646) >= 0.04) return kFALSE;
  if (*fPtds_d <= 2.5) return kFALSE;
  (*fIk)--; //original fIk used f77 convention
  ...
  if (fNlhpi.At(*fIpi) <= 0.1)  return kFALSE;
  (*fIpis)--; if (fNlhpi.At(*fIpis) <= 0.1) return kFALSE;
  if (*fNjets < 1)              return kFALSE;
}
...
//fill some histograms
hdmd->Fill(*fDm_d);
h2->Fill(*fDm_d,*fRpd0_t/0.029979*1.8646/ *fPtd0_d);
```