Recent Developments in the ROOT I/O and TTrees

CANAL, Philippe (FERMILAB)
BRUN, Rene (CERN)
FRANK, Markus (CERN)
RADEMAKERS, Fons (CERN)
RUSSO, Paul (FERMILAB)

ROOT I/O History

- Version 2.25 and older
  - Only hand coded and generated streamer function, Schema evolution done by hand
  - I/O requires : ClassDef, ClassImp and CINT Dictionary

- Version 2.26
  - **Automatic schema evolution**
  - **Use TStreamerInfo (with info from dictionary) to drive a general I/O routine.**

- Version 3.03/05
  - **Lift need for** ClassDef and ClassImp for classes not inheriting from TObject
  - **Any non TObject class** can be saved inside a TTree or as part of a TObject-class
- Version 4.00/00
  - Automatic versioning of 'Foreign' classes
- Version 4.00/08
  - Non TObject classes can be saved directly in TDirectory

- Version 4.01/02
  - Large TTrees, TRef autoload
- Version 4.04/02
  - TTree interface improvements, Double32 enhancements

- Version 5.08/00
  - Fast TTree merging, Indexing of TChains, **Complete STL support.**

Outline

- General I/O
  - STL Collections
  - Data compression using reduced precision
  - Alternatives to default constructors
  - Other I/O improvements
- Trees
  - New Features
  - Fast Merging
  - Indexing of TChains
  - TTree Interface enhancements
  - TRef and pool::Reference
  - Browsing

I/O Improvements – Outline

- STL collections
- Data compression using reduced precision
- Alternatives to default constructors
- Other I/O improvements

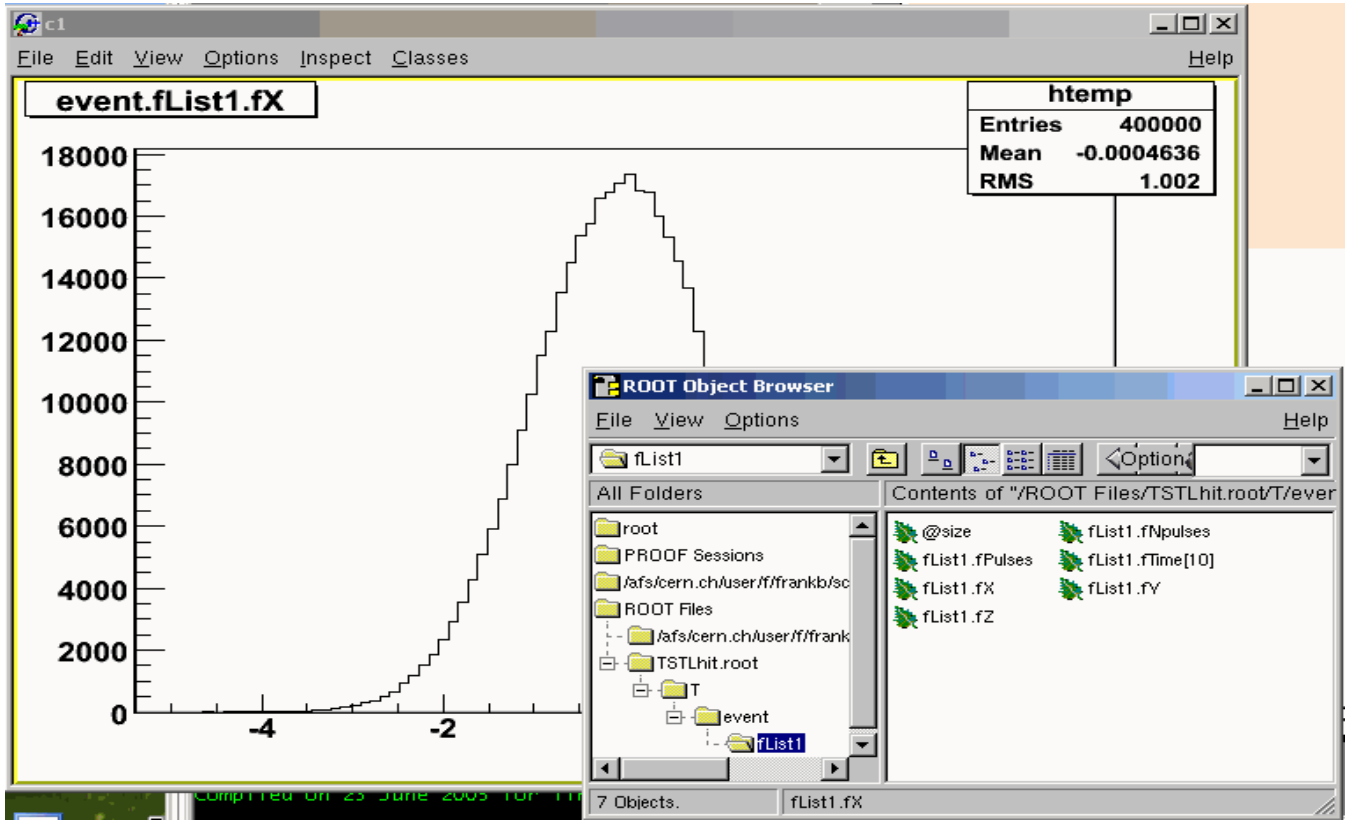
ROOT I/O: STL Collections

- Support for all STL containers
  - vector, list, set, multiset, deque, map, multimap, queue and stack
  - Also the non portable:
    - hash_map, hash_multimap, hash_set, hash_multiset
- Support for schema evolution between container

```
TClonesArray ←→ vector<T>
TClonesArray ←→ list<T>
list<T>      ←→ vector<T>
map<T,K>     ←→ list<pair<T,K>>
```

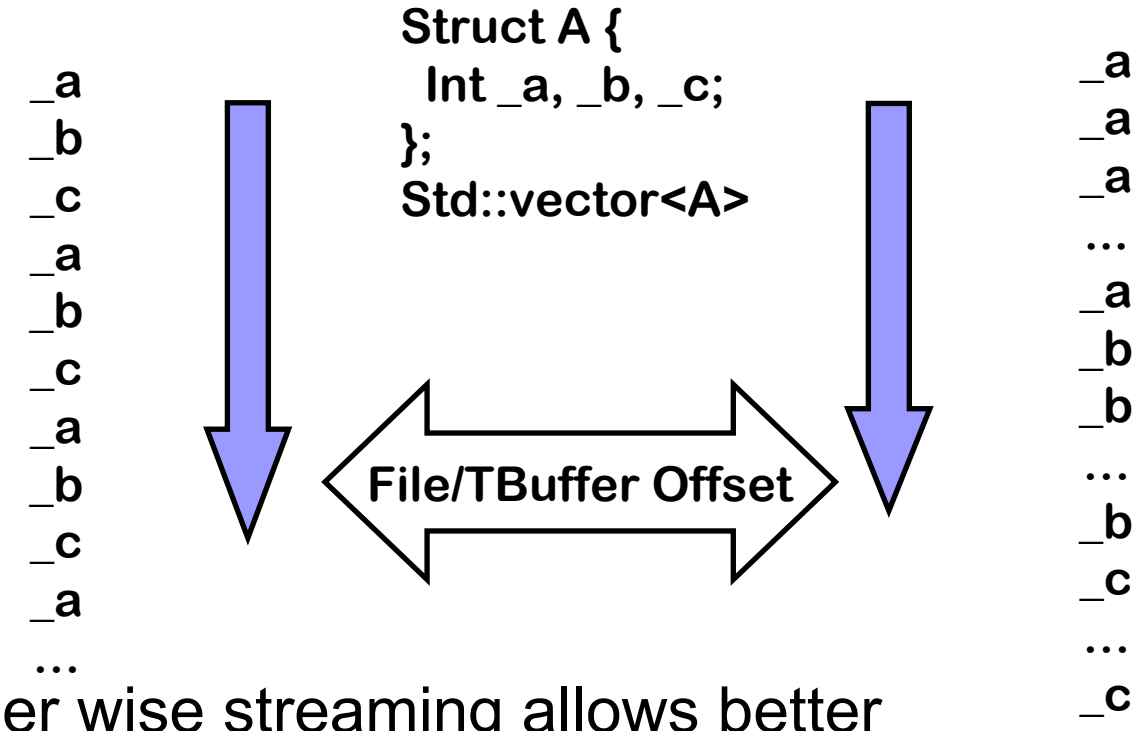
- STL collections can be saved in split mode
  - Objects (not pointers) are splitable
  - Quick pre-selections on trees
  - Interactivity: Trees can be browsed
  - Save space (see $ROOTSYS/test/bench):
    std::vector<THit>: compression 5.38
    std::vector<THit*>: compression 3.37
- Can be extended to ***any*** iterable collections via an implementation of the interface TVirtualCollection

ROOT I/O: STL Collections (2)



ROOT I/O: STL Collections (3)

- **Streaming: Object- & member wise**



- Member wise streaming allows better compression (zip works more efficient)
- **Bool_t TStreamerInfo::SetStreamMemberWise(Bool_t enable)**

Float, double and space…

- Math operations very often require double precision, but on saving single precision is sufficient…
- Data type: **Double32_t**

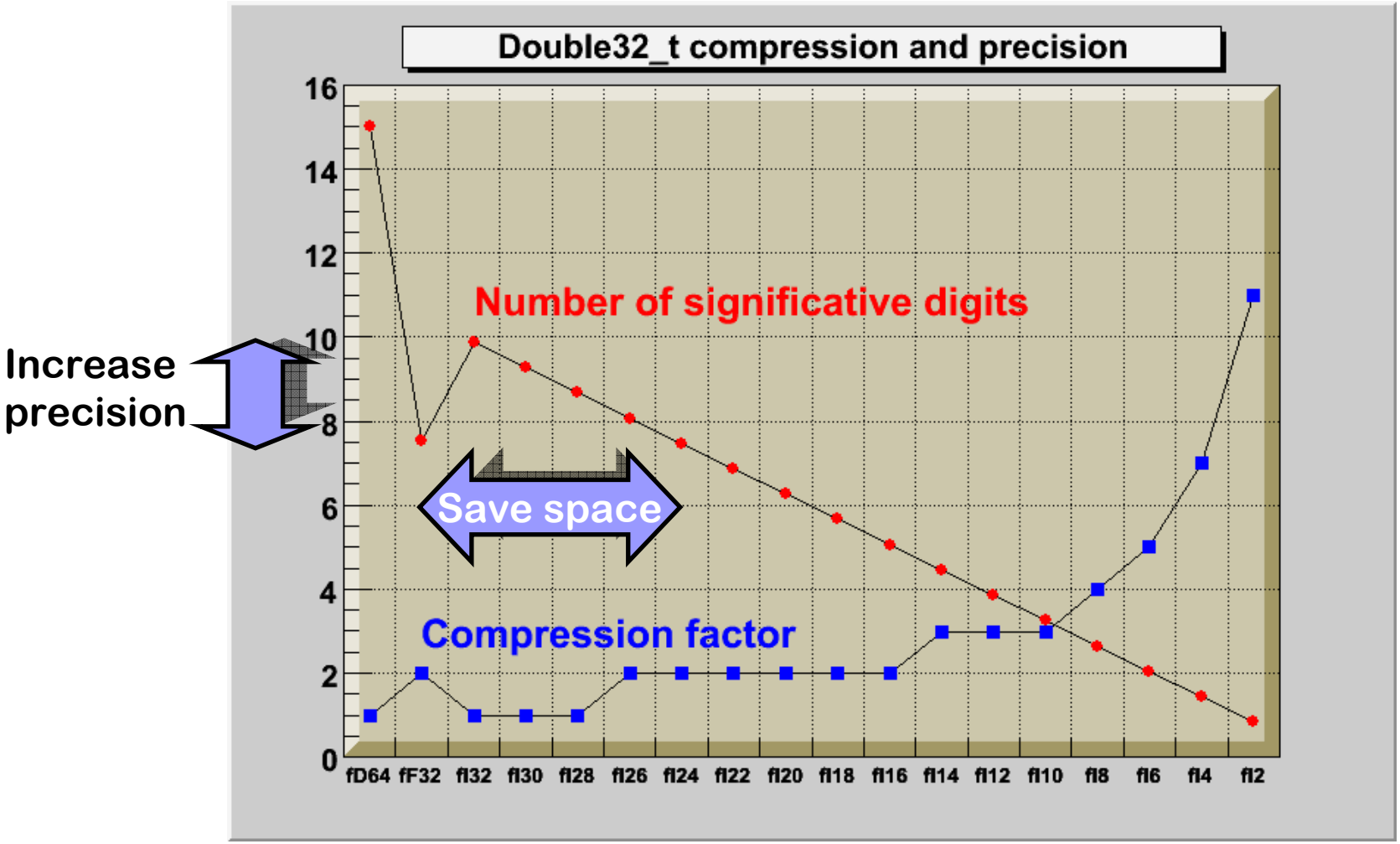
  In memory: double
  On disk: float or integer

- Usage (see tutorials/double32.C):

```
Double32_t m_data; //[min,max<,nbits>]
```

  - No nbits,min,max
    - saved as float
  - min, max
    - saved as int 32 bits precision explicit values or expressions of values known to Cint (e.g. “pi”)
  - nbits present
    - saved as int with nbit precision higher precision than float for same persistent space

Float, double and space… (2)



Default Constructors

- ROOT requires a "*default*" constructor for reading
- Not all classes can provide a constructor with no parameters.
- Alternative: I/O constructor customization

```
#pragma link C++ class MyClass;
#pragma link C++ ioctortype UserClass1;
#pragma link C++ ioctortype UserClass2;
```

  - Constructor search order:

```
MyClass(UserClass1*);
MyClass(UserClass2*);
MyClass(TRootIOCtor*);
MyClass(); // Or constructor with all args defaulted.
```

Other I/O improvements

- Thread Safety
  - Reduce reliance on *gFile/gDirectory* in internal code
  - Improve thread safety of internal code
- Variable size array of 'Foreign' Object:

```
Obj *fArr; //[n]
```

- New Class TFileMerger
  - Copying and/or Merging two or more files using the many TFile plugins.

```
TFileMerger m;
m->Cp("srcUrl", "destUrl");
```

```
m->AddFile("url1");
m->AddFile("url2");
m->Merge();
```

TTree extensions - Outline

- New Features
- Fast Merging
- Indexing of TChains
- TTree Interface enhancements
- TRef and pool::Reference
- Browsing

New Features

- Circular TTree
  - Memory TTree buffers wrap after specified number of entries

```
gROOT->cd(); //make sure that the Tree is memory resident
TTree *T = new TTree("T","test circular buffers");
. . .
T->SetCircular(20000);
for (i = 0; i < 65000; i++) { . . . }
```

- Importing ASCII data
  - Long64_t TTree::ReadFile(filename,description)
  - 'description' gives column layout following 'leaflist' format

```
TTree *T = new TTree("ntuple","data from ascii file");
Long64_t nlines = T->ReadFile("basic.dat","x:y:z");
```

Fast Merge of TTrees.

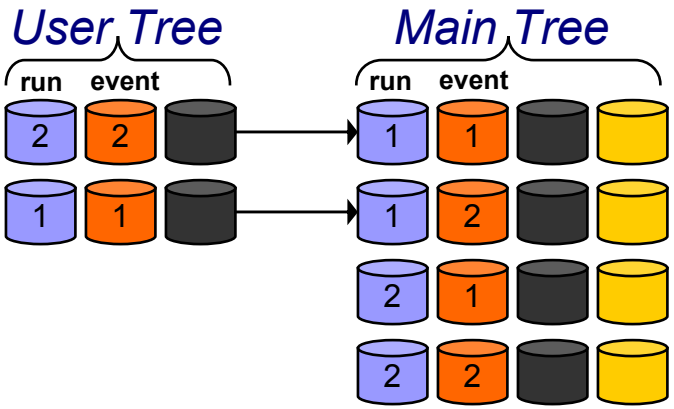
- New option, "fast“ for CloneTree and Merge.
  - No unzipping, no un-streaming.
  - Direct copy of the raw bytes from one file to the other.
  - Much higher performance.
  - Only available if the complete TTree is being copied.
  - Can also sort the baskets by branch.

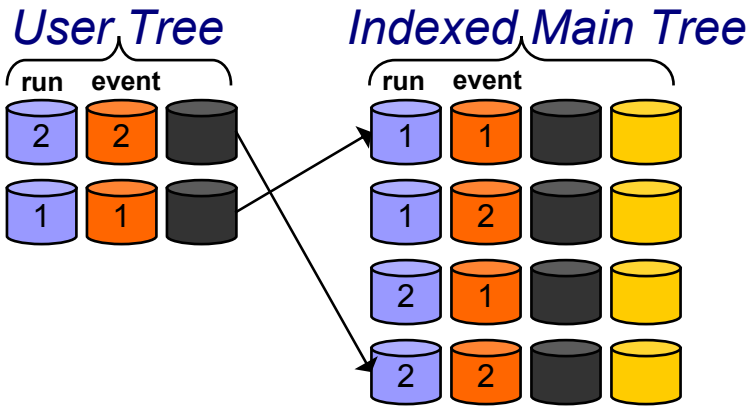
```
myChain->CloneTree(-1,”fast”);
myChain->Merge(filename,”fast”);
```

TTree Indices

- Use to connect friend TTrees.
- Extended for TChains
  - Re-use its TTrees' indexes
  - Requires the TTrees to be sorted

```
// Create index using Run and Event numbers
tree.BuildIndex("Run","Event");
// Read entry for Run=1234 and Event=56789
tree.GetEntryWithIndex(1234,56789);
```





TTree Interface

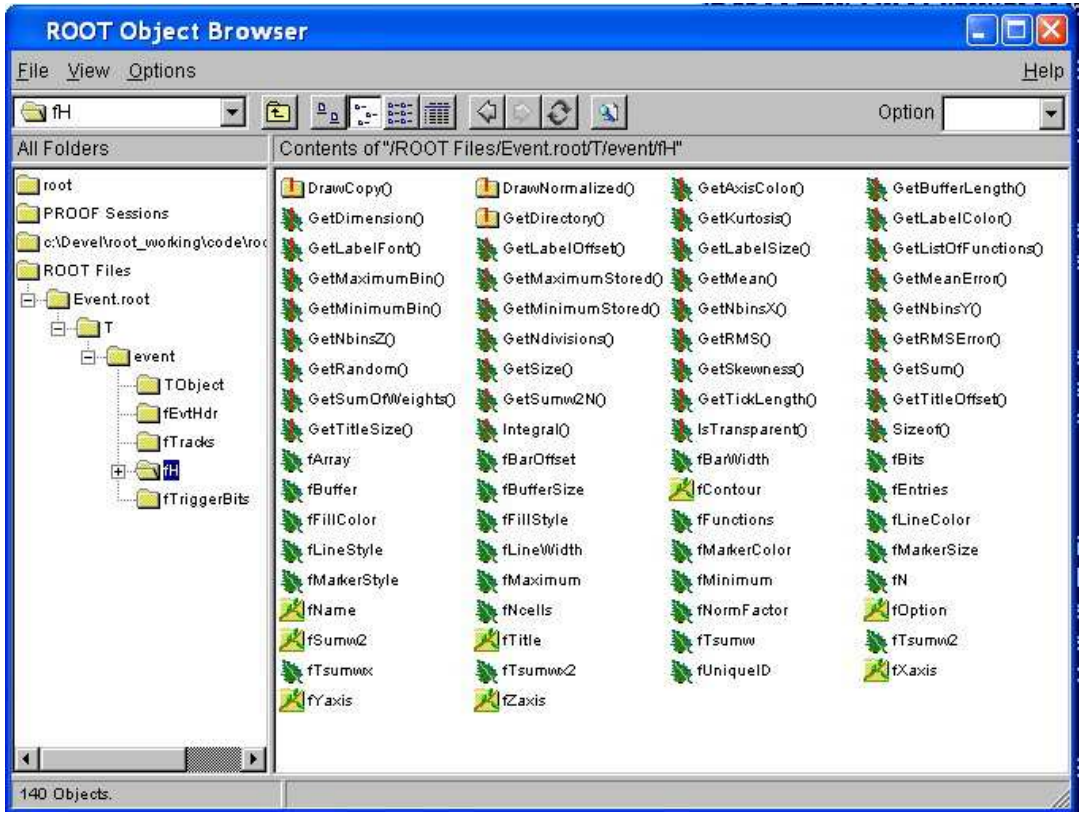
- TTree::SetBranchAddress(object)
  - Speed improvements by ~ factor 20
  - ☺ Purists no longer need to reuse objects
    Objects can quickly bound for each Fill() call
- New overloaded call to TTree::Branch
  - Better type safety
  - Saves additional argument with the class name
  - ☺ No more typos of class names for templated classes

```
template <class T> TBranch *Branch(name, T **obj_address,…);

example:          MyObj* pObj = …;
                  myTree->Branch("Branch",&ptr);
```

Browsing extension

- Can now Browse:
  - Split objects
  - Unsplit objects
  - Collections
- And can now see
  - Simple member functions
  - Transient members
  - Persistent members



Ongoing: Object Reference Support

- TBranch* TTree::BranchRef()
  - Creation of optional branch containing all information to find the branches of referenced objects.
  - Enabling this branch at write time saves the additional info
- ROOT and POOL support references to objects
  - ROOT: TRef
  - POOL: pool::Reference
- Need for automatic, implementation independent reference follow mechanism
  - TTree::Draw will automatically follow TRefs

Other Improvements

- Consolidations, consolidations
- Improved thread safety
- Improve **ACLiC** dependency checking
- Extended TBits interface
- Enhanced TFormula's run-time performance (by Marian Ivanov)

Upcoming Features

- References
  - Will implement a TVirtualRefProxy providing a generic interface for reference objects (including *GetObject, GetObjectType*). This will be used by TTree::Draw to be able to dereference TRefs and pool::ref
- MakeProxy
  - Add support for STL containers
  - Add support for CINT-interpretation
- TTree
  - Indexing using bitmap algorithm (TBitMapIndex) from LBL (See John Wu's talk)
  - TVirtualCut
  - TTree::Draw performance

Posters

- **92 - ROOT 2D graphics visualisation techniques**
  - Poster - Monday 13 February 2006 11:00
  - **Presenter: BRUN, Rene (CERN)**
- **91 - ROOT 3D graphics overview and examples**
  - Poster - Monday 13 February 2006 11:00
  - **Presenter: BRUN, Rene (CERN)**
- **189 - Recent User Interface Developments in ROOT**
  - Poster - Monday 13 February 2006 11:00
  - **Presenter: Mr. RADEMAKERS, Fons (CERN)**
- **186 - ROOT/CINT/Reflex integration**
  - Poster - Monday 13 February 2006 11:00
  - **Presenter: Dr. ROISER, Stefan (CERN)**
- **228 - The structure of the new ROOT Mathematical Software Libraries**
  - Poster - Wednesday 15 February 2006 09:00
  - **Presenter: Dr. MONETA, Lorenzo (CERN)**
- **249 - XrdSec - A high-level C++ interface for security services in client-server applications**
  - Poster - Wednesday 15 February 2006 09:00
  - **Presenter: GANIS, Gerardo (CERN)**
- **408 - xrootd Server Clustering**
  - Poster - Wednesday 15 February 2006 09:00
  - **Presenter: HANUSHEVSKY, Andrew (Stanford Linear Accelerator Center)**

Presentations

- **446 - ROOT in the era of multi-core CPUs**
  - Plenary - Wednesday 15 February 2006 12:00
  - **Presenter: BRUN, Rene (CERN)**
- **98 - PROOF - The Parallel ROOT Facility**
  - Distributed Data Analysis - Monday 13 February 2006 15:00
  - **Presenter: GANIS, Gerardo (CERN)**
- **187 - ROOT GUI, General Status**
  - Software Tools and Information Systems - Monday 13 February 2006 16:40
  - **Presenter: RADEMAKERS, Fons (CERN)**
- **188 - From Task Analysis to the Application Design**
  - Software Tools and Information Systems - Monday 13 February 2006 17:00
  - **Presenter: Mr. RADEMAKERS, Fons (CERN)**
- **129 - ROOT I/O for SQL databases**
  - Software Components and Libraries - Monday 13 February 2006 17:40
  - **Presenter: Dr. LINEV, Sergey (GSI DARMSTADT)**
- **185 - Reflex, reflection for C++**
  - Software Components and Libraries - Tuesday 14 February 2006 14:00
  - **Presenter: Dr. ROISER, Stefan (CERN)**
- **227 - New Developments of ROOT Mathematical Software Libraries**
  - Software Components and Libraries - Tuesday 14 February 2006 16:00
  - **Presenter: Dr. MONETA, Lorenzo (CERN)**
- **383 - New features in ROOT geometry modeller for representing non-ideal geometries**
  - Software Components and Libraries - Wednesday 15 February 2006 14:00
  - **Presenter: BRUN, Rene (CERN)**
- **93 – ROOT 3D graphics**
  - Software Components and Libraries - Wednesday 15 February 2006 16:00
  - **Presenter: BRUN, Rene (CERN)**
- **407 - Performance and Scalbility of xrootd**
  - Distributed Data Analysis - Wednesday 15 February 2006 17:00
  - **Presenter: HANUSHEVSKY, Andrew (Stanford Linear Accelerator Center)**

Conclusions

- Even after 10 years of ROOT:
- The I/O area is still improving
- There were quite a number of developments
  - Full STL support
  - Data compression
  - Tree I/O from ASCII, tree indices
- There will be certainly some developments in the I/O area
- The “classical” stuff however is intended to be kept stable
- Main focus:
  Consolidation (Thread Safety)
  Generic Object Reference support
  - User defined reference objects supported by
  - User defined reference handlers (proxies)