

ROOT 6 Features

Fons Rademakers

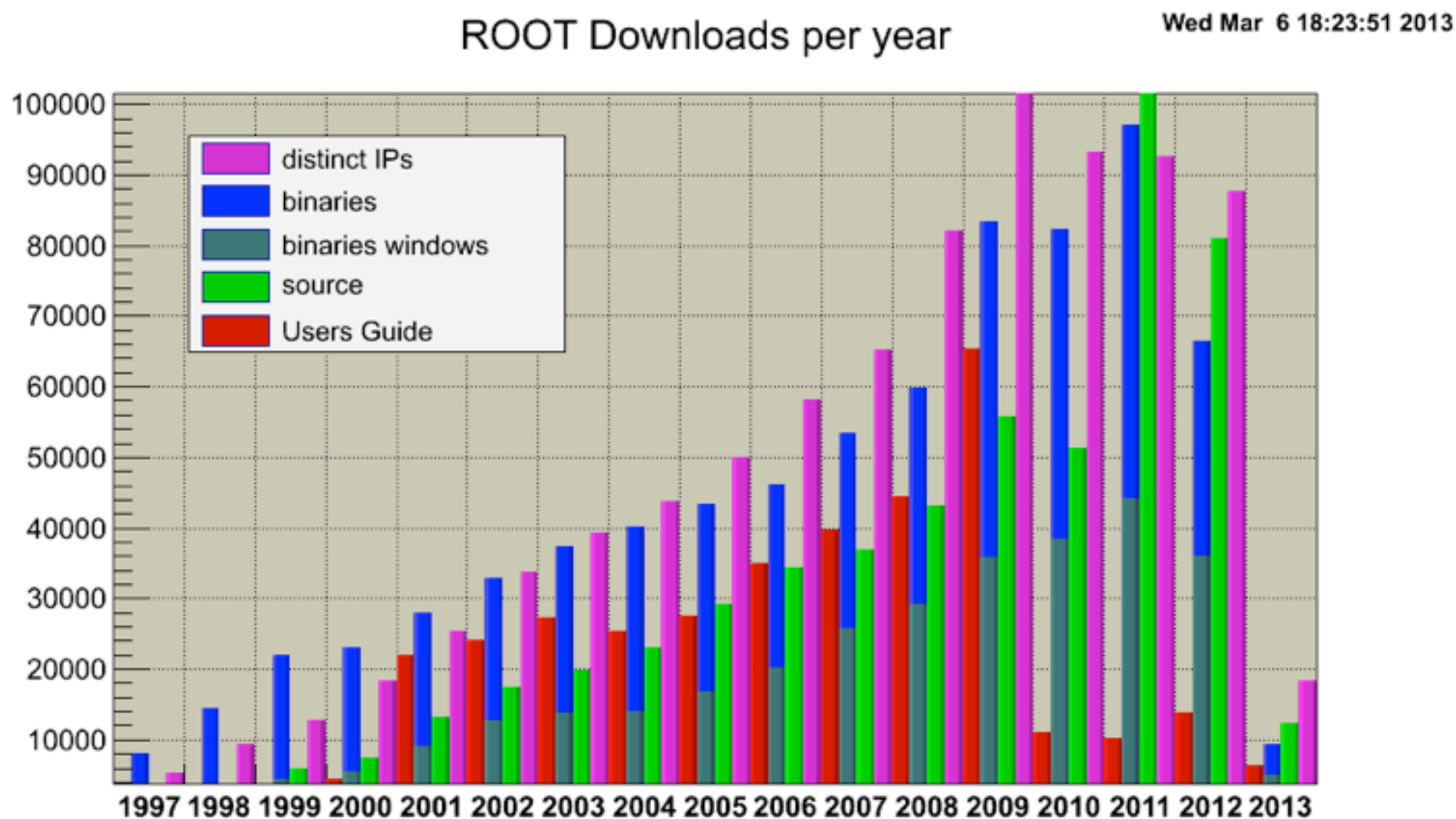
INFN Workshop
Genova, 27-31 May 2013

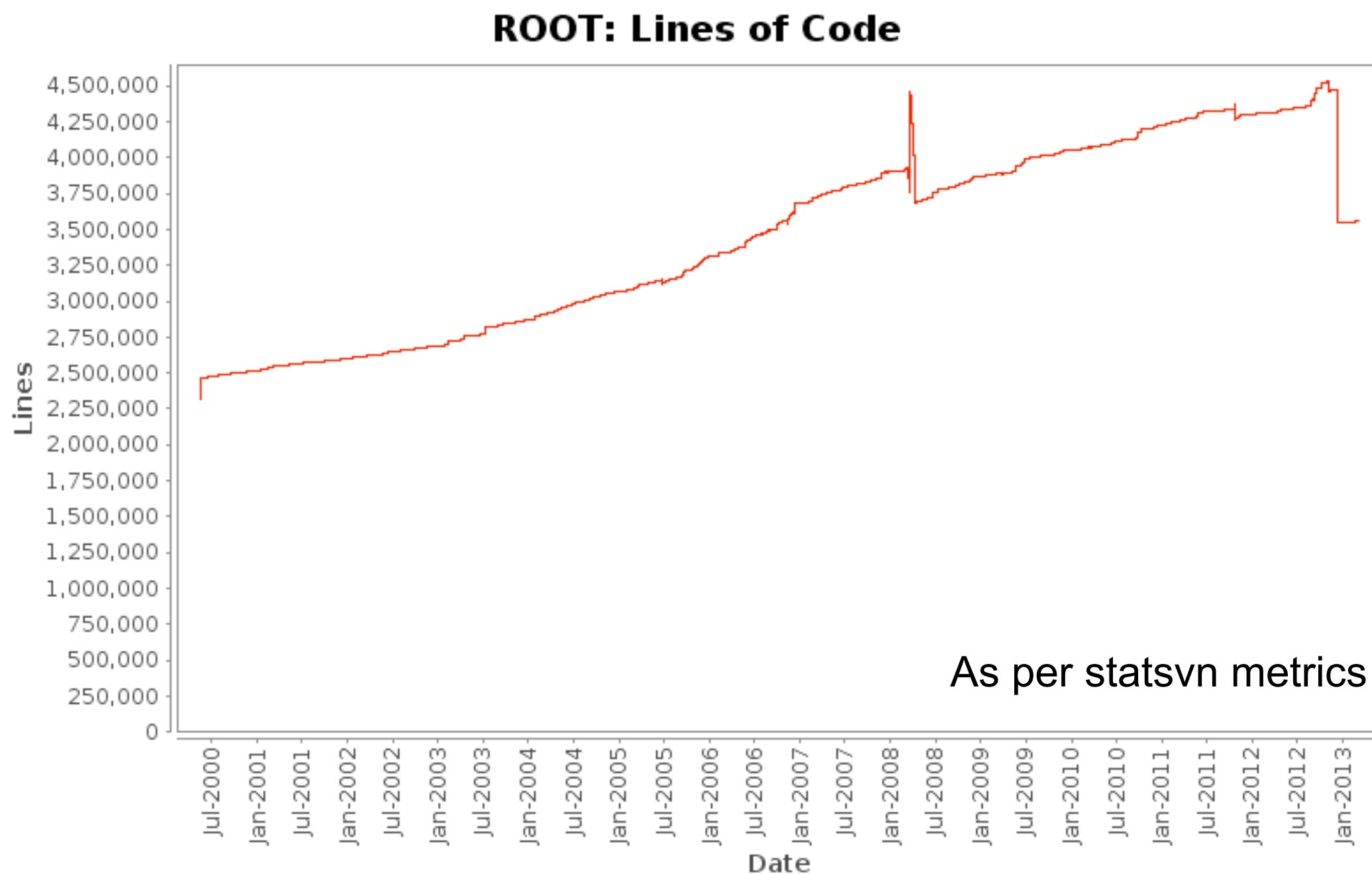
The Current ROOT Team





- Ever increasing number of users
 - 6800 forum members, 68750 posts, 1300 on mailing list
 - Used by basically all HEP experiments and beyond

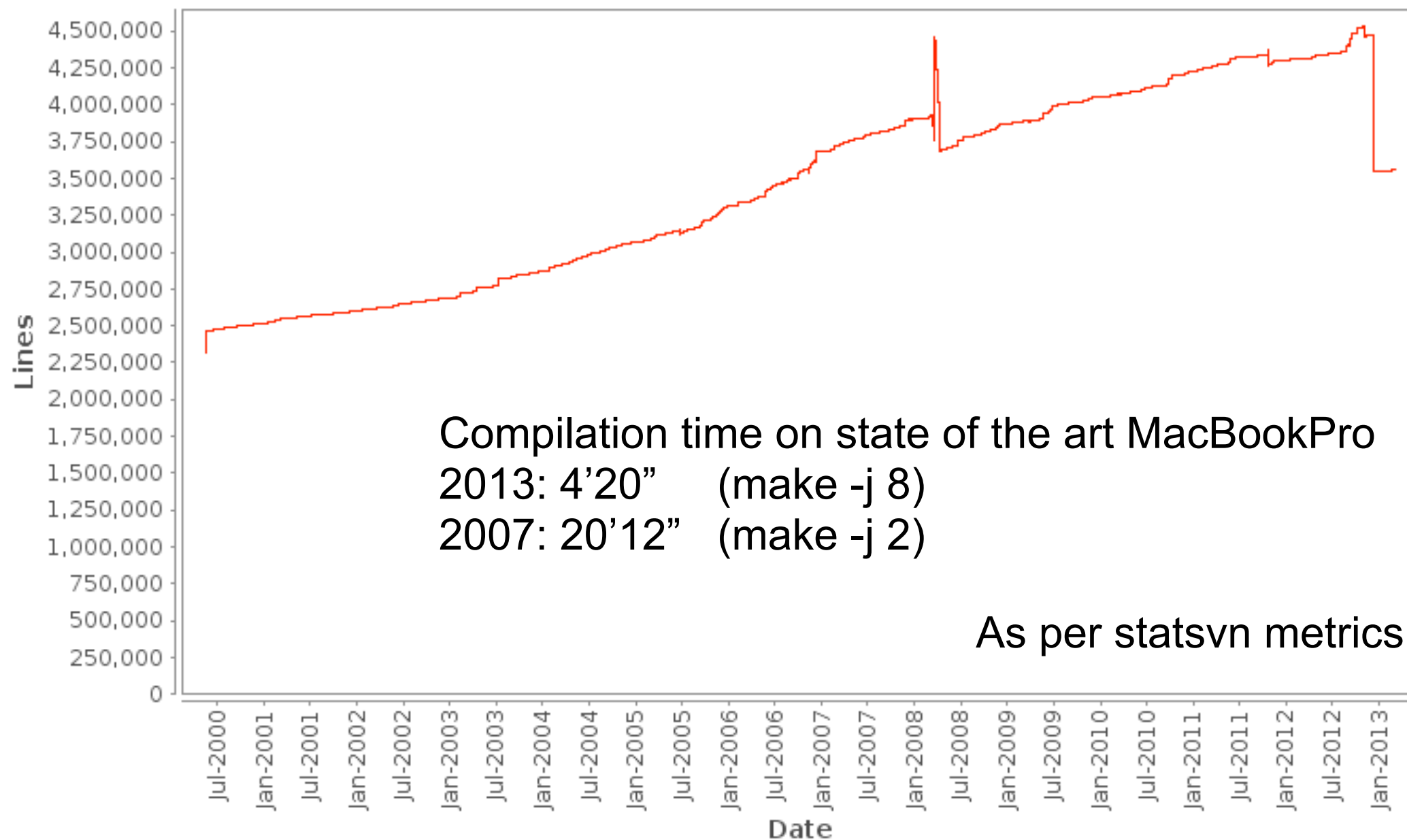




Real MLOCs: 2.75, was 3 with CINT



ROOT: Lines of Code



Real MLOCs: 2.75, was 3 with CINT



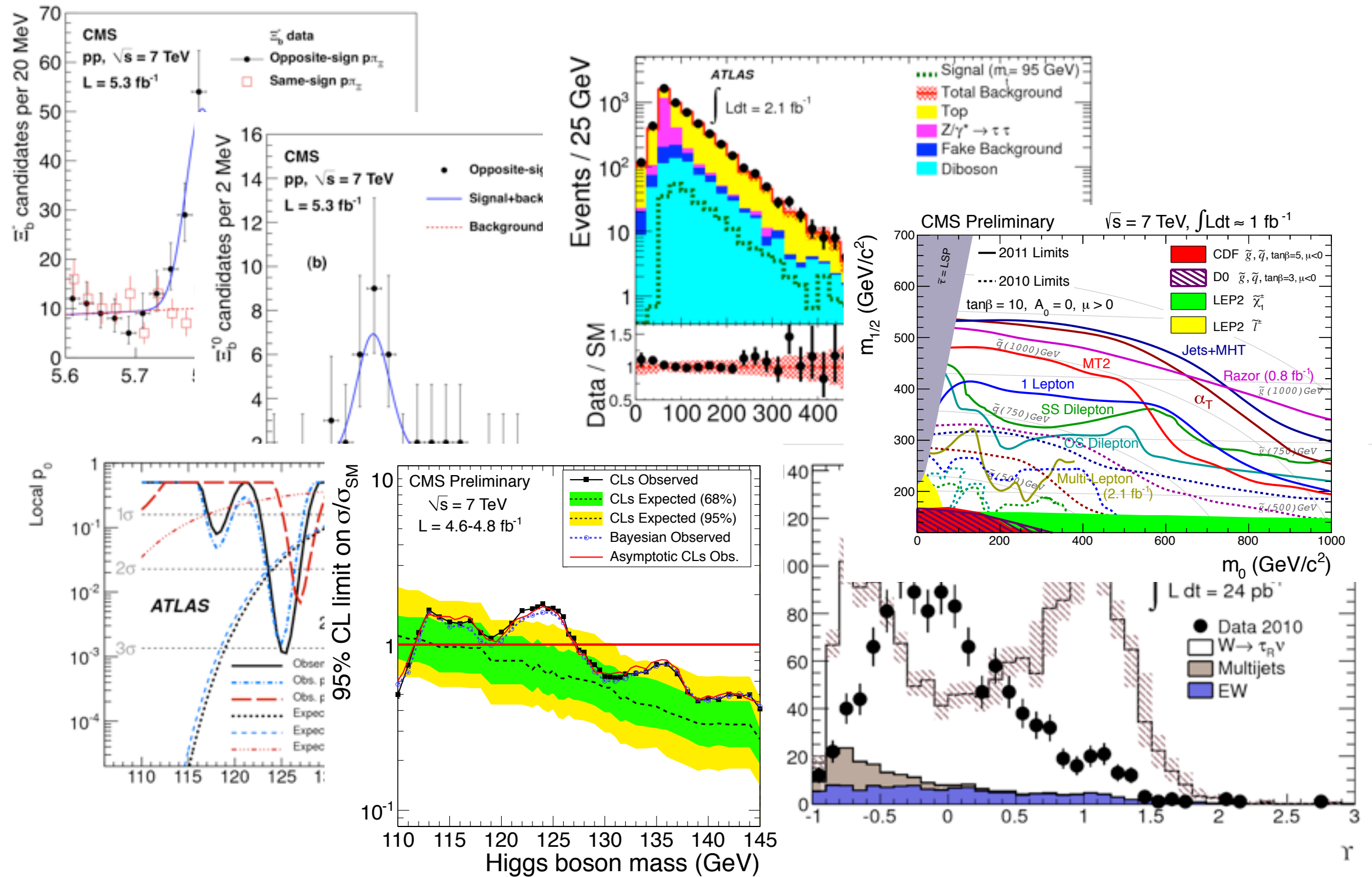
As of today
177 PB
of LHC data
stored in ROOT format

ALICE: 30PB, ATLAS: 55PB, CMS: 85PB, LHCb: 7PB



ROOT - In Plots








- Replacing CINT by new cling: correctness, full C++
- Cling based on LLVM and clang compiler libraries
- “New” open source compiler suite, default on OSX and FreeBSD
- Compiler-grade C++ on its way to C++11
- Best diagnostics on the market:




- Replacing CINT by new cling: correctness, full C++
- Cling based on LLVM and clang compiler libraries
- “New” open source compiler suite, default on OSX and FreeBSD
- Compiler-grade C++ on its way to C++11
- Best diagnostics on the market:




```
Error: Can't call map<string,const char*,less<string>,allocator<const
string,const char*> > >::operator[]((char*)0x255a9e8) in current scope err.C:19:
Possible candidates are...
(in map<string,const char*,less<string>,allocator<const string,const char*> > >)
Error: improper lvalue err.C:19
```


- Replacing CINT by new cling: correctness, full C++
- Cling based on LLVM and clang compiler libraries
- “New” open source compiler suite, default on OSX and FreeBSD
- Compiler-grade C++ on its way to C++11
- Best diagnostics on the market:



```
Error: Can't call map<string,const char*,less<string>,allocator<const
string,const char*> > >::operator[]((char*)0x255a9e8) in current scope err.C:19:
Possible candidates are...
(in map<string,const char*,less<string>,allocator<const string,const char*> > >)
Error: improper lvalue err.C:19
```

```
err.C:19:15: error: assigning to 'mapped_type' (aka 'const char *') from
      incompatible type 'double'
      myMap["A"] = 12.3;
                  ^ ~~~~
```



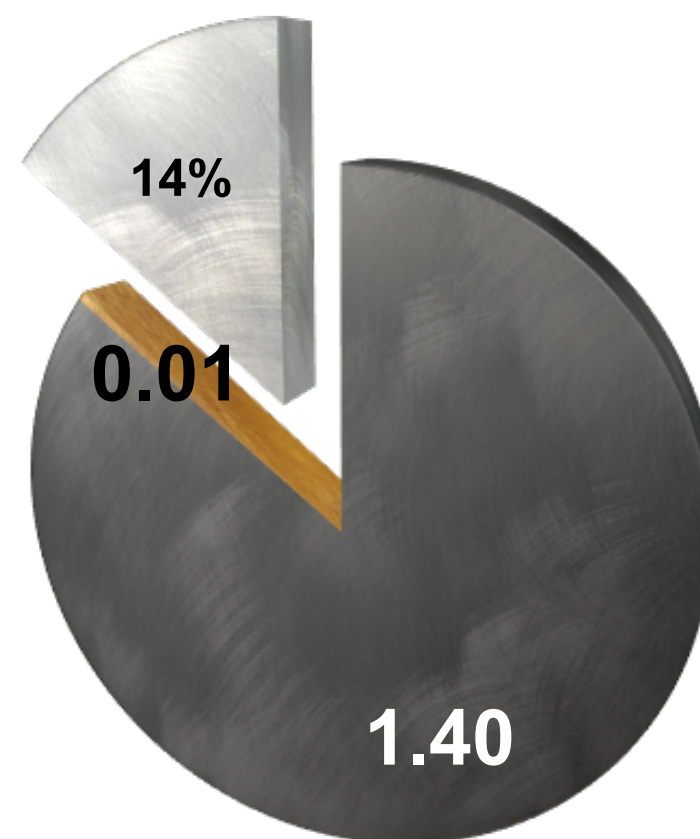
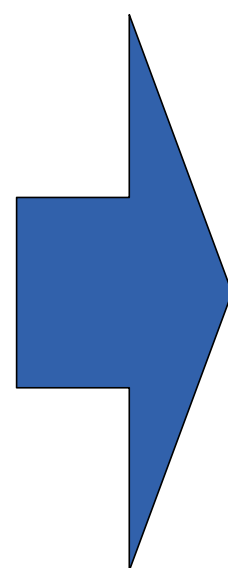
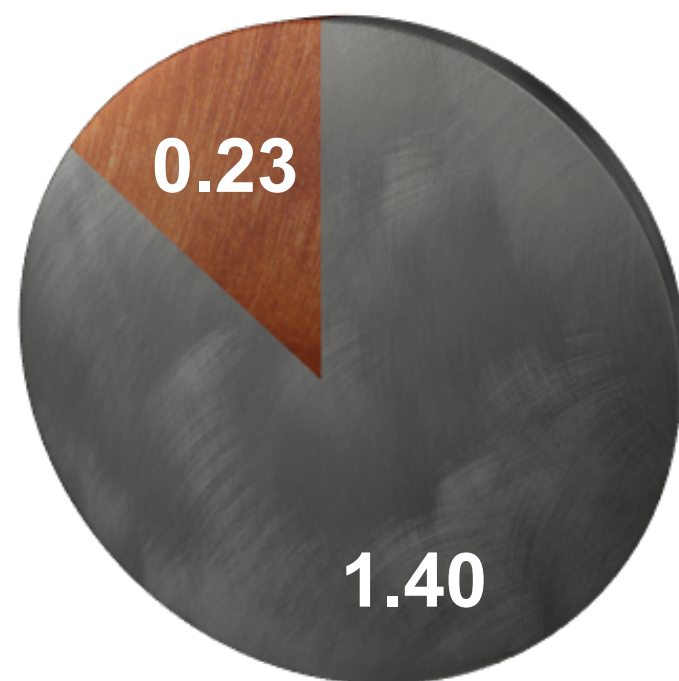


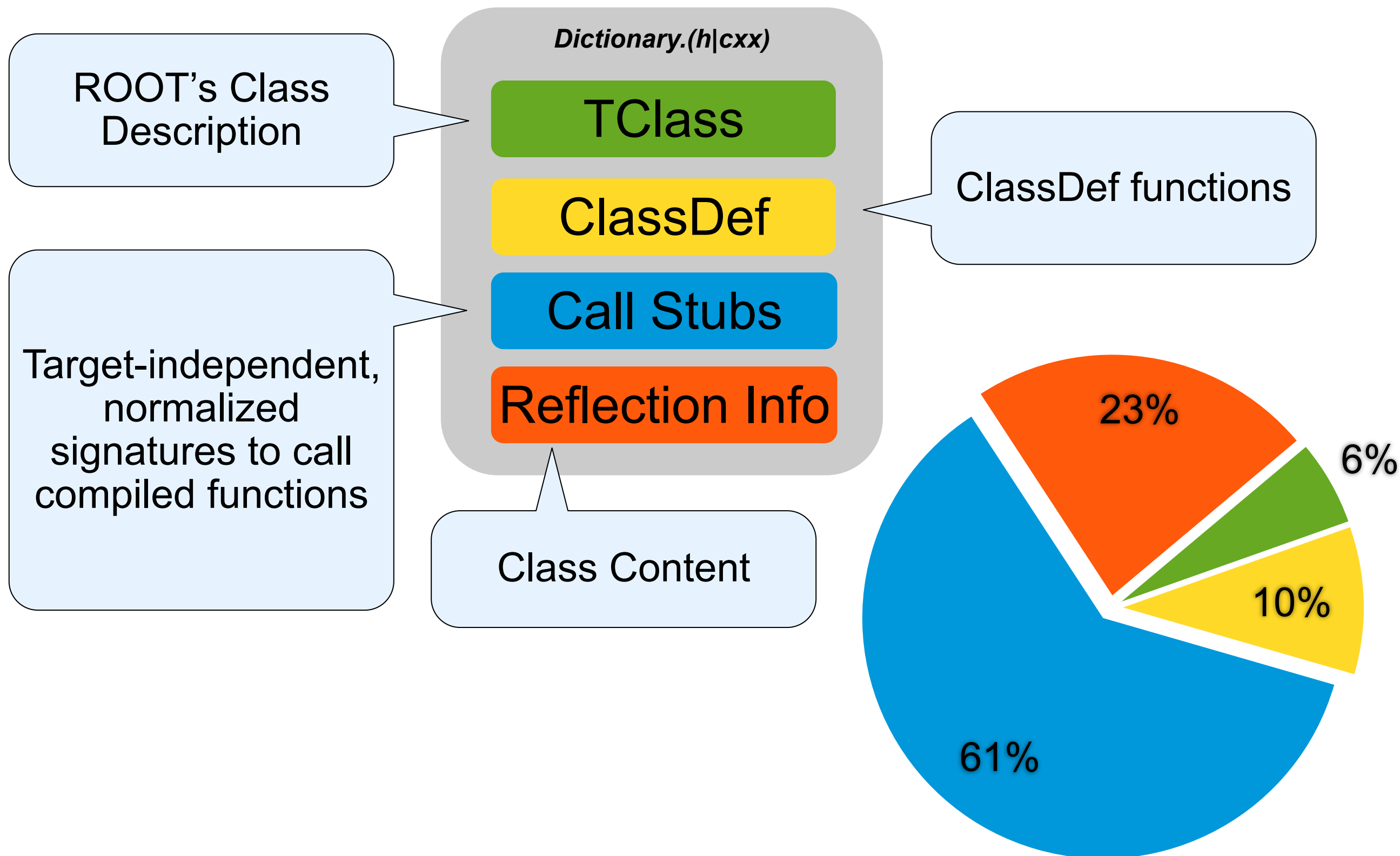
- Cling released in July 2011
- Fully functional C and C++11 interpreter
- Just-in-Time compilation, i.e. always ACLiC
- Still a few issues to solve, e.g. reloading of code
- Support multiple interpreter objects (c.f. multi-threaded context)
- Integration with ROOT close to be finished
- Interfaces via TCling and TClass
- Precompiled headers as dictionary, no huge source files

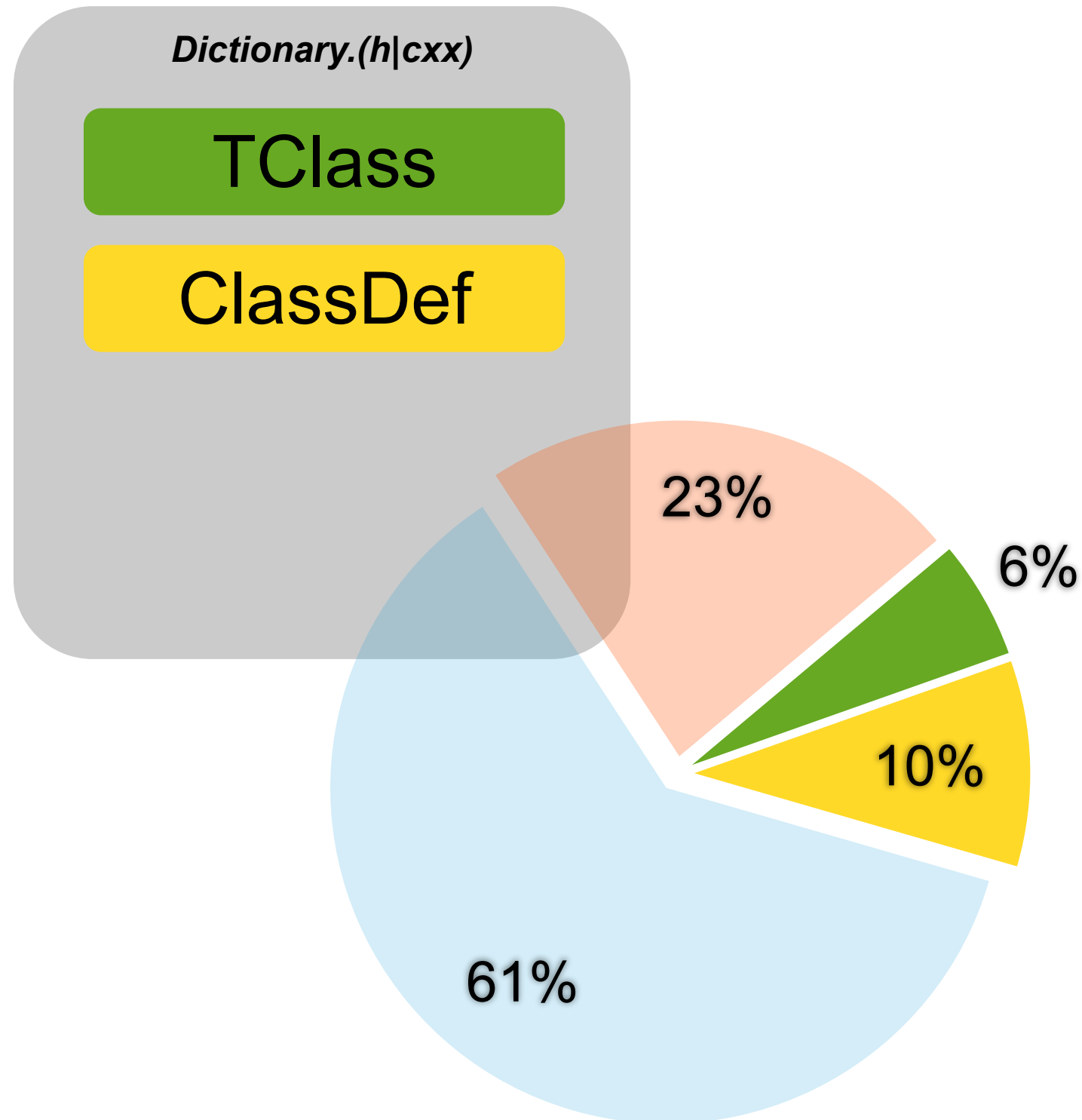
 ROOT
 CINT+Reflex

 ROOT
 saved

 cling







- Reflection data in precompiled headers (“modules”)
- JIT for function calls

Dictionary Sizes ROOT 5 vs 6

- Optimized, 64bit, Linux, v5-34 / trunk *with PCH* loading all libraries from root-config --glib

	v5-34	trunk
Dictionary .o (MB)	124	43 + 25 (PCH) = 68
G__Base2.o	2.2	0.6
G__Smatrix.o	5.1	0.8
Memory (MB)		
RSS	26	97
VSS	84	156
Startup Time		
CPU	0.06	0.5
SYS	0.02	0.03



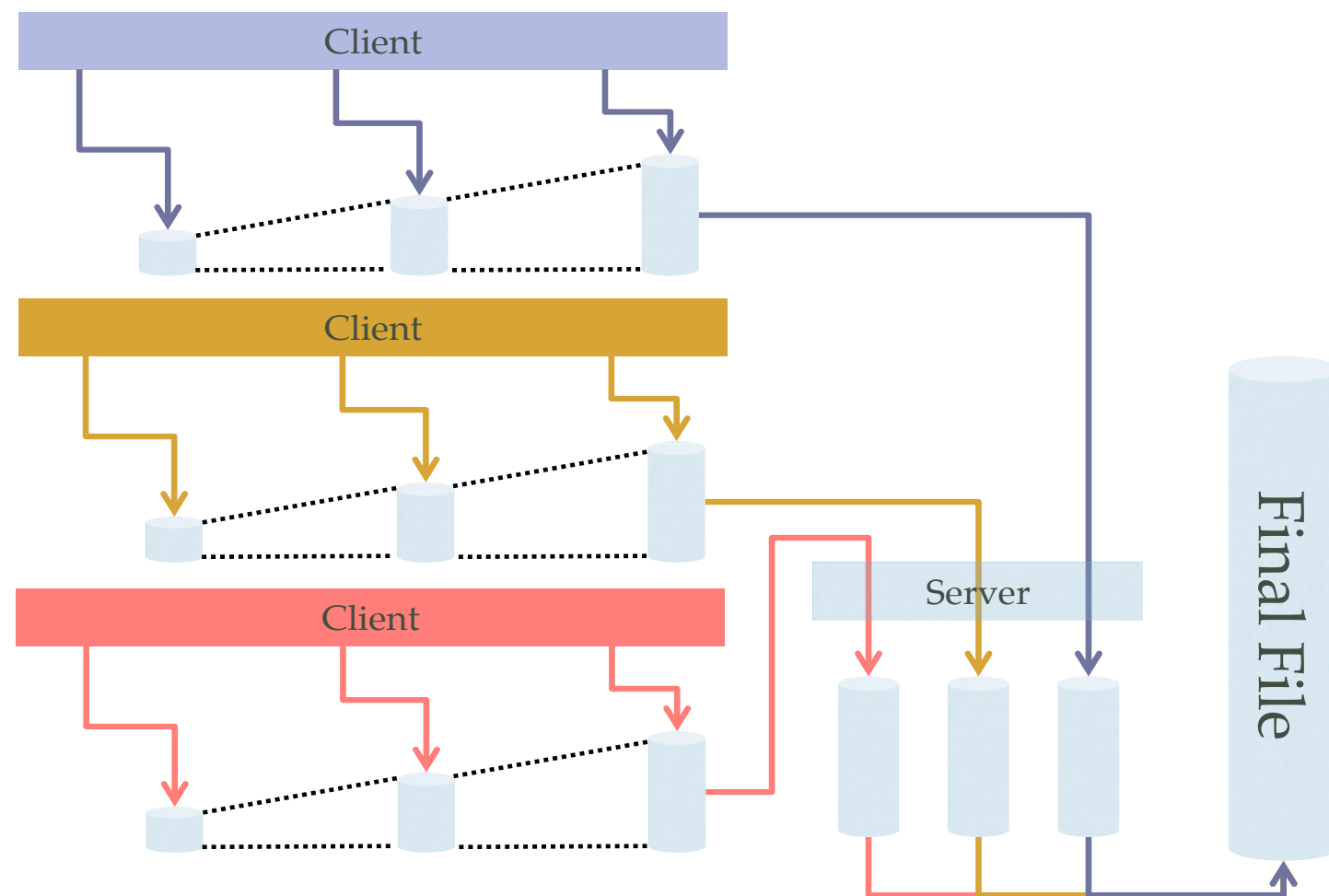
- rootcint becomes rootcling
- Writes dictionary modules, sourcefiles
- Same parameters
- Keep LinkDef.h
- Future:
 - Also act as genreflex replacement
 - Same parameters
 - Keep selection.xml
 - Further reduce compiled dictionary size
 - Leverage JIT



- Parallel Tree merging (see next slides)
- Asynchronous prefetching
 - When enabled, a separate thread will fetch the TTreeCache blocks that are likely to be needed next and the main thread continues the normal data processing
- Automatic support for more than one TTreeCache per file
 - `TTTree::SetCacheSize(Long64_t)` no longer overrides nor deletes the existing cache. Each cache is completely independent
- Support for HDFS, Amazon S3, CloudFront and Google Storage
- Support for LZMA compression

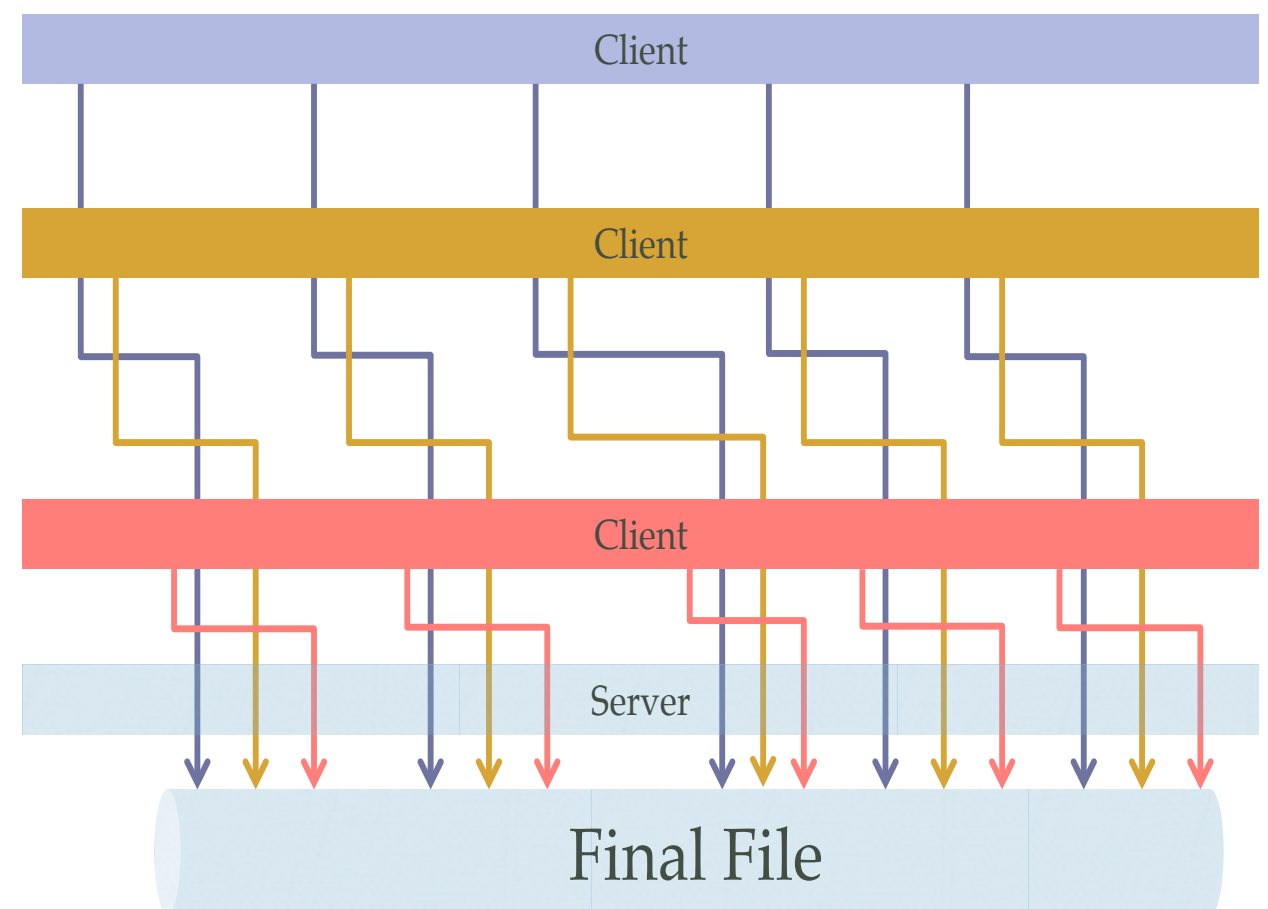


- Existing solution:
 - Processing and storing the partial output on each local node and then only at the end of the process, upload to the server and only when all slaves are done, read all files on the server and write to a single output file.





- New TFile implementations:
 - TMemFile: a completely in-memory version of TFile.
 - TParallelMergingFile: a TMemFile that on a call to Write will upload its content and reset the TTree objects.
- New solution:
 - Increase parallelism by having the workers start uploading the TTree clusters directly to the server which immediately starts saving them in the final output file.





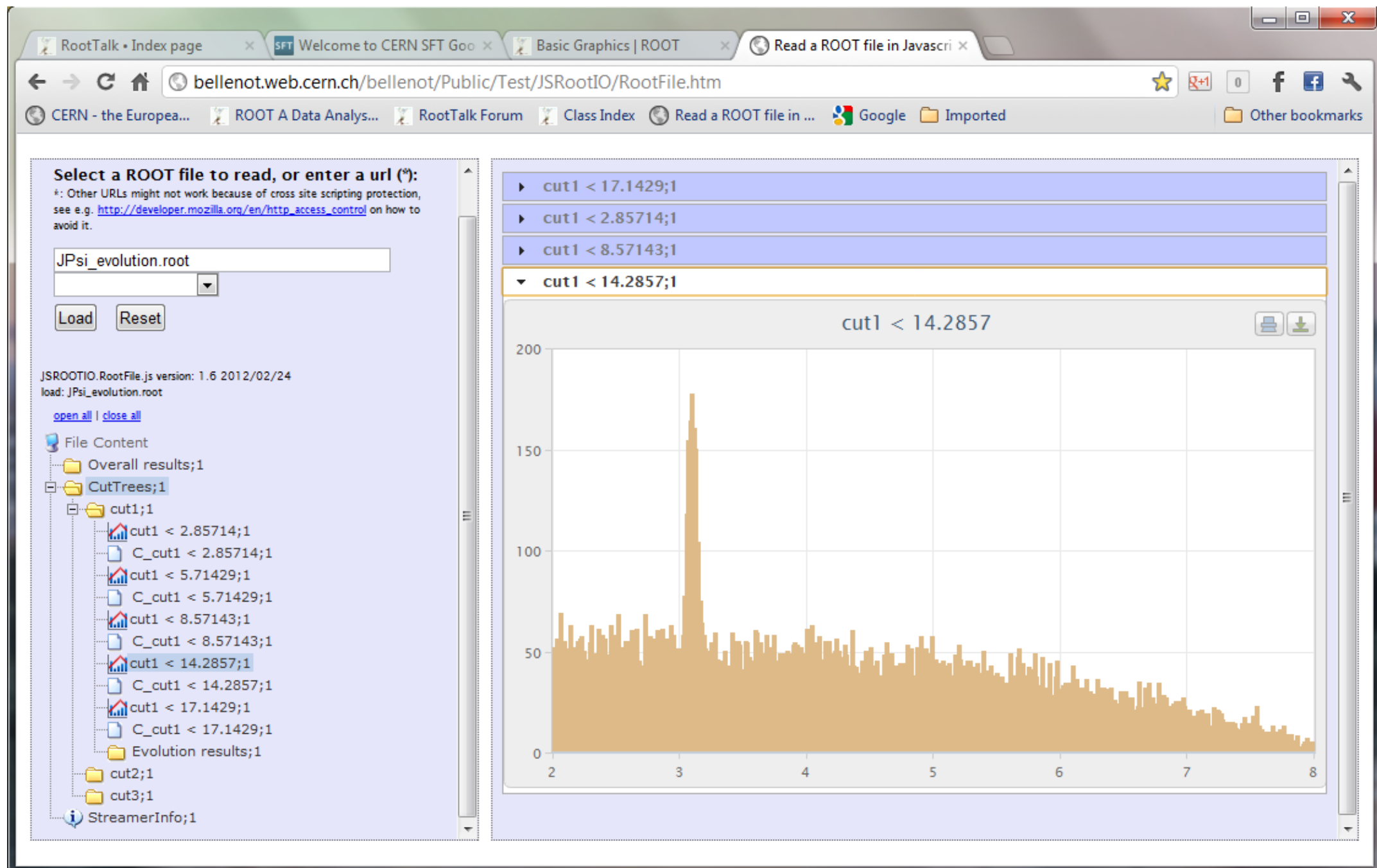
- Continuous review of I/O performance in close collaboration with experiment experts
- Reimplementation of OptimizeBaskets
- Explore changing the on-file byte format to little endian
- Extension of the parallel merger
- Improvement of thread safety
- I/O Customization: Nested Objects



- 3 approaches:
 - Parallel Merge via external process
 - Support for one TFile per thread
 - Currently requires meta data (TClass, TStreamerInfo) to be fully build before starting parallel operation
 - Planning to lifting this restriction in ROOT 6 (requires cling)
 - Internal use of spare cores
 - Ability to read multiple TBranch data in parallel
 - Top level branches can be uncompressed and un-streamed independently
 - Prefetching of remote file content as a separate thread
 - Threaded zipping and unzipping of buffers

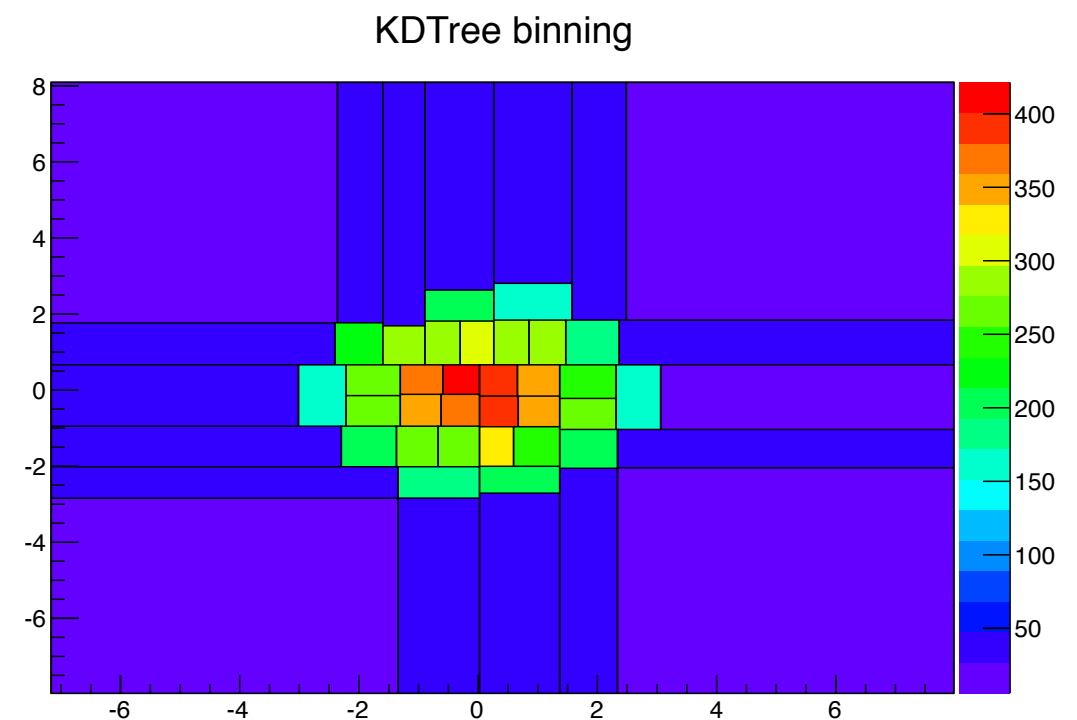


- Provide ROOT file access entirely locally in a browser
 - ROOT files are self describing, the “proof of the pudding...”





- New class for binning multi-dimensional data using a kd tree (TKDTreeBinning)
 - Automatic binning of data
 - Every bin will have same entries or same weight (content)
 - Efficient multi-dimensional histogram
 - Can be used to compare MC and data (make bins with MC and compare with the data)
 - Could be used for non-parametric density estimation using kernels



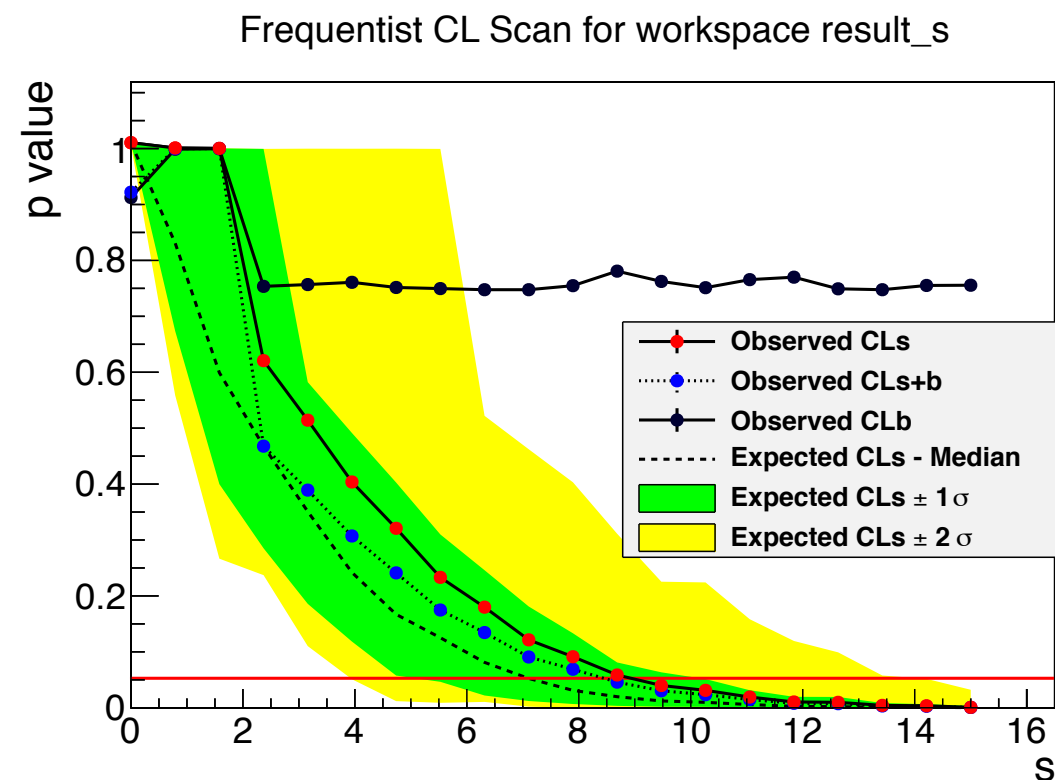
- Also new THn class
 - Multidimensional histograms with equal bins
 - For non-sparse multi-dimensional data



- Framework for statistical calculations
 - Standard tool for producing physics results at LHC
 - Parameter estimation (fitting)
 - Interval estimation (e.g limit results for new particle searches)
 - Discovery significance (quantifying excess of events)
 - Implement several statistical methods (Bayesian, Frequentist, Asymptotic)
 - New tools for model creation and combinations
 - Histfactory: make RooFit models (RooWorkspace) from input histograms
 - See the presentations from ROOT Users Workshop
 - RooFit:
 - <http://indico.cern.ch/contributionDisplay.py?contribId=19&confId=217511>
 - RooStats:
 - <http://indico.cern.ch/contributionDisplay.py?contribId=23&confId=217511>

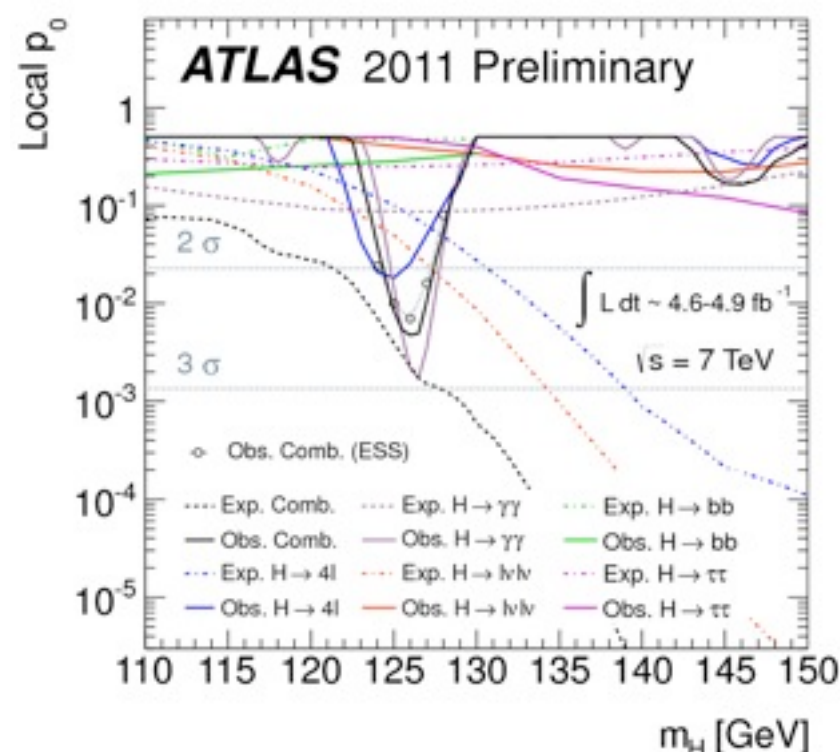
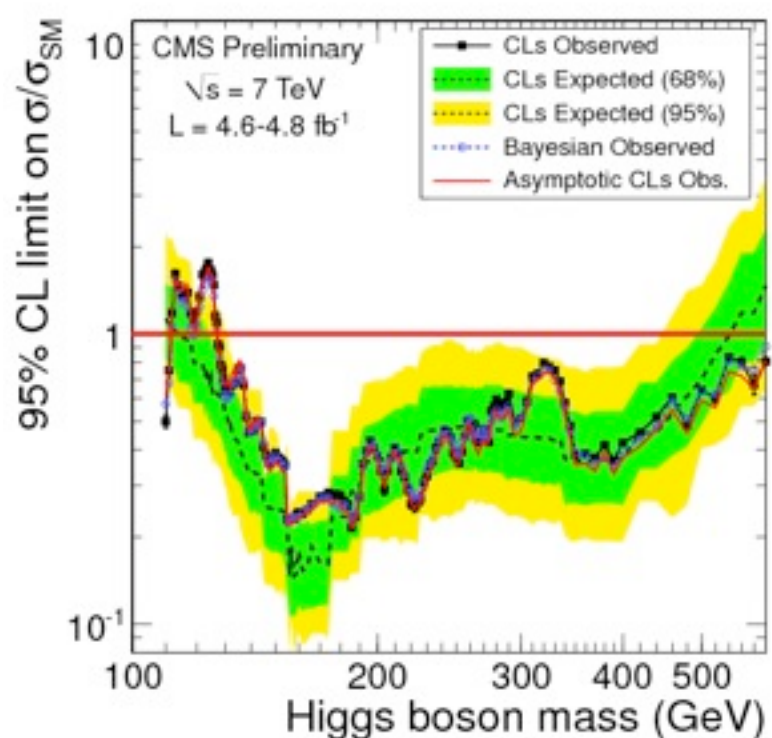


- Classes for computing exclusion limits and discovery significance using the procedure endorsed and agreed by ATLAS and CMS (CLs method for limits)
 - Using both pseudo-experiments or asymptotic formulae
 - Compute observed limits, expected limits and bands
- Can be very time consuming, many pseudo-experiment need to be generated and fitted
- Working on improving performances
 - More efficient fitting for multi-parameter problems (improve the new version of Minuit)
 - Investigate parallelization techniques (PROOF, Multi-threads or porting on GPU)





- Higgs combination results of ATLAS and CMS
 - Make combined model with several channels and more than 200 parameters
 - RooStats used successfully for producing results:
 - Could use different statistical methods for comparison
 - Required a lot of CPU for fitting each pseudo-experiment
 - Use Grid for submitting all jobs
 - On-going work on improving fitting performances





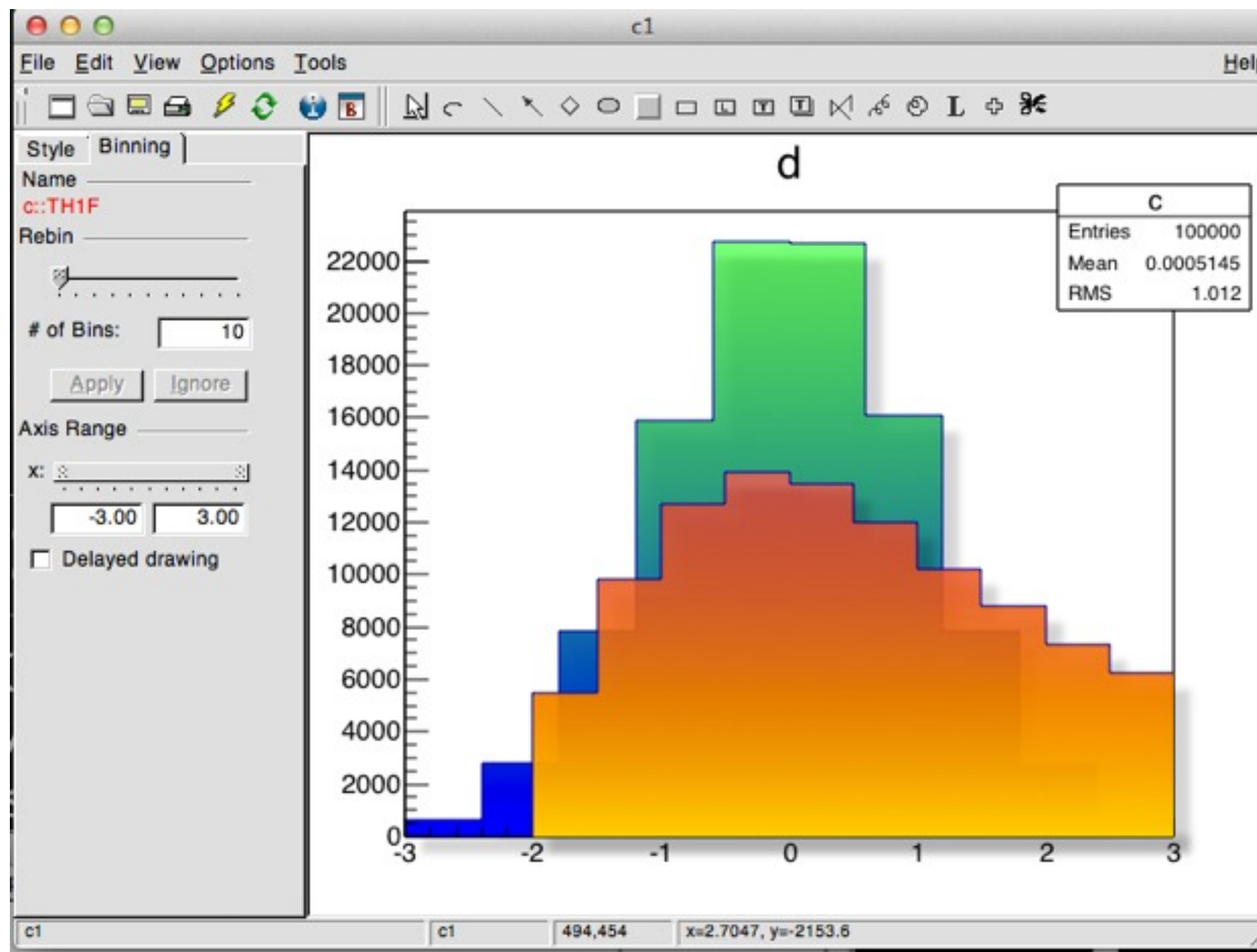
- The ROOT solution to most Big Data problems
- Increasing number of sites and users deploying PROOF
- Deployment on ad-hoc PROOF clusters is made easy by PoD (see talk tomorrow by Dario Berzano)
- The Virtual Analysis Facility - PROOF cluster on a cloud (see also Dario's talk tomorrow)
- An extensive PROOF bench suite has been developed allowing to test many aspects of PROOF related parameters (network, disk, cpu, etc.)
- Many stability improvements



- Improvement of the connection layer and of the work distribution engine (packetizer)
- Result merging optimization (see also parallel tree merging in I/O)
- Improvement of memory management (in merging of results, in sharing common data)
- Support for dynamic addition of workers
- Towards a fully-functional multi-master setup
 - Dataset management across clusters
 - Dynamic load-balancing across sub-masters
 - Dynamic re-organization of a set of workers in multi-tier structures to scale up to $O(1000)$ cores
 - Needed for extreme platforms like IBM's BG/Q



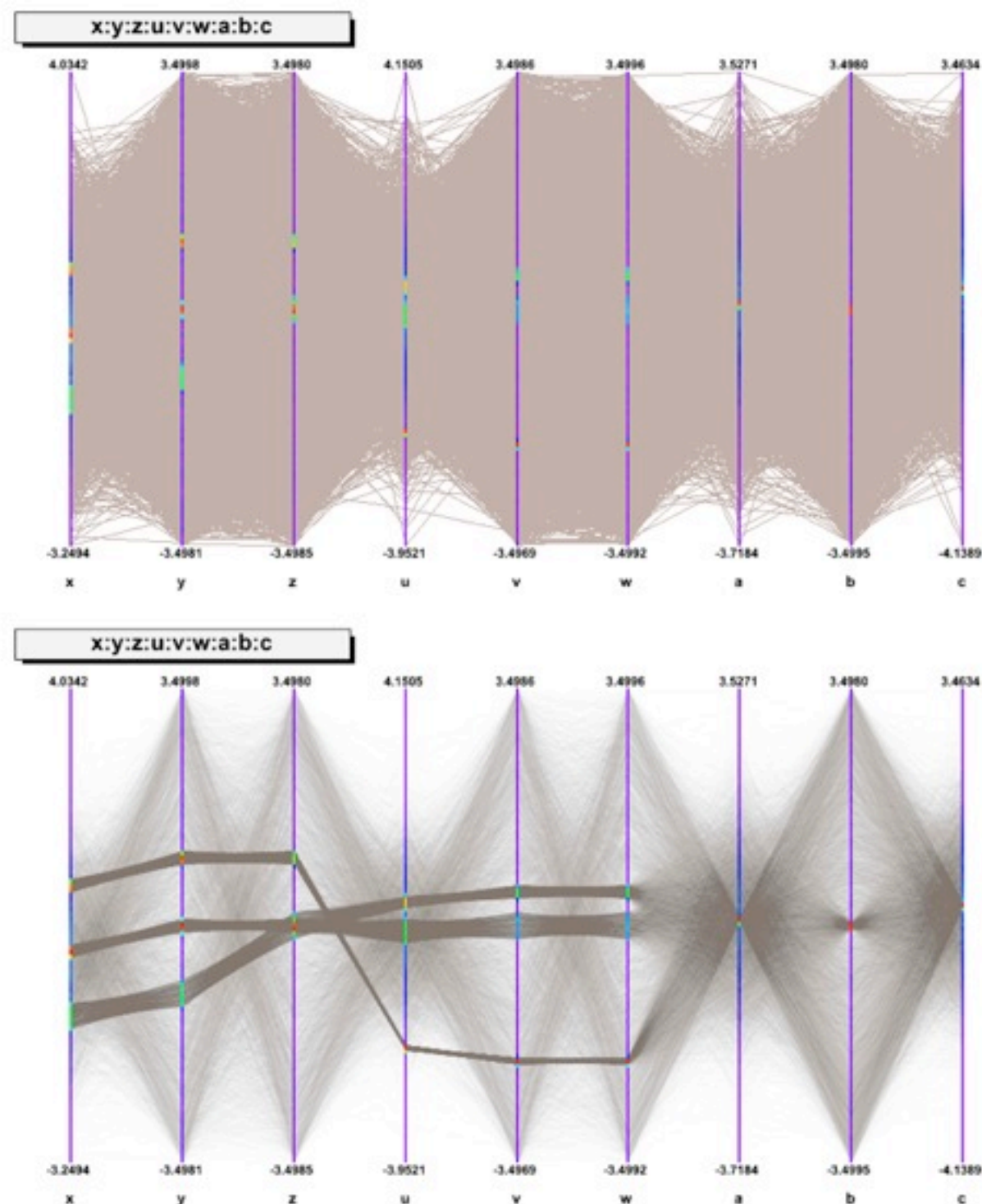
- Uses native OSX Cocoa and Quartz for graphics
 - Supports easily new features like transparency, gradients and shadows



- In parallel to OSX Cocoa and Quartz, transparency is also supported in PDF and TImage outputs.

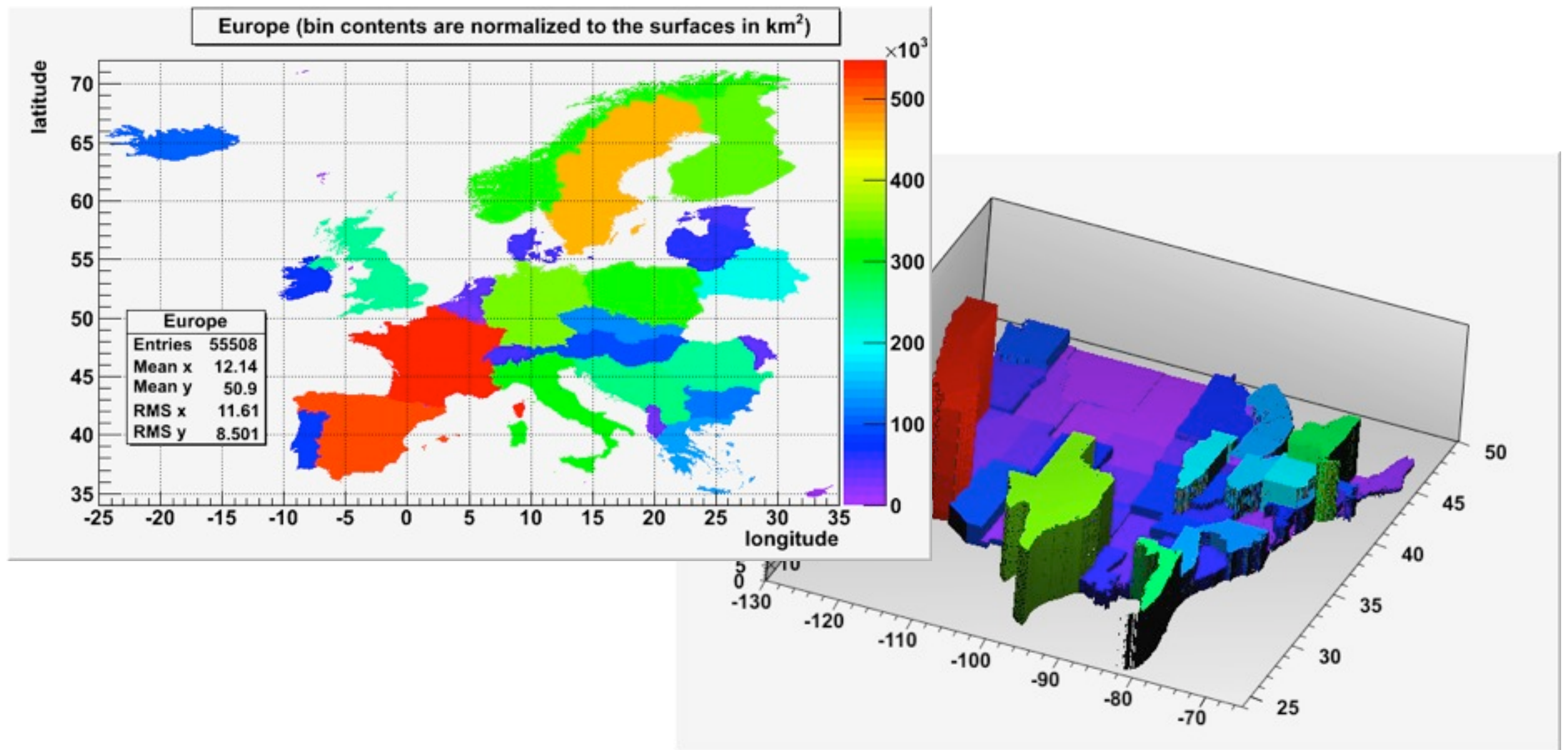
This provides new possibilities to visualize large data set. As shown on this parallel coordinates plots.

These two plots are the same. On top without transparency: useless. Bottom with transparency: all the clusters clearly appear.



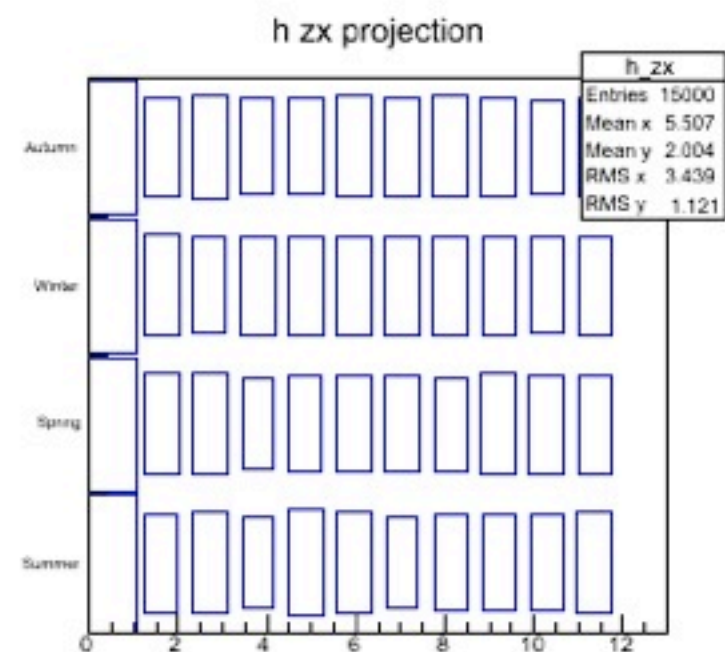
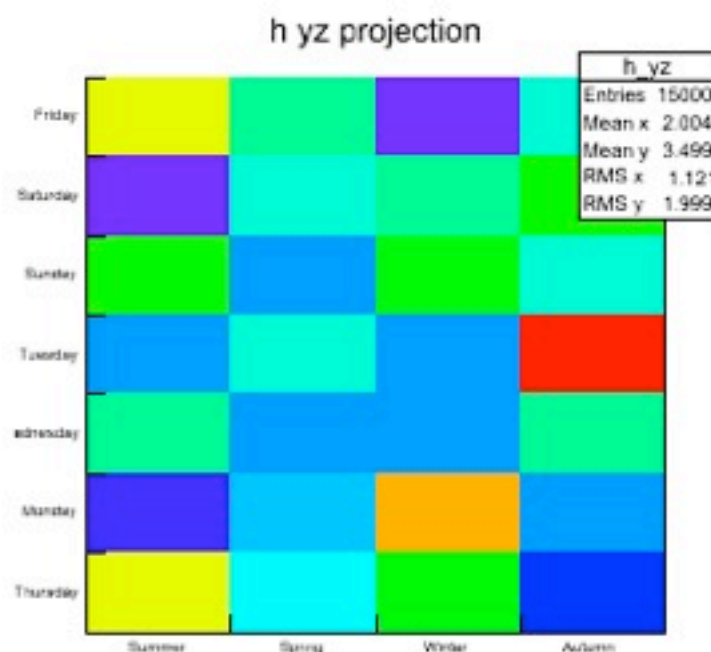
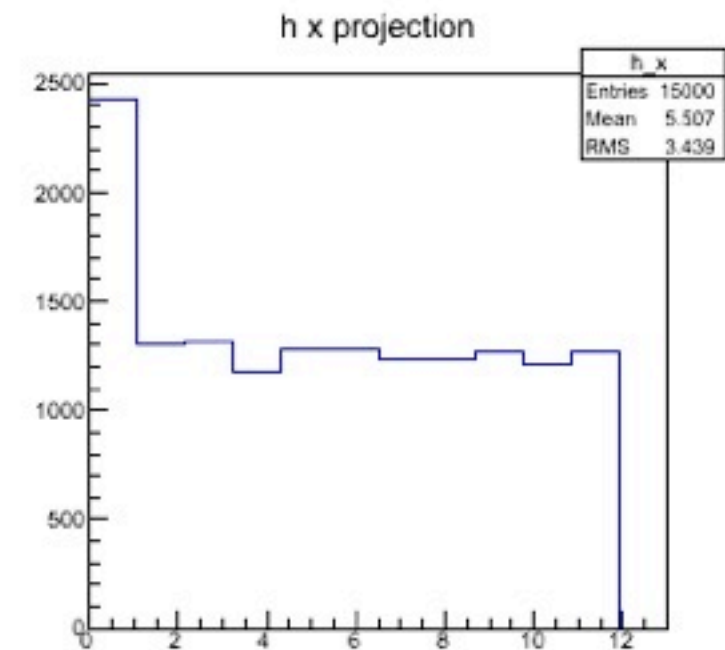
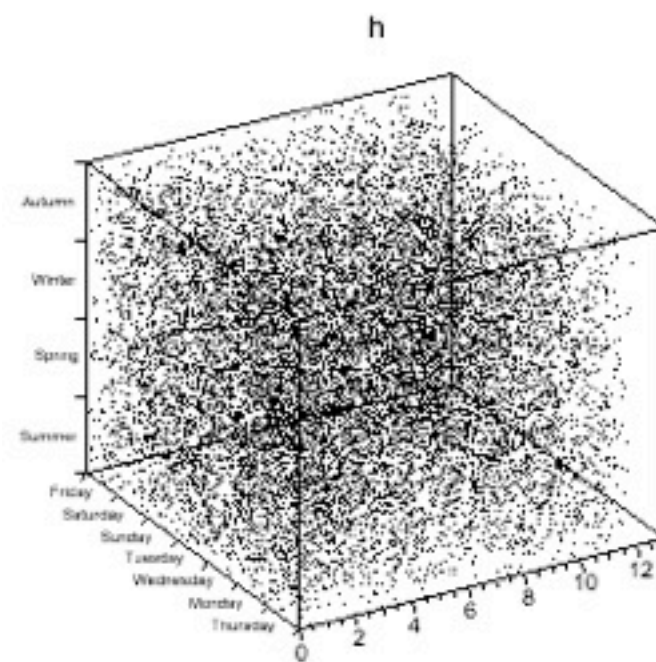
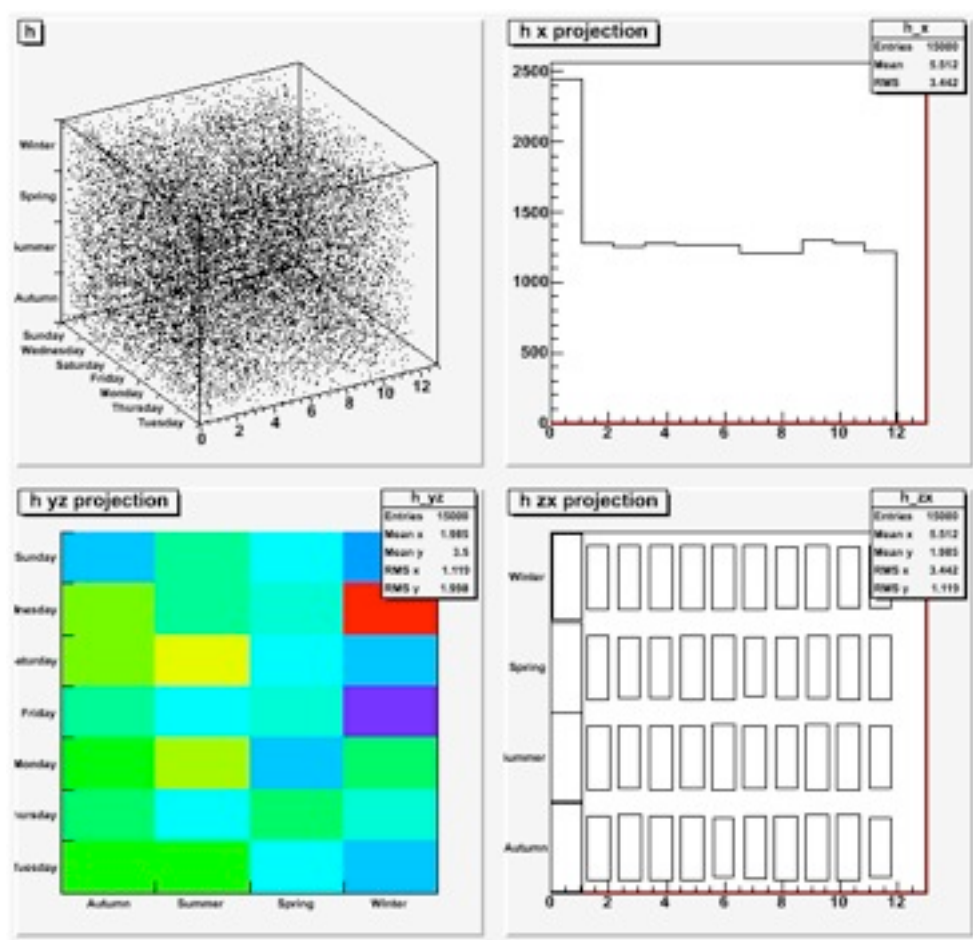


- TH2Poly is a 2D Histogram class, inheriting from TH2, allowing to define polygonal bins of arbitrary shape.





- The new style “Modern” is a synthesis of different custom styles used by several major experiments.





T_{La}tex

$$\prod_{j \geq 0} \left(\sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left(\sum_{k_0, k_1, \dots \geq 0} a_{0k_0} a_{1k_1} \dots \right)$$

$$W_{\delta_1 \rho_1 \sigma_2}^{3\beta} = U_{\delta_1 \rho_1 \sigma_2}^{3\beta} + \frac{1}{8\pi^2} \int_{\alpha_1}^{\alpha_2} d\alpha'_2 \left[\frac{U_{\delta_1 \rho_1}^{2\beta} - \alpha'_2 U_{\rho_1 \sigma_2}^{1\beta}}{U_{\rho_1 \sigma_2}^{0\beta}} \right]$$

$$d\Gamma = \frac{1}{2m_A} \left(\prod_f \frac{d^3 p_f}{(2\pi)^3} \frac{1}{2E_f} \right) |M(m_A - \{p_f\})|^2 (2\pi)^4 \delta^{(4)}(p_A - \sum p_f) \\ 4 \operatorname{Re} \left\{ \frac{2}{1 - \Delta\alpha} \chi(s) \left[\hat{g}_v^e \hat{g}_v^f (1 + \cos^2 \theta) + 2 \hat{g}_a^e \hat{g}_a^f \cos \theta \right] \right\}$$

$$p(n) = \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k} A_k(n) \frac{d}{dn} \frac{\sinh \left\{ \frac{\pi}{k} \sqrt{\frac{2}{3}} \sqrt{n - \frac{1}{24}} \right\}}{\sqrt{n - \frac{1}{24}}}$$



T Latex

$$\prod_{j \geq 0} \left(\sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left(\sum_{k_0, k_1, \dots \geq 0} a_{0k_0} a_{1k_1} \dots \right)$$

$$W_{\delta_1 \rho_1 \sigma_2}^{3\beta} = U_{\delta_1 \rho_1 \sigma_2}^{3\beta} + \frac{1}{8\pi^2} \int_{\alpha_1}^{\alpha_2} d\alpha'_2 \left[\frac{U_{\delta_1 \rho_1}^{2\beta} - \alpha'_2 U_{\rho_1 \sigma_2}^{1\beta}}{U_{\rho_1 \sigma_2}^{0\beta}} \right]$$

$$d\Gamma = \frac{1}{2m_A} \left(\prod_f \frac{d^3 p_f}{(2\pi)^3 2E_f} \right) |M(m_A)|^2 \prod_{j \geq 0} \left(\sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left(\sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} a_{0k_0} a_{1k_1} \dots \right)$$

$$4 \operatorname{Re} \left\{ \frac{2}{1 - \Delta \alpha} \chi(s) [\hat{g}_v^e \hat{g}_v^f (1 + \right.$$

$$W_{\delta_1 \rho_1 \sigma_2}^{3\beta} = U_{\delta_1 \rho_1 \sigma_2}^{3\beta} + \frac{1}{8\pi^2} \int_{\alpha_1}^{\alpha_2} d\alpha'_2 \left[\frac{U_{\delta_1 \rho_1}^{2\beta} - \alpha'_2 U_{\rho_1 \sigma_2}^{1\beta}}{U_{\rho_1 \sigma_2}^{0\beta}} \right]$$

$$p(n) = \frac{1}{\pi \sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k} A_k(n) = \frac{1}{2m_A} \left(\prod_f \frac{d^3 p_f}{(2\pi)^3 2E_f} \right) |\mathcal{M}(m_A - \{p_f\})|^2 (2\pi)^4 \delta^{(4)}(p_A - \sum$$

$$4 \operatorname{Re} \left\{ \frac{2}{1 - \Delta \alpha} \chi(s) [\hat{g}_v^e \hat{g}_v^f (1 + \cos^2 \theta) + \hat{g}_a^e \hat{g}_a^f \cos \theta] \right\}$$

$$p(n) = \frac{1}{\pi \sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k} A_k(n) \frac{d}{dn} \frac{\sinh \left\{ \frac{\pi}{k} \sqrt{\frac{2}{3}} \sqrt{n - \frac{1}{24}} \right\}}{\sqrt{n - \frac{1}{24}}}$$

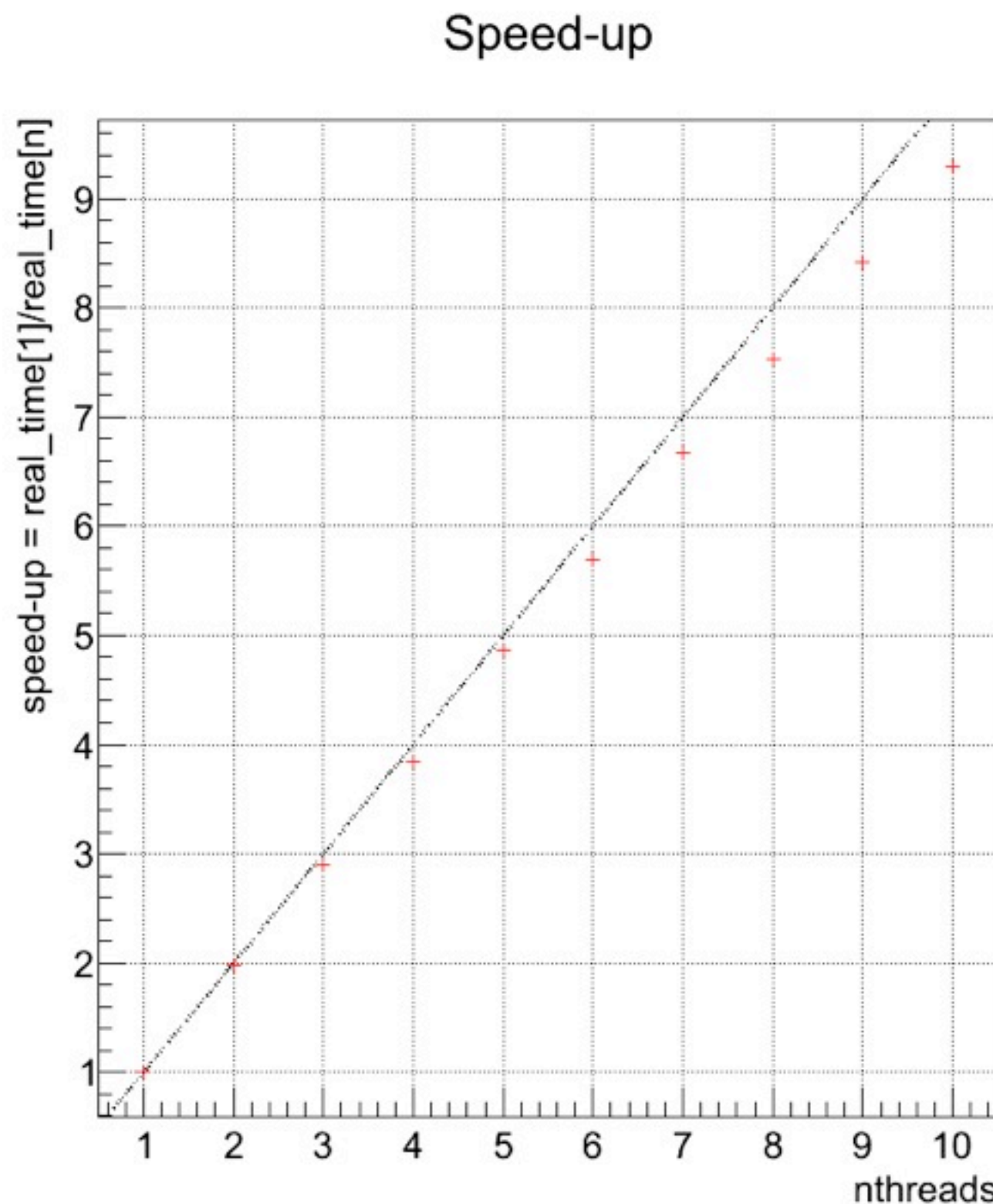
RHIC スピン物理

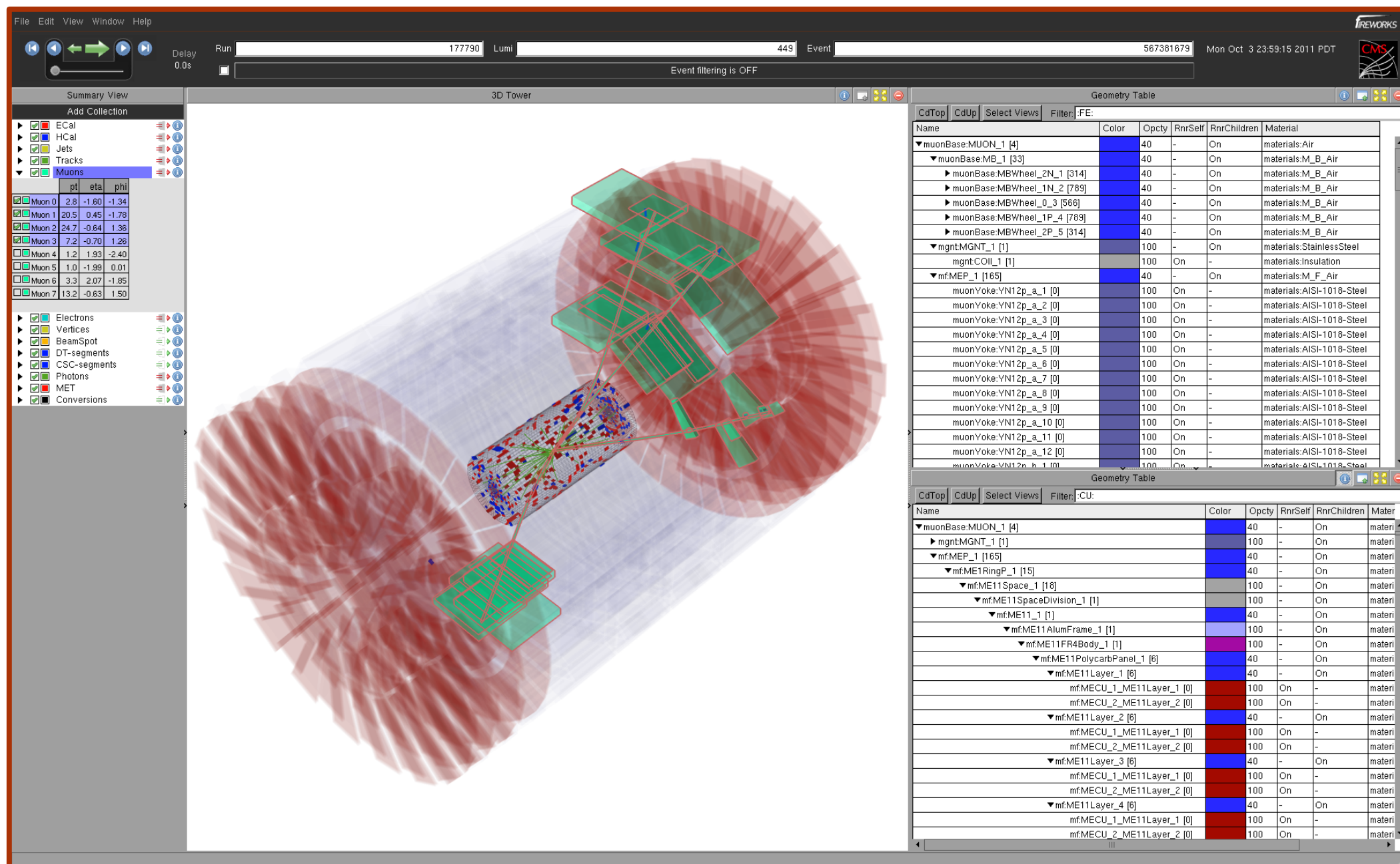


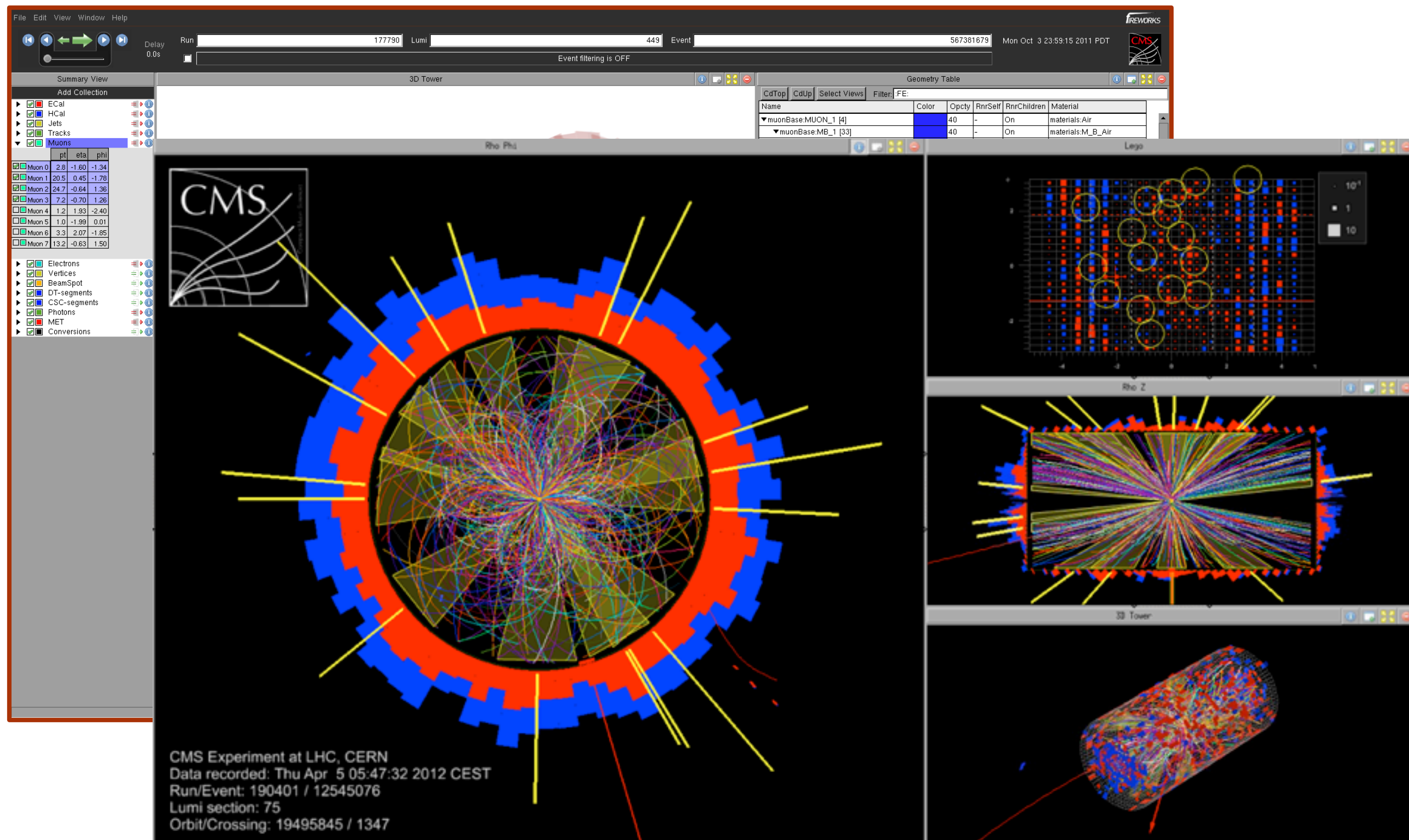
- Enabled via: `gGeoManager->SetMaxThreads(N)`
 - Fast thread ID retrieval implemented via TLS
- Each thread works with its own navigator
 - Navigators holding all state-dependent data
- No locks, no race conditions
 - Stateful data structures migrated to thread data arrays
- Small overhead by replacing direct access
 - Additional calls to `TGeoManager::ThreadId()`
 - Partially compensated by percolating stateful data in the calling sequence of some internal methods
- Excellent scalability
- See the presentation on Thursday @ 2:20 pm
 - <http://indico.cern.ch/contributionDisplay.py?contribId=12&confId=217511>



- Enabled via: gGeoMan
 - Fast thread ID retrieval
- Each thread works with
 - Navigators holding all s
- No locks, no race cond
 - Stateful data structures
- Small overhead by repl
 - Additional calls to TGec
 - Partially compensated k calling sequence of son
- Excellent scalability
- See the presentation o
 - <http://indico.cern.ch/contributionD>









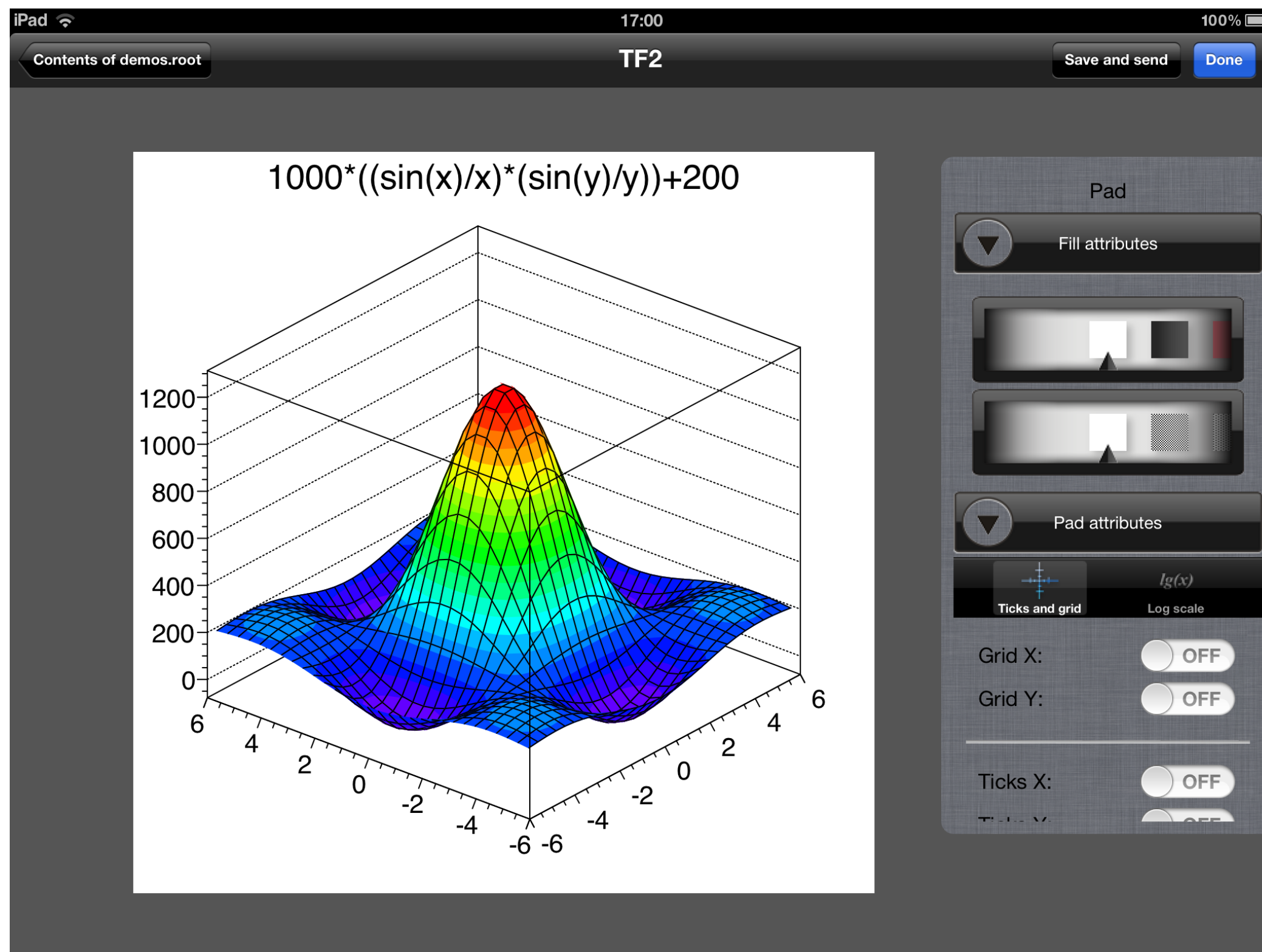
- Concurrent tasks
 - TTask+TThread and TTaskManager
- Concurrency in I/O
 - Read-ahead thread
 - Parallel merger
 - Threaded zipping and unzipping of buffers
- Concurrency and vectorization in math
 - Introduction of the Vc matrix library
- Concurrent geometry navigation
- Thread safe cling
 - Multiple concurrent cling instances



- Currently over 500 million iOS devices, of which 90 millions iPads
- All of ROOT ported to iOS (little endian, 32-bit, ARM)
 - To cross compile on OSX to iOS native version:
./configure ios
 - To compile on OSX to iOS simulator version:
./configure iossim
- Android will come later



- Ported ROOT's non-GUI graphics on iOS
 - TVirtualPad interface in ROOT::iOS::Pad using Quartz 2D library
- Writing ROOT-based native applications for iOS
 - Demo of ROOT's graphics
 - Browser for ROOT files
- Future developments
 - Components to support users who want to develop for iOS
 - Port ROOT's OpenGL code to GL/ES
 - Use HCP (head coupled perspective) technology for event display (Eve)
 - Uses front facing camera
 - Glasses free 3D display





- Build using “./configure;make” and now also cmake
- Code managed now in Git
- Static code analyzer
 - Coverity (commercial)
- New continuous build system
 - Electric Commander (commercial)
 - Incremental and full builds and nightly build snapshots for 14 platforms/configurations
- New source code convention checker
 - Eclair (commercial, but uses LLVM inside)
- New documentation system
 - From MS Word to Docbook to Markdown



```
# Main title
```

```
To make a paragraph just skip one line.
```

```
## A sub-section
```

```
Now an itemized list:
```

- item 1
- item 2

The basic idea behind this mark-up language is to be as close possible to what you would write in a email.

```
<sect1>
<title>Main title</title>
<para>
To make a paragraph just skip one line.
</para>
<sect2>
<title>A sub-section</title>
<para>
Now an itemized list:
</para>
<itemizedlist>
  <listitem><para>item 1</para></listitem>
  <listitem><para>item 2</para></listitem>
</itemizedlist>
<para>
The basic idea behind this mark-up language is
to be as close possible to what you would write
in a email.
</para>
</sect2>
</sect1>
```




- Current version is v5-34-05
 - It is an LTS (Long Term Support) version
 - New features will be back ported from the trunk
- Version v6-00-00 is scheduled for when it is ready
 - It will be Cling based
 - It will not contain anymore CINT/Reflex/Cintex
 - GenReflex will come in 6-02
 - It might not have Windows support (if not, likely in 6-02)
 - Several “Technology Previews” will be made available
 - Can be used to start porting v5-34 to v6-00



- Users and experiments want an absolutely stable ROOT so their carefully tuned analysis are not disturbed when moving to a new version
- At the same time we need to extend to new platforms and incorporate new technologies to make life easier for the users and to keep ROOT on the bleeding edge
- A careful balancing act, but a very exciting one