Booting ROOT with BOOT

René Brun, Fons Rademakers

CERN

Geneva, Switzerland

Plan

- The BOOT ideas were introduced at CHEP06 in Mumbay.
- Meanwhile these ideas have matured and many are now a reality.

- In this talk we will describe:
  - Achievements already in released version 5.16 (June 07)
  - Developments in current CVS head -> in December release 5.18
  - Plans for short and medium term.

- Note that BOOT is the name of the project and not necessarily the name of an executable module.

BOOT: Ideas

- The goal is to facilitate the life of an average user
  - Making the installation trivial
  - Providing automatic updates
  - Loading only what you need, ie minimizing memory use
  - Dynamic linking via plug-in manager
  - Making the UI (in particular the GUI widgets) more dynamic to accommodate extensions (ROOT or user classes).

Some Facts

10 shared libs

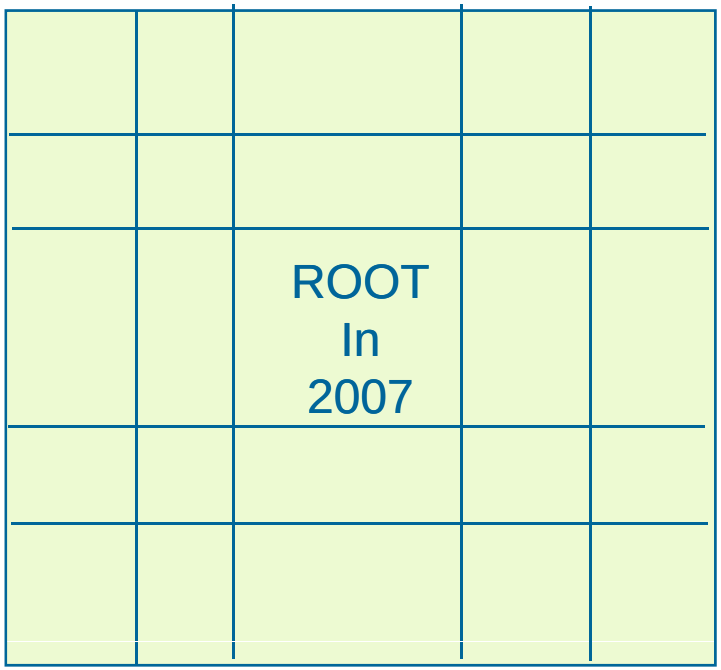
200 classes

ROOT
In
1995

PAW model

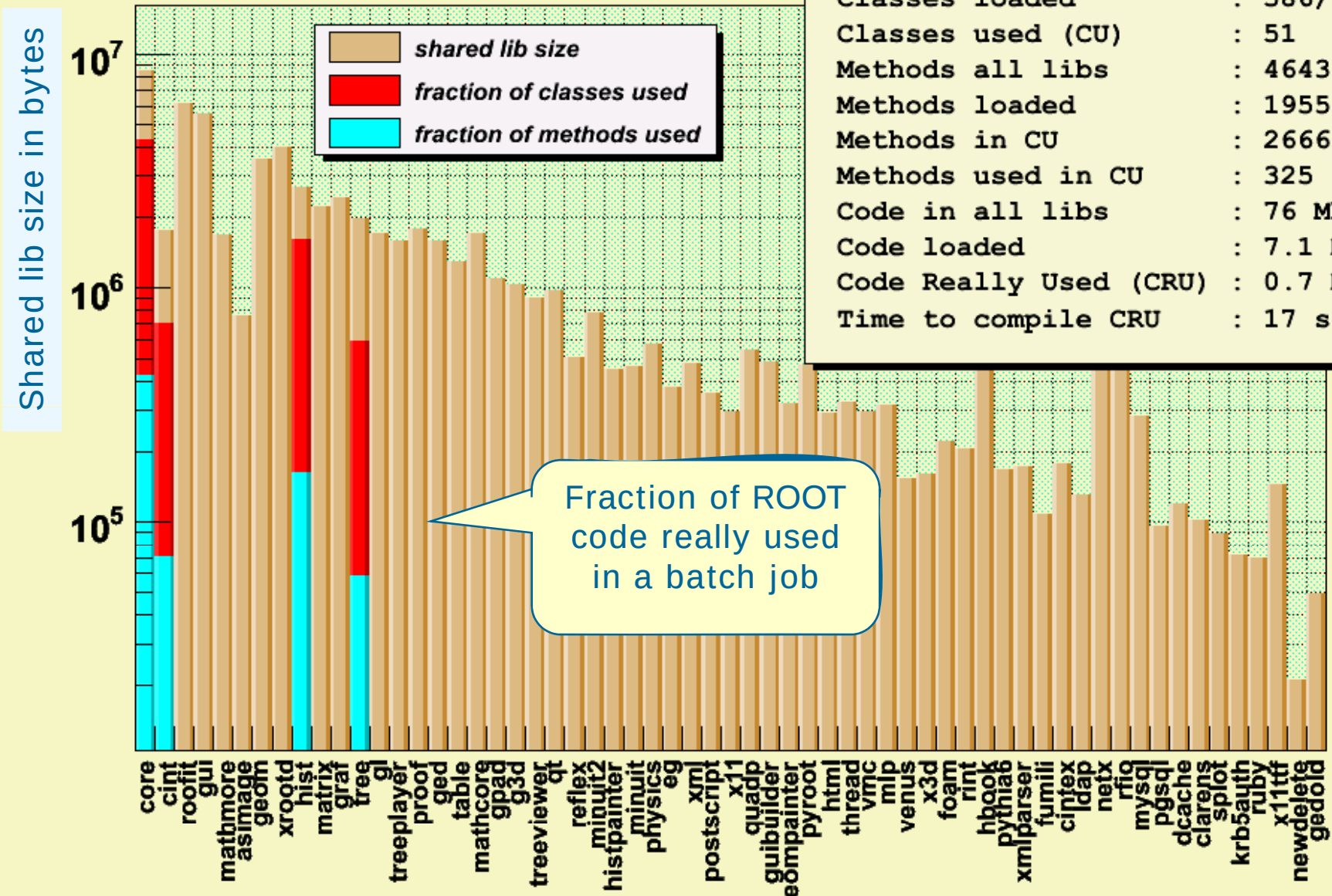
100 shared libs

1800 classes

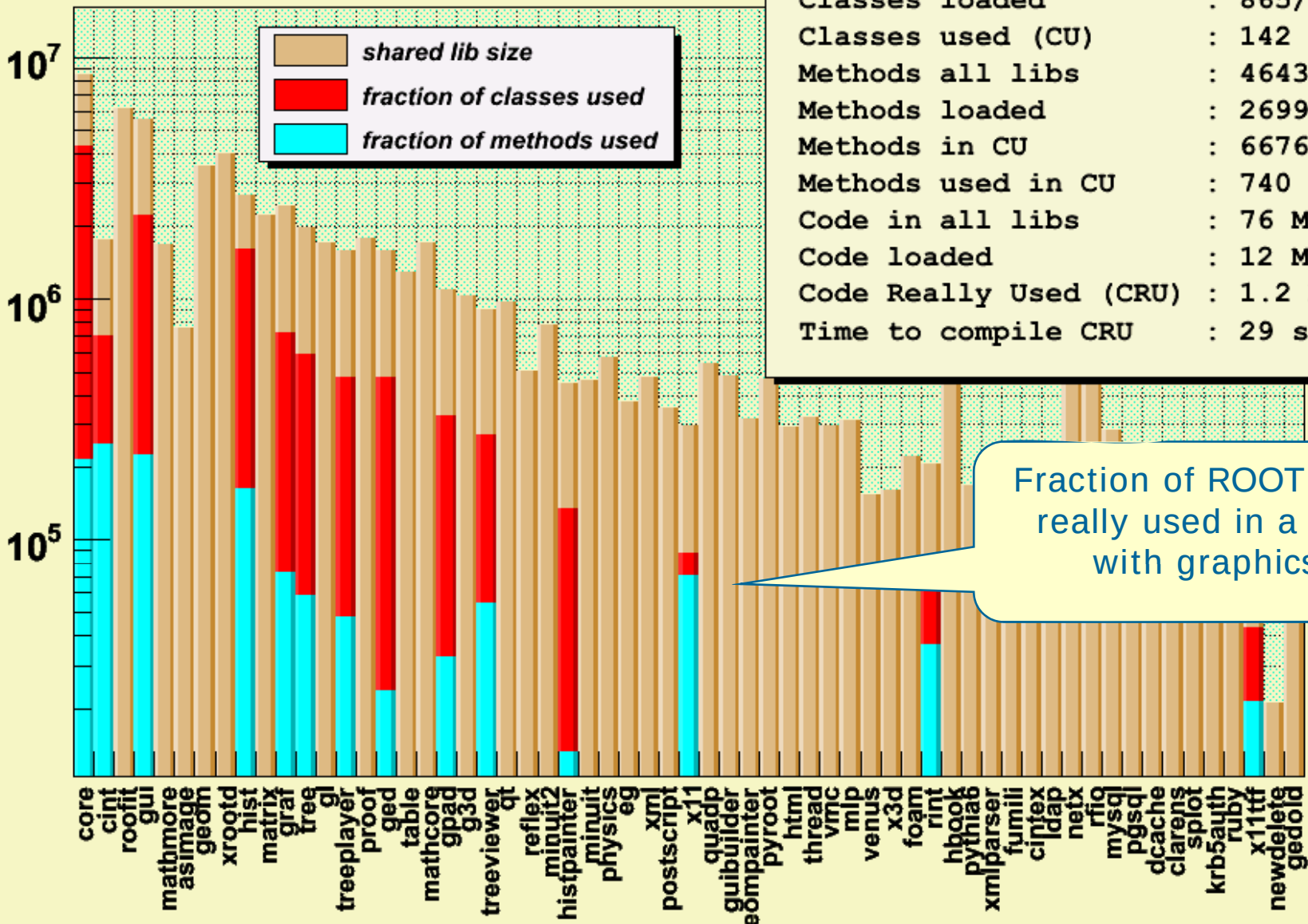


Plug-in manager

code used in a batch use case



code used in a graphics use case

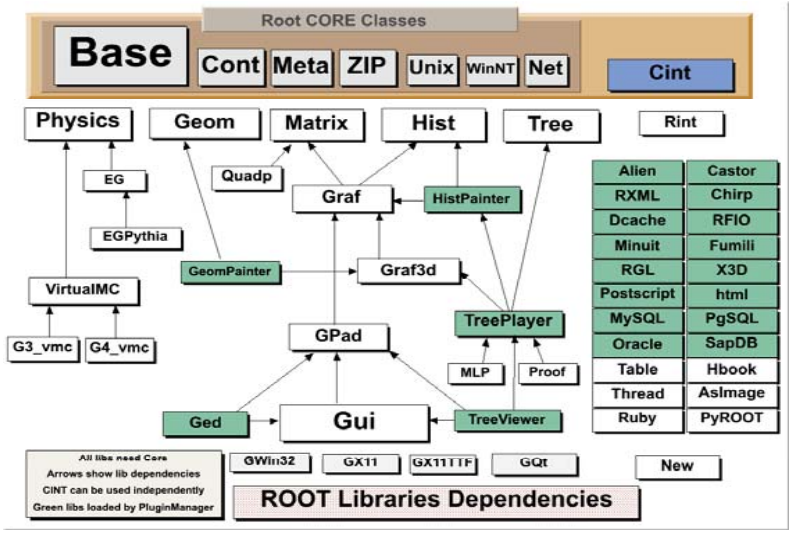


Libs used	:	14/86
Classes loaded	:	865/1459
Classes used (CU)	:	142
Methods all libs	:	46438
Methods loaded	:	26996
Methods in CU	:	6676
Methods used in CU	:	740
Code in all libs	:	76 Mb
Code loaded	:	12 Mb
Code Really Used (CRU)	:	1.2 Mb
Time to compile CRU	:	29 s

Fraction of ROOT code really used in a job with graphics

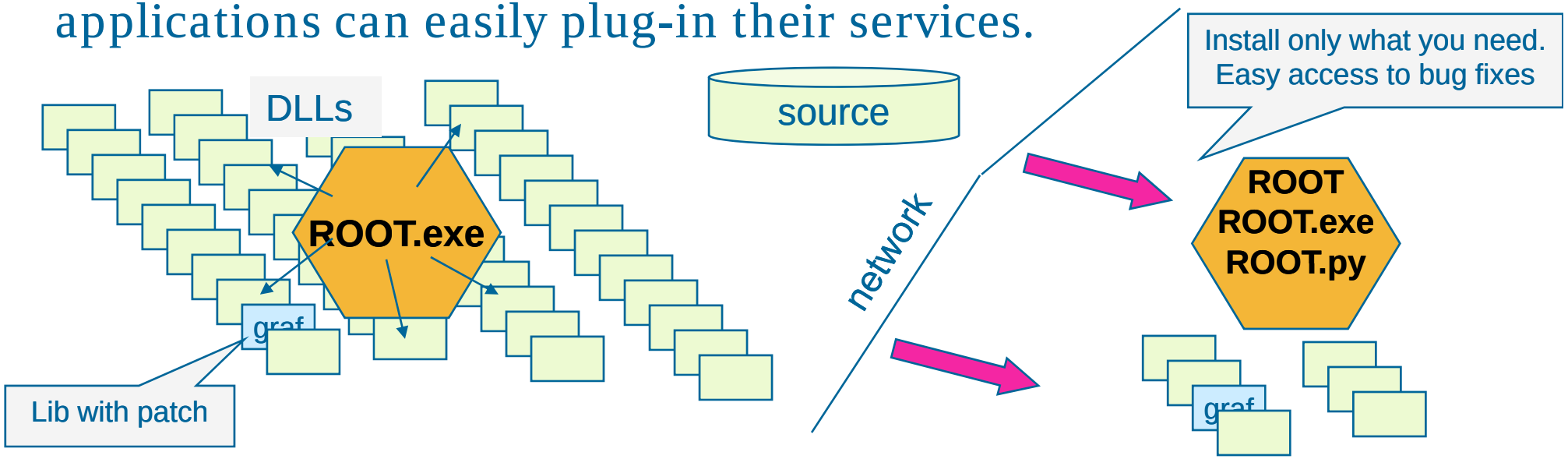
Can we gain with a better packaging?

- Yes and no. Going beyond the recent ROOT restructuring is difficult.
- One shared lib per class implies more administration, more dictionaries, more dependencies.
- 96 shared libs for ROOT is already a lot. 1500 would be non sense
- A small CORE library is essential.
- Plug-in Manager helps



Requirements for BOOT

- Reorganize the internal structure with more granularity and reduce dependencies as much as possible.
- Provide a system kernel as small as possible (code and memory) easy to install, possibly a stand-alone executable module less than a few Megabytes in memory (say <10).
- Reorganize several components (eg TBrowser) such that other applications can easily plug-in their services.



Phase 1 (nearly completed)

Libraries reorganization
Plug-in manager extensions
Improvements in dictionary size
Speed-up file access in WANs

Improving file access in WANs

- One of the BOOT requirements was to provide fast file access to files in WANs (in particular source files to be compiled locally).
- Techniques to package the source from CVS/SVN have been tested.
- Accessing sources (eg in ROOT compact binary files) requires to minimize the number of network transactions (key point on WANs).
- This requirement forced us to implement a ROOT cache that turned out to be a major improvement in I/O in general (in particular for trees) (see talk on this topic by Leo Franco).
- This improvement has required developments in I/O servers like rootd, xrootd and Dcache.

Disk cache improvements with high latency networks

- The file is on a CERN machine connected to the CERN LAN at at 100MB/s.
- The client **A** is on the same machine as the file (local read)
- The client **B** is on a CERN LAN connected at 100 Mbits/s with a network latency of 0.3 milliseconds (P IV 3 Ghz).
- The client **C** is on a CERN Wireless network connected at 10 Mbits/s with a network latency of 2 milliseconds (Mac Intel Coreduo 2Ghz).
- The client **D** is in Orsay (LAN 100 Mbits/s) connected to CERN via a WAN with a bandwith of 1 Gbits/s and a network latency of 11 milliseconds (P IV 3 Ghz).
- The client **E** is in Amsterdam (LAN 100 Mbits/s) connected to CERN via a WAN with a bandwith of 10 Gbits/s and a network latency of 22 milliseconds (AMD64 280).
- The client **F** is connected via ADSL with a bandwith of 8Mbits/s and a latency of 70 milliseconds (Mac Intel Coreduo 2Ghz).
- The client G is connected via a 10Gbits/s to a CERN machine via Caltech latency 240 ms.
- The times reported in the table are realtime seconds

client	latency(ms)	cachesize=0	cachesize=64KB	cachesize=10MB
A	0.0	3.4	3.4	3.4
B	0.3	22.0	6.0	4.0
C	2.0	11.6	5.6	4.9
D	11.0	124.7	12.3	9.0
E	22.0	230.9	11.7	8.4
F	72.0	743.7	48.3	28.0
G	240.0	>1800s	125.4s	9.9s

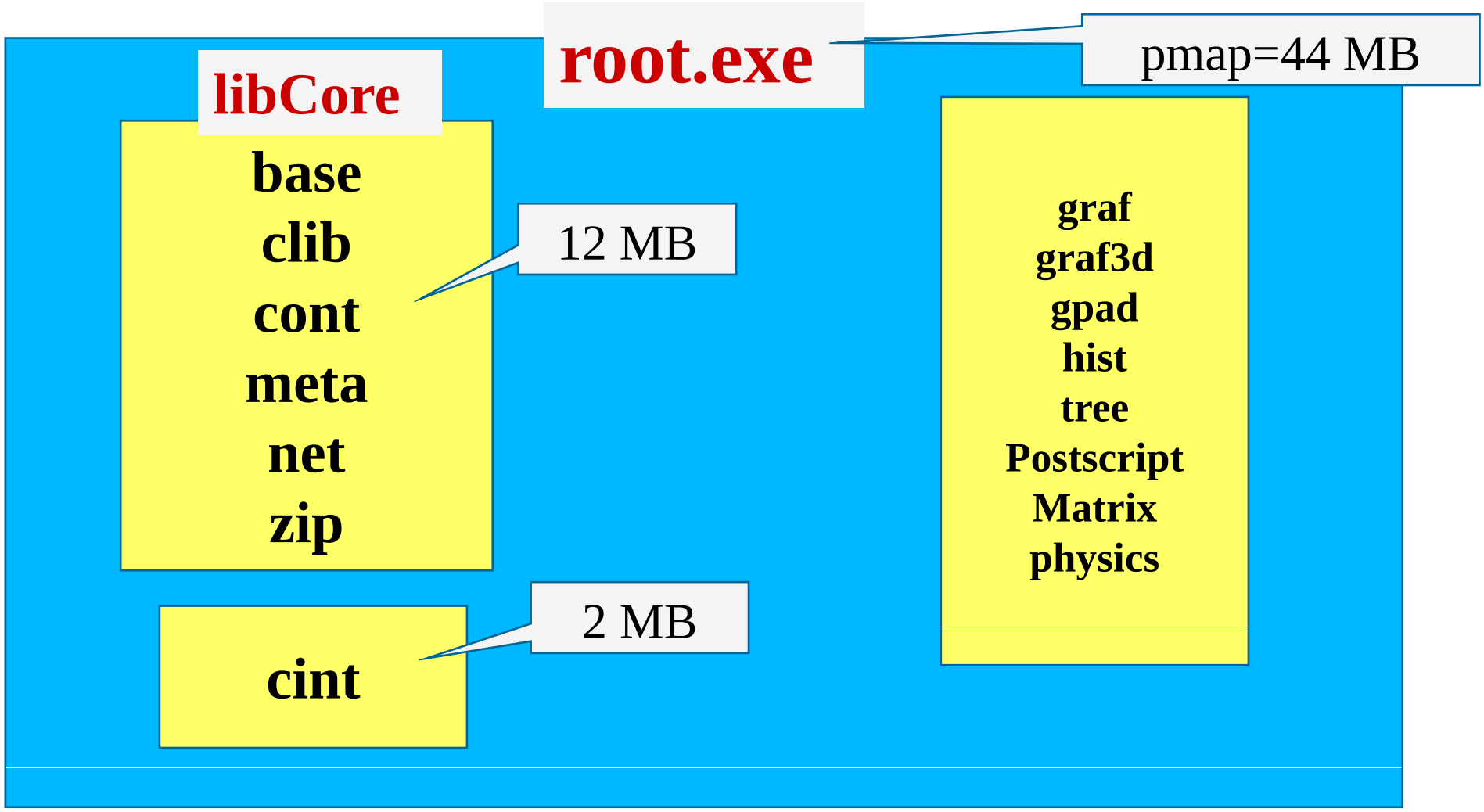
One query to a 280 MB Tree I/O = 6.6 MB

Downloading only a few classes from a main source tree is a similar problem

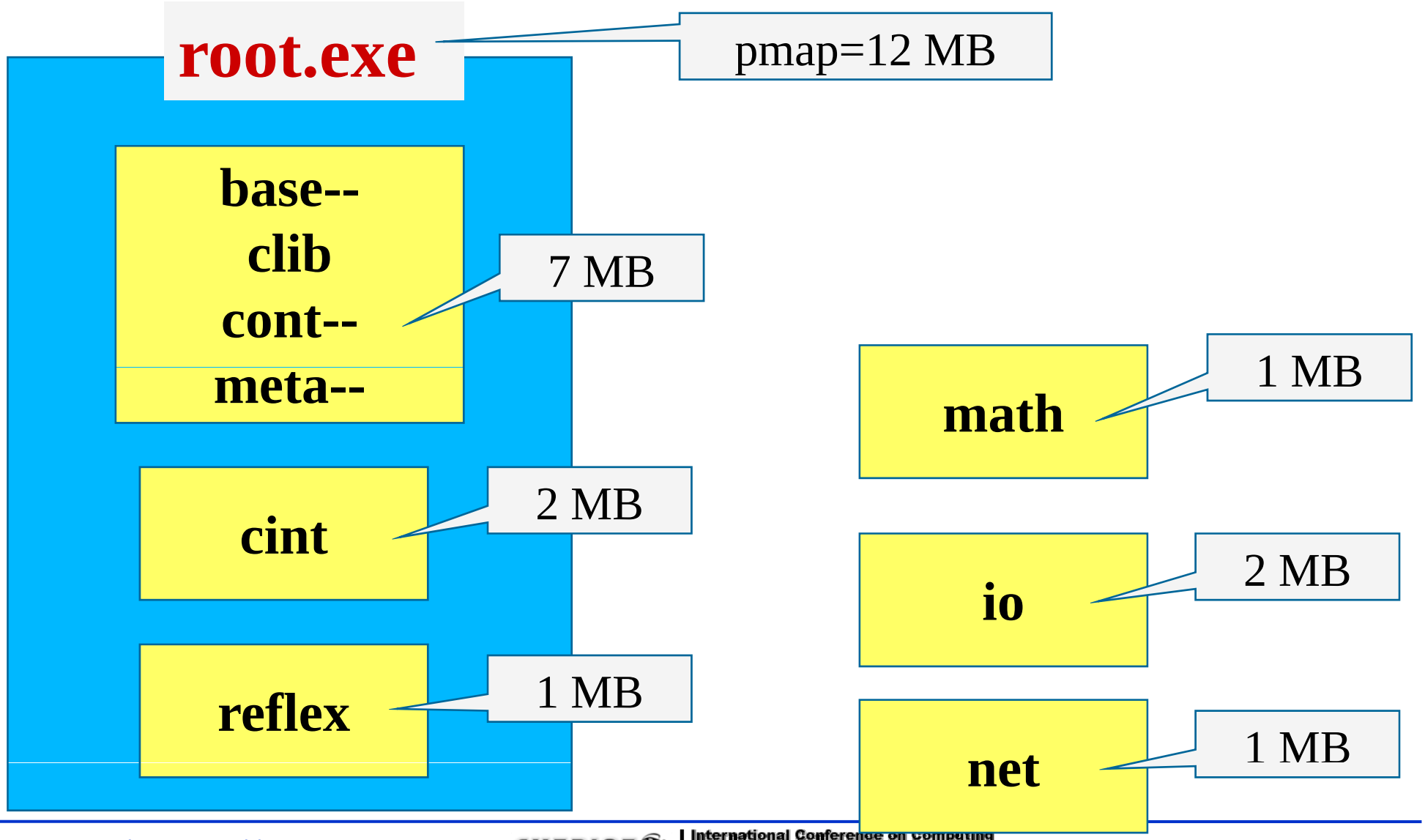
More Modularity

- Improve modularity with new subdirs “io”, “math”, ”net”
- Reduce the size of libCore
- Reduce libraries dependencies
- Reduce compilation time when changing the most important header files.
- Better documentation in sources

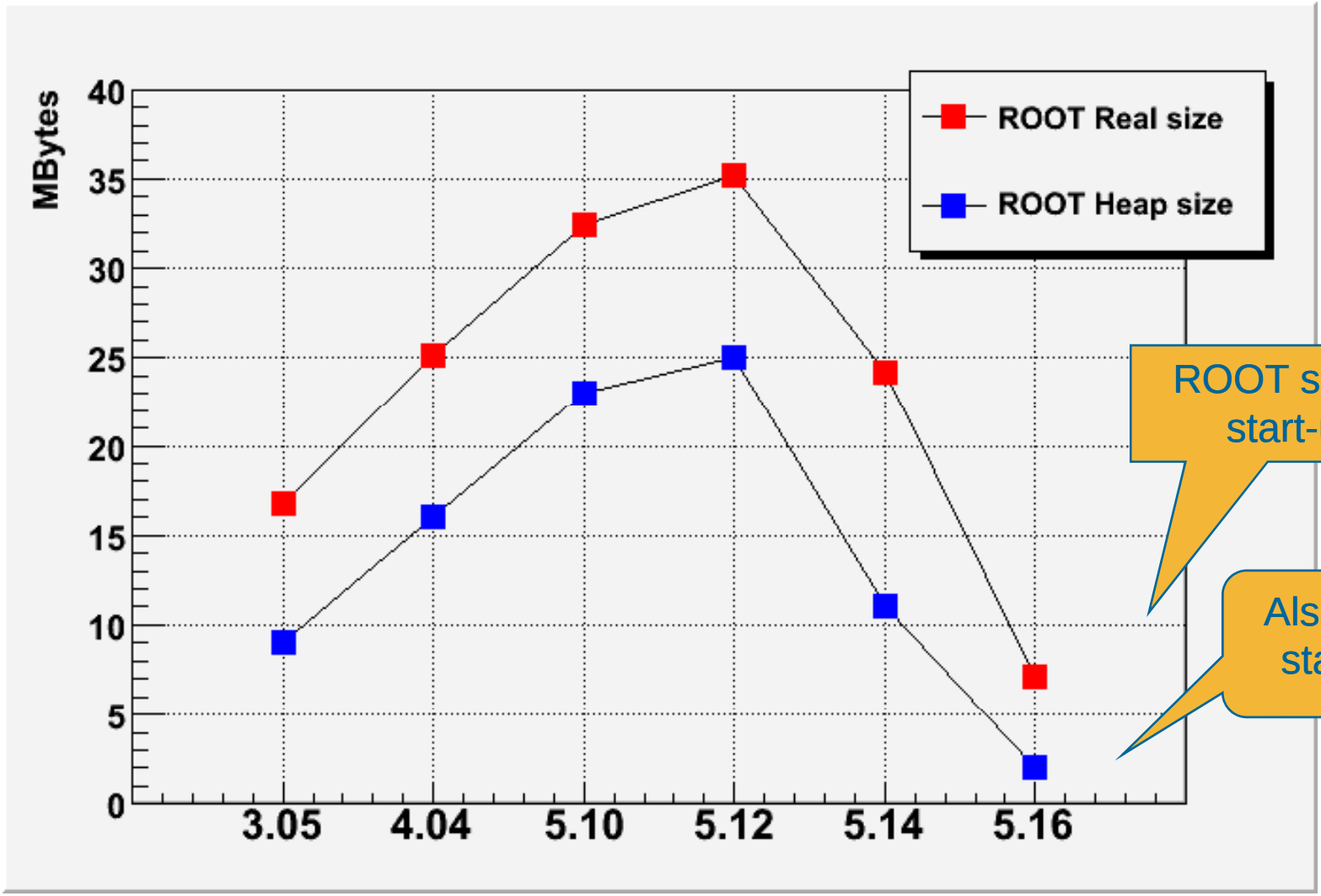
Libraries Organization (5.14)



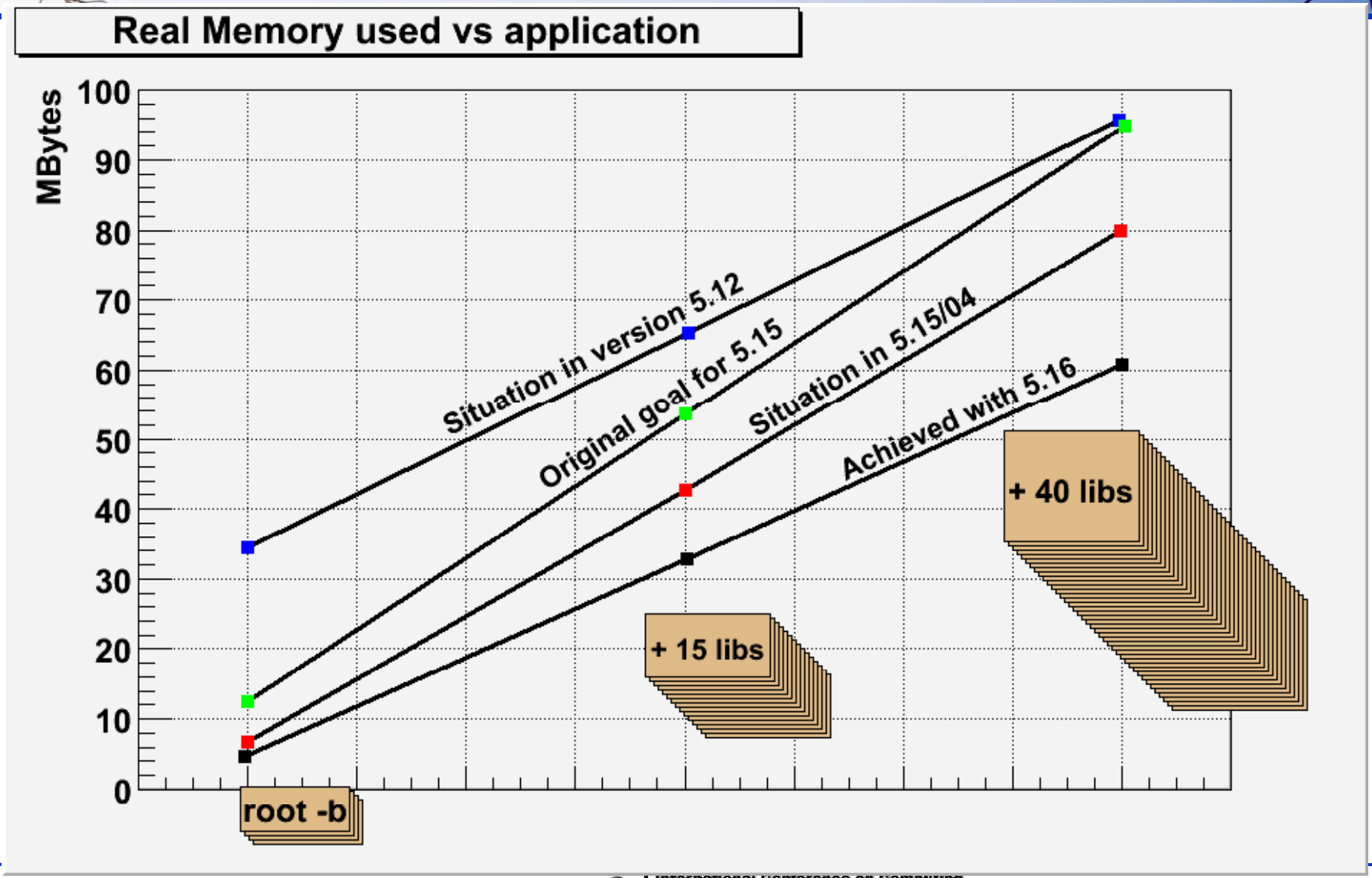
Libraries Reorganization



Large Heap Size Reduction



Substantial gain in memory



Phase 2 (completed in December 07)

More libraries reorganization

Improvements in dictionary size

Substantial reduction of code in dictionaries

A more powerful and dynamic browser

Problem with Dictionaries

- Today cint/reflex dictionaries are machine dependent.
- They represent a very substantial fraction of the total code.

	*.o	G_*.o	Dict %
mathcore	2674520	2509880	93.8%
mathmore	598040	451520	75.5%
base	6920485	4975700	71.8%
physics	786700	558412	71.0%
treeplayer	2142848	1495320	69.8%
geom	4685652	3096172	66.1%
tree	2696032	1592332	59.1%
g3d	1555196	908176	58.4%
geompainter	339612	196588	57.9%
graf	2945432	1610356	54.7%
matrix	3756632	2020388	53.8%
meta	1775888	909036	51.2%
hist	3765540	1914012	50.8%
gl	2313720	1126580	48.7%
gpad	1871020	781792	41.8%
histpainter	538212	204192	37.9%
minuit	581724	196496	33.8%

What is inside a dictionary file

- Code describing the C++ classes (inheritance, members, functions)
- Additional code for I/O: member offsets calculation, interface to class streamers.
- Stubs to call the member functions from the interpreters (CINT or Python)

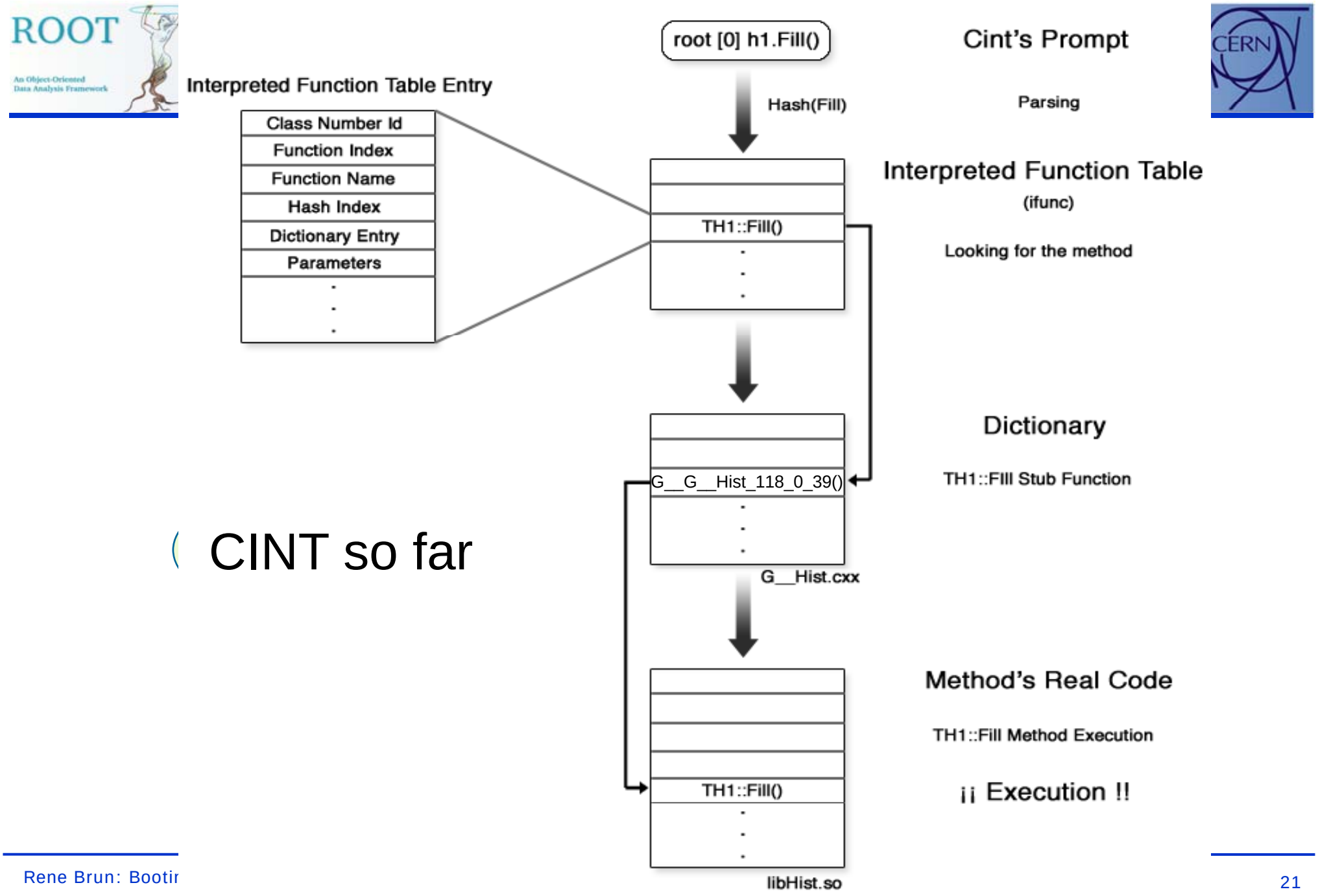
CINT Stubs

- At loading time we have one "G__memfunc_setup" call for each stub function (in a file like G__Hist.cxx).
- This function creates a new field in the G__ifunc_table with attributes like name, hash, type, parameters, etc. (Including the pointer to the stub function!)

Generated code in the dictionary... for every single method!!!

```
G__memfunc_setup("Fill",391,G__G__Hist_118_0_39, 105, -1, G__defined_typename("Int_t"),
                 0, 1, 1, 1, 0, "d - 'Double_t' 0 - x", (char*)NULL, (void*) NULL, 1);
```

```
static int G__G__Hist_118_0_39(G__value* result7, G__CONST char* funcname, struct G__param* libp, int hash)
{
   G__letint(result7, 105, (long) ((TH1*) G__getstructoffset())->Fill((Double_t) G__double(libp->para[0])));
   return(1 || funcname || hash || result7 || libp) ;
}
```


root [0] h1.Fill()
Cint's Prompt
Hash(Fill)
Parsing
Interpreted Function Table Entry
Class Number Id
Function Index
Function Name
Hash Index
Dictionary Entry
Parameters
TH1::Fill()
Interpreted Function Table
(ifunc)
Looking for the method
Dictionary
G__G__Hist_118_0_39()
TH1::FIll Stub Function
G__Hist.cxx
Method's Real Code
TH1::Fill Method Execution
TH1::Fill()
¡¡ Execution !!
libHist.so

CINT so far

Interpreted Function Table Entry

Class Number Id
Function Index
Function Name
Hash Index
Dictionary Entry
Parameters
Virt. Address Pointer
.
.
.

root [0] h1.Fill()

Hash(Fill)

Cint's Prompt

Parsing

Interpreted Function Table

(ifunc)

Looking for the method

TH1::Fill()

Dictionary

Direct Calling !!

TH1::Fill Stub Function

G__Hist.cxx

Method's Real Code

TH1::Fill Method Execution

Execution !!

TH1::Fill()

libHist.so

New Schema

Only one Function for all methods !!

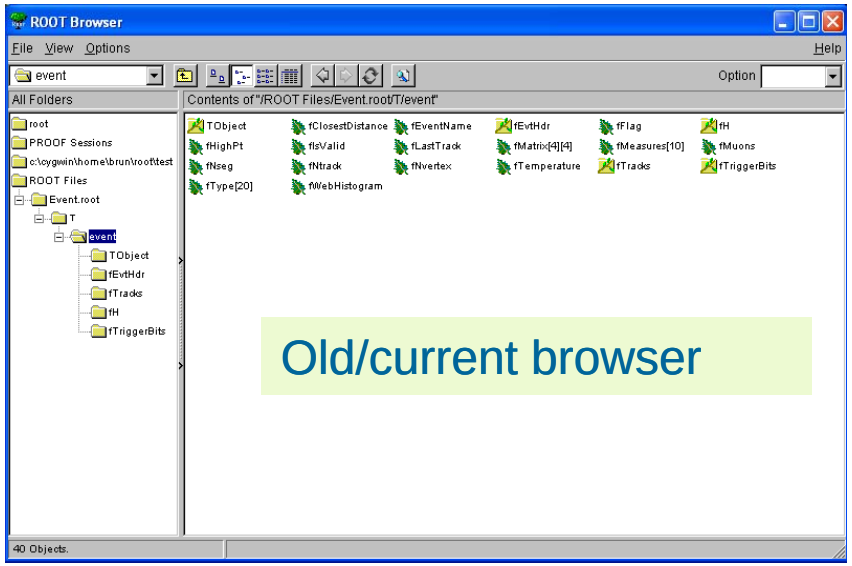
Dictionaries: Direction

- The new algorithm with automatic stubs (no code generated) has been developed for gcc only (need to implement it for vc++ and icc too).
- We would like to suppress a very large fraction of the code in dictionaries (if not all), instead storing the same information in a ROOT file.
- In this way we minimize the amount of information to be loaded in memory, and this decouple the dictionary from the real code.
- However we will still have to support the old and new schema in case of exotic compilers/linkers.

Browser Improvements

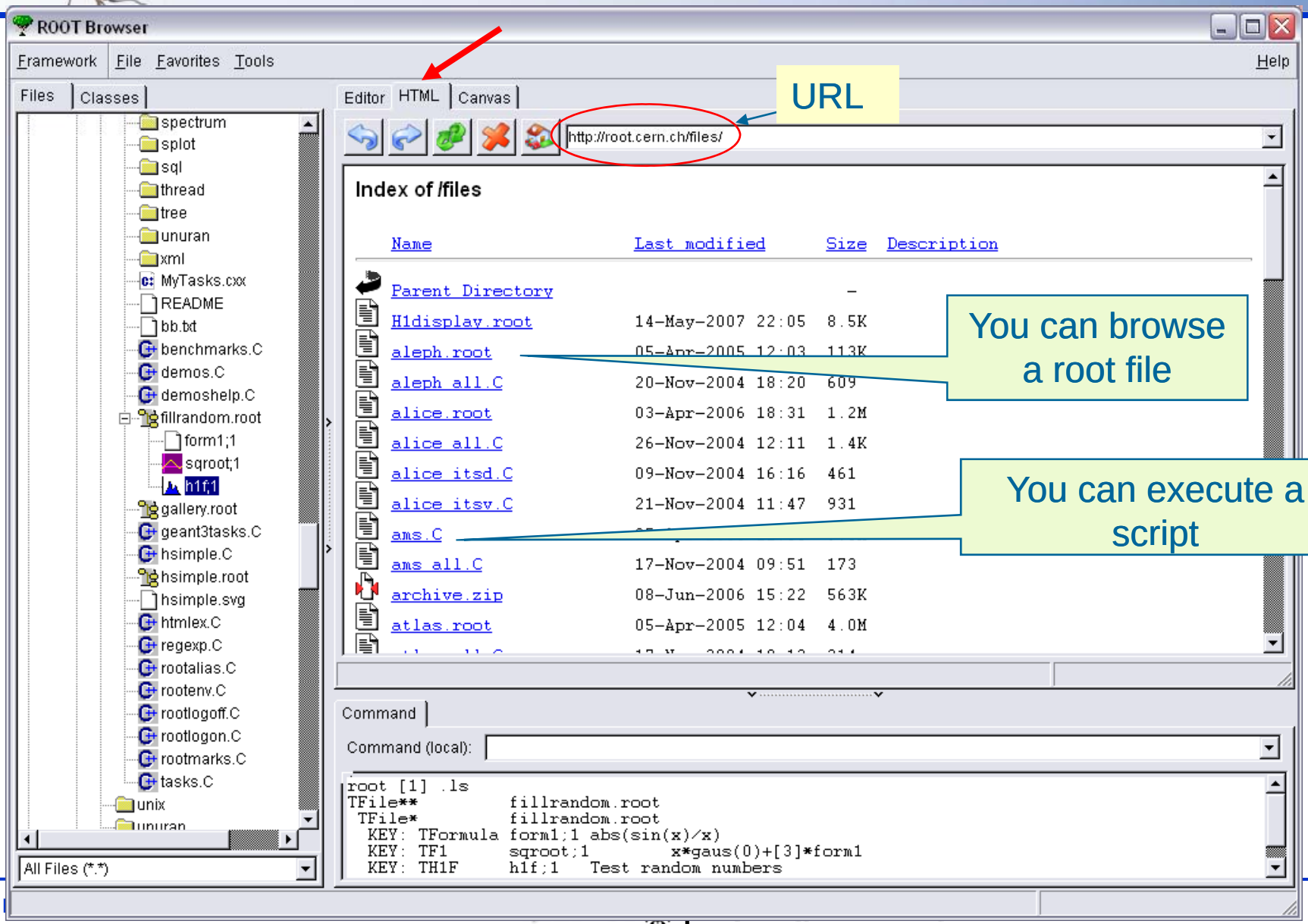
- The browser (TBrowser and derivatives) is an essential component (from beginners to advanced applications).
- It is currently restricted to the browsing of ROOT files or Trees.
- We are extending TBrowser such that it could be the central interface and the manager for any GUI application (editors, web browsers, event displays, etc).

The new browser will be an essential component of BOOT
See next slides for some examples.
Thanks to Bertrand Bellenot

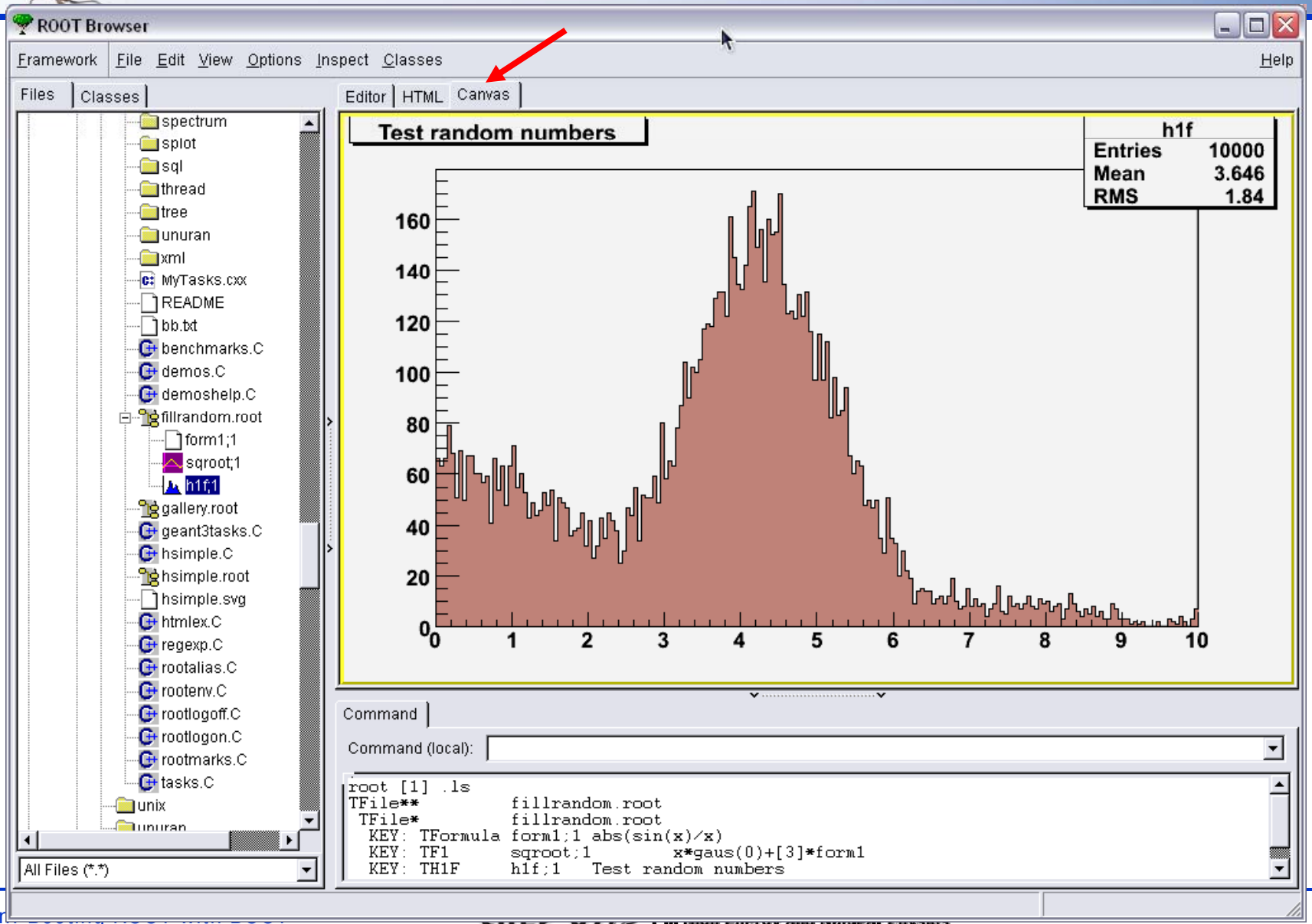


Old/current browser

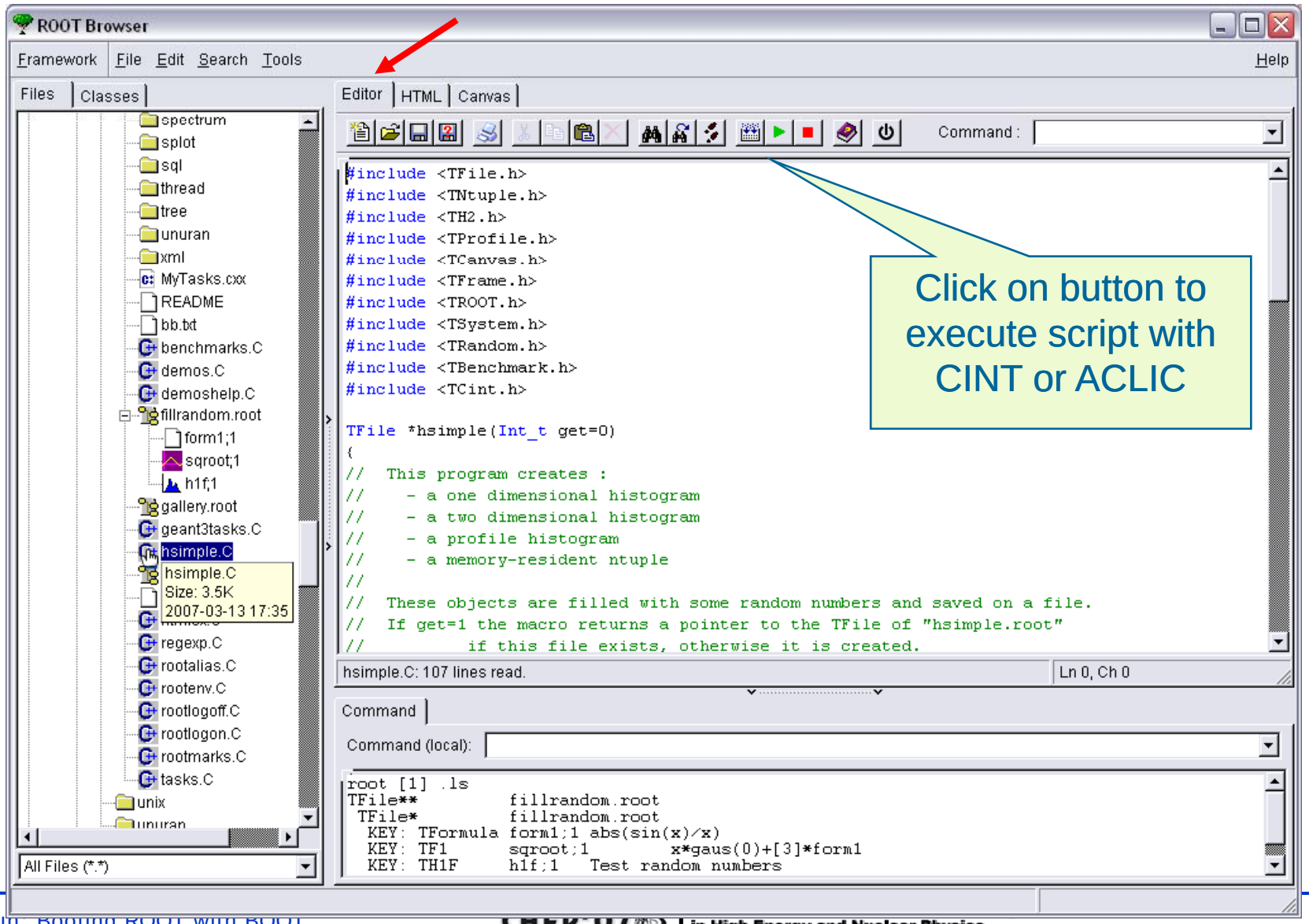
The TGhtml web browser plug-in



Hist Browser + stdin/stdout

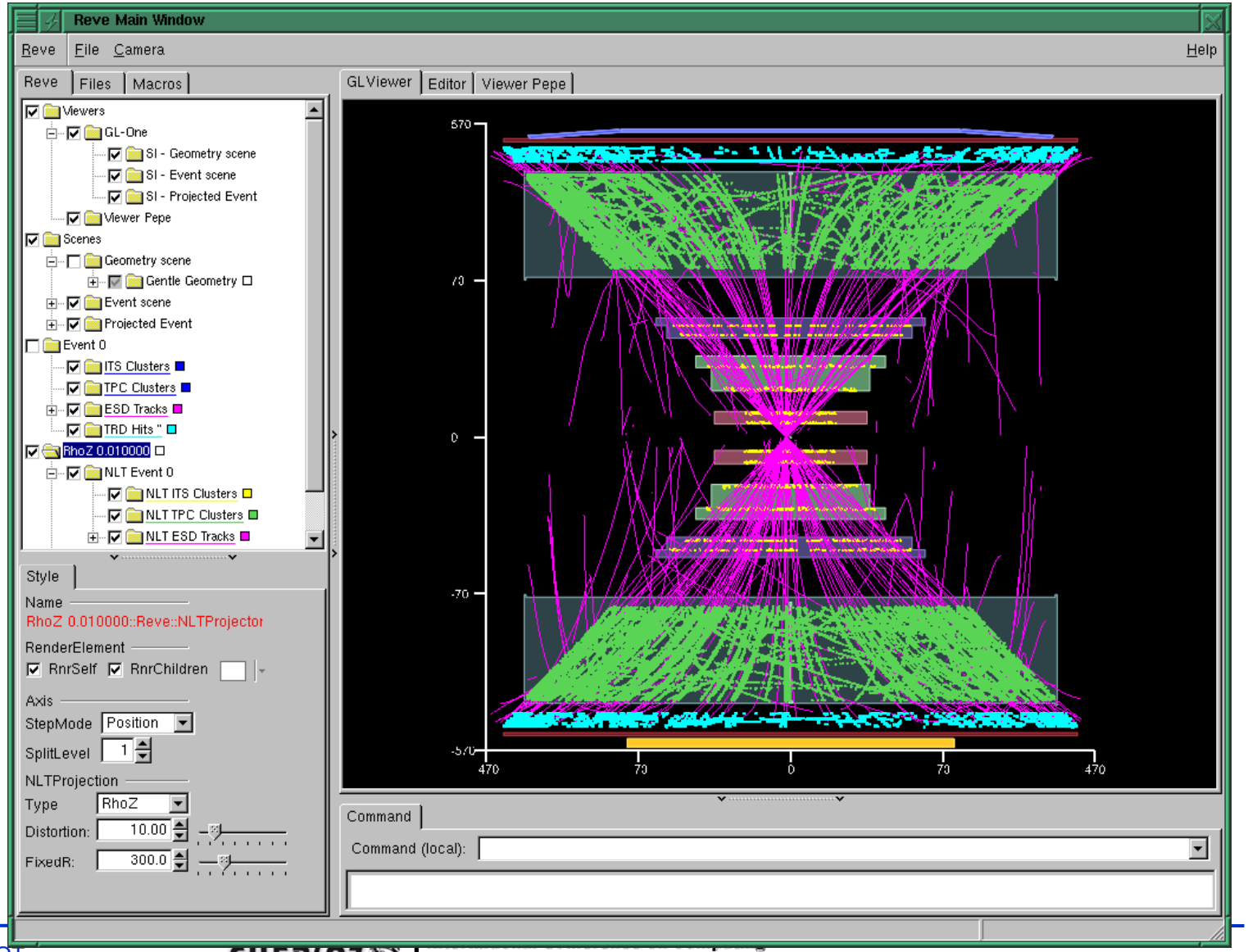


Macro Manager/Editor plug-in



GL Viewer plug-in

Alice event display prototype using the new browser (Matevz Tadel)



Phase 3 (early next year)

Substantial reduction of code in dictionaries

Improved ACLIC

A simple installation procedure

Automatic Updates

ACLIC Improvements

- The ACLIC system is becoming a class (TACLIC) .
  - root > **.x script.C**
  - root > **.x script.C++**
  - root > **.x script.C+**
- The new ACLIC will be used more and more to compile dynamically code generated on the fly (eg tree proxies).
- It could be used as a Makefile replacement in most situations with data analysis (including PROOF).
- Trade-off between compilation time and interpretation time

BOOT Installation Script

- Manage installation of the ROOT system
- Users download small script (python/perl)
  - This script then downloads the actual minimal ROOT system for the desired platform, i.e. **root.exe**
  - If binary for platform does not exist, or if platform optimized version is desired, script downloads sources and initiates compilation of root.exe
  - The script also allows the easy installation of any tagged version
- All downloaded and installed versions are stored in a local user space cache
- To avoid server overload Squid can be used for proxying and caching
- To avoid the installation of malicious code all installed material is digitally signed

Examples

```
$ wget http://root.cern.ch/root (or via web browser)
$ root --versions
Binary versions available for platform macosx:
  v5-16-00
  v5-14-00g
$ root --v5-16-00
Downloading v5-16-00…
Installing v5-16-00…
Welcome to ROOT v5-16-00 for macosx
root [0].q
$ root
Welcome to ROOT v5-16-00 for maxosx
root [0]
```

Examples (2)

```
$ root --source-versions
Source versions available:
  v5-17-01
  v5-16-00
  v5-14-00g
$ root --v5-17-01 --macosxicc  [build with Intel icc]
Downloading v5-17-01…
Compiling v5-17-01…
Installing v5-17-01…
Welcome to ROOT v5-17-01 for macosxicc
root [0].q
$ root
Welcome to ROOT v5-17-01 for macosxicc
root [0]
```

Examples (3)

```
$ root --local-versions
Local versions available:
  v5-17-01 for macosxicc  [default]
  v5-16-00 for macosx
$ root
Welcome to ROOT v5-17-01 for macosxicc
root [0].q
$ root --v5-16-00
Welcome to ROOT v5-16-00 for macosx
root [0].q
$ root
Welcome to ROOT v5-16-00 for macosx
root [0]
```

Examples (4)

```
$ root
Welcome to ROOT v5-17-01
root [0]  TTree *T = (TTree*)file->Get("T");
root[1]  TMultiLayerPerceptrom mlp("var1,var2",T);
Do you want to download the libMLP library? (y)
Downloading libMLP.
Compiling libMLP.
Installing libMLP.
root [2]  mlp.Train(10000);
root [3]  mlp.Export("example");
root [4]  .q
```

Patch/Plug-ins Management

- The same download, proxy, cache and security infrastructure will be used to manage the installation of patch versions and to announce new releases
- Patches can be installed automatically or on demand
  - ROOT patch releases will be binary compatible and not require recompilation of user code.
  - Ie, you can install only the patched shared lib.
- Instead of distributing always the almost 100 ROOT plug-ins we will use the same download infrastructure to retrieve (and optionally compile) plug-ins on demand

Summary

- We are progressing with the BOOT project:

- More modularity in splitting large libraries and taking advantage of the improved plug-in manager.
- Reduction of the size of dictionaries (memory and generated code).
- Making the UI/GUI more general with the possibility to plug-in user extensions.
- Simplifying the installation procedure and providing an automatic update procedure.