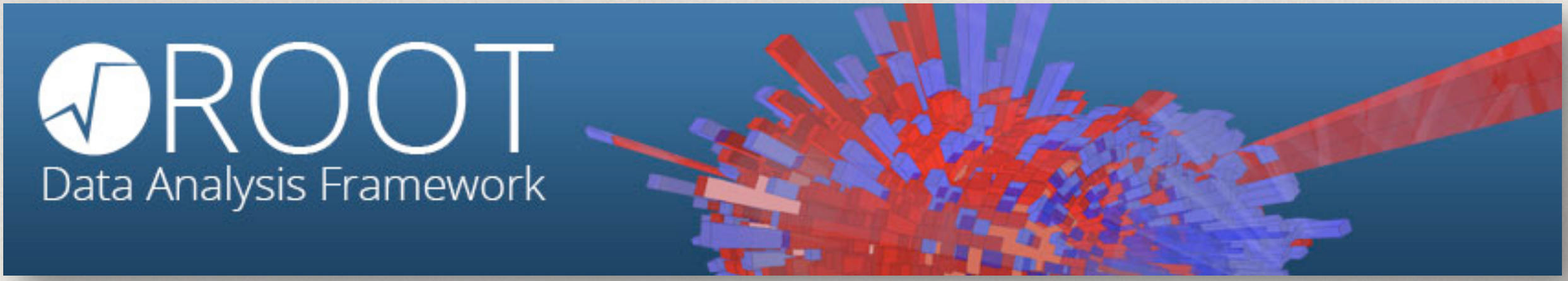


ROOT Development Roadmap

"ROOT Turns 20" Users' Workshop, 15-18 September 2015, Saas-Fee, Switzerland

Pere Mato for the ROOT team

Outline

- ROOT 6 current status
  - Completion work
- Plans and ideas beyond version 6
  - Technical and Collaboration challenges
  - Main development axes
- Conclusions

What is ROOT 6?

- An evolution of ROOT5
  - "Like before, but better"
- Old functionalities preserved, new ones added
- ROOT6 and ROOT5 are compatible:
  - Old ROOT files are readable with the 6 version
  - New ROOT files are readable with the 5 version
  - Old macros can be executed with ROOT6 (if written in proper C++, see next slides)

CLING

- Replaces CINT: a radical change at the core of ROOT
- Based on LLVM and CLANG libraries.
  - Piggy back on a production quality compiler rather than using an old C parser
  - Full support for C++11/14 with carefully selected extensions
  - Script's syntax is much stricter (proper C++)
  - Use a C++ just in time compiler (JIT)
  - A C++11 package (e.g. needs at least gcc 4.8 to build)
- Will allow support for more architectures (ARM64, PowerPC64)

What is changing for the User?

implicit “auto”

```
root [0] h = new TH1F("myhisto","thetitle",10,0.,10.)
(class TH1F *) 0x7fa89c093bd0
root [1] h.Draw()
ROOT_prompt_1:1:2: error: member reference type 'TH1F *' is a
pointer; maybe you meant to use '->'?
h.Draw()
~^
 ->
```

Proper C++

```
root [0] #include <map>
root [1] std::map<int,std::string> stringMap;
root [2] stringMap[0] = "zero";
root [3] stringMap[0]
(std::__1::map<int, std::__1::basic_string<char>,
std::__1::less<int>, std::__1::allocator<std::__1::pair<const int,
std::__1::basic_string<char> > > >::mapped_type &) "zero"[4]
```

C++

What is changing for the User?

```
root [0] auto vars = gROOT->GetListOfGlobals()
(class TCollection *) 0x7f8df2b36690
root [1] for (auto && var : *vars) cout << var->GetName() << endl;
gROOT
gPad
gInterpreter
...
```

C++11

```
root [0] auto f=[](int a){return a*2;}
(class (lambda at ROOT_prompt_0:1:8) &) @0x109b7751c
root [1] f(2)
(int) 4

root [2] #include <random>
root [3] #include <functional>
root [4] std::mt19937_64 myEngine;
root [5] std::normal_distribution<float> myDistr(125.,12.);
root [6] auto myGaussian = std::bind(myDistr,myEngine);
root [7] myGaussian()
(typename __bind_return<_Fd, _Td, tuple<> >::type) 1.344781e
```

C++11

Python without Dictionaries

A.h

```
#include <iostream>
class A {
 public:
  A(const char* n) : m_name(n){}
  void printName() {std::cout<< m_name
                    << std::endl;}
 private:
  const std::string m_name;
};
int dummy() {return 42;}
```

python

```
>>> import ROOT
>>> ROOT.gInterpreter.ProcessLine('#include "A.h"')
0L
>>> a = ROOT.A('my name')
>>> a.printName()
my name
>>> ROOT.dummy()
42
```

- Great potential with many 3rd party libraries!!

ROOT
Data Analysis Framework

Not every thing has been great

- Initial version of ROOT 6 (6.00) suffered from excess of memory utilization and slow startup time
  - Main cause: C++ PCMs no delivered on time by Clang, many headers files needed to be parsed at load time
  - Required a change in the design to minimize the number of files to be parsed: classes involved in I/O do not require their headers to be parsed
- Issue solved already in Autumn 2014
  - 6.02, 6.04 series not affected!
  - The memory requirements are comparable (+- few %) to ROOT 5 for most of the use cases.
    - Still ongoing work to understand the ALICE case.

How did we get to this point?

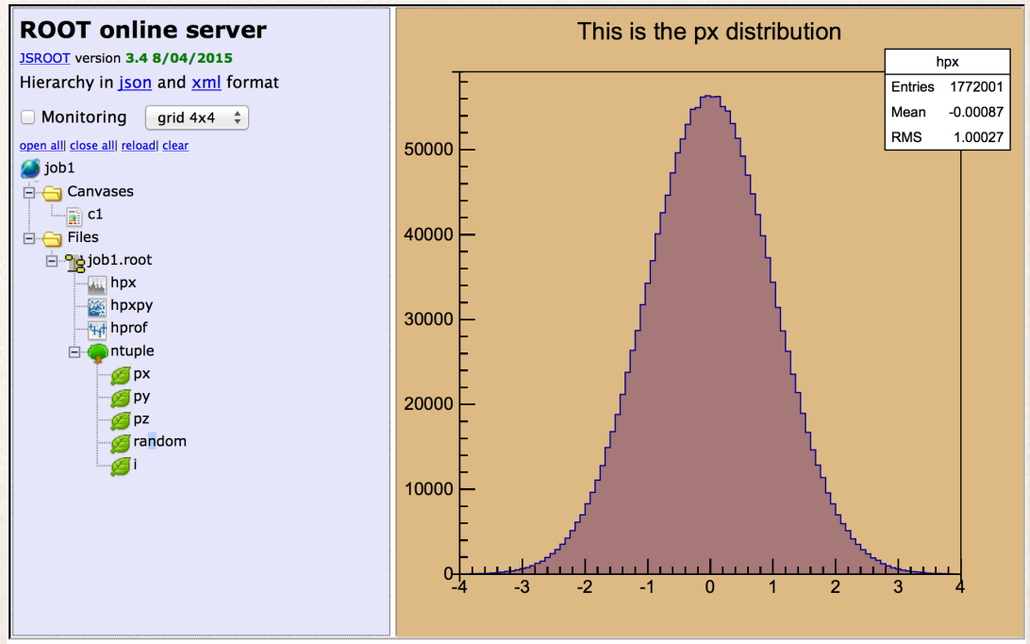
- Integration with the LHC experiments' environments during the past 18 months
  - Weekly planning meetings to report progress and issues by the experiments
- And other non-LHC experiments (FairROOT, …)
- Integrating feedback from single users since May 2014
- Some issues reported (e.g. via the ROOT forum) and already fixed
- Usual patch and development branches available
- Overall very satisfactory feedback

ROOT 6 could not possibly have reached its current state without the help of all the LHC experiments

Remote Access

- New plugin based on Davix is the default for http: file access
- JSON is now a supported I/O backend
- JavaScript library JSRootIO
  - Continue support and on-demand extension including drag and drop, context menus, etc.
- HttpServer introduced for easy web display development:
  - Can be used for custom (live) display by using javascript and JSRootIO

```
THttpServer serv;
serv.Register("abc/fold1", hpx);
// Then browse via http://localhost:8080
```



ROOT
Data Analysis Framework

Build and Testing System

- ROOT uses the **CMake** cross-platform build-generator tool as a primary build system
  - Native windows builds, support for many build tools: GNU make, Ninja, Visual Studio, Xcode, etc
  - See instructions at https://root.cern.ch/building-root
    - Classic **configure/make** will still be maintained, but it will not be upgraded with new functionality, platforms or modules.
- Unit and Integration tests (~1200) have been migrate to **CTest**
- Binary installations are packaged with **CPack**
- Nightly and Continuous integration builds are automated and scheduled with **Jenkins,** as well as all the release procedures

Version 6 Releases Timeline

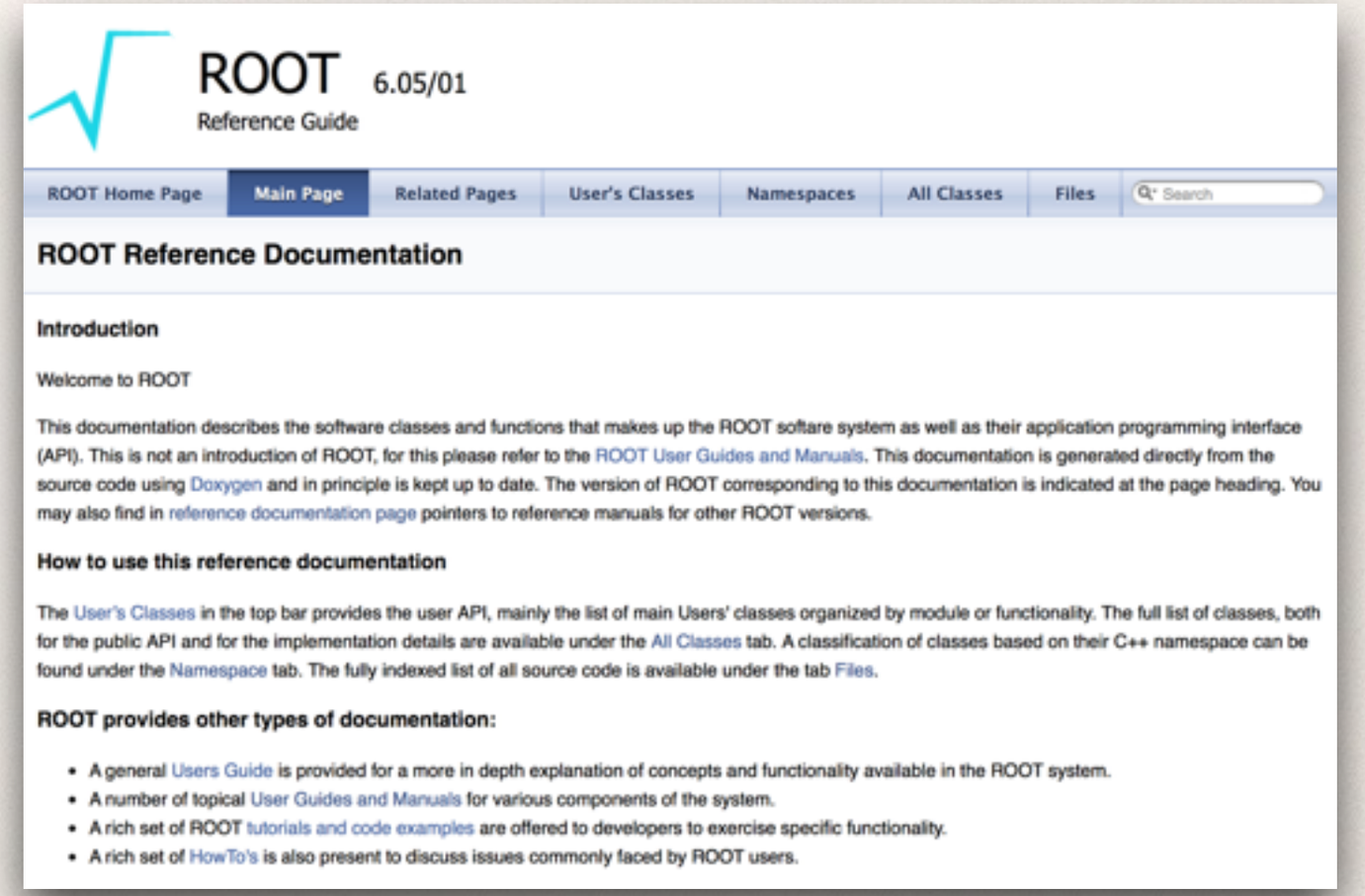
- Last ROOT workshop - technology preview
- 6.00 - May 2014
  - End-user preview for the ROOT6 features
- 6.02 - November 2014
  - Usable for the LHC experiments
- 6.04 - May 2015
  - New JIT (solving exceptions, inline assemble code, Aarch64, etc.)
  - New TFormula (leveraging from CLING)
  - ROOT-R interface
  - I/O for C++11 containers
  - New platforms on the way (Aarch64, PowerPC)

ROOT 6 Completion

- **Introduction of PCMs** (Pre-compiled Modules)
  - Minimize parsing of headers (the biggest source of extra memory consumption)
  - Avoid to need of headers deployment
  - To achieve a smooth integration of PCMs to the experiments software systems will require some work
    - Still some technical decisions to be taken
- **Windows support**
  - New versions of LLVM should work on Windows
- Aiming for completion for version 6.08 in May 2016

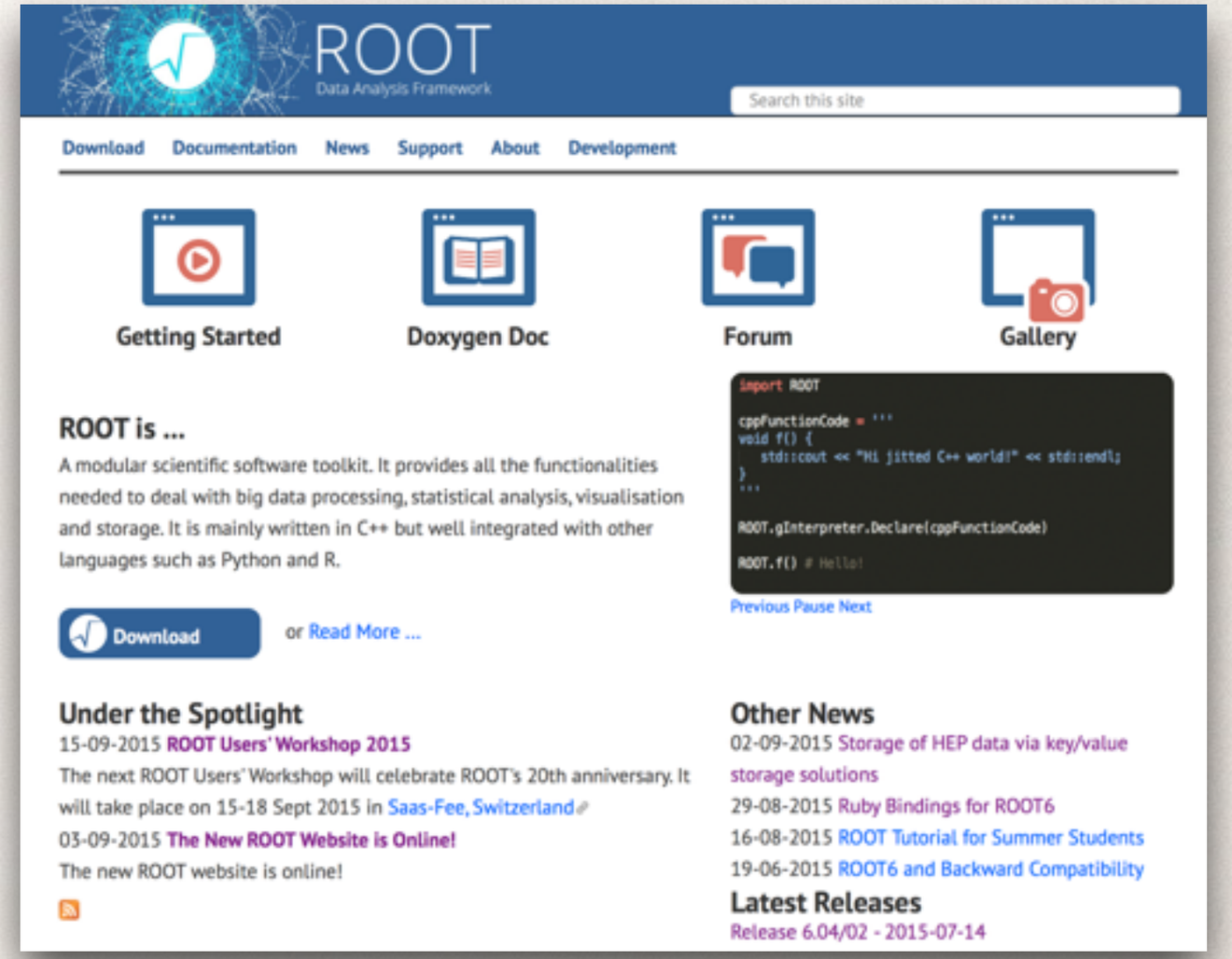
Doxygen Reference Guide

- ROOT Reference Documentation is now generated with **Doxygen**
  - **https://root.cern.ch/doc/master/index.html**
- **Work in progress!!**
- To achieve this, the comments in the source code needed to be formatted and written specifically for Doxygen to generate proper documentation.
  - Time consuming!



New ROOT Web

- ROOT website migrated to Drupal 7
  - hosted in CERN web infrastructure
- Took the opportunity to revise the content, to revise the organization and to give a new look



✎ Nefeli's presentation Thursday morning

Development Beyond ROOT 6

Technical Challenges

- ROOT is 20 years old, and some parts requires re-engineering and modernization
- Exploit modern hardware (many-core, GPU, etc.) to boost performance
- Modernize implementations (C++11 constructs, use existing libraries, etc.)
- Need to reflect on the needs and eventually solve the backward / forward compatibility

Collaboration Challenges

- ROOT is 20 years old, and requires your collaboration to ensure evolution and sustainability
- We would like to facilitate contributions to ROOT without engaging our responsibility in the maintenance and user support
  - layered software modules or plugins that can bring new functionality to the end-users
  - e.g. systems like Jenkins/Drupal/R provides a platform for developers to contribute in an easy manner
- Contributions can be on extending existing functionality at the beginning and later on the core functionality

HEP Software Foundation

- The HEP Software Foundation (HSF) aims to facilitate coordination and common efforts in high energy physics (HEP) software and computing internationally
  - sharing expertise, raising awareness of existing software and solutions, catalyzing new common projects, promoting commonality, etc.
- ROOT being one of the essential software packages in basically 100% of HEP experiments software stacks, **it must to be part of it**
- We need to better integrate all HEP software components, and ROOT is an essential ingredient
- ROOT has played the role of 'hosting' contributions that are useful to the HEP community
  - Providing build & testing infrastructure, integration, distribution, licensing, support infrastructure, etc.
  - HSF should just generalize what ROOT has been doing so far

ROOT
Data Analysis Framework

Training

- The ROOT team are preparing 3 ROOT courses for inclusion in the CERN Training Programme
  - see : https://root.cern.ch/root-training-proposal
- Basic Course
  - the interpreter, histograms, files, trees, fitting, python interface, GUI
- Advanced Analysis Course
  - RooFit, RooStats, multi-variate analysis, PROOF
- Advanced Developers Course
  - rootcore, geometry, event display, httpserver, javascript (JSROOT), ROOT as a service (ROOTaaS)

Development Main Directions

- **Cling Interpreter and its full exploitation**
  - C++11/14, JIT compilation opens many possibilities (e.g. TFormula, automatic differentiation, improved interactivity, etc.)
- **Modern C++ interfaces**
  - Explore better C++ interfaces making use if new standards (C++14, C++17)
- **Parallelization**
  - Seek for any opportunity in ROOT to do things in parallel to better exploit the new hardware (e.g. Ntuple processing, I/O, Fitting, etc.)

Development Main Directions (2)

- **Packaging and modularization**
  - Incorporate easily third party packages (e.g. VecGeom in TGeom)
  - Build/install modules and plugins on demand. Facilitate contributors to provide new functionality
- **Re-thinking user interface**
  - Explore new ways to provide thin-client web-based user interfaces
- **ROOT as-a-service**
  - Thin client plugged directly into a ROOT supercomputing cloud, computing answers quickly, efficiently, and without scalding your lap

Friendliness and Standardization

- Many interfaces can be improved in C++14,17
  - Ownership, type safe containers, string options
  - Resulting in improved user productivity
    - Dramatically reduce memory errors, wrong results, etc.
- Extent support for and more extensively use of new C++ constructs
  - std::string, std::string_view
  - std::array, std::shared_ptr, std::unique_ptr
- Gradual introduction of new backward incompatible interfaces

✎ Axel's presentation this morning

Development:Parallelization

Seek for any opportunity in ROOT to do things in parallel to better exploit the new hardware

- Re-engineer Proof-Lite or develop something new for executing parallel tasks in both **multi-process** and **multi-thread**
- Prototype solution(s) for a number of use cases:
  - Histogram/ntuple filling, TTree processing (TTreeDraw), I/O pipeline, Minimization/Fitting, etc.
- Make parallelization transparent when possible, provide user-friendly means otherwise
- Solve problems for merging efficiently the output objects produced by the parallel tasks: (histograms, trees, etc….)
- Introduce thread-safety where needed (e.g. I/O)

Multi-process

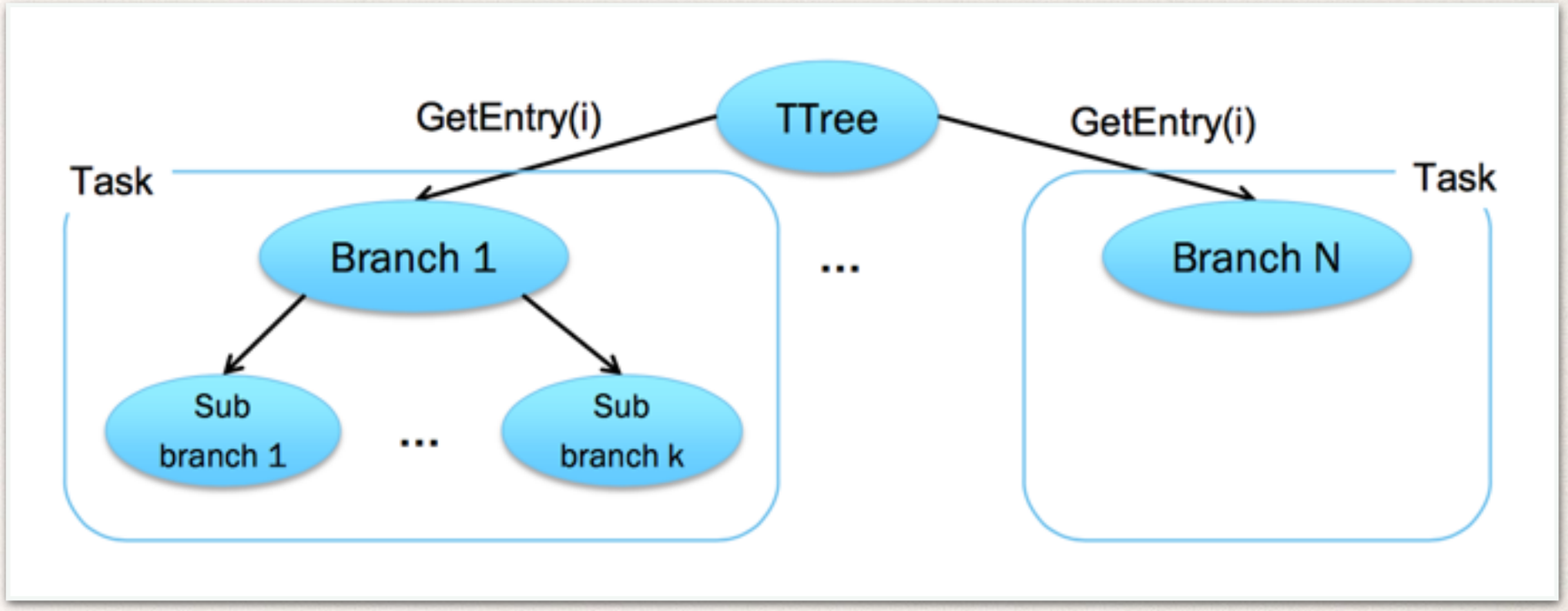
- Developing a new lightweight framework for multi-process applications
  - Inspired by the Python *multiprocessing* module
  - Idea to re-implement Proof-Lite using it
- Distribute work to a number of fork()'d *workers,* then collect results
  - Main advantage: workers have access to complete 'master' state

```
TPool pool(8)
auto res = pool.Map(
   [ ] (string f) {return myMacro("opt", 12, f);},
   {"file1","file2","file3"}
)
```

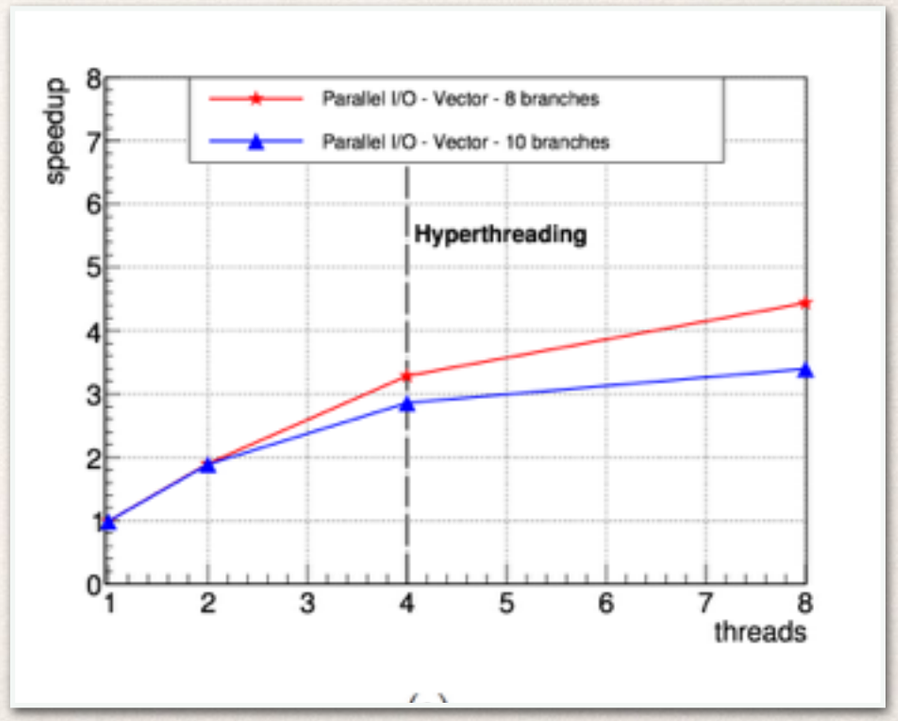
✎ Gerri's presentation on Thursday afternoon

ROOT
Data Analysis Framework

Parallel TTree Reading



- Started prototyping a parallel TTree reading using a "task programming model" (e.g. TBB)
  - speeding up the TTree:GetEntry(i)



✎ Enric's presentation on Friday morning

ROOT
Data Analysis Framework

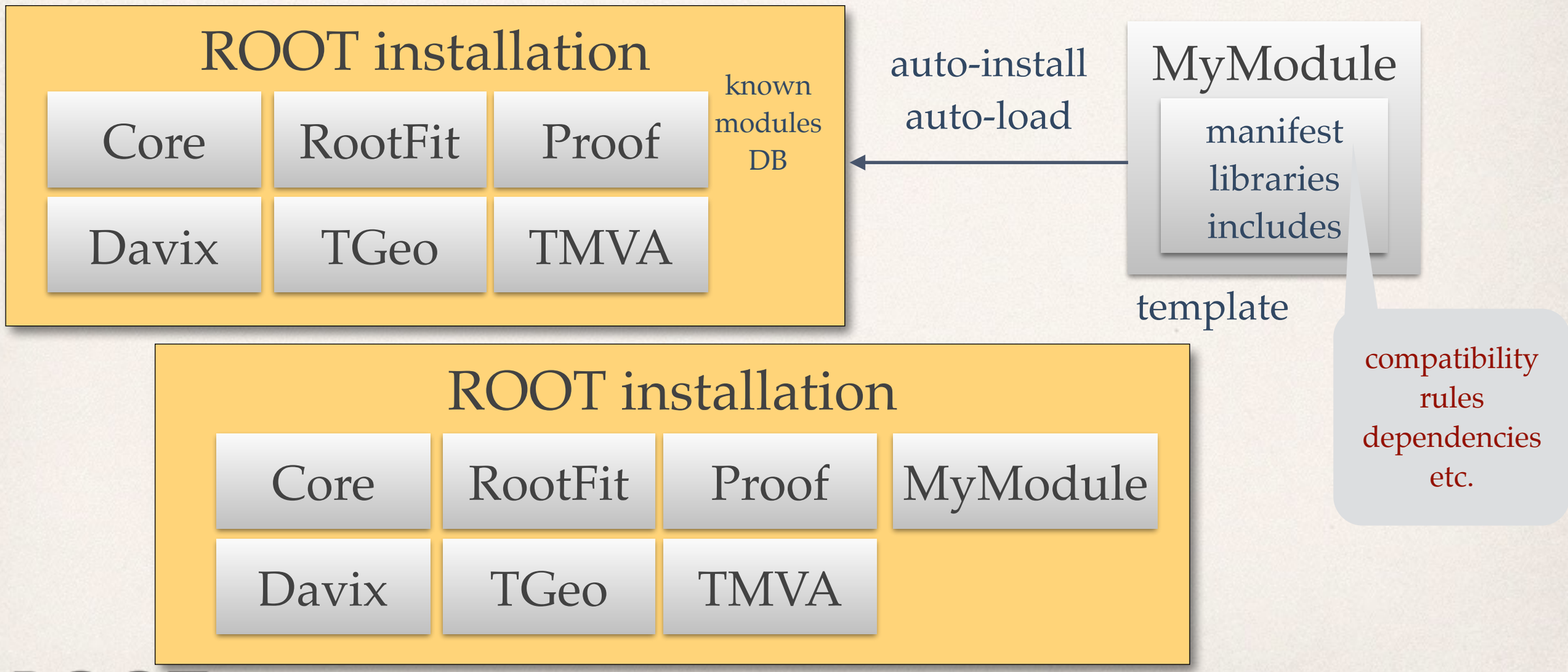
Development: Packaging

Easy use third party packages
Build/install modules and plugins on demand
Slimed down initial ROOT installation (BOOT)

- Need to incorporate new external packages in the core of ROOT
  - e.g. VecGeom, vc, vdt, TBB, new random lib, ….
  - streamlined procedures for building, testing and deploying
  - optional functionality will require external libraries to be either installed previously or be included as part of the build/installation
- Develop model for building/installing modules on demand and evolve ROOT into BOOT
  - Essential for contributors

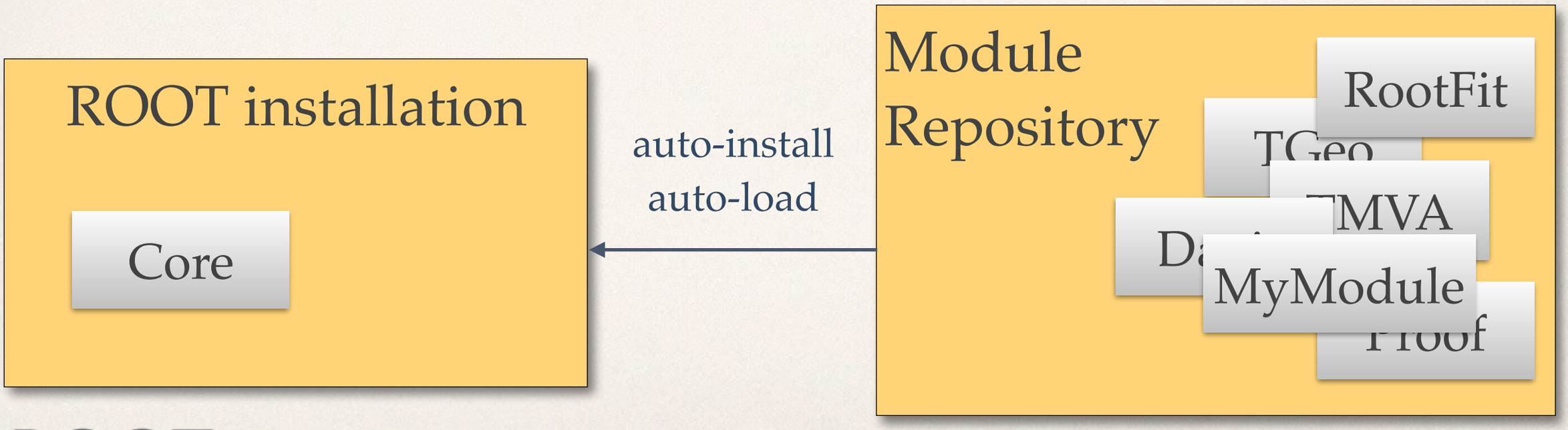
Step1: Auto-install

- Explicit or implicit installation of **known modules** when required
  - type/namespace <—> module name



Step2: BOOT

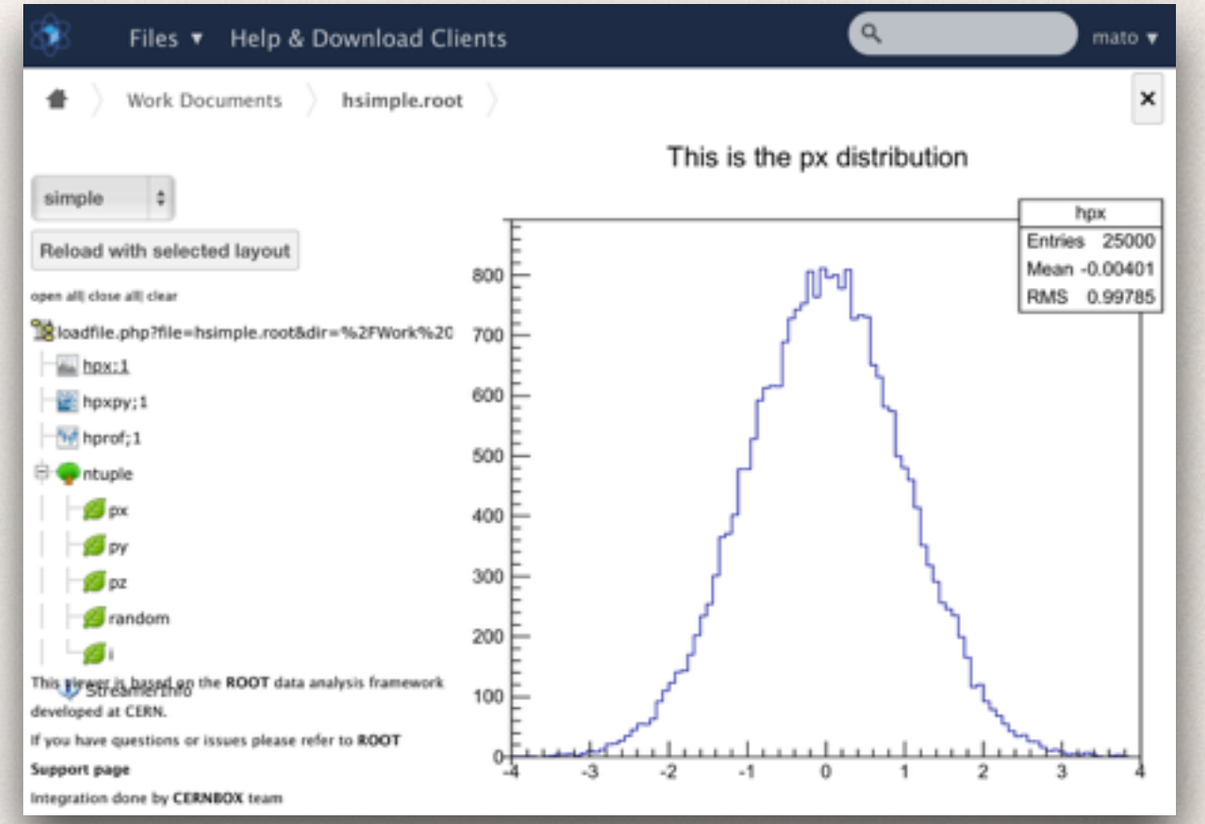
- Once the auto-install system is well understood and working adequately we could factorize the standard ROOT installation in a number of modules
  - Minimal installation and memory requirements
  - Contributors can easily provide modules



Development: Rethinking UI

Explore new ways to provide thin-client web-based user interfaces

- Increase interactivity using modern web technology (javascript) in a client-server model
  - No need to install anything in the client side
  - 3D geometry viewer
- Built on the HttpServer of Sergei Linev and JSROOT of Bertrand
- CERNBox Example

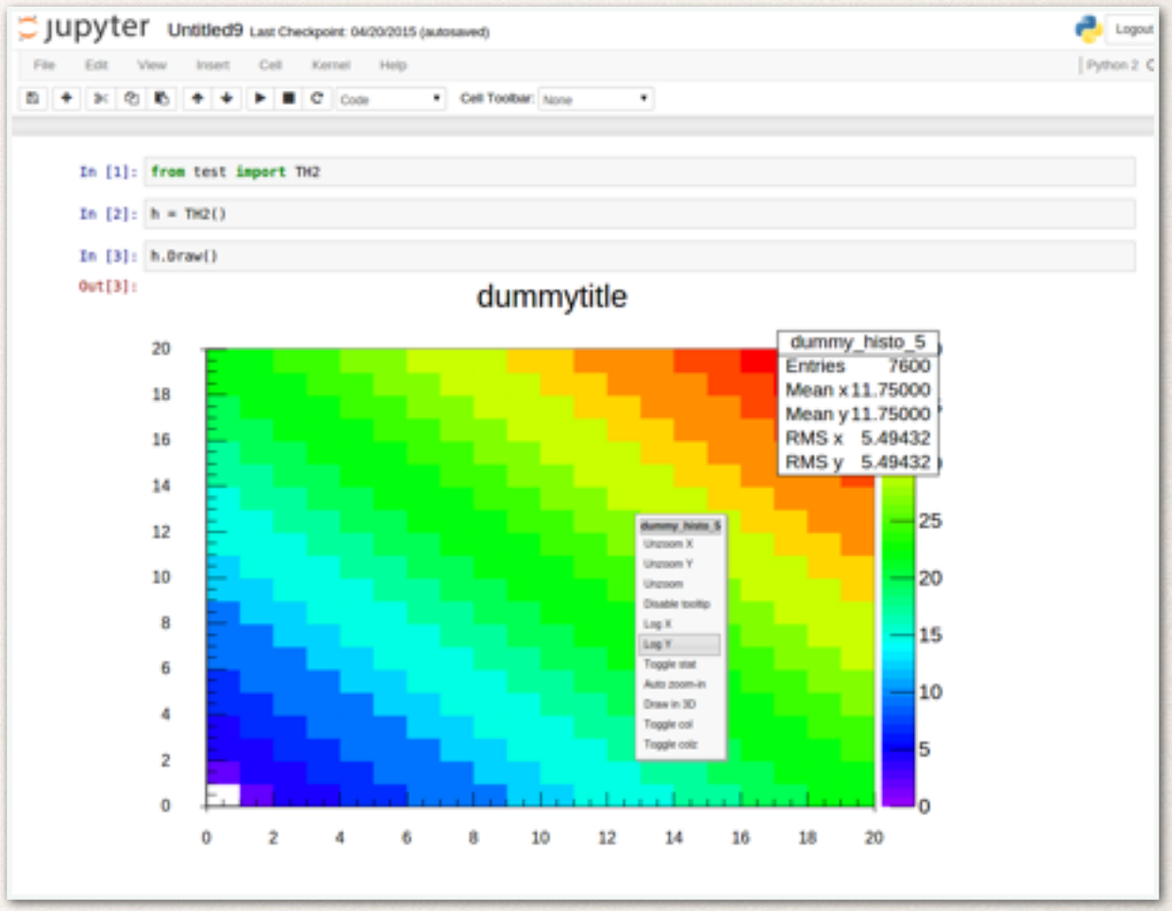


✎ Bertrand's presentation on Wednesday afternoon

ROOT
Data Analysis Framework

Exploring Jupyter Notebooks

- Jupyter offers a browser-based **notebook** with support for code, rich text, mathematical expressions, inline plots and other rich media
  - Ideal for training material
  - Possible way to document and share analysis
- Built-in client-server support
  - User 'sends commands' (python, C++) and gets objects back (textual, graphics, etc.)



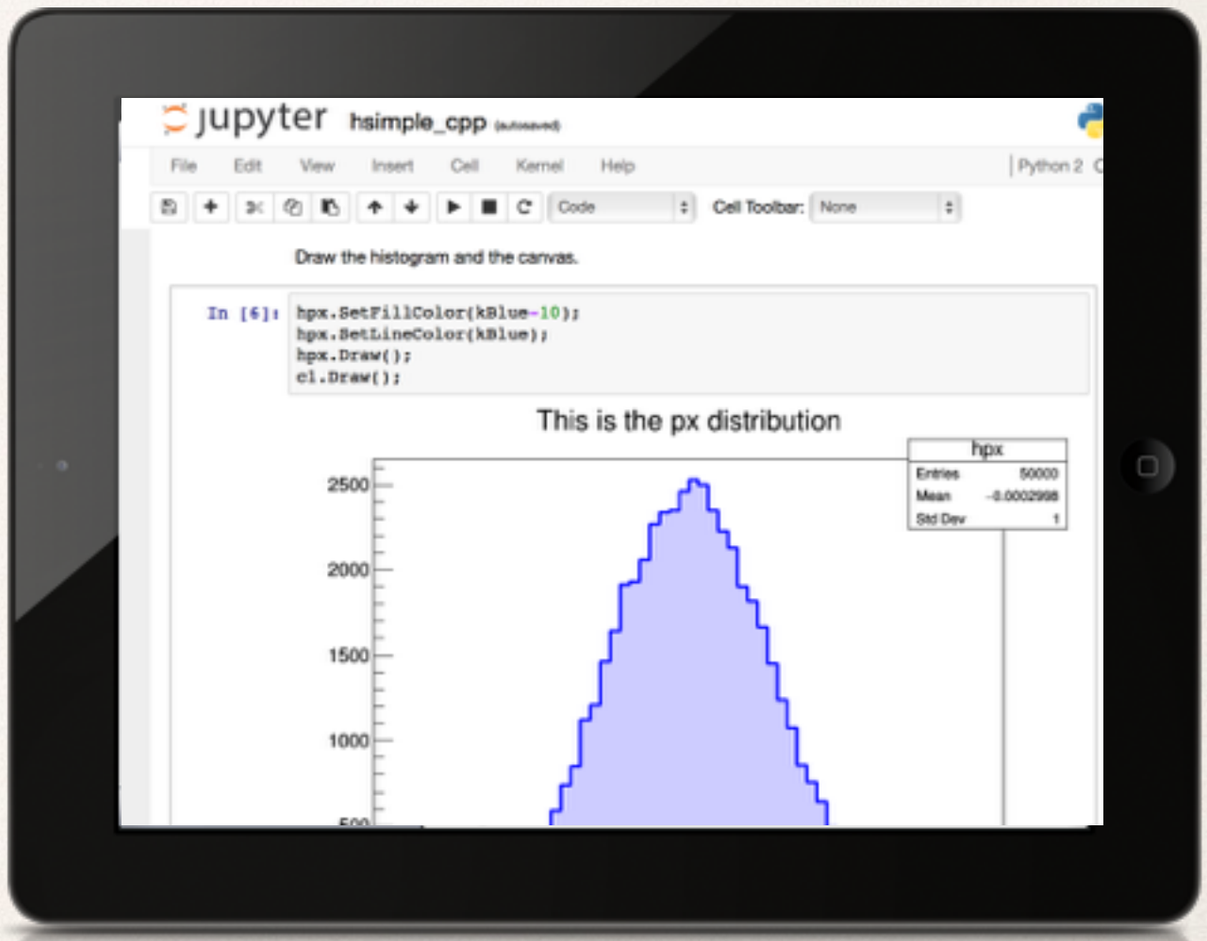
✎ Danilo's presentation on Wednesday afternoon

ROOT
Data Analysis Framework

ROOT as-a-Service

Thin client plugged directly into a ROOT supercomputing cloud, computing answers quickly, efficiently, and without scalding your lap

- Natural evolution of modern applications
- Computations run on a backend Cloud infrastructure
  - Scale on demand
  - VMs + Containers?
- User with a web-based interface
  - No local ROOT installation
- Combines the work on **parallelization** to exploit many cores and nodes together with the new **web-based interface** to provide a modern and satisfying user experience



ROOT
Data Analysis Framework

ROOT Versions

- **5.34 - current production**
  - ROOT 5 is frozen except for critical bug fixes
- 6.02
  - First functionally complete ROOT 6 version. Superseded by 6.04
- **6.04 - current production**
  - Used [or soon used] by the LHC experiments in production
- 6.06
  - Scheduled for November 2015
  - 6.05/02 - development release
- 6.08 - target for PCMs and Windows support
  - Scheduled for May 2016

Goals for this Workshop

- **Feedback on the long-term directions** introduced in the next few slides and subsequent presentations during the workshop
- **Identify collaboration opportunities** within the long list of wishes from the developers and the user community
- **What changes users would like to see** in areas such as user support, documentation, training etc.

Conclusions

- ROOT6 is used in production by most LHC experiments
- Long list of development ideas following the following axes:
  - Standardization and friendliness
  - Parallelization
  - Packaging and modularization
  - Re-thinking user interface
  - ROOT as-a-service
- Your feedback is most welcome