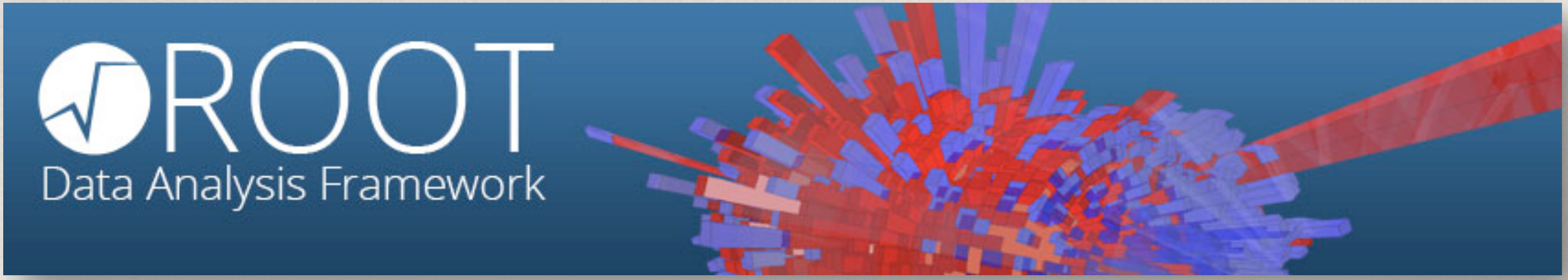


Summary of "ROOT Users' Workshop”

SFT Group Meeting, 12 October 2015

Pere Mato for the ROOT team

ROOT Users' Workshop

- The ROOT Users' Workshop took place in Saas-Fee
  - https://indico.cern.ch/event/349459/
  - ~70 participants, 45 presentations
- Goals:
  - **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.

Agenda

- 20150915_Tuesday
  - 0900_Opening_Session
    - 0910_ROOT_Development_Roadmap
    - 0955_Fons_and_ROOT
    - 1015_Rene_and_ROOT
  - 1130_Presentations
    - 1130_Highlights_and_Analysis_of_the_Answers_to_the_ROOT_Users_Survey
    - 1150_Modern_C_Interfaces_for_ROOT
    - 1400_Root-Based_Analysis_in_ATLAS
    - 1425_Web-_and_Grid_based_ROOT_analysis_and_national_ROOT_analysis_tutorials_in_ATLAS
    - 1445_Moving_CMS_developers_and_analysis_community_to_ROOT6
    - 1510_The_Data_Intensive_ANAlysis_DIANA_HEP_project
    - 1600_ROOT_in_ATLAS_TDAQ
    - 1625_Experiences_with_ROOT_for_ALICE_analyses
    - 1650_FairRoot
    - 1710_ROOT_on_C_Modules
- 20150916_Wednesday
  - 1130_Presentations
    - 0900_Writing_good_C14
    - 1100_JavaScript_ROOT
    - 1125_ROOTaaS_ROOT_as_a_Service
    - 1150_Python_bindings_for_C_using_PyROOT_cppyy_the_experience_from_PyCool_in_COOL
    - 1210_Project_Everware_running_complicated_analysis_pipelines_made_easier

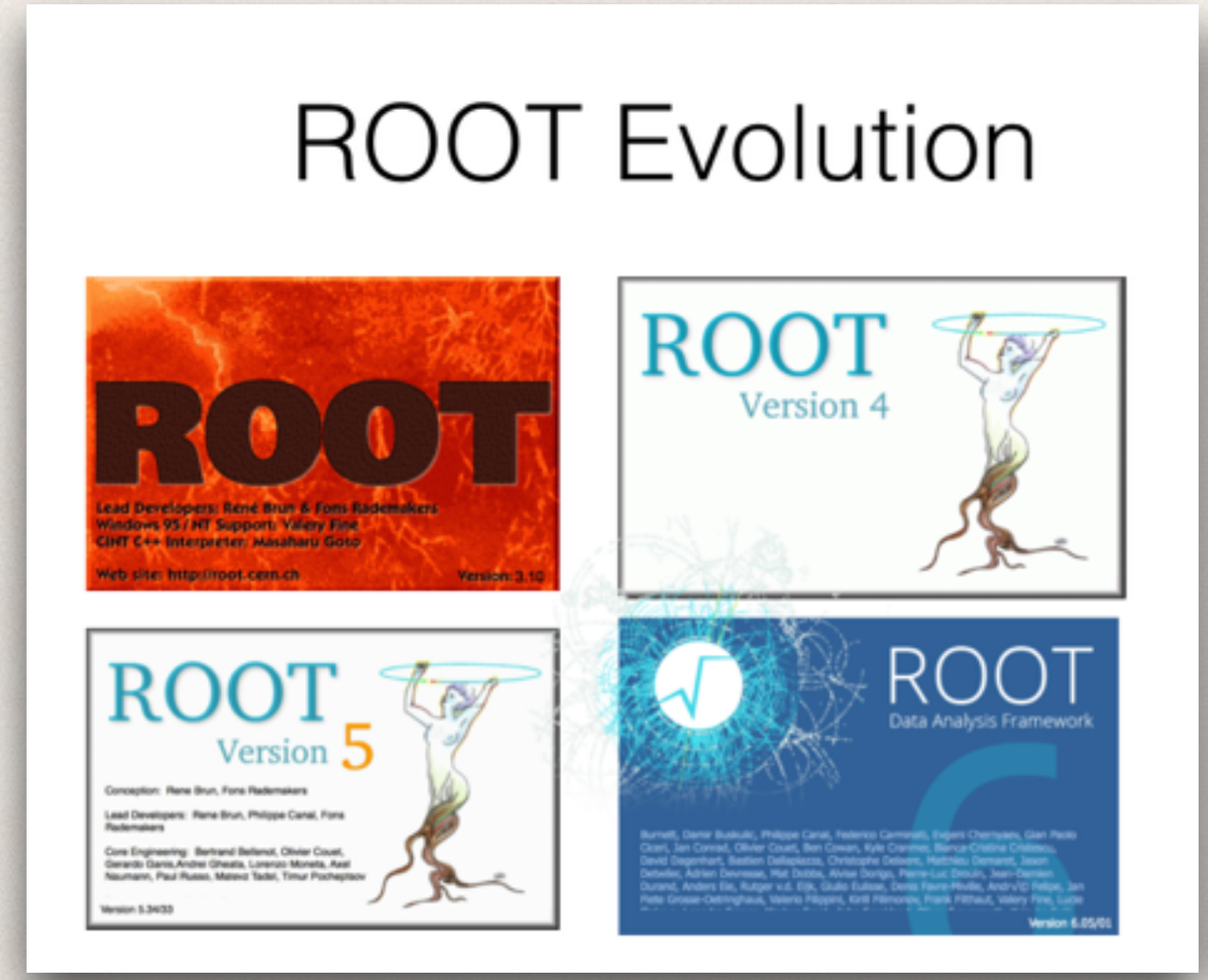
Agenda (2)

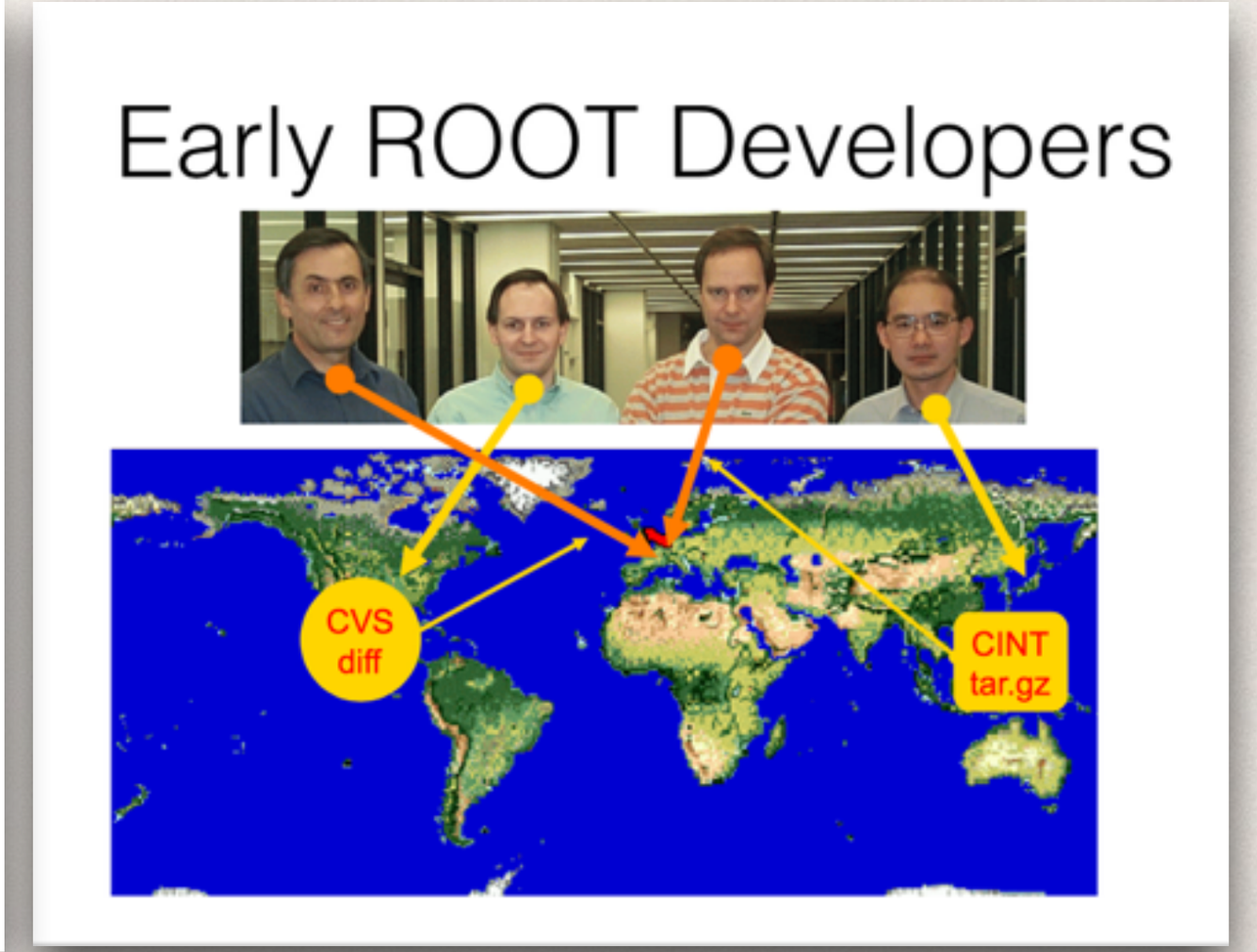
- 20150917_Thursday
  - 1130_Presentations
    - 0900_Powering_a_Player-First_Culture_with_Massive_Gameplay_Data_A_Sneak_Peek_at_Data_and_Electronic_Arts
    - 0945_Explicitly_Data-Parallel_Programming_with_C
    - 1100_Collaborative_development_of_software_and_methods_for_genomic_data_analysis
    - 1140_The_future_of_ROOT_with_R
    - 1200_Simulating_Grid_Cells_using_ROOT
    - 1400_How_to_bring_Modern_Machine_Learning_to_HEP
    - 1420_RooFit_status__development
    - 1440_TFormula_random_numbers_and_more_news_from_the_Math_Department
    - 1505_Graphics_News_new_Palettes_Transparency_Interactive_editing_LaTex_Dump
    - 1600_Evolution_of_multiprocessing_in_ROOT
    - 1620_PROOF_Analysis_Framework
    - 1640_TGeo_reloaded_-_beyond_the_legacy
    - 1715_DD4hep_a_Detector_Description_Solution_for_High_Energy_Physics_Experiments
- 20150918_Friday
  - 1130_Presentations
    - 0900_ROOT_I_O_Status_and_Perspectives
    - 0930_First_experiments_with_TTree_I_O_parallelisation
    - 0950_XRootD_and_ROOT_Considered
    - 1010_Analyzing_LHC_experiment_software_in_terms_of_obsolete_memory_utilization_with_a_focus_on_ROOT_objects
    - 1100_THttpServer_class_in_ROOT
    - 1120_Packaging_ROOT_for_Fedora_and_EPEL
    - 1140_Go4_Version_5_-_a_ROOT_based_online_and_offline_analysis_environment
    - 1200_Julia_a_fast_dynamical_language_for_technical_computing_and_data_analysis
    - 1220_Event_Visualisation_Environment_of_ALICE
    - 1400_ROOT_and_NASA
    - 1420_ALFA_Next_generation_concurrent_framework_for_ALICE_and_FAIR_experiments
    - 1440_The_Belle_II_Experiment_ROOT_6_at_the_High-intensity_Frontier
    - 1500_Object_oriented_data_analysis_at_the_BGO-OD_experiment

Outline

- 20th Anniversary
- User Feedback
- ROOT Current Status
- Ongoing Developments
- User Support
- New Ideas

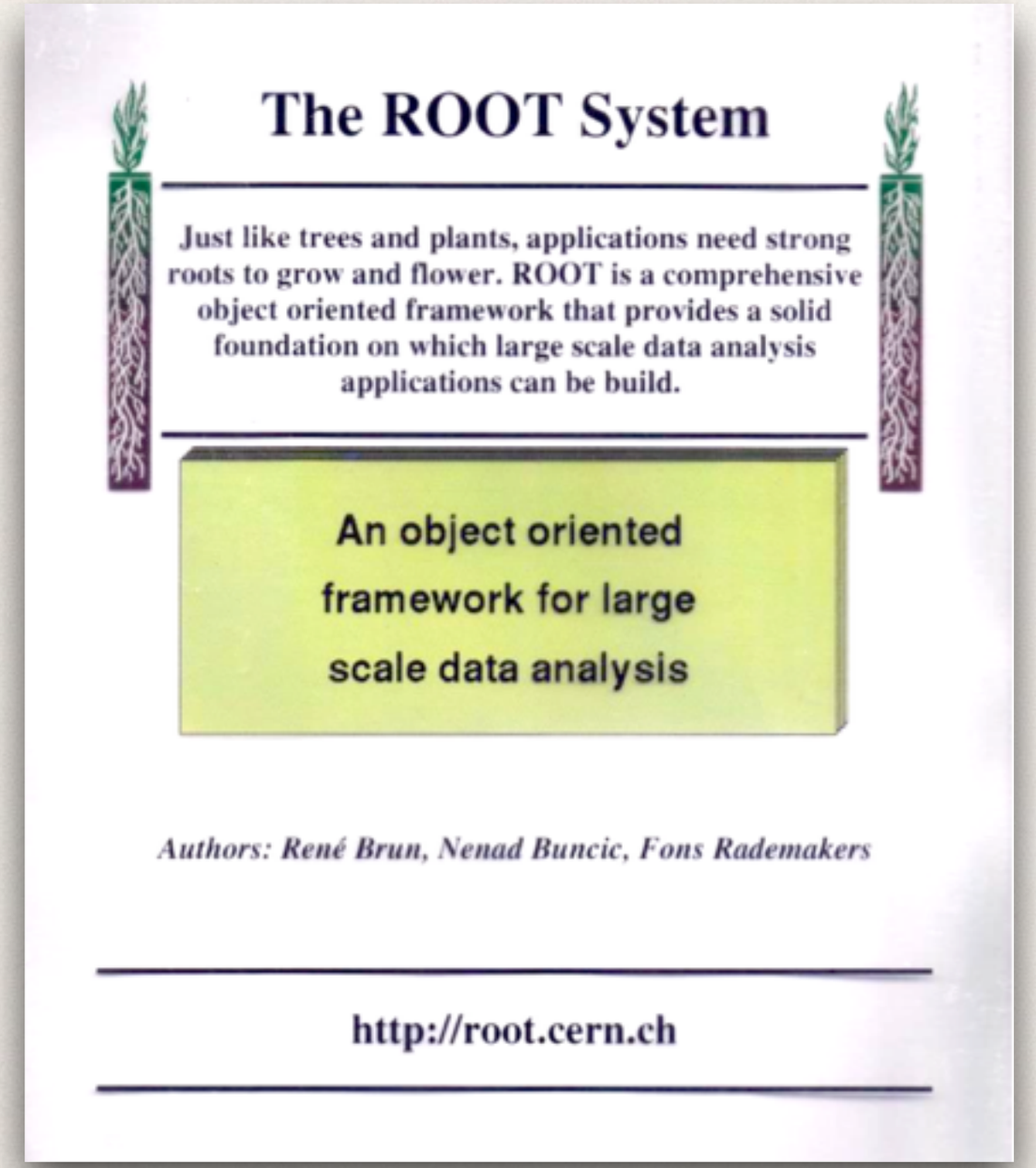
20th Anniversary Presentations

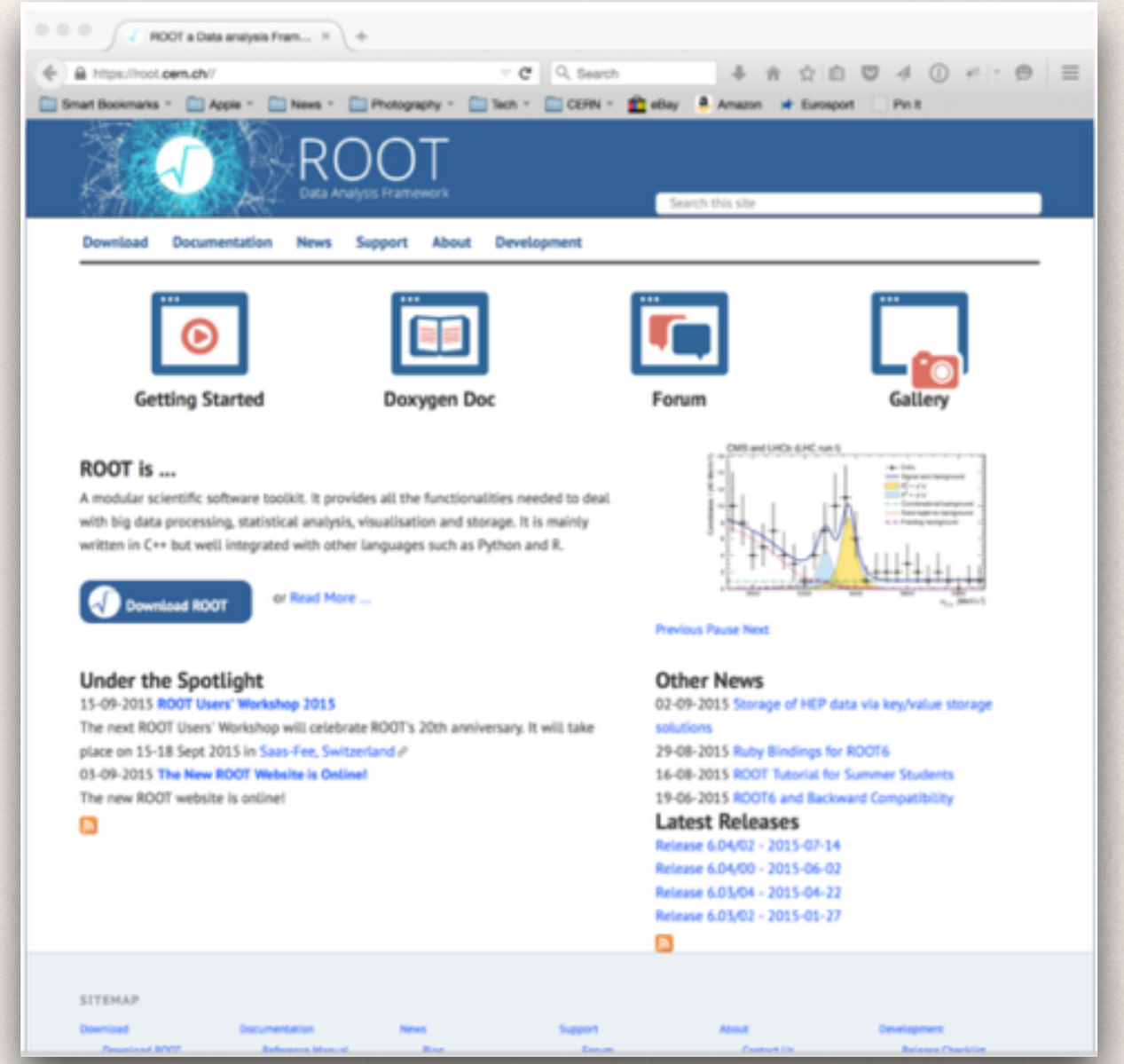




Rene Brun, Fons Rademakers

First web page (1995) to last (2015)





Rene Brun, Fons Rademakers

Technology and User Support

Technology Evolution

- Version control systems:
  - CMZ -> CVS -> Subversion -> Git
- Build systems:
  - CMZ -> configure;make (non-recursive makefile) -> cmake
- Documentation system:
  - MS Word -> DocBook -> Markdown
- Website technology:
  - Plain html -> CSS and html -> Drupal
- Continuous integration system:
  - Electric Commander -> Jenkins

Focus on the User, User, User

- ROOT's success mainly due to prompt and courteous user support
- Solicit user feedback via
  - Mailing lists
  - Web fora
  - Bug reporting system
  - Private e-mails
- React to any form of feedback within minutes

Fons Rademakers

ROOT
Data Analysis Framework

ROOT Users Survey

Profile of respondants

3

- 353 people filled in the questionnaire – thank you!
- Experience: <2y: 4% 2-5y: 28% 5-10y: 37% >10y: 30%
- Frequency: every day 73%; several times/week 25%
- Essentially all HEP research programmes are represented
  - LHC, ν physics, b physics, …
- Nuclear and plasma physics
  - RHIC, GSI,…
- Astronomy and astrophysics
  - Fermi-LAT, Fermi-HAWC, HESS, MAGIC, P

Comments on support

8

- Most users are either fully (45%) or partially(45%) satisfied
  - Usually I find answer to my problem on forum
  - I use forum a lot but almost never need to post questions
- BUT 5-10% are clearly not happy
  - 'Improve the documentation'
  - 'I wish tutorials were kept up-to-date and were more relevant to my needs'
  - 'It can take very long from bug reports to fixes'
  - 'Some questions left completely unanswered'
  - 'I sometimes get "that's the way it is – live with it"! '

John Havey

ROOT
Data Analysis Framework

Lots of Experiment Feedback

ROOT as library

- monolithic design
- ROOT encourages to be used as main process
- often it would be nice to use ROOT libraries separately possible but lacks documentation and support
- better interfaces to other libraries

in particular file I/O

- lots of data available as ROOT files
- simple reading of data from ROOT files would be nice e.g. reading ROOT file on embedded system

Conclusion

- ROOT has proven an extremely successful toolkit for both CMS developers and users
- We find the weekly meeting with the ROOT team essential
  - We have ROOT6 for Run2 because of the long collaboration between CMS and ROOT developers
  - Should this become a more widely advertised meeting for "customers" of ROOT?

CMS: David Lange
Alice: Jochen Klein

ROOT
Data Analysis Framework

More Experiments

ROOT 6 arrived - Lessons learned

KIT Karlsruhe Institute of Technology

The Belle II collaboration moved its framework to ROOT 6 on the 3.9.2015.

First release with ROOT 6 expected in October 2015

Lessons learned:

- The help and fast feedback provided by the ROOT team was essential - thank you very much!
- Serious preparation necessary before doing the switch - if you don't want your code base to be in limbo for some time
- Include the developers early and actively in the migration process
- Report bugs and problems you find to the ROOT Jira so everyone can profit

Performance

Example from Panda experiment
ROOT 6.04/00

	sim_complete.C		digi_complete.C		reco_com		d_complete.C	
Geant3	ROOT5	ROOT6	ROOT5	ROOT6	R	OT6	ROOT5	ROOT6
1 event	44s	25s	18s	11s		15s	8s	5s
10 events	47s	28s	20s		30s	17s	12s	7s
100 events	82s	42s		24s	81s	45s	62s	35s
1000 events	464s	2	147s	140s	642s	309s	616s	286s

ROOT6 up to 2x faster

Belle II: Thomas Hauth
FairROOT: Florian Uhlig

ROOT
Data Analysis Framework

ROOT Status

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**

Version 6 Releases Timeline

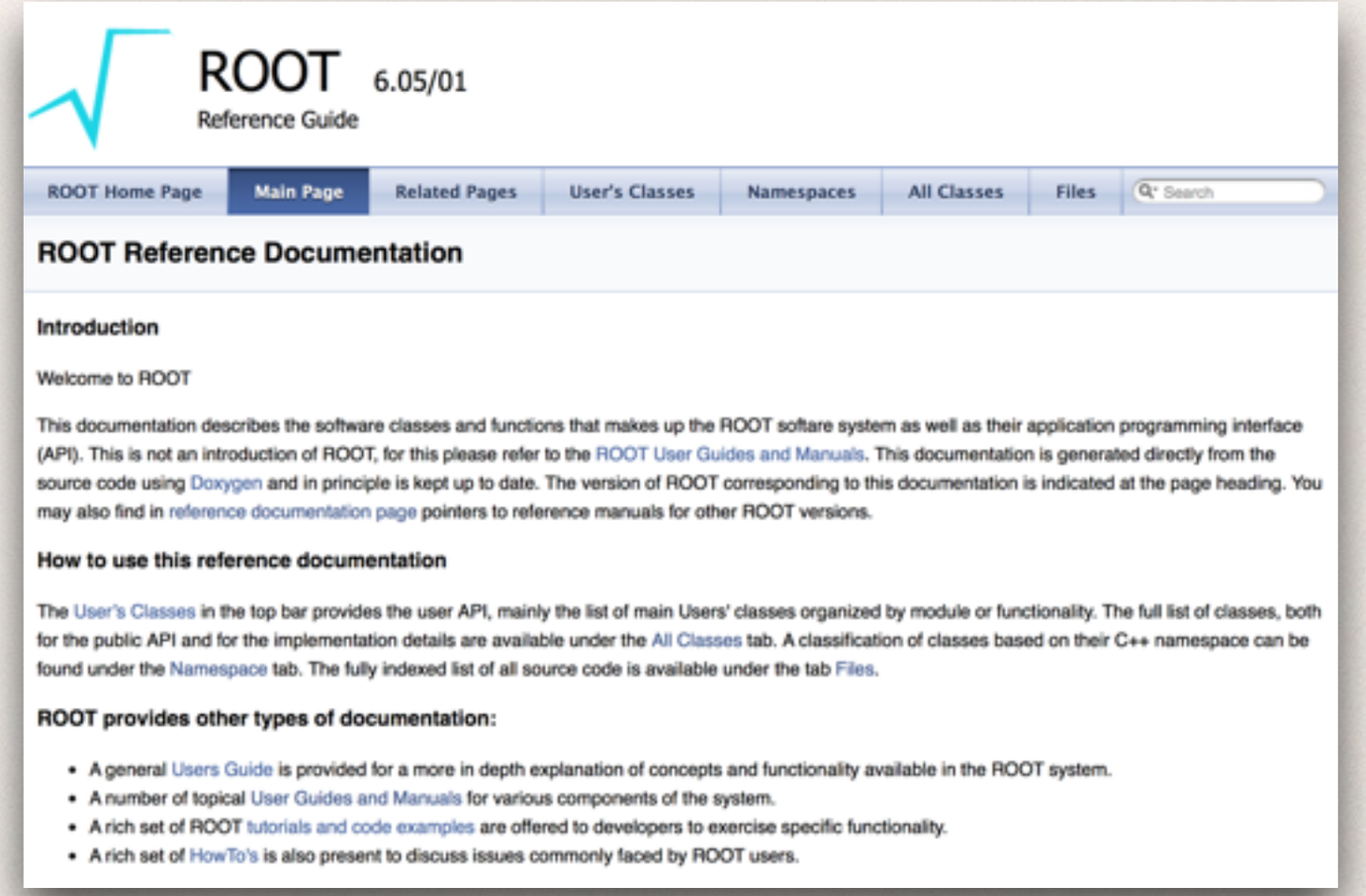
- Last ROOT workshop - technology preview
- 6.00 - May 2014
  - End-user preview for the ROOT6 features
- 6.02 - November 2014
  - Usable by the LHC experiments
- 6.04 - May 2015 - **Current production**
  - New JIT, new TFormula, new platforms on the way (Aarch64, PowerPC), ROOT-R, etc.
- 6.06 - Scheduled for November 2015
  - In time for 2016 running
- 6.08 - Scheduled for May 2016
  - Targeting PCMs and Windows support

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 migrated 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

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!



ROOT 6.05/01
Reference Guide

ROOT Home Page | Main Page | Related Pages | User's Classes | Namespaces | All Classes | Files | Search

ROOT Reference Documentation

Introduction

Welcome to ROOT

This documentation describes the software classes and functions that makes up the ROOT softare system as well as their application programming interface (API). This is not an introduction of ROOT, for this please refer to the ROOT User Guides and Manuals. This documentation is generated directly from the source code using Doxygen and in principle is kept up to date. The version of ROOT corresponding to this documentation is indicated at the page heading. You may also find in reference documentation page pointers to reference manuals for other ROOT versions.

How to use this reference documentation

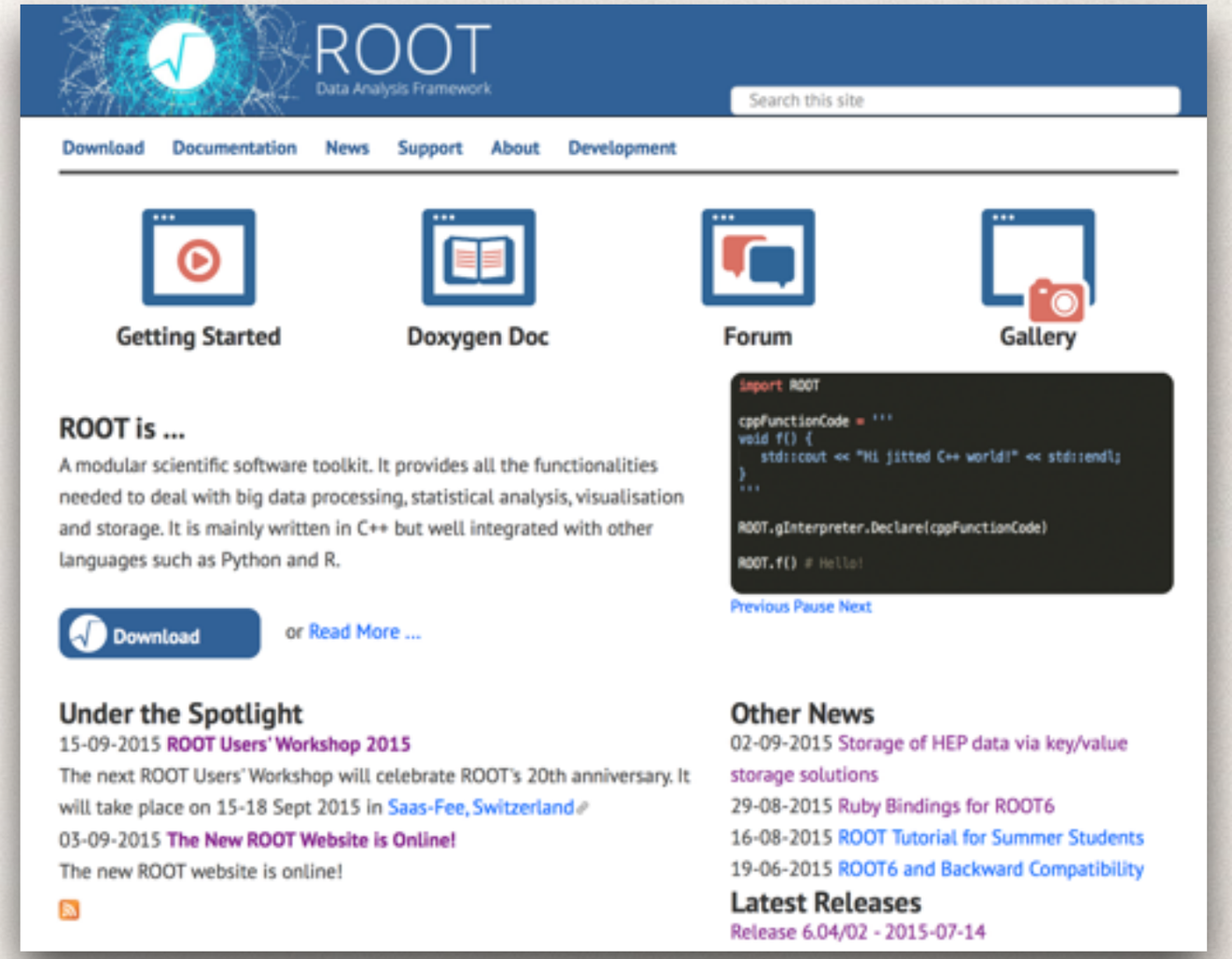
The User's Classes in the top bar provides the user API, mainly the list of main Users' classes organized by module or functionality. The full list of classes, both for the public API and for the implementation details are available under the All Classes tab. A classification of classes based on their C++ namespace can be found under the Namespace tab. The fully indexed list of all source code is available under the tab Files.

ROOT provides other types of documentation:

- A general Users Guide is provided for a more in depth explanation of concepts and functionality available in the ROOT system.
- A number of topical User Guides and Manuals for various components of the system.
- A rich set of ROOT tutorials and code examples are offered to developers to exercise specific functionality.
- A rich set of HowTo's is also present to discuss issues commonly faced by ROOT users.

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 Kousi

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)

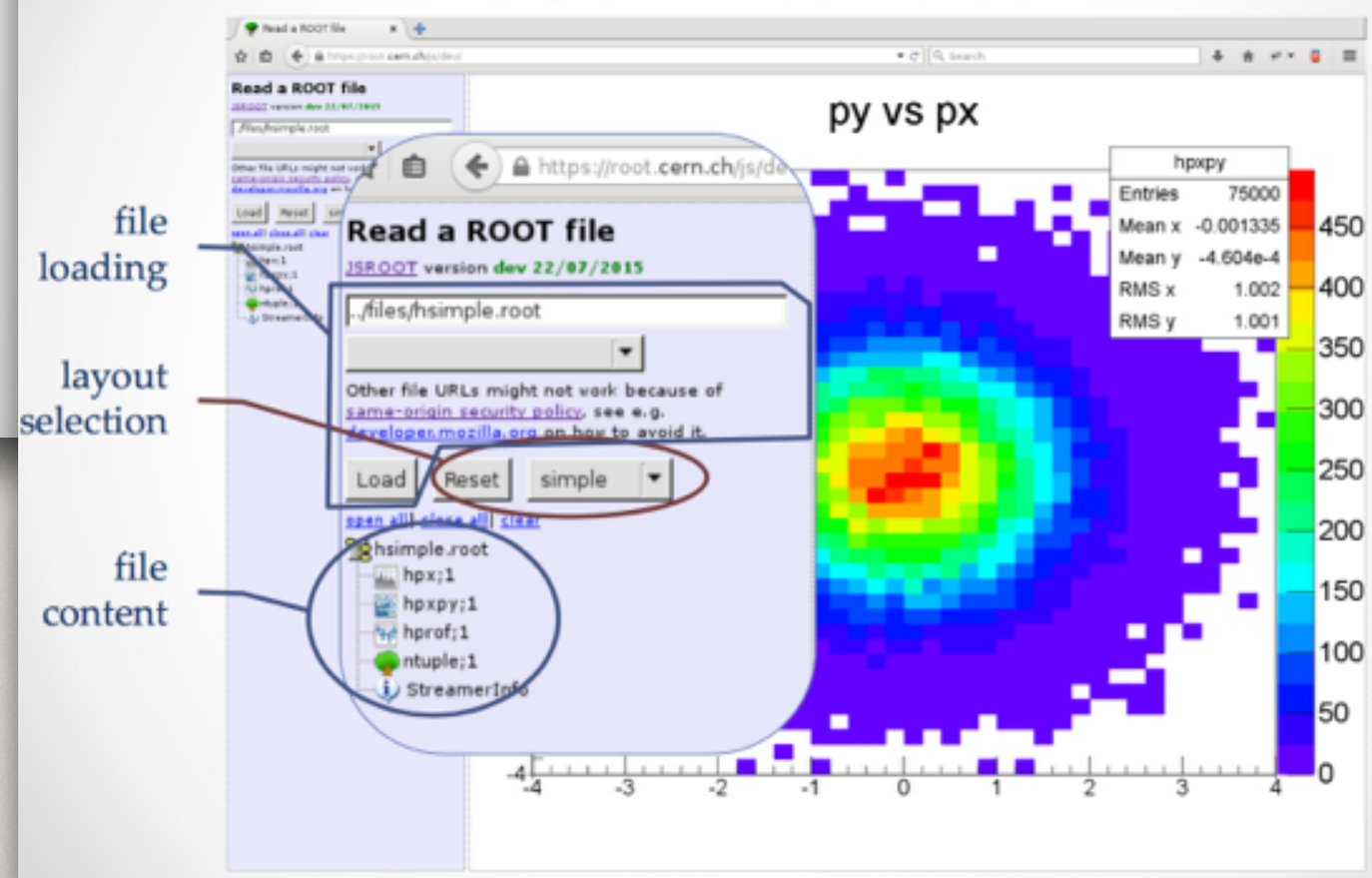
JavaScript ROOT

Main features

JavaScript ROOT provides:

- Objects reading from binary and JSON ROOT files
- Display for popular ROOT classes in web browsers
- Flexible API for usage in other projects

User interface



Betrand Bellenot, Sergey Linev

ROOT
Data Analysis Framework

RooFit

Pushing the boundary on RooFit model complexity

- MINUIT minimization (still) works well with 4200 parameters.
  - Had to disable default MINUIT2 feature to save intermediate covariance matrix at every VariableMetric step (each V takes ~70 Mb. 100 steps = 7 Gb…)
- Some tuning of memory model and code optimization needed. ATLAS/CMS model consumes ~6 Gb, minimizes w.r.t 4200 params in ~5 hours
  - Profiling with callgrind, memcheck, massif
  - 40% used by objects representing functions, 30% on links between objects, 30% on caches of various types
  - Majority of CPU time spent in probability functions doing the 'actual work' (morphing transformations)
- Work on scalability improvements going
  - Most scaling issues in model manipulation (setup phase for fit) - usually fixed with lookup tables etc

Further development plans

- Documentation (yes I know…) Holy grail project - develop a guide to statistical analysis with hands-on RooFit implementation [big project!]
- Improved internal optimization of likelihood calculation & parallelization (many ideas - not so much time yet)
- Keep working on scalability and performance - so far has never been a showstopper
- Incorporate new tools and concepts that emerge from collaborations (a posteriori trimming of model complexity - 'pruning')
- Replace old core code with modern STL implementations (big help here so far from Manuel Schiller!)

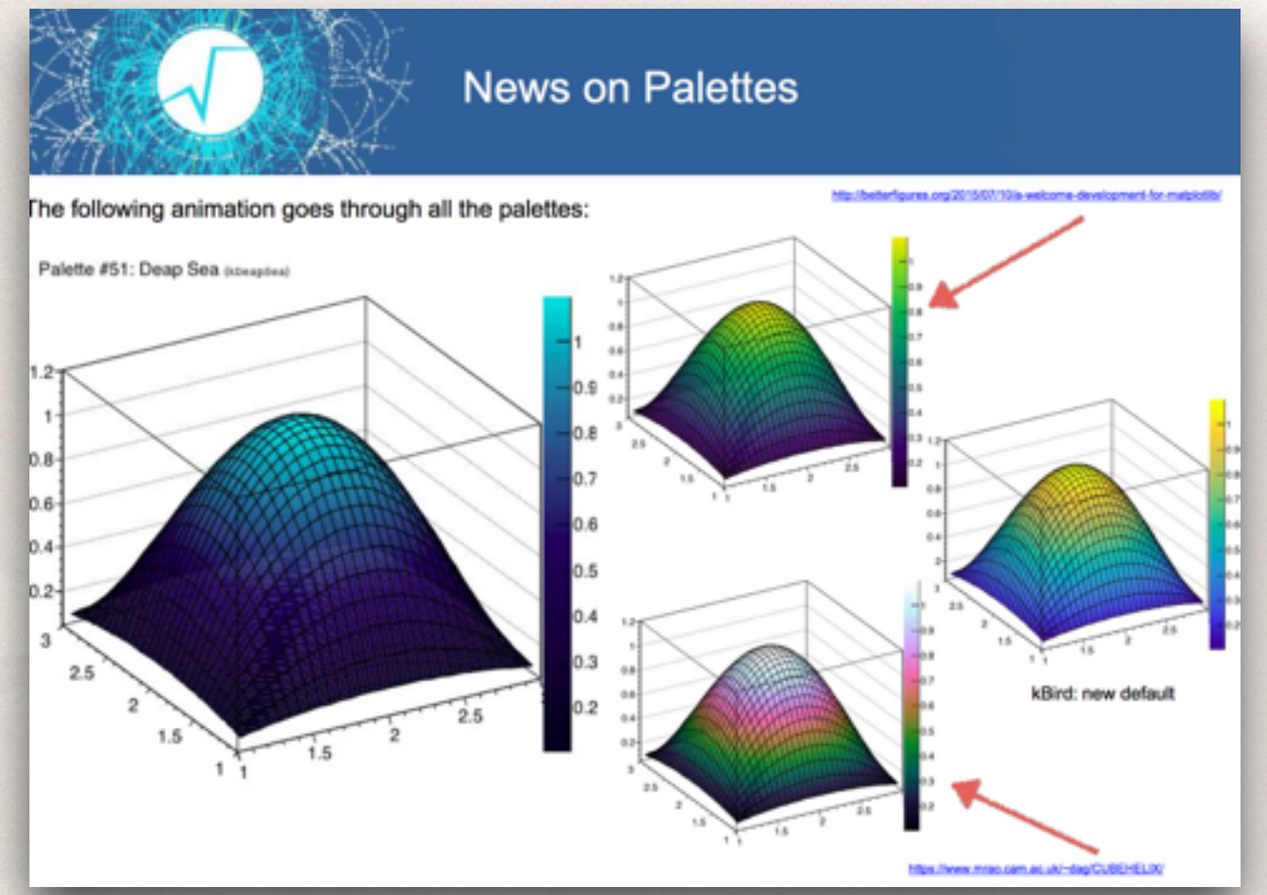
Wouter Verkerke

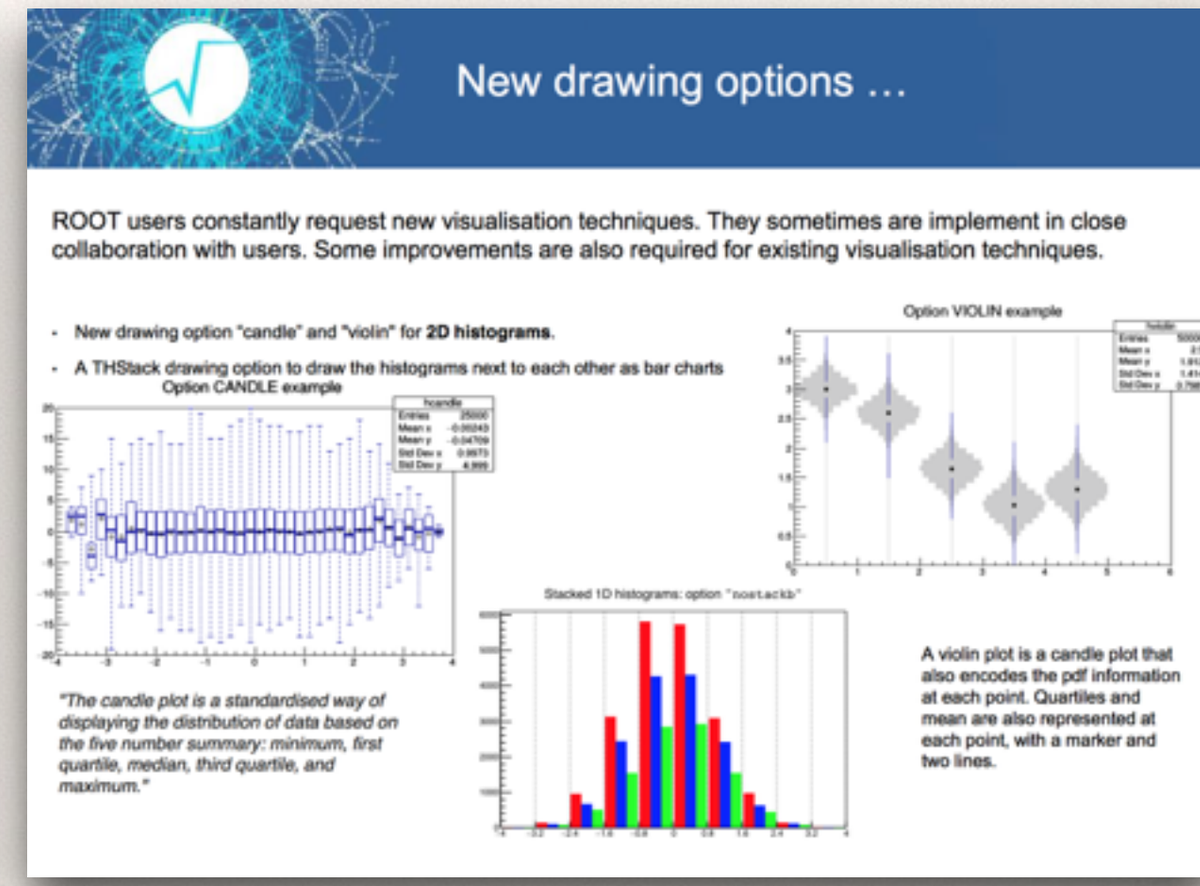
Math Libraries

- New TFormula class
- Improvements in TF1
- New Random number classes
  - new MIXMAX generator
- Recent developments in TMVA (redesign and interfaces to R and Python)

Lorenzo Moneta at al.

Graphics





Olivier Couet

ROOT
Data Analysis Framework

Packaging ROOT

Building ROOT for Fedora

- The ROOT build that is packaged should be as complete as possible
- But installation should be modular – the build is split up in 90+ separate packages
- When building the Fedora version of ROOT, some rarely used modules get compiled, often using different versions of dependencies than was used before
- This results in bug reports and patches which sometimes surprise the ROOT developers

Things that would make things easier

- Make it possible to run the root binary compiled for installation in the build tree to make it easy to generate documentation and run the test suite
- Avoid having to choose between being able to build a complete set of modules (using configure) and being able to easily run the test suite (using cmake) – make at least one of them feature complete
- Add missing --disable-builtin-xxx flags

Mattias Ellert

Ongoing Developments

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

Modern C++ interfaces

- Aiming to improve friendliness and standardization
- Many interfaces can be improved in C++14, 17
  - Type-safety instead of runtime crashes, for instance ownership and drawing options; simple and focused classes, etc.
  - 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

New C++ Interfaces Proposal

The Goal

- The world has changed, ROOT needs to adapt
- Successful maintenance, yet need for evolution
- Can only convince through features, robustness, simplicity: usability

Interoperability
Simplicity
Task-Parallel
Robustness
etc.

The Path

- Small steps enable organic growth: enable feed-back loop
- Early involvement and adoption has proven a key ingredient to success of ROOT 6 (and v1, v2,…), much more for ROOT 7
- In time for Run 3!

Axel Naumann

ROOT
Data Analysis Framework

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)

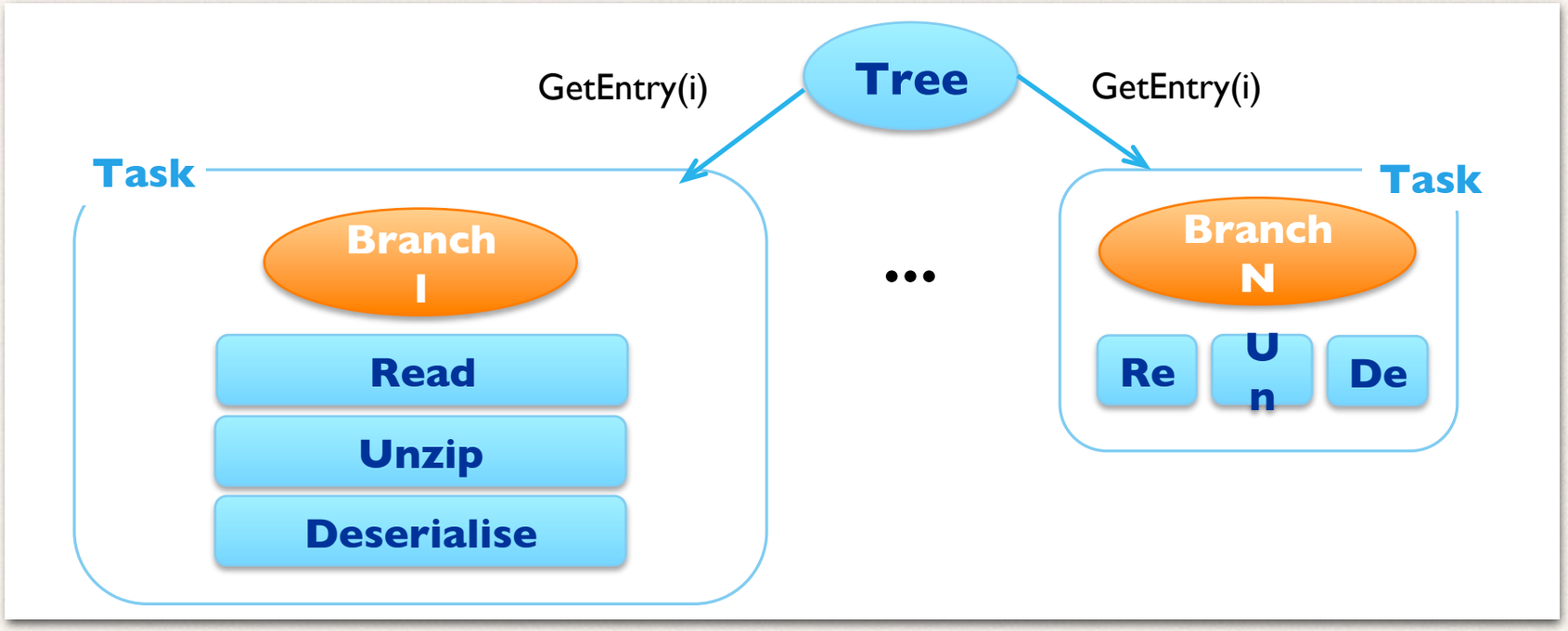
Multi-process

- 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"}
)
```

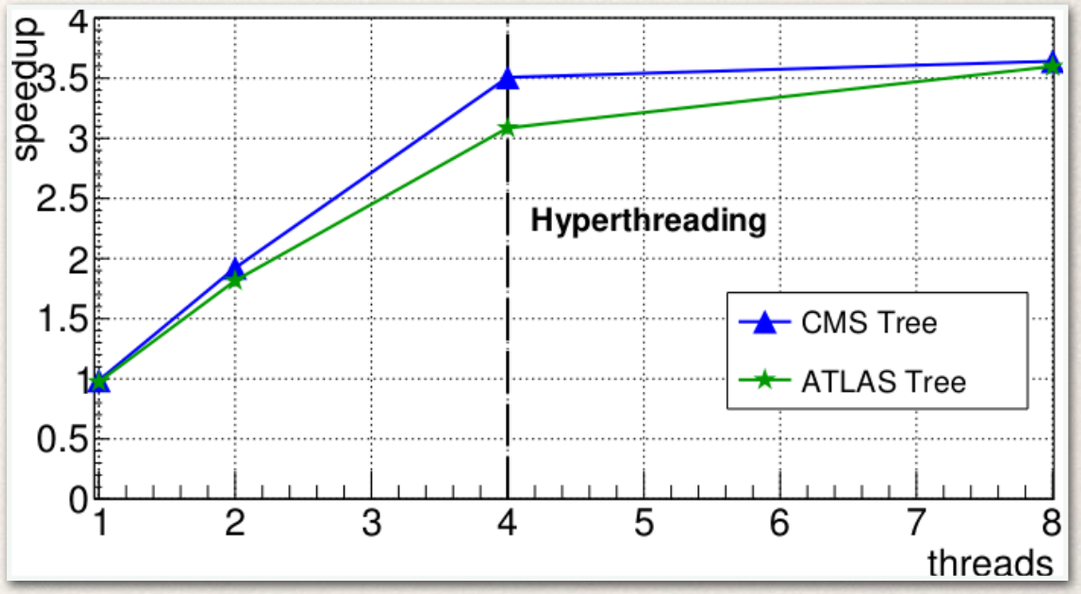
Gerri Ganis, Enrico Guiraud

Parallel TTree Reading



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

Enric Tejedor



ROOT Data Analysis Framework

Development: Packaging

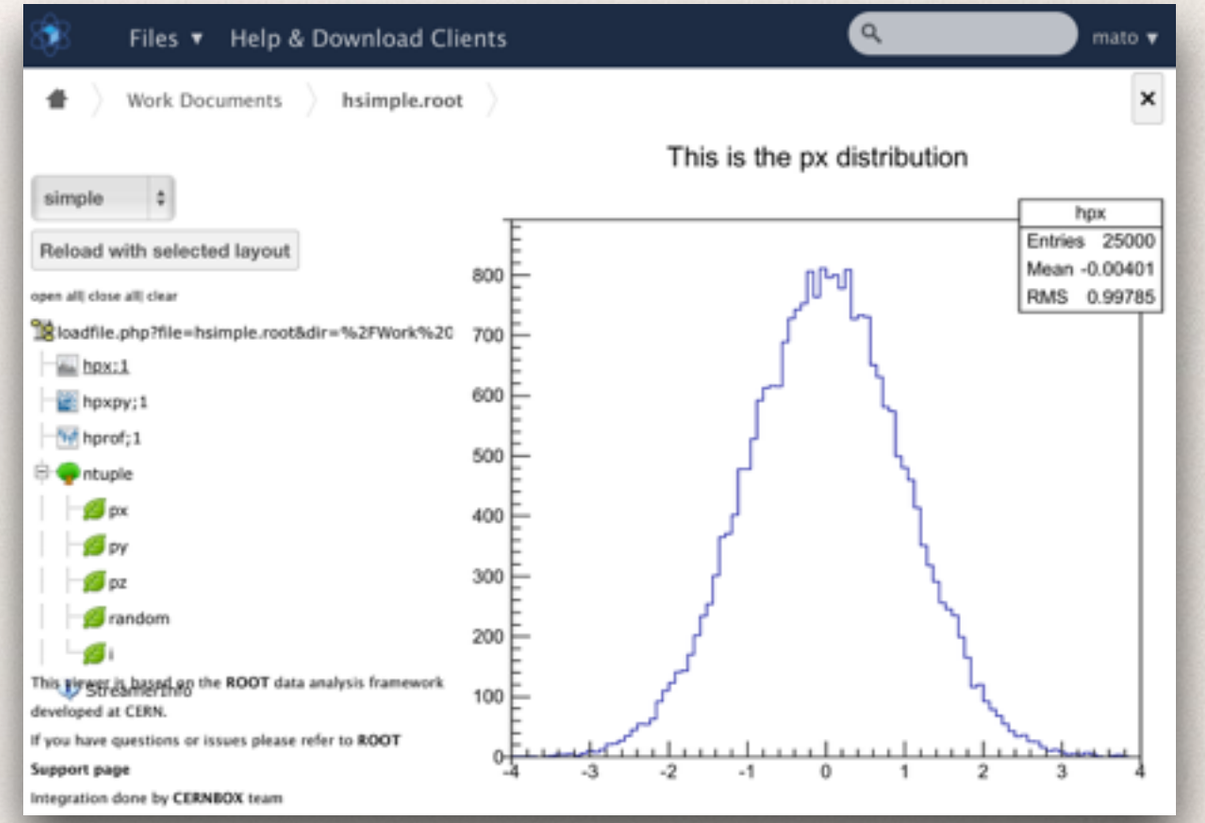
Easy use third party packages
Build/install modules and plugins on demand
Slimed down initial ROOT installation (BOOT)

- 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**

Development: Rethinking UI

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

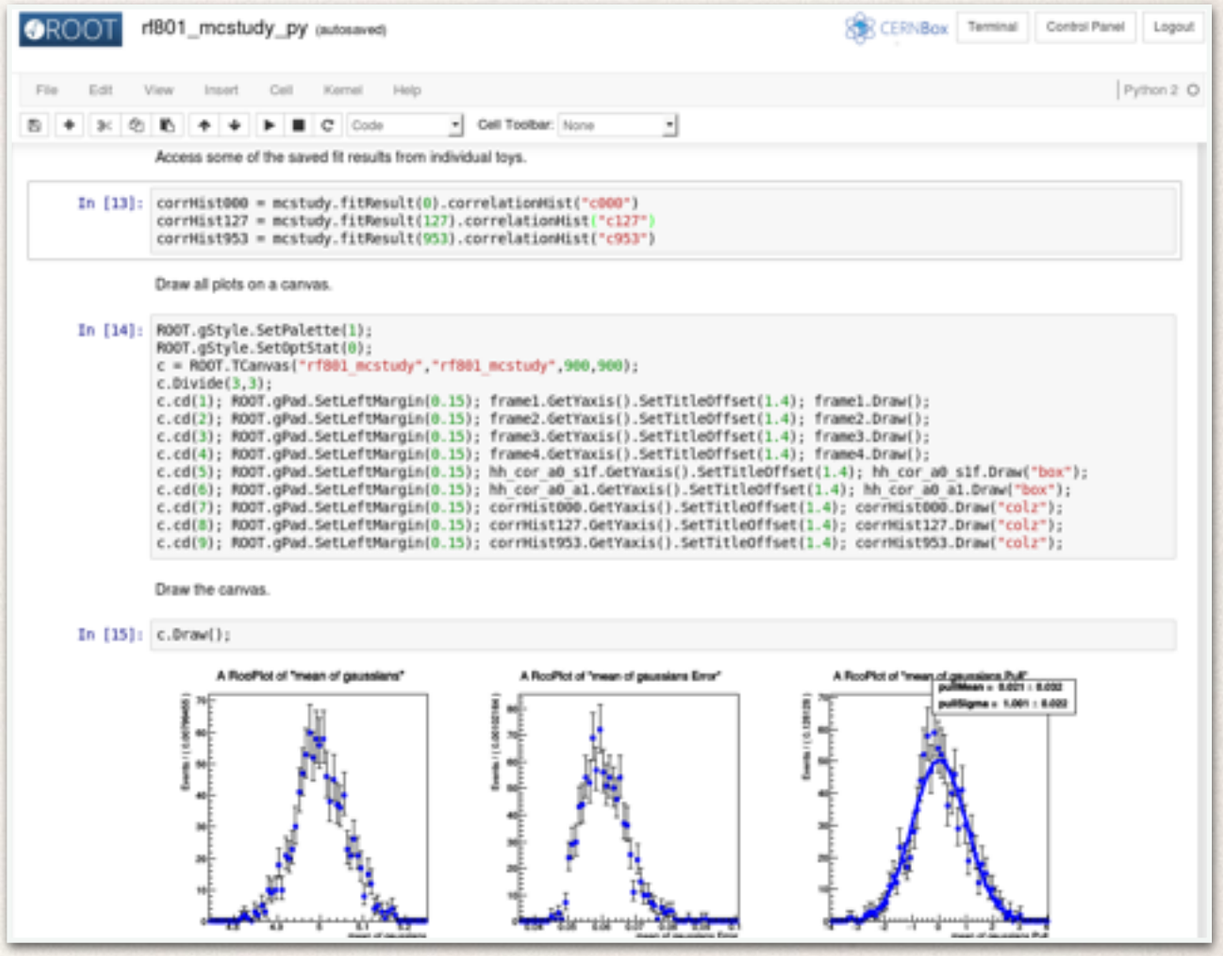
- Increase interactivity using modern web technology (**javascript**) in a client-server model
  - No need to install anything in the client side
  - 3D geometry viewer
- Built on the HttpServer of Sergei Linev and JSROOT of Bertrand
- CERNBox Example



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.)

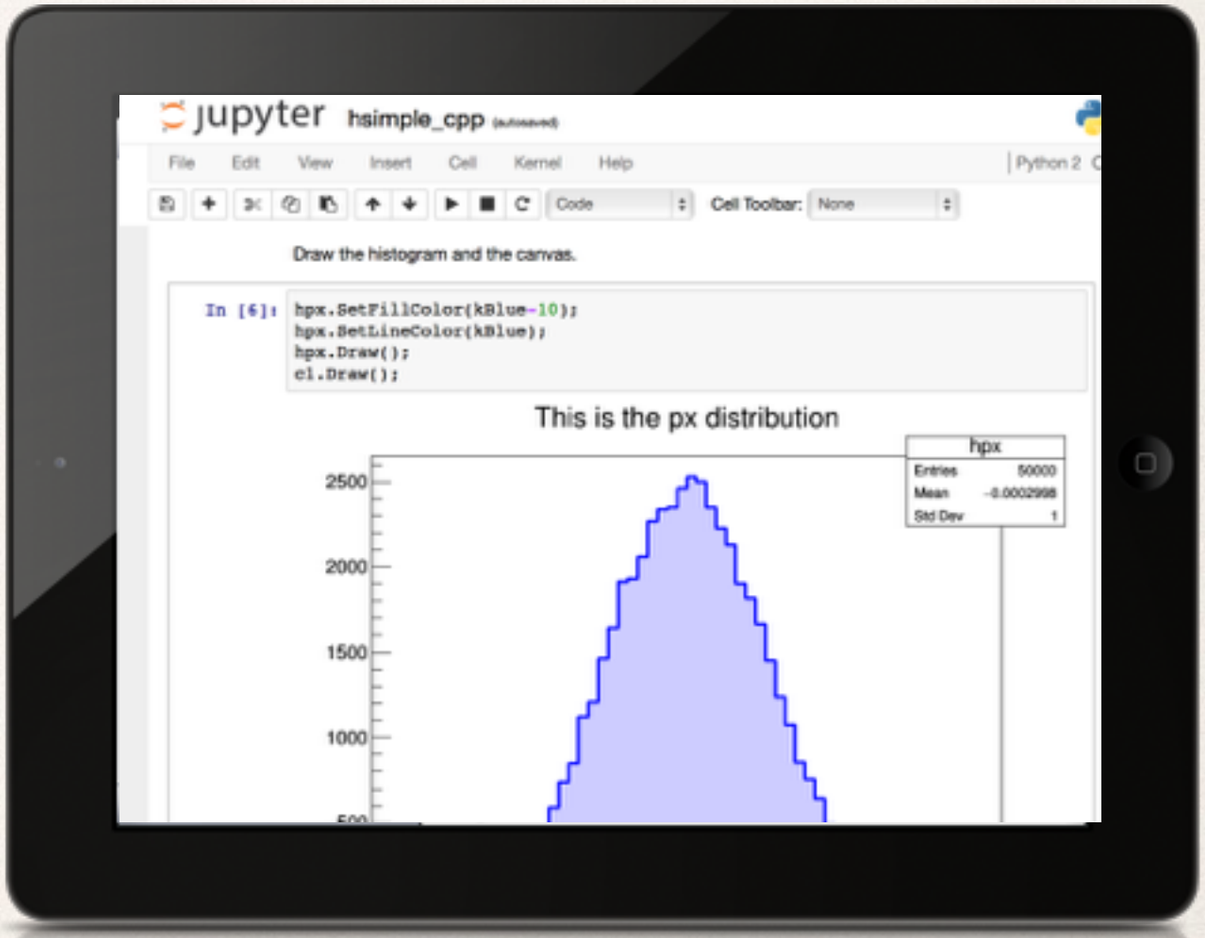


ROOT
Data Analysis Framework

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

Proposing ROOTaaS Pilot

Conclusions

- ROOT is now integrated with notebooks
  - Python and C++ interactive shells
  - Tab completion, C++/Python integration, syntax highlighting, graphics inlining, shell commands
  - Available now (6.05/02)!
- Integration with the CERN services portfolio
  - Collaborating with IT department: started to capitalise on interplay with storage services
  - Work in progress, usable demo available to be tried at the ROOT workshop!
  - Bright future ahead of us: e.g. r&d on containers scheduling, job submission steering from notebook (e.g. with Ganga), software provision models.

Got very positive feedback and suggestions

Danilo Piparo, Enric Tejedor

New Ideas

Writing Good C++ 14

Coding guidelines

- Let's build a ***good*** set!
  - Comprehensive
  - Browsable
  - Supported by tools (from many sources)
  - Suitable for gradual adoption
- For modern C++
  - Compatibility and legacy code be damned! (initially)
- Prescriptive
  - Not punitive
- Flexible
  - Adaptable to ***many*** communities and tasks
- Non-proprietary
  - But assembled with taste and responsiveness
- Teachable
  - Rationales and examples

Stroustrup - Guidelines - Root'15

Current status

- Available
  - About 350 Rules (https://github.com/isocpp/CppCoreGuidelines)
  - GSL for Clang, GCC, and Microsoft (https://github.com/microsoft/gsl)
  - First tools: October for Microsoft; ports later (November?)
  - MIT License
- We need help
  - Review of rules
    - More examples and refinements for existing rules
  - Specialized rule sets
    - For particular application areas, projects, …
    - For concurrency
    - For libraries
    - …
- Continuous development
  - "forever"

Stroustrup - Guidelines - Root'15

Bjarne Stroustrup

Everware

Everware let's you edit and run code that has complex setup instructions with one click, from your browser.

Tim Head

Architecture

Docker + jupyterhub = everware

If the code's environment is specified in a Dockerfile, everware will build it, launch it and connects you to it via your browser.

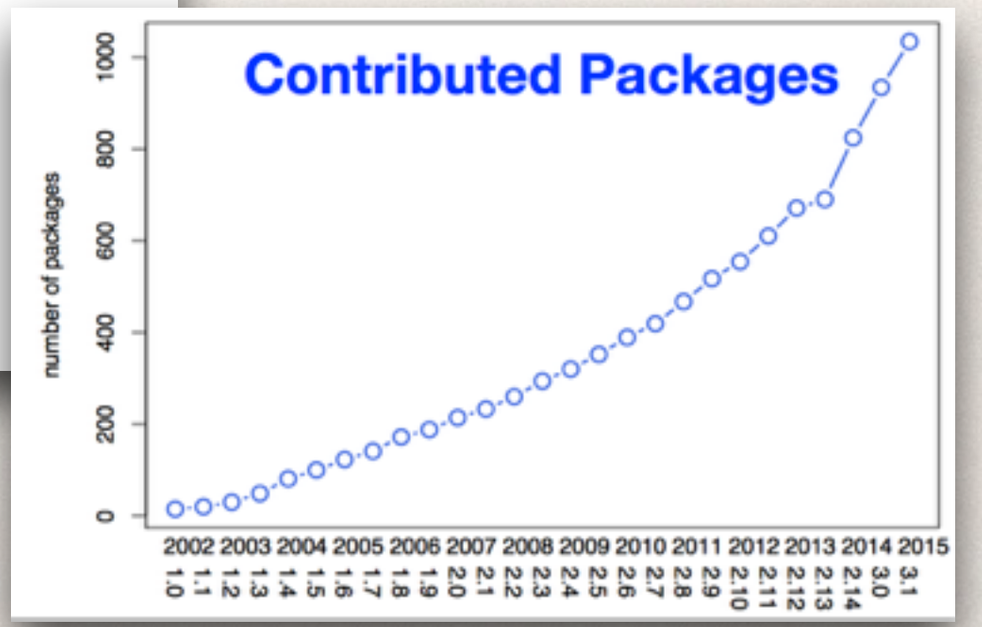
Bioconductor

Open source and open contribution software project for the understanding of genomic data

- Lower the barrier of entry to adequate statistical methods
- Distributed development of interoperable components
- Integration of many data types and experiments
- Rapid development and code re-use
- Robust and durable publication & distribution of software
- Facilitate computational reproducibility of scientific claims
- Training
- Turn users into developers
- Based on statistical language R

world's largest bioinformatics project with
10,000s users, >11,000 references in Pubmed Central

the HSF for bioinformatics !



Wolfgang Huber

ROOT
Data Analysis Framework

R

Why R?

- high-level, interpreted programming language
- rapid prototyping, creativity, flexibility and reproducibility
- scientific and statistical computing capabilities
- graphics
- mature package management system
- inter-language interfaces (C, C++, Java, JavaScript)

- LISP inside

Wolfgang Huber

Omar Zapata

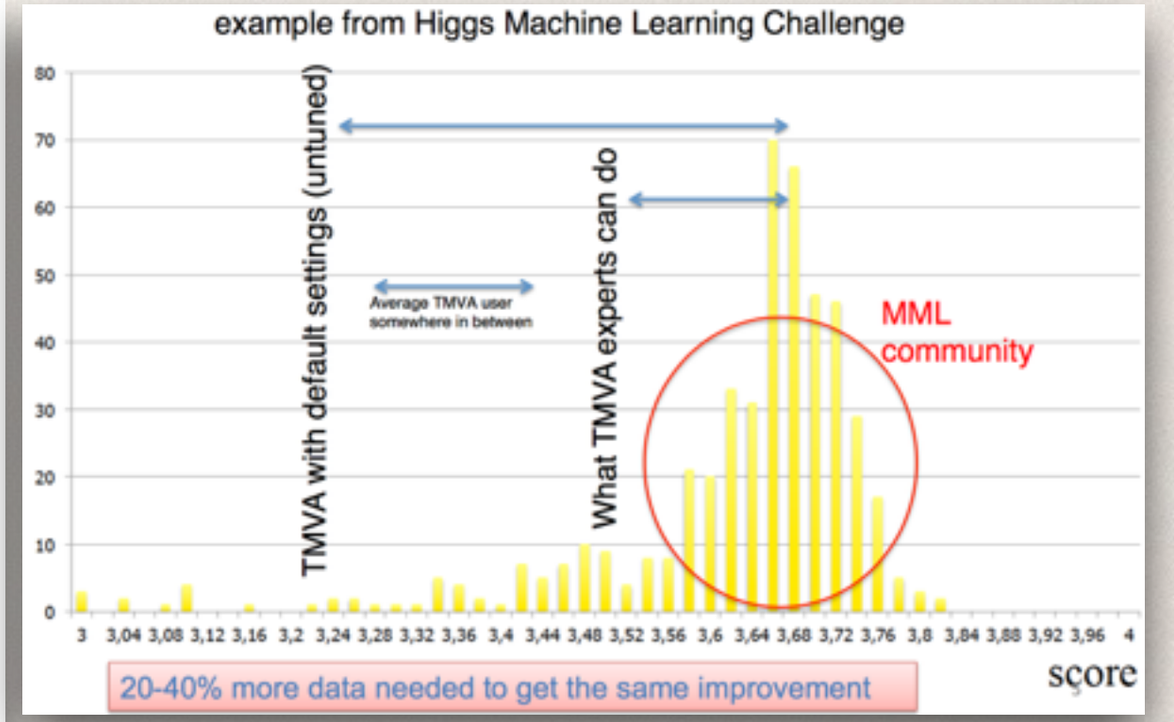
ROOT
Data Analysis Framework

- ROOT with R
- Examples
- RMinimizer
- RMVA (R with TMVA)
- Future directions

Machine Learning and HEP

Machine Learning (ML) usage in HEP

- Many HEP problems can be posed in form of a classification or regression problem
- Best signal-background discrimination, both high and low level
  - Kinematic selection for physics analysis (used in many Run 1 results, HiggsML Challenge, pheno papers, e.g. 1402.4735 ,…)
  - Object identification: b-tagging (e.g. MV1 in ATLAS, ATLAS-CONF-2014-046), boosted objects…
  - Track reconstruction (NN clustering for ATLAS pixel: 1406.7690, connecting the dots 2015 WS: https://indico.physics.lbl.gov/indico/conferenceDisplay.py?confId=149)
  - Trigger level (LHCb example: http://cds.cern.ch/record/2019813?ln=ru)
  - Idea to use ML in FPGA's for phase 2 upgrade
  - Most conventional algorithms are already black boxes to most users
  - Many of these applications are in production software
  - Many other ideas & plans…
  - I am sure there are similar use cases at the level of the LHC machine



- Interest HEP & ML community to bring MML to HEP – Very promising initial results / work done
- Still a lot to do: data-MC comparison, systematics, etc.
- Better long-term support for TMVA

Tobias Golling

ROOT Data Analysis Framework

The SSD Challenge

The ROOT I/O SSD Challenge

- If by 2018 SSD's become active storage
  - Either in a hierarchy or primary storage
- ROOT I/O may be insufficient
  - Object layout & access algorithms HD-oriented
    - SSD's have their own peculiarities
      - For example, large page read-out size
- Time to start rethinking ROOT I/O!
  - How to get the most out of SSD's

XRootD & SSD

- XRootD already is SSD ready
  - Already supports tiered storage (i.e. SSD+HDD)
    - Used by SLAC for the ATLAS Tier 2
- Reasonable approach until SSD prices drop
  - HD rival is estimated by 2020
    - Based on improvements to 3D NAND technology
- So, 2020 may mean primarily SSD access
  - Will ROOT IO be ready?

Andrew Hanushevsky

ROOT
Data Analysis Framework

New Languages: Julia

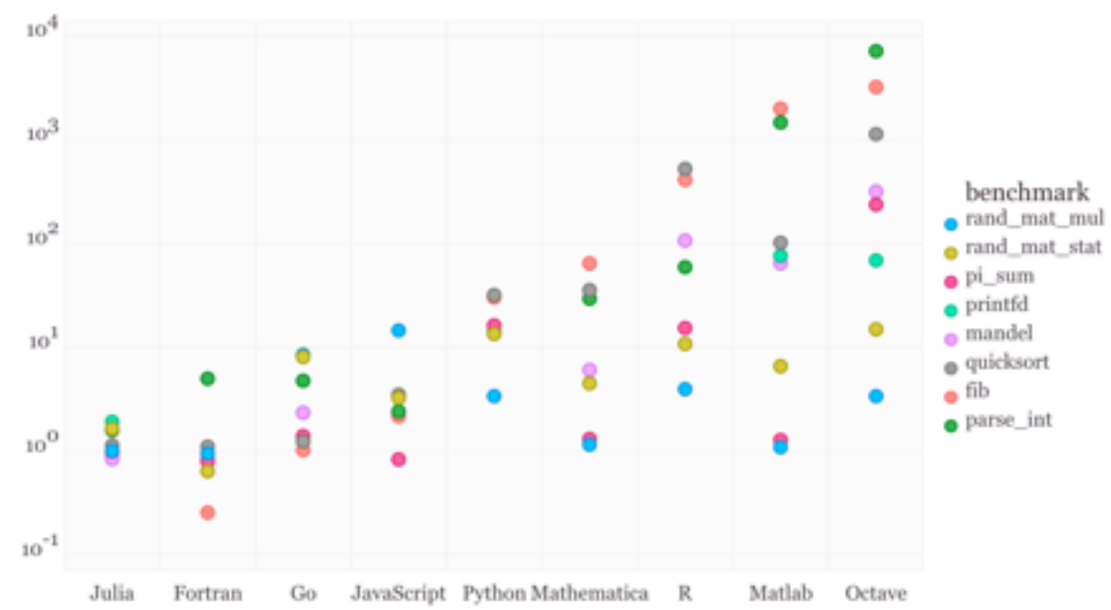
The solution

julia

High-level, fast, dynamically compiled numerics.

- Started at MIT CSAIL in 2011, now open-source, worldwide activity.
- Easy to use (like MATLAB, R), for generic numerical computing
- Used for physics, bio-informatics, statistics, image processing, finance
- Modular design: well-tested core + packages
- Code and issues tracked on github, (too) easy to contribute.
- Based on LLVM, OpenBLAS/Intel MKL

Speed comparisons



Naive julia implementation often similar to or better than C / Fortran,

Joosep Pata

ROOT
Data Analysis Framework

Digesting Feedback and Preparing Programme of Work

ROOT Team Retreat

- Organized a 1 day discussion meeting to give answer to a number of questions to guide our development
- Main Topics
  - User Support
  - Collaboration and Contributors
  - Modularization and BOOT
  - Interoperability with other languages and libraries
  - Documentation.
  - Training
  - New C++ interfaces
  - Expressing Parallelism
  - Input and output (partially treated also in the parallelisation section)
  - Interpreter, reflection, core and typesystem:
  - Graphics, Gui and Visualisation
  - ROOT as a Service
  - ROOT 7 PLANS

Main Decisions

- Made quiet a lot of progress and took a number of decisions, although we didn't manage to reach agreement on everything
  - Different viewpoints and priorities from team membe
- The main decisions are documented at:
  https://root.cern.ch/root-retreat-2015-minutes
- ROOT 7 plans:
  - timescale for ROOT 7 is end of Run II
  - prepare a document with proposed feature-set of ROOT 7
  - use Wednesday meeting to discuss this with users/experiments