

ROOT Plans for 2017 and Beyond

11th January 2017, SFT Group Meeting

Outline

- ❖ What did we manage from last year's plans?
 - ❖ Quick review of tasks
- ❖ Main areas of work for 2017
 - ❖ Maintenance and support
 - ❖ Development tasks
 - ❖ Foreseen tasks in each ROOT sub-system
 - ❖ Documentation
 - ❖ Building and packaging
 - ❖ Infrastructure
 - ❖ Tutorials and courses

The ROOT Core Team

- ❖ Bertrand BELLENOT
- ❖ Philippe CANAL (FNAL)
- ❖ Oliver COUET
- ❖ Gerri GANIS
- ❖ Enrico GUIRAUD (DOCT)
- ❖ Pere MATO
- ❖ Lorenzo MONETA
- ❖ Axel NAUMANN
- ❖ Danilo PIPARO
- ❖ Enric TEJEDOR (FELL)
- ❖ Xavi VALLS (DOCT)
- ❖ Vassil VASSILEV (Princeton)

Review Last Year Plan

Cling Interpreter and PCMs

- ✦ Upgrade to latest LLVM/CLANG
 - ✦ Support new ABI (GCC 5)
- ✦ Optimizations and improvements
 - ✦ reduce and CPU and memory consumption
 - ✦ transparently be able to create functions and the ROOT prompt
 - ✦ tab-completion based on multi-interpreters
- ✦ PCMs for ROOT's dictionaries
 - ✦ be able to replace the today unique PCH by a number of PCMs



- ✦ Major upgrade of LLVM/Clang to support new ABI
- ✦ Building ROOT with 'clang modules'

Parallelization: Multi-Threading

- * Continue tackling the parallelization use cases
- * In particular, the next steps are:
 - * Investigate the `TTree::Draw` parallelization and its combination with the `TThreadLocalObject` class for the parallel histogram filling
 - * `TTree::Process(lambda)`
 - * Investigate how to read multiple events in parallel
 - * Investigate how to write multiple events/branches in parallel
- * Benchmark different prototypes
 - * Speedup, understand the CPU overheads, memory increase for the different cases
- * Investigate how to interface to an externally provided **scheduler**



- * Parallel processing (reading) of TTree being explored with *Functional Chains* (see later)

Parallelization: Multi-Process

- ✦ Ensure the same (compatible) interface between the multi-threading and multi-processor solutions
- ✦ Improve the new **core/multiproc** package
 - ✦ Complete support for `TSelector` and provide integration with `TTree::Draw`
 - ✦ Provide detailed documentation and more complex examples
 - ✦ Investigate the extension of the `multiproc` interfaces to a cluster of machines (à la iPython parallel)
- ✦ PROOF in maintenance mode



- ✦ *TProcessExecutor* and *TThreadExecutor* implements *TExecutor* interface
 - ✦ used internally in Math and TVMA
- ✦ Investigating the use of Spack clusters




Parallelization: Math Use Cases

- ✦ Deploy parallelization for ROOT fitting using both multi-threads and multi-processes
- ✦ Optimize the low-level implementation of **multiproc** to be useful for replacing the RooFit specific solution
- ✦ Use multi-process solution for RooStats calculators (e.g. frequentist studies which requires large generation of pseudo-experiments)






- ✦ MT/MP fitting is almost done
- ✦ Replacement of ProofLite by **multiproc** is postponed for this year

Vectorization

- ✦ Use types in the VecCore library, which embeds low-level support for vectorization, in the ROOT Linear Algebra classes. 
- ✦ VecGeom could be an external dependency, but it is very likely to be directly used by core ROOT components, in which case it would need to be included as a base package. 
- ✦ Interface to Vector Function evaluation to be used mainly for fitting. 

- ✦ Mainly work in progress (Xavi's PhD work) to be completed soon
- ✦ Separate release of VecCore would be desirable

New C++ Interfaces (ROOT 7)

- ✦ Many interfaces can be improved with C++14,17
 - ✦ Ownership, type safe containers, string options
 - ✦ Improved user productivity, by dramatically reducing memory errors, wrong results, etc.
 - ✦ Bi-weekly meetings with users from experiments
 - ✦ started discussing iteration and ownership issues
 - ✦ Starting with histograms + visualization, and TFile
 - ✦ Investigation of TTree and POD storage interface
- 
- 
- 
- ✦ Continued bi-weekly [monthly] meeting with users from experiments
 - ✦ Testing, suggestions, feedback, etc.
 - ✦ 'root7' build option to start playing with the new classes

I/O Performance

- ✦ Better support concurrent I/O operations
- ✦ Code (algorithm) optimization
 - ✦ New version of Optimize Basket
 - ✦ Prefetching in fast cloning
 - ✦ I/O implementation improvements
- ✦ Optimization via change in (low level) file format
 - ✦ Endianness leading to R&D on “memcpy to disk”, i.e. persistent memory layout (POD)
 - ✦ compression windows, eliminating metadata redundancies
- ✦ Provide help for benchmarking I/O performance in presence of automatically generated POD data models (e.g. HepMC3, PODIO)



I/O New Features

- ✦ Support for C++11 `std::unique` and `std::array`
- ✦ Investigate how to support C++11 shared and weak pointers
- ✦ Streamlining of I/O interface (mostly in the context of the v7 new interfaces)
- ✦ Open up the interface of TFile to be able to make use of key-value storage
 - ✦ Investigate the Kinetic key-value store technology
- ✦ Support for JIT-compiled collection proxies
 - ✦ Enable transparently the serialization of collections
 - ✦ Stream collection without dictionary with MultiProc



Geometry

- * Provide the users of **TGeo package** and its derived navigation interfaces (VMC) automatic support for vectorized navigation based on the **VecGeom package**
 - * Add **VecGeom** library as CMake-configurable external module
 - * Phase-1: Implementation of a TGeoShape-derived bridge class (TGeoVGShape) delegating the navigation interface to the **VecGeom** solid
 - * Phase-2: Implementation of a **VecGeom-aware** navigation interface that can redirect the current navigation API of TGeoManager / TGeoNavigator to native **VecGeom** navigators



Math Libraries

- * Treat VC and VDT libraries like all the other external libraries
 - * Look for them externally, otherwise build them internally (builtin_XXX option)
- * Integrate new minimizers based on libcmaes (new external)
- * Re-implement TRandom classes using ROOT::Math::Random
- * Complete **MixMax** testing and make it the default random generator
- * Extend RooStats asymptotic calculator to 2 dimensions
- * Integrate in RooFit new class for multi-dimensional kernel density estimator



- * RooStats and RooFit tasks removed since it didn't receive request from users

Machine Learning

- * Active participation to the Inter-Experimental LHC Machine Learning Working Group IML launched recently



- * Foreseen tasks and activities:

- * Complete TMVA re-design
- * Add support for cross-validation in TMVA
- * Add possibility for parallel execution in TMVA methods
- * Evaluate and Integrate a new deep learning neural network in TMVA
- * New SVM (support vector machines) models in TMVA



- * Ongoing re-engineering of TVMA. Performance improvements.
- * Added additional interfaces to main stream tools (e.g. SciKitlearn, Keras, R)

Graphics

- * Explore integration with JSROOT, i.e. avoid parallel developments
- * Development of new plots
 - * Helper class for Ratio Plots (e.g. Data/MC, MC/MC)
 - * Spider plot not based on TTrees (e.g. Data Frames)
- * Consolidation of the present functionalities, e.g. fixed size fonts.
- * Style object containing all style elements, e.g. legend placement applicable to other graphics objects
- * The graphics UI needs some rethinking and/or simplification
 - * leaner interface, helper tools (classes/functions) for plotting, automatic color/line/marker/legend schemes
- * Many requests in JIRA



JSROOT

- * Complete 2D drawing options
- * Add missing 3D drawing & options
- * Add simple GUI (overlay buttons) to access some functionality (e.g. saving as .png)
- * Finalize geometry
 - * Add (many) missing shapes and constructive solid geometry (CSG)
- * Add GUI for interactivity with THttpServer



Python bindings and ecosystem

- ❖ Python3 fully supported
 - ❖ Few pending issues to be fixed
- ❖ Release PyROOT as a genuine Python package
 - ❖ Co-existing within Python2 and Python3
 - ❖ build / install based on *pip* package management?
- ❖ Prototype tree / ntuple analysis which adopts a Data Frame-like interface
 - ❖ Learn from interfaces used by crowds of data scientists



- ❖ Work in progress to re-structure Python packages within ROOT
- ❖ *Functional chains* started a python prototype (summer student). See later

Packaging and Modularization

- ✦ Converge on a factorization of the logical components of ROOT and produce an analysis of the current interdependencies among them
 - ✦ Estimate the effort needed to reduce the module coupling
 - ✦ Demonstrate that the coarse grained factorisation is possible in our build system
- ✦ Develop model for building/installing modules on demand and evolve ROOT into BOOT (à la R)
- ✦ Decide what distribution mechanism for the single packages is adequate for our needs
 - ✦ Collaborate with the HSF packaging working group



- ✦ Very little progress has been achieved. Design ideas mainly.
- ✦ Coherent treatment of bundled / unbundled dependent ROOT packages

ROOTbooks

- ✦ Integration of ROOT and Jupyter notebooks is well advanced, although there is still some work to do in the following areas:
 - ✦ Disseminate and advertise the use of ROOTbooks
 - ✦ produce ROOTbook tutorials, examples
 - ✦ use of Binder as a try-out demo
 - ✦ Make the JS visualization the default and improve it
 - ✦ save as button, back to initial state button, save 3D geometry as image
 - ✦ Add the R magic to use ROOT-R and combine it with Python and C++
 - ✦ Rollback feature for cells (specially for C++)
 - ✦ Cling kernel integrated in JupyROOT
- ✦ Demonstrate the possibility to integrate with the Akhet VNC system



ROOT as-a-service

- * Deliver in 2016 a **pilot service** that is able to serve requests for a medium number of users ($O(100)$). These could be the milestones to achieve:
 - * Consolidate the single-node prototype
 - * Crash test it: feedback from early users
 - * Puppetise it for replication
 - * Allow the user to customize their container through JupyterHub forms, e.g. with respect to the LCG release to be used.
 - * Security check
 - * Evolve to a distributed setup where containers are created in multiple VMs.
 - * Service monitoring and management (e.g. clean dead sessions)



- * SWAN service in production!!

Documentation

- ❖ Website
 - ❖ “Try me” Binder button for ROOTbooks in front page
- ❖ Doxygen
 - ❖ Finish it for good and improve the current graphical look and feel
- ❖ Tutorials
 - ❖ Entirely review the full set of tutorials, modernize them, eliminate non-didactical ones, add new from common questions
- ❖ ROOTbooks
 - ❖ Migrate tutorials to ROOTbooks when possible / useful
 - ❖ Transform ROOT primer into a ROOTbook - with input data
- ❖ User Guides
 - ❖ Update the revise the User Guide and Topical Manuals
- ❖ O'Reilly book
 - ❖ If additional effort is found, write a O'Reilly book for ROOT



New Platforms

- ✦ Make ARM64 and PPC64 1st class citizens
 - ✦ Broken tests have to be fixed and kept green
- ✦ Support for new ABI coming with GCC 5
- ✦ Windows (Visual Studio 2015)
 - ✦ New (latest) version of llvm / clang
 - ✦ Have ROOT 6 fully ported and running
 - ✦ Help porting DAVIX to Windows



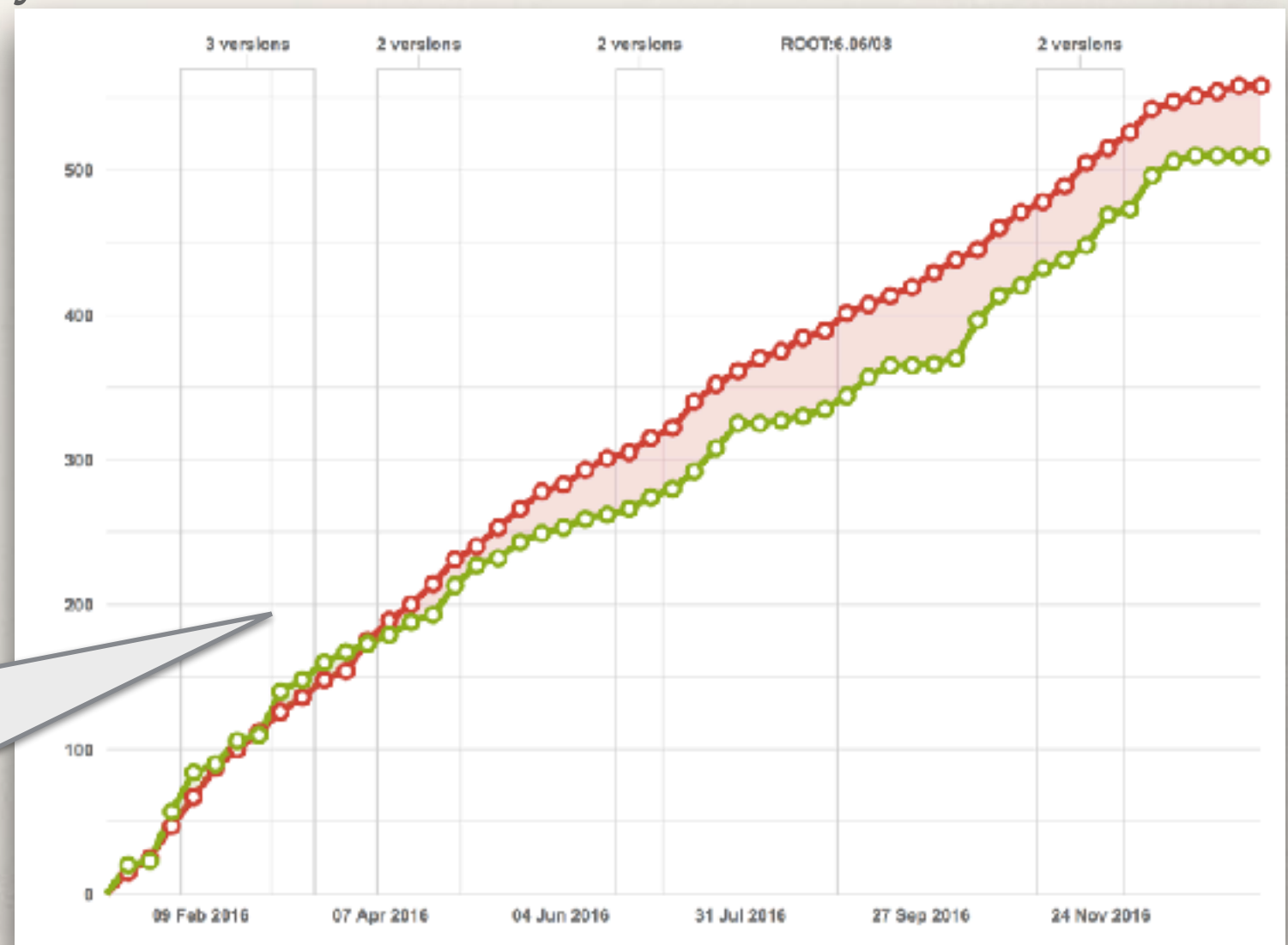
Infrastructure

- * Jenkins
 - * new Jenkins build nodes (e.g. 32 bit nodes, ...)
 - * proper Jenkins report emails
- * Migrate RootTalk forum to the *discourse* platform
- * Coding guidelines checker
 - * Define new coding guidelines
 - * Implement automatic checker with Jenkins



User Support: JIRA

- ❖ Bugs: 558 created and 510 resolved
 - ❖ backlog increased by 48 issues (of total about 1000)
 - ❖ ~4 new bugs / working day
- ❖ Definitely needed a campaign to triage and eventually close 'obsolete' bugs and tasks.



10 releases in 2016 !

User Support: RootTalk

- ❖ In addition to JIRA we have the RootTalk Forum:
 - ❖ Total posts 100112
 - ❖ Total topics 22546
 - ❖ ~ 20 posts / day
 - ❖ ~ 5 new topics / day
- ❖ Weekly shifts to ensure that no post gets unanswered
 - ❖ A lot of effort ~1 FTE
 - ❖ Latency reduced dramatically!



GENERAL	TOPICS	POSTS	LAST POST
Announcements General Announcements. Moderator: rootdev	100	128	by Axel G Wed Jan 20, 2015 12:39
ROOT	TOPICS	POSTS	LAST POST
ROOT Support Discuss installing and running ROOT here. Please post bug reports here . Moderator: rootdev	14695	66326	by moneta G Thu Jan 29, 2015 17:03
ROOT Documentation Discuss the ROOT documentation here. Moderator: rootdev	240	730	by coael G Tue Dec 02, 2014 11:12
Users' Contributions Some general interest ROOT macros and programs provided by ROOT users. If you have such macros or programs you can put them here. Moderator: rootdev	5	18	by schilner G Mon Dec 08, 2014 18:10
PROOF Support Discuss PROOF, the Parallel ROOT Facility, here. Moderator: rootdev	455	2148	by Dmitry G Thu Jan 29, 2015 11:54
Stat and Math Tool Support Discuss RooFit, TMVA and other statistical and mathematical tools here. Please post bug reports here . Moderators: cranmer, rootdev	1542	4416	by neam G Thu Jan 29, 2015 9:36
PyROOT Support Discuss PyROOT, the Python ROOT language binding, here. Moderators: wlas, rootdev	775	3795	by joel.vogl G Fri Jan 16, 2015 11:40
My ROOT App Discuss your own ROOT application. Moderator: rootdev	124	358	by dlapina G Mon Jan 26, 2015 8:17

Main areas of work for 2017

Maintenance and Support

- ❖ JIRA
 - ❖ We have 1035 open bugs with no way to converge
 - ❖ It takes effort to report a bug; they currently stay open for about 1 year in average
- ❖ User contributions
 - ❖ We have 35 pull requests, with several open since >1 year
- ❖ Static Analysis
 - ❖ We have 3800 open Coverity issues
- ❖ Most of team members have a long list of cleanup tasks in the existing code

I/O Functionality

- ❖ Support of `shared_ptr<T>`
 - ❖ Try to do better than the straightforward `shared_ptr<T> → T*`
- ❖ Assess impact of byte swapping on runtime performance
 - ❖ If significant, change the file format ensuring backwards compatibility
- ❖ Zero-copy I/O
 - ❖ New API to deserialize multiple events
- ❖ Enhance morphing capabilities of schema evolution write rules
- ❖ Implement inlined `ClassDef` (ROOT-6284)
- ❖ I/O without dictionaries
 - ❖ Collection proxies (`std::COLL<T>`), interpreted classes, etc.

I/O Performance

- ❖ Multi-treaded file operations
 - ❖ Introduce a new TFile class, potentially less performant than the existing one, but featuring thread safe methods
 - ❖ Parallel write
- ❖ Optimizations
 - ❖ Improved compression of branches holding a non-split collection
 - ❖ Per entry compression
 - ❖ Reduction of overhead of deep inheritance chains
- ❖ Provide help for benchmarking I/O performance in presence of automatically generated POD data models (e.g. HepMC3, PODIO)
- ❖ Comparisons with other data formats (Parquet, Avro, Kudu, etc.)

CLING Interpreter

- ✧ Incorporate LLVM/Clang patches upstream
 - ✧ such that ROOT can use external installation of LLVM/Clang
 - ✧ collaborate with LLDB team to find common points of interest
- ✧ Windows native compatibility
- ✧ Allow to parse and compile functions at the prompt in all cases
- ✧ Lazier compilation (less memory and faster)
 - ✧ E.g. virtual function tables
- ✧ Set JIT optimization level at runtime
- ✧ Revisit current 'work arounds' in roottest/ctest cling driver
- ✧ Improve autocompletion
- ✧ Improve current unloading functionality
- ✧ Make more use of the clang driver

Precompiled Modules

- ❖ Facilitate and support the use C++ modules for the compilation of large software projects (e.g. FCC->LHCb->CMS / Atlas)
- ❖ Use C++ modules as ROOT 'dictionaries'
 - ❖ When modules are sufficiently stable and feature complete enough, start to use them in combination with dictionaries therewith avoiding parsing of large amounts of code at runtime
 - ❖ Ensure dictionary generation backward compatibility

New C++ Interfaces

- ❖ New histograms and graphs
- ❖ New TFile interface
- ❖ New drawing approach “a la RooFit” (no Draw method)
- ❖ Backward compatibility studies:
 - ❖ Assess needs for complete backwards compatibility for I/O
 - ❖ Investigate code-to-code transformations to ease user code migration

The SWAN Service*

- ❖ Migration of the service to IT
- ❖ Interfacing SWAN with Spark resources
- ❖ Improve the user experience when sharing in SWAN
- ❖ Productize the current SWAN service for deploying it in private instances

(*) In collaboration with IT

New Analysis Approaches

- ❖ Improve ROOT notebook interface
 - ❖ Tab-completion
 - ❖ Definition of functions in code cells (without magic)
 - ❖ JSROOT as default graphics
- ❖ Exploration and prototyping of **functional chains**
 - ❖ Both C++ and Python interfaces
 - ❖ Usability, expressibility, performance, etc.
- ❖ Assess the Apache Spark as scheduler in combination with the PyROOT interface to complement PROOF for distributed calculations
 - ❖ Distributed processing of large TTrees
 - ❖ Interface with Spark clusters
 - ❖ Interface IT container service

Machine Learning

- ❖ Finish and release all the parallelization work
 - ❖ External to methods (cross-validation / hyperparameter tuning / envelope methods)
 - ❖ Method-internal (BDT / DNN / SVM)
 - ❖ Multi-threading, multi-process and spark prototypes
- ❖ Regression
 - ❖ Develop multi-objective regression
 - ❖ Expand loss function class to all methods
- ❖ Hyperparameter Tuning
 - ❖ Fully integrate with cross-validation, add hill-climbing, benchmark with Bayesian optimization
- ❖ Preprocessing, Unsupervised Learning and Additional DNN architectures
 - ❖ Develop deep auto-encoders from DNN base
 - ❖ CNN
 - ❖ RBM from DNN base
 - ❖ Additional pre-processing functions

Machine Learning (2)

- ❖ Performance improvements
 - ❖ Identify hotspots (both training and evaluation)
 - ❖ Memory optimization (overall framework and BDT)
- ❖ Interfaces
 - ❖ Improve Keras interface (data I/O)
 - ❖ Assess interfacing C++ to python C interface versus a pure python approach
- ❖ Documentation
 - ❖ Updated examples (+notebooks in gallery)
 - ❖ For beginners
 - ❖ More advanced plus new features

Math Libraries

- ❖ Integrate vectorization and parallelization in ROOT fitting
- ❖ Power TMath with VecCore types (augment it, no replacement)
- ❖ Vectorization of TFormula (using VecCore)
- ❖ Parallel integration (e.g. Vegas)
- ❖ Use vectorization for numerical algorithms (e.g. integration)
- ❖ Investigate automatic differentiation using CLING

RooStats and RooFit

- ❖ Investigate the Hydra package (fitting and MC toys)
- ❖ Use new TFormula
- ❖ Browser for large models to change parameters values via a UI (if effort available)
- ❖ Performance improvements
 - ❖ Improvement of runtime performance in presence of large models starting from real use cases, e.g. from CMS
 - ❖ Use multiprocessing for RooStats calculators
 - ❖ Take advantage of vectorisation, parallelisation on CPUs and / or GPU where it makes sense
 - ❖ Removal of virtual functions (define models at compile time)

Python Bindings

- ❖ Improved type-system (e.g. addition of missing entities)
- ❖ Assess the limitations, usability and necessary improvements in presence of the modernised C++ interfaces
- ❖ “Initializer-list” support (i.e. convert a python collection to a C++ collection)
- ❖ Release PyROOT as a genuine Python package
 - ❖ Co-existing within Python2 and Python3
 - ❖ build/install based on pip package management?

TTree

- ❖ Sparse reading
 - ❖ Compression of individual entries
- ❖ Bring parallel tree merger to production quality
- ❖ Jitted TTreeFormula
 - ❖ Similar to the new TFormula
- ❖ Type-safe TTree::SetBranchAddresses
- ❖ Vectorised TTree::Draw and other interfaces

Parallelism

- ❖ Boost performance of analysis expressed via **functional chain** with multicore based parallelism
- ❖ Take advantage of implicit multi-threading for writing large TTrees
 - ❖ Parallel basket compression
- ❖ Parallel (vectorised and threaded) fitting
- ❖ Interface to distribute calculations on Spark resources following the map-reduce pattern (python, then C++)
- ❖ Unify the forking framework for RooFit and TProcessExecutor
- ❖ Investigate how to interface to an externally provided **scheduler**
 - ❖ typically provided by experiments' frameworks

Graphics

- ❖ Many of the JIRA reports are related to graphics
- ❖ **Design a web-based, platform independent UI and graphics to be the main ROOT graphics platform**
- ❖ Auto / aided placement of legends and axes titles
- ❖ Fixed size fonts by default
- ❖ Complete the TMathText class
- ❖ Spider plot not based on a TTree
- ❖ Slim-down procedure for the JSONs used for visualisation (e.g. in notebooks)
- ❖ Make JSROOT the default visualization for notebooks

Modularization

- ❖ Develop a model for building, distributing and deploying **ROOT modules** that can extend an existing installation of ROOT on demand
 - ❖ Evolve ROOT into `BOOT` (à la R)
- ❖ Package the current ROOT I/O plugins as ROOT modules
- ❖ To cope with package/module dependencies, interface ROOT with popular package managers (e.g. Spack, Conda)

Documentation

- ❖ Review of the Users' Guide, now very outdated
- ❖ Modernisation of Tutorials and inclusion of Python as a first class citizen in all categories
- ❖ documentation / manual / user guide for core / multiproc, ML
 - ❖ Release new users guide; Doxygen

Building and Packaging

- ❖ New build modes
 - ❖ cross-building ROOT
 - ❖ static build of ROOT
- ❖ Embed cloudflare zlib rather than the traditional one
- ❖ Make ROOT.py less monolithic, allow for several python packages

Infrastructure

- ❖ Migrating master ROOT repository to GitHub
 - ❖ Adapt contribution procedures (PR, etc.)
 - ❖ Integration with SFT's Jenkins CI
- ❖ Testing infrastructure improvements
 - ❖ Add coverage, memory checks, etc.
 - ❖ Performance regressions
 - ❖ Test guidelines and documentation
- ❖ Using containers for building / testing in Jenkins
- ❖ Migrate RootTalk forum to the *discourse* platform
- ❖ Get rid of <hash>.cern.ch / reverse proxy with shibboleth

Tutorials and Courses

- ❖ Continue to guarantee the Summer Student' workshop
- ❖ CERN Technical Training
 - ❖ Preparation of the proposed courses (introductory and advanced)
- ❖ ROOT Users' workshop
- ❖ Participation to HSF Community White Paper
- ❖ CERN School of Computing (CSC, tCSC, iCSC)
- ❖ Reinforce engagement with analysis community, most notably with CERN groups