ROOT's 2019 Plan of Work

Axel, ROOT/Experiments meeting, 2019-02-06

The Themes

- **User support**
- **Robustness** (see e.g. new interfaces, testing, bug fixes)
- **Ease of use** (see e.g. RDF, TMVA)
- **Performance** (see e.g. RooFit, benchmarking)

- Whatever actually helps analyses!

TMVA

- Ease of use: interoperability (ROOT, numpy, current TMVA, sklearn, ...); support inference with externally trained models
  - Python-like C++ vector with shape info; Numpy arrays as input in Python
  - modularize Factory / Reader steps
- Performance: fast inference/application, high-throughput and low-latency, consistent across models / tools
- Focus on HEP-specific statistical compatibility checks (Kolmogorov–Smirnov), externalize / deprecate others (TMVA GUI)

RooFit

- Ease of use: improved python and collection interfaces
- Performance: less virtual calls, cache friendly, bulk data; parallelization of sub-tasks (e.g. PDF normalization)
- Robustness: test coverage, benchmarks
- More minimizers (GSL, scipy.optimize?)

Math

- Implement fitting for new histograms; collect usage feedback
- PRNG: VecMath-based, common for ROOT and Geant4/V; vectorized + MT-enabled; all standard algorithms; wrapped by ROOT
    - for ROOT, add RanLux++ and update RunLux implementation
- Investigate new minimizers, see scipy.optimize

WebGraphics, WebGUI, WebEve

- Provide "minimum viable product" for histogram graphics
- Decide on new default graphics style
- RFitPanel, RBrowser, file dialog; embedding in JupyterLab
- Eve7: a working prototype, coordinating with CMS Fireworks-Web development
- Testing

I/O

- Parallelism bottlenecks, TTreeReader performance fixes; parallel merger
- Consolidate tests and benchmarks
- RForest prototype and tutorials
- Explicit error handling for I/O
- Compression: rationalize compression setting, LZ4 (+Bitshuffle) default, CloudFlare zlib, investigate ZSTD
- std::shared_ptr, std::variant

Analysis tools: RDataFrame

- Nested loops, multi-dim (esp 2D) arrays for C++: RTensor
- Improved integration with Python, RHist, RForest
    - prototype CUDA RDaraFrame kernel "fed" from RForest!
- Bulk-I/O RDataFrame; dissolve event boundary
- Multi-dim category operations: pT-bins, sample, data/MC,...; including dedicated weights and systematic uncertainties

More modern and pythonic PyROOT

- Experimental pyroot (with current cppyy) to become default
- Improved pythonizations
    - extensible decoration mechanism; provide more + document + test
- Pythonic RooFit
- Multi-version PyROOT installs: multiple Python versions, multiple ROOT versions

Modules

- Serve as performance improvement for frameworks
- Complete features to enabled by default: all of ROOT, all platforms, incremental builds
- Optimize performance
- Help adoption by experiments' frameworks

Lazy builds, CMake, Platforms

- Ease of use, robustness: build / update parts of ROOT on top of existing build
- Robustness: re-define CMake interface for enabling / disabling parts
  - make fail-on-missing default, feature-based ("xmlio") not dependency-based ("libxml2")
- Ease of use: update packaging for deb (+ Ubuntu PPA?), rpm, MacOS
- Robustness: benchmark coverage + tracking, enable (+fix!) sanitizer builds
- Windows: finalize 32bit; prototype 64bit

Documentation + Training

- One day "train the trainer" event to spread the news + collect feedback
- `root [0] .help TTree`
- Rewrite crucial classes' documentation, rewrite old tutorials
- Prepare ROOT website generated from git (move away from Drupal)
  - focus on crucial, updated pages

Summary

- **User support**
- **Robustness** (see e.g. new interfaces, testing, bug fixes)
- **Ease of use** (see e.g. RDF, TMVA)
- **Performance** (see e.g. RooFit, benchmarking)

Whatever actually helps analyses!

Plan

- Presentation in EP-SFT: Feb 11
- Updates during the year @ ROOT/Experiments meetings
  - progress, change of focus
  - your requests!
- As always :-)

Discussion

- What would you benefit from for your framework?
- Any input related to analyses?