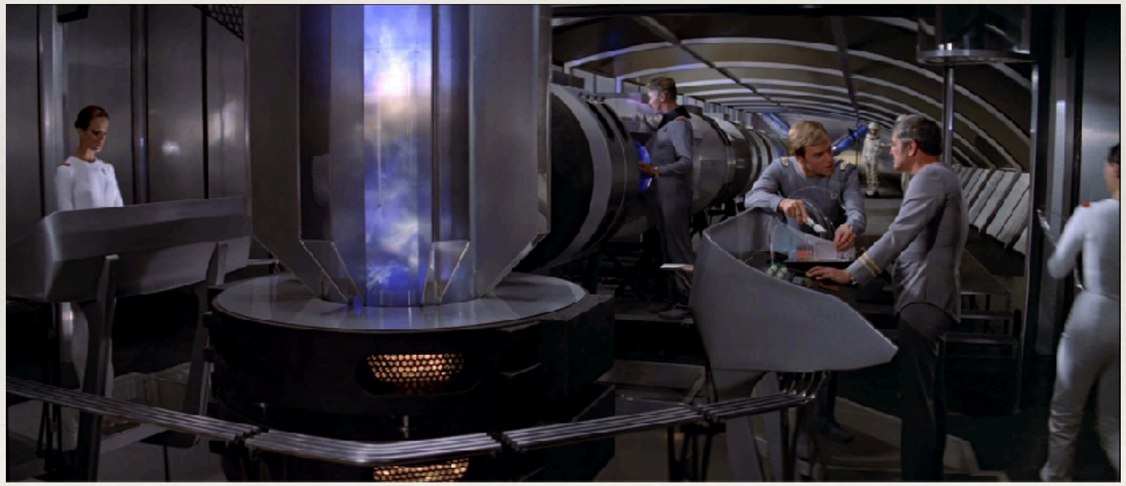
EP-SFT Meeting, 2018-03-19

ROOT in 2020

Axel Naumann axel@cern.ch
For the ROOT Team

Role

- ROOT is the warp core of HEP data analysis
  - I/O
  - Math
  - Visualization
  - Interactivity, analysis, documentation
- Not guaranteed - there are competitors!



Context

- ROOT is at the center of data analysis since +/- 15 years
  - today, the world offers lots of open source tools for big data
- ROOT provides expertise to the community by the community
  - sustainable tools by ensuring in-house expertise
  - many solutions specific and optimized for HEP
  - alternatives are often not direct solutions
- And yet: 20 years of success is not a guarantee for the future!

The Mission

- Need to convince:
  - simplicity
  - robustness
  - speed
  - features
- Good code + good code + good code.

Simplicity

- Focus on physics, reduce time spent on coding (and debugging!)
  - clear, consistent C++ interfaces (modern C++ helps)
  - excellent Python support (more pythonic ROOT a la rootpy)
  - do things perfectly by default, but allow for customization
- Separation of concerns (e.g. histograms from graphics from I/O)

Robustness

- ROOT's memory model based on PAW:
  - directories own named objects etc
  - causes crashes due to raw pointers and implicit ownership
- ROOT 6: arrays are pointers, configuration through strings
- Instead: let the compiler check where possible, substituting runtime errors

Speed

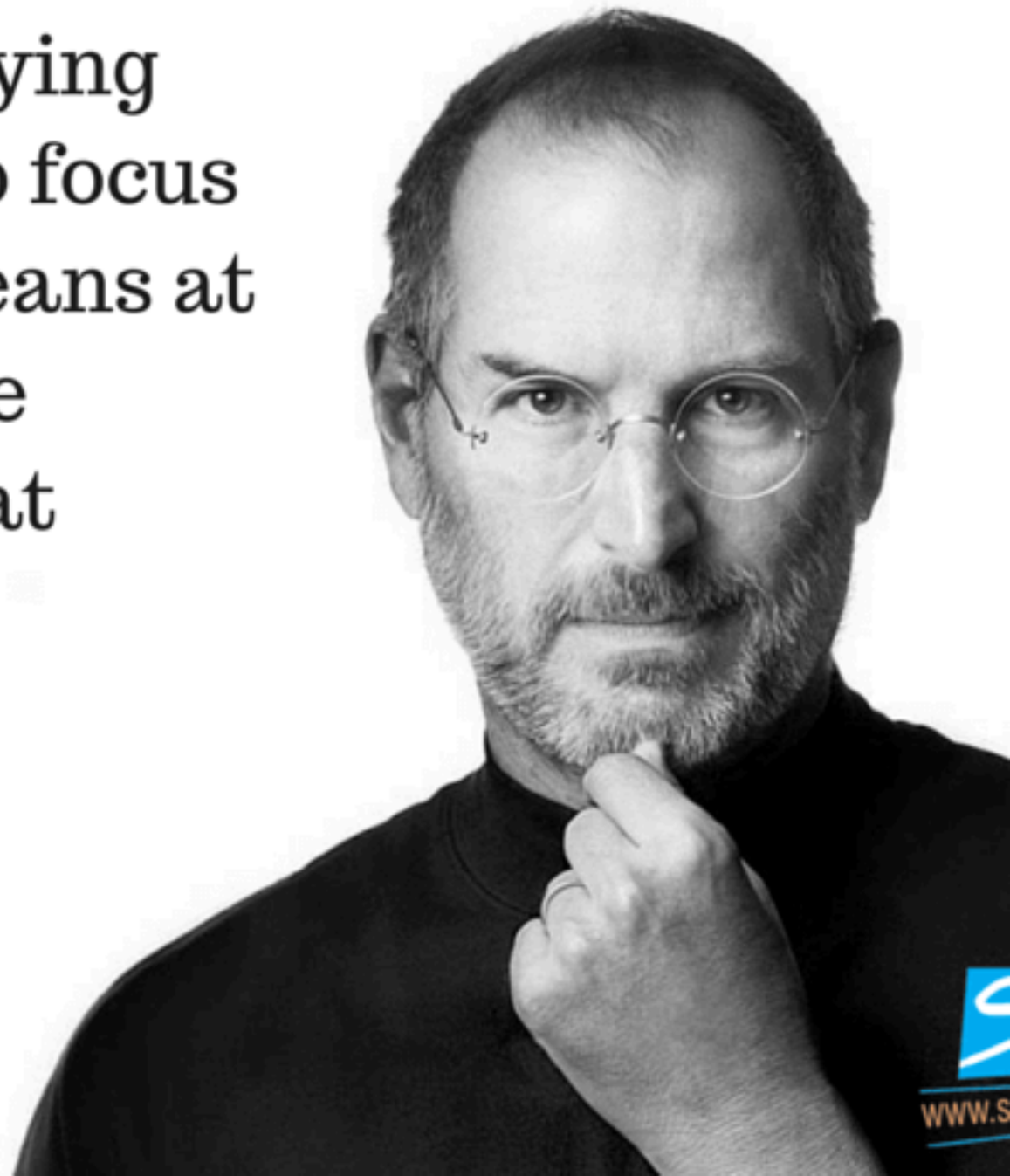
- C++ from 1995 was all about object oriented code
  - has proven to incur a performance cost
- Instead: modern design, bulk operations where possible, less virtual functions / more vectorization and cache locality
- Thread-safe, context-free objects

Features

- ROOT's smallest issue :-) But:
- Simple install
- Simple (i.e. "non-detail") programming model
- Using all cores by default whenever possible, vectorize internal operations
- Continue to follow needs of physicists before everyone is aware (think TMVA, RooFit)
- Focus!

People think focus means saying yes to the thing you've got to focus on. But that's not what it means at all. It means saying no to the hundred other good ideas that there are. You have to pick carefully.

Steve Jobs



“ROOT 7”

- New interfaces, using modern C++ for simplicity, robustness and speed
  - change interfaces after 20 years, and then freeze them again
- Keep interfaces readable for current ROOT users
  - `canv->cd(); hist->Draw();` becomes `canv->Draw(hist);`
- Expose new interfaces early, release gradually
  - see ROOT::Experimental [link], available with -Dcxx14=On

I/O, TTree

- Want to to be extremely performant:
  - 0-copy where possible
  - I/O using all cores, best compression algorithms
  - multi-thread-friendly: one tree, many entries analyzed by multiple threads
- Robust interfaces: type-safe (no void*), explicit memory ownership
- Optimized for I/O devices of 2020: SSD, 3D XPoint, network

Histograms

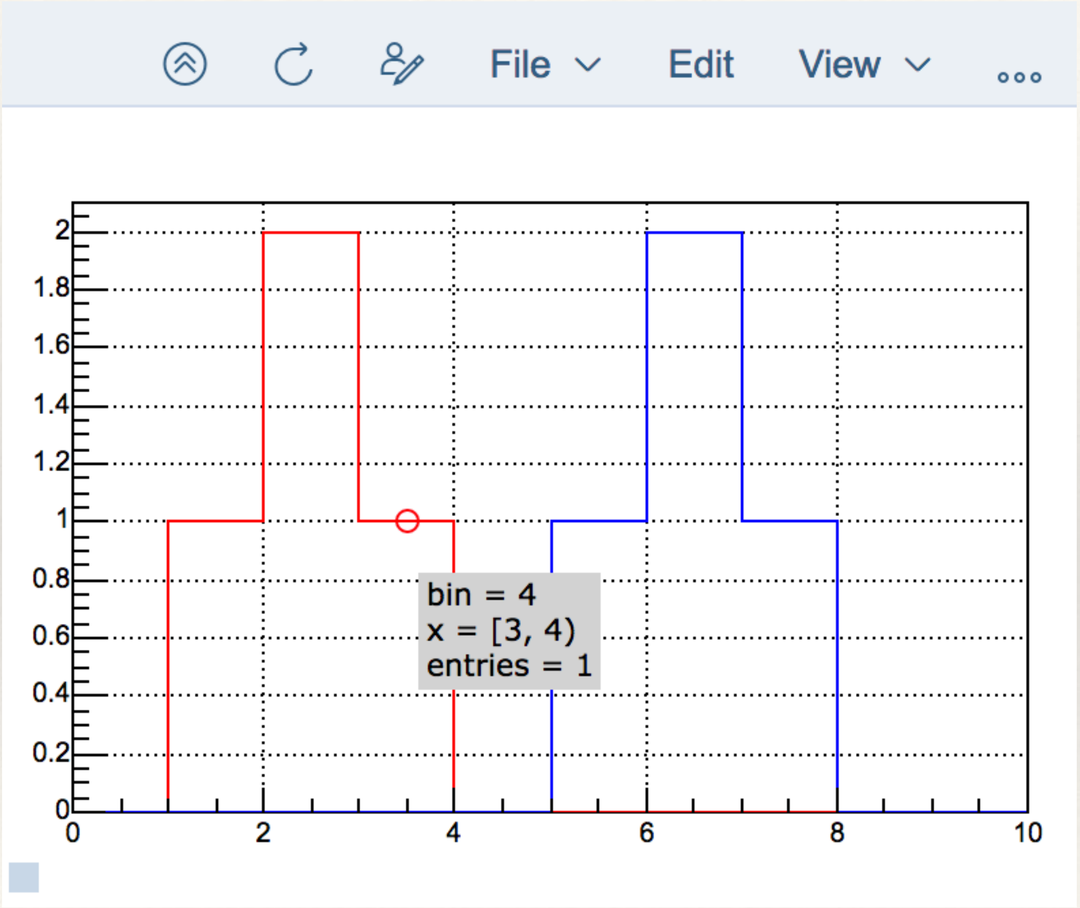
```
// Create a 2D histogram with an X axis with
// equidistant bins, and a y axis with irregular
// binning.
Experimental::TH2D hist({100, 0., 1.},
                        {{0., 1., 2., 3., 10.}});

// Fill weight 1. at the coordinate 0.01, 1.02.
hist.Fill({0.01, 1.02});
```

- Fast and simple
  - shield advanced features
    from basic ones: offer both high-performance interface and usability layer
- Composable and configurable, enabling histogram algorithm library, operating on “any” histogram
- Transform embedded histogram concepts into first-class citizens:
  - axis definition, histogram range, iteration, bin index, bin content storage

Parallel, Simple Analysis

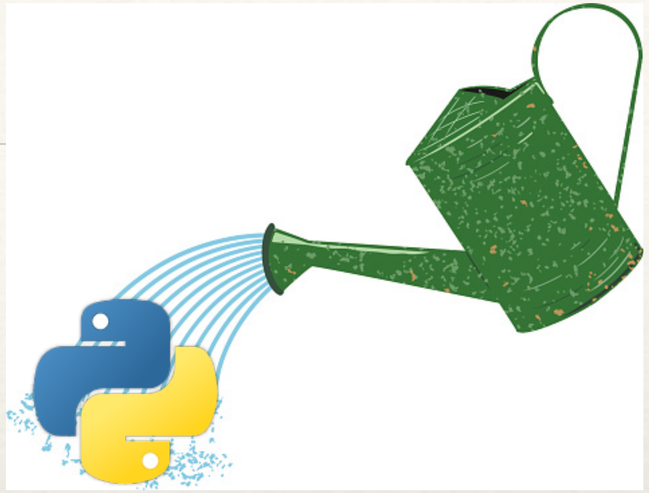
- Currently: you specify reading, looping, selections, output / slimming / skimming (= caching), histogramming; always run everything and on one core - or have smart code and spend time on infrastructure (or TSelector)
- What we want:
  - you focus on the selection, projections, algorithms
  - ROOT takes care of the boring stuff: reading, looping, scheduling, parallelizing - as efficiently as possible
  - with beautiful and efficient Python interfaces

WebGUI Graphics



- WebGUI Graphics = HTML + JavaScript + CSS + OpenUI5 + three.js plus D3.js
- Replace custom GUI with Win32 GDK, X11, Cocoa and GL back-ends (and what about Wayland?!)
- Local and remote interaction, extensible painters, future-proof, beautiful graphics

PyROOT



- It's unique - the world is jealous.
  Move from maintaining it to growing it!
- Expand it beyond "C++ to Python":
  - add "pythonic" interfaces a la rootpy
- Enable fast-path interfaces, e.g. to numpy arrays
- Design C++ interfaces such that they play nicely with Python
  - ownership, type-safety, compact code

Build and Install

- Make binary installs simple for users
  - easy install means happy physicists
  - install core parts, build extensions as needed, on demand
- Make it simple to build
  - allow for e.g. experiment's of physics group's extensions
  - ROOT "package manager"

What to do with a Wonderful ROOT

- If we are happy, why should *only* we be happy?
  - impressive amount of patches, feedback, general applause for cling
  - also warm fuzzy feeling :-) - motivation from public praise
- We want to reach out!
  - we must track us vs them, anyway
  - then tell the world about it

ROOT @ World

- Several features are relevant for the world, but also for HEP
  - time-window TTree
  - documentation
  - stability: every physicist's curse corresponds to an uninstall in bio-physics
- Candidate areas to track are bio-physics, genome, finance, ITER
  - we need to reach out ourselves: invest

Conclusion

Summary: ROOT @ 2020

- New ROOT core(e.g. histograms, TTree): simpler and more robust
- Simple, efficient and composable analysis using all your cores
- Passing data efficiently into machine learning tools, be it TMVA or external
- Web-based graphics

Bottom Line

- ROOT's main goals:
  - simplicity
  - robustness
  - speed
- Keep ROOT at the heart of physicists' data analysis, and make it nice!
- Focus on physicists: efficiency: brain / second, more than CPU / second