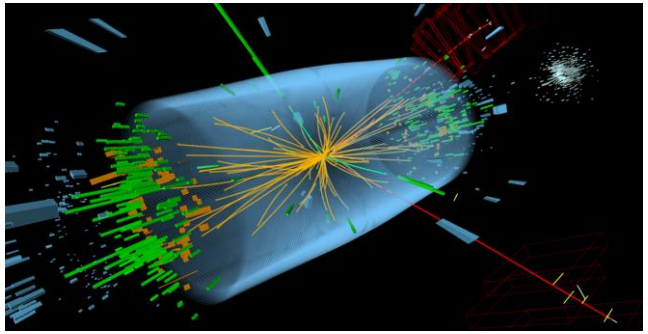
Python
in
High Energy Physics

Pratyush Das

(Institute of Engineering and Management)

What is High Energy Physics?

- Branch of physics that studies the nature of particles that constitute matter and radiation.
- Also known as Particle Physics.
- Asks the big questions - What is our Universe made of? What forces govern it?
- Has the largest scientific collaborations - CMS, ATLAS, etc..



CMS
Compact Muon Solenoid

ATLAS
EXPERIMENT

The first computers were built for Physics

(excluding secret code breaking computers)

1944: John Mauchly (physicist) and J. Presper Eckert (electrical engineer) designed ENIAC to replace mechanical computers for ballistics.

ENIAC was one of the first computers driven by machine code instructions, stored as a program in memory.

Los Alamos group led by Nicholas Metropolis, developed Monte Carlo techniques for physics problems.

Visible influence of Physics

1945: John von Neumann learned of the work on ENIAC and suggested using it for nuclear simulations (H-bomb).

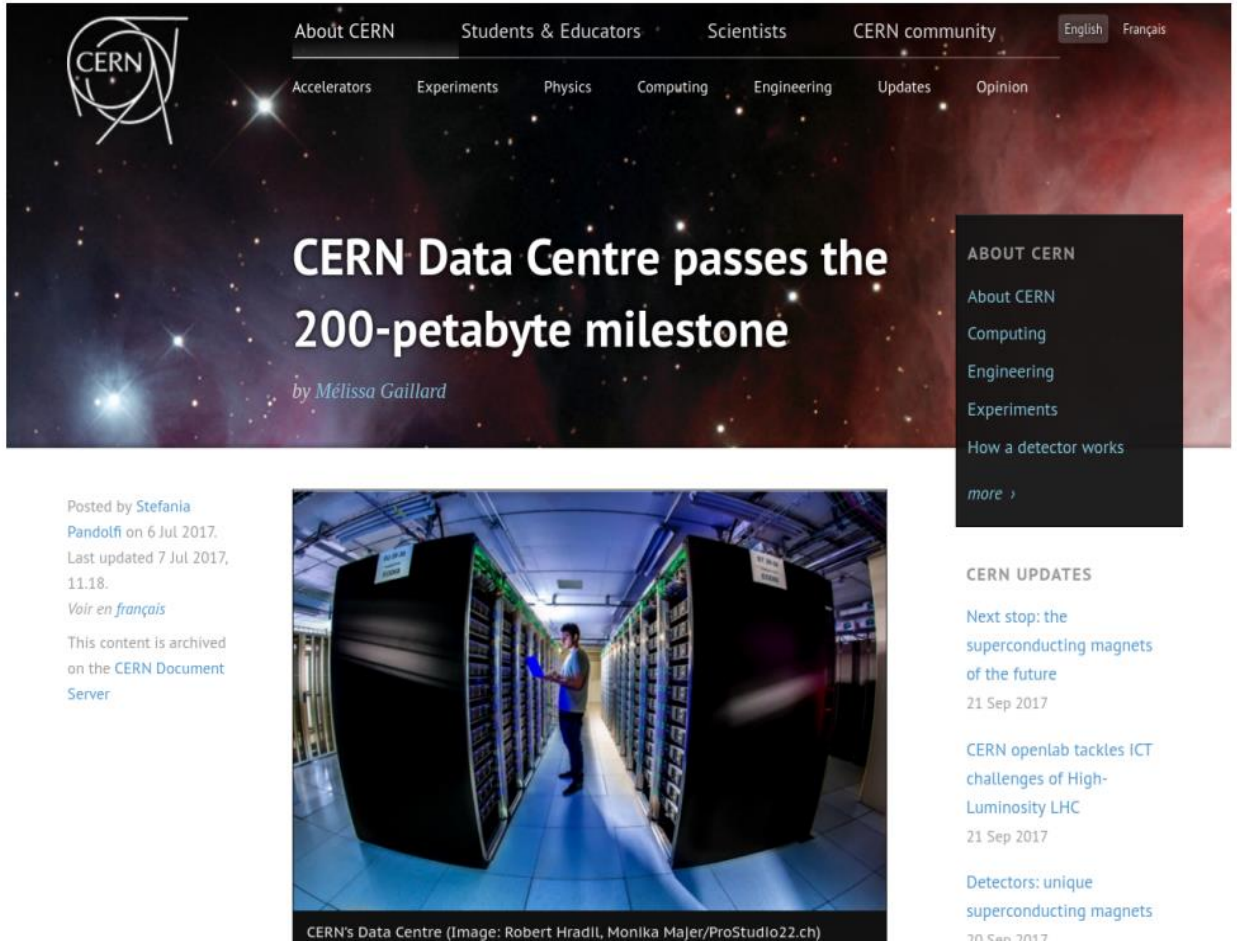
His internal memo was leaked; now known as “Von Neumann architecture.”

1952–1959: At Remington Rand, Grace Hopper developed a series of compiled languages, ultimately COBOL.

1991: Tim Berners-Lee invents the World Wide Web at CERN.

Computing problems in HEP

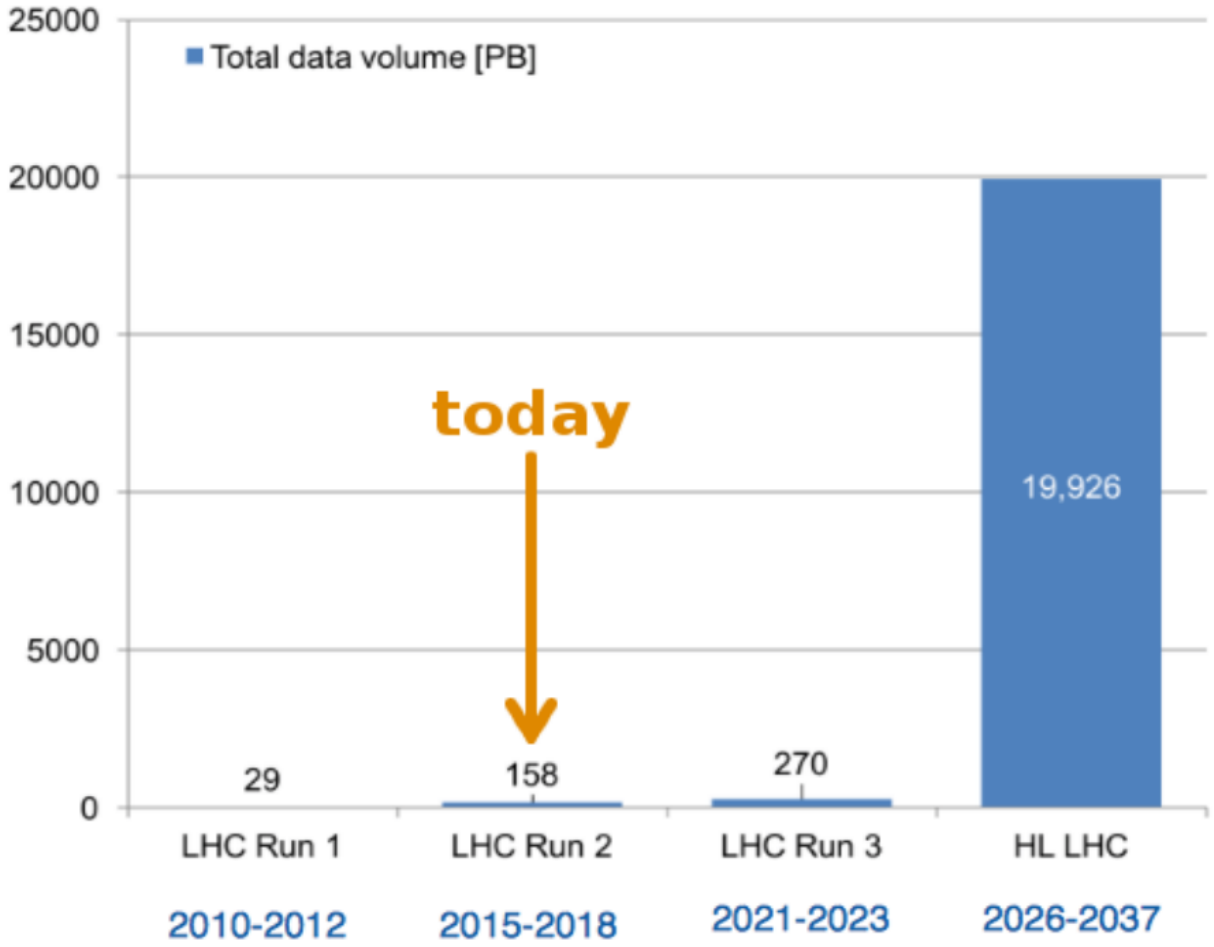
It's big data...



- Jim Pivarski, Strange Loop 2017

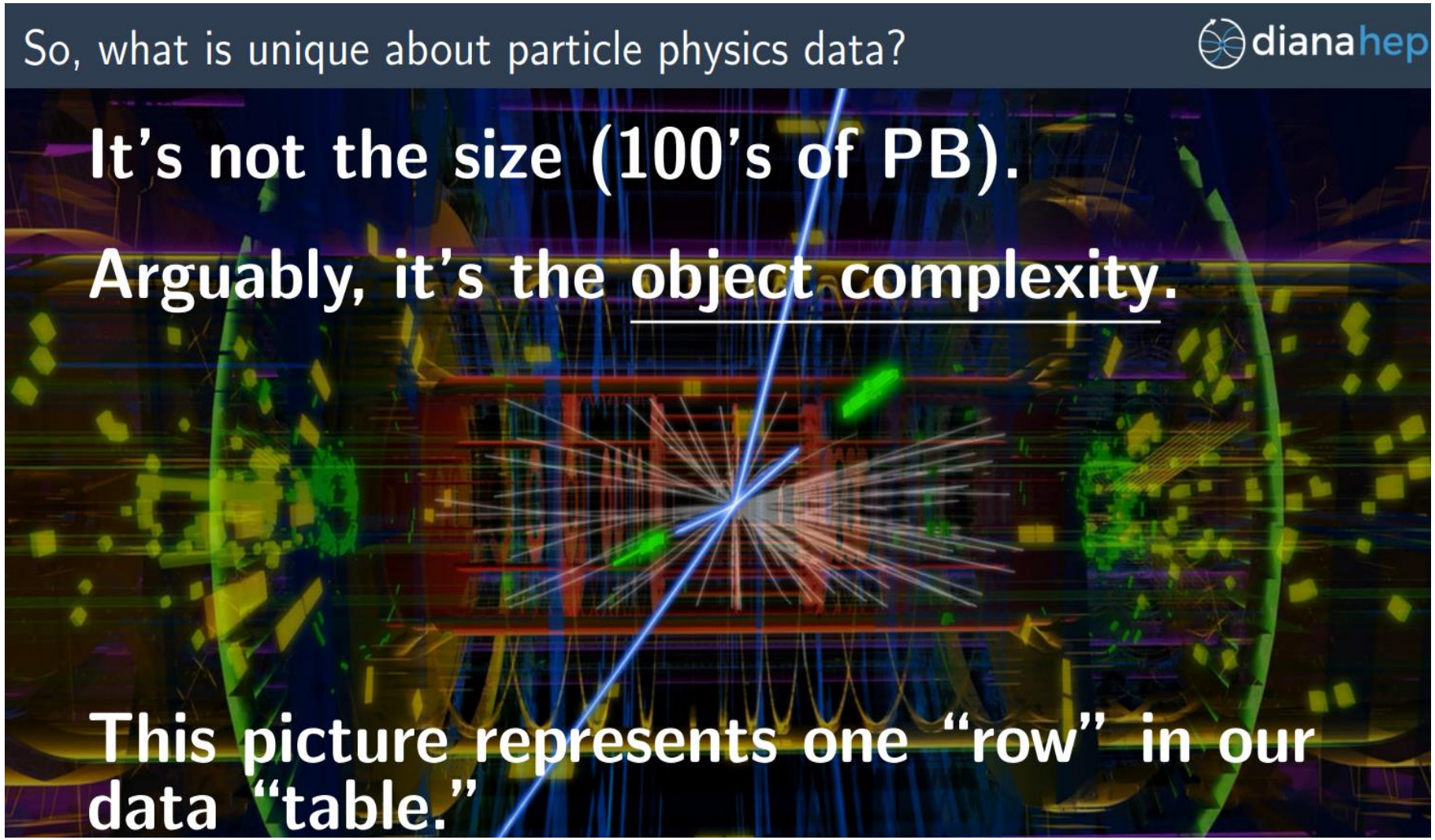
Computing problems in HEP

hand, it will be getting bigger. . .

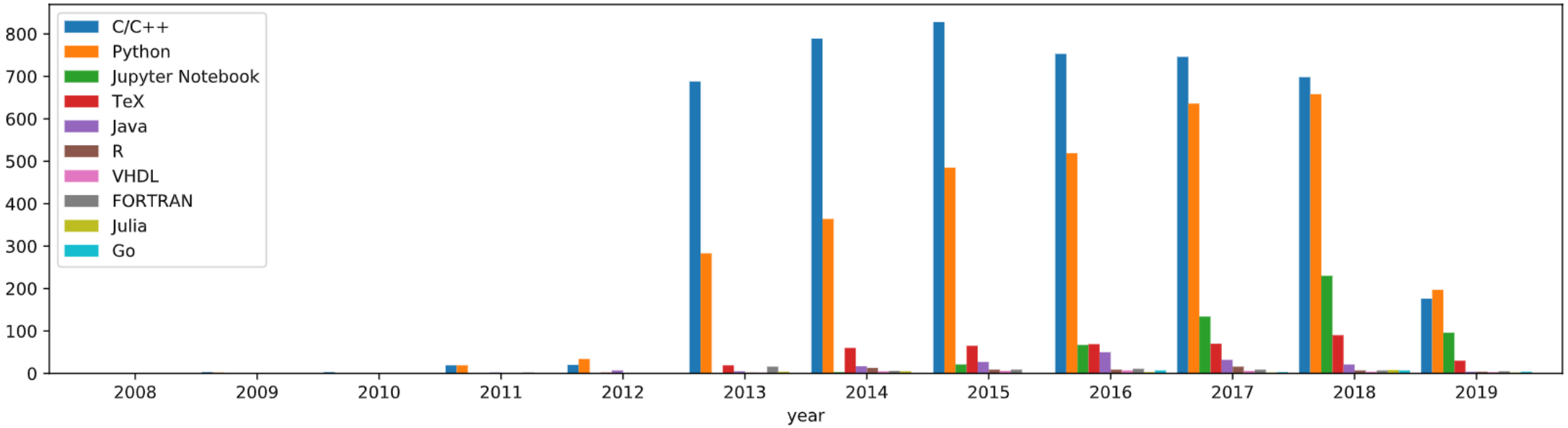


- Jim Pivarski, Strange Loop 2017

Computing problems in HEP



- Jim Pivarski, Strange Loop 2017



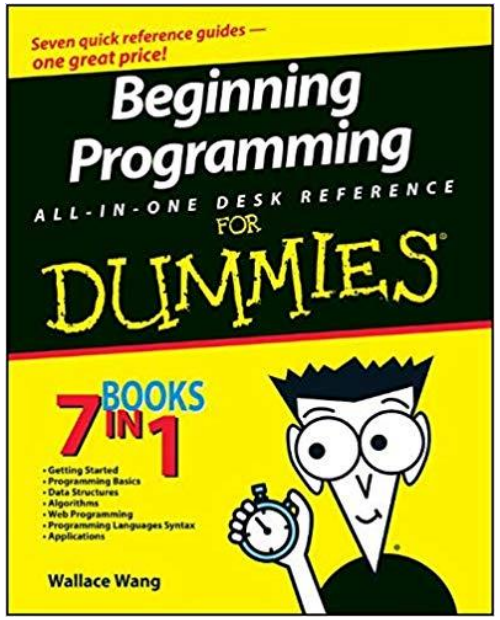
Programming Languages in High Energy Physics

Two programming languages have dominated in the field of High Energy Physics.

- Upto early 1990s - Fortran
  - PAW, HBOOK, ZEBRA
- Early 1990s to Present Day - C++
  - ROOT
- Future - Python?

Physicists drove programming language development in the 1940's and 1950's but stuck with FORTRAN until the 21st century.

Requirements of a language to be used in HEP



Easy

Cython

NumPy

Fast

Worldwide, Aug 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.73 %	+4.5 %
2		Java	20.0 %	-2.1 %
3		Javascript	8.35 %	-0.1 %

Mainstream

How do HEP physicists work with data?

Every HEP physicist uses ROOT.

ROOT is ...

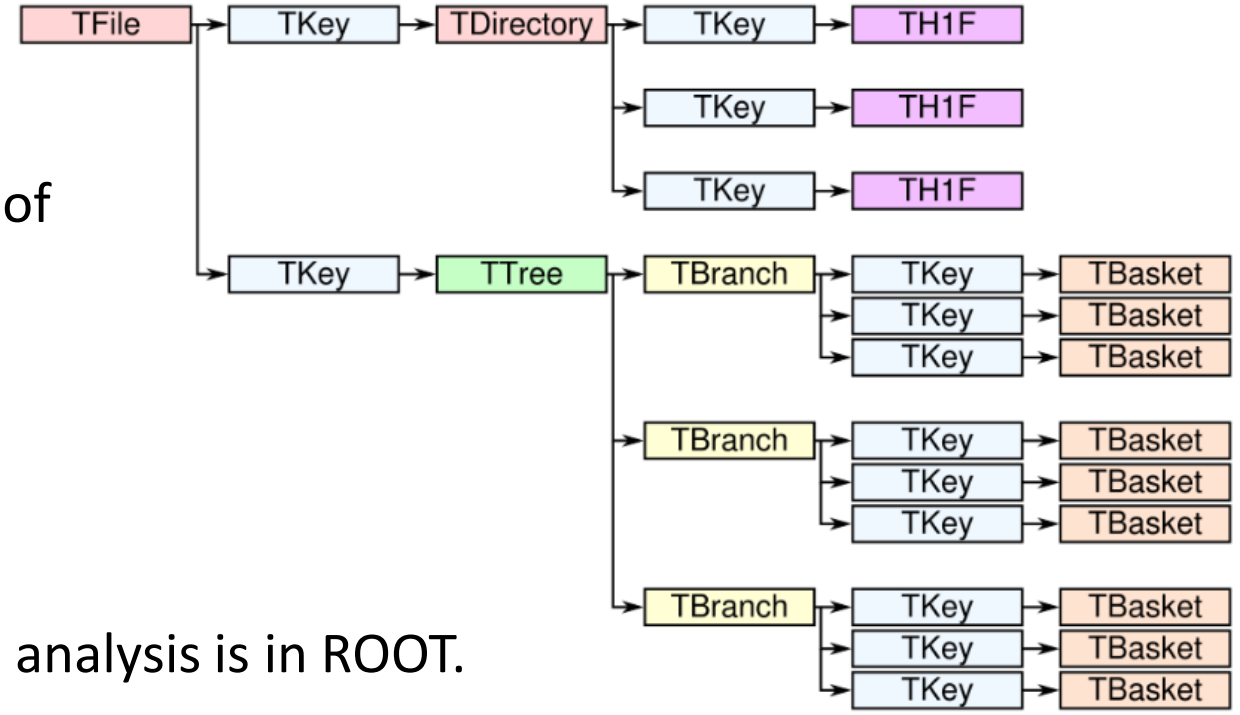
A modular scientific software toolkit. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++

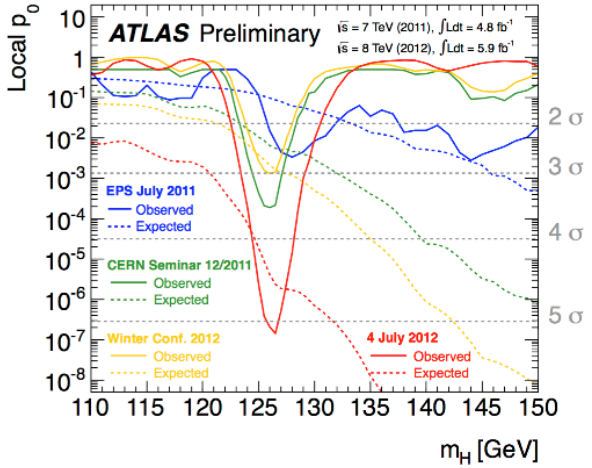
It really provides all the functionalities - From plotting graphs to machine learning libraries, all in one monolithic package.

Primary reason for dominance of C++ in High Energy Physics.

What is ROOT?

- What is all this petabytes of data? – ROOT Files.
- It is a file format used for storing physics data – one of the largest open source file formats.
- Computing in HEP ***is*** ROOT.
- The whole HEP ecosystem from detector collision to analysis is in ROOT.





Discovery of Higgs Boson

ROOT and Python

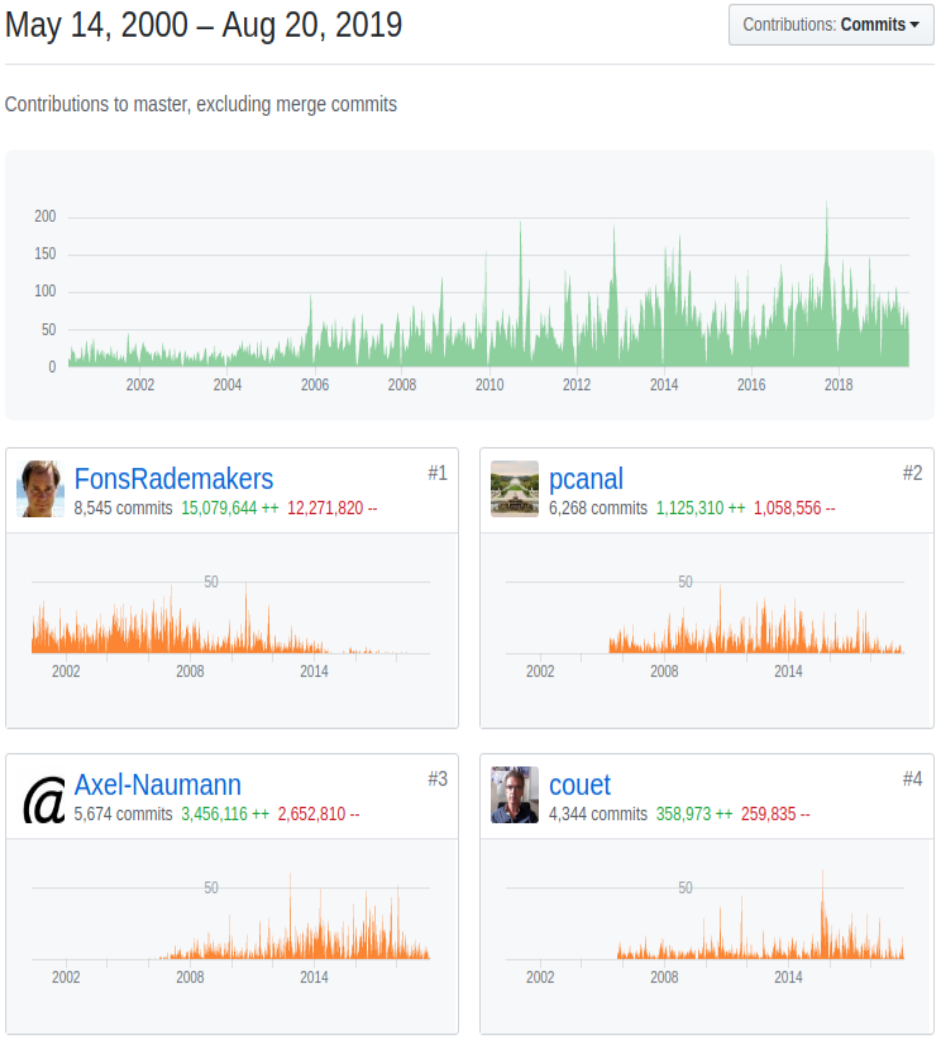
- ROOT has Python wrappers around its C++ code - PyROOT.
- But ROOT has a huge codebase with rapid development - Tedious to add python bindings.
- Dynamic Python bindings - **cppyy**

High-performance Python-C++ bindings with PyPy and Cling

Updated August 24, 2018; see also: http://cppyy.readthedocs.io/en/latest/
Performance results collected for cppyy1.2.4, with PyPy2 6.1 (pre-release), and Python3.7

Wim T.L.P. Lavrijsen
Lawrence Berkeley National Laboratory
Berkeley, California
Email: WLavrijsen@lbl.gov

Aditi Dutta
Nanyang Technological University
Singapore
Email: aditidutta137@gmail.com

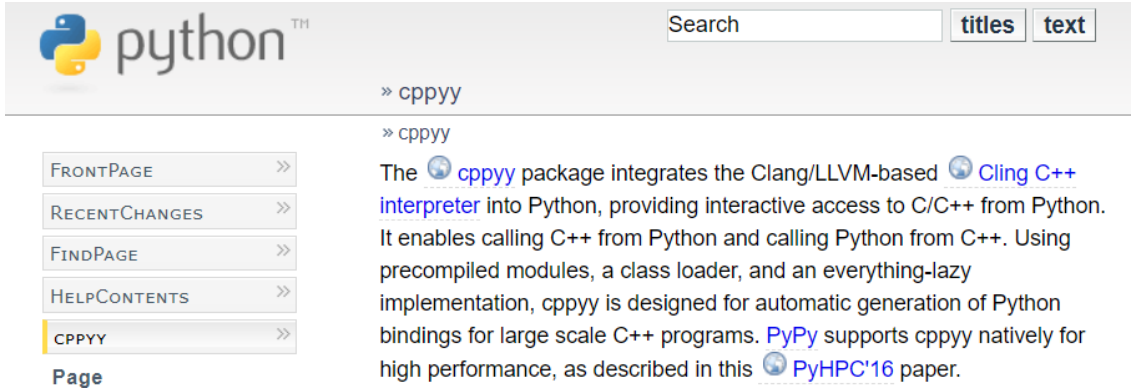


cppyy

- ROOT doesn’t have separate files for each C++ class to link it to Python.

- PyROOT comprises of just 3 main files for generating python bindings from C++ -
  - ROOT.py
  - Cppyy.py
  - _pythonization.py

- Initially developed deeply integrated with ROOT.



Being re-written by the author(Wim Lavrijsen) as a stand-alone library.

Alternate implementation of ROOT in Python

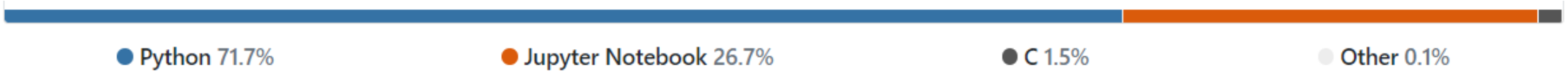
Although PyROOT is improving with cppyy support, it still lacks in some things -

1. Object ownership issues between C++ and Python
2. Not completely Pythonic
3. Slow to deal with certain types of data - jagged arrays

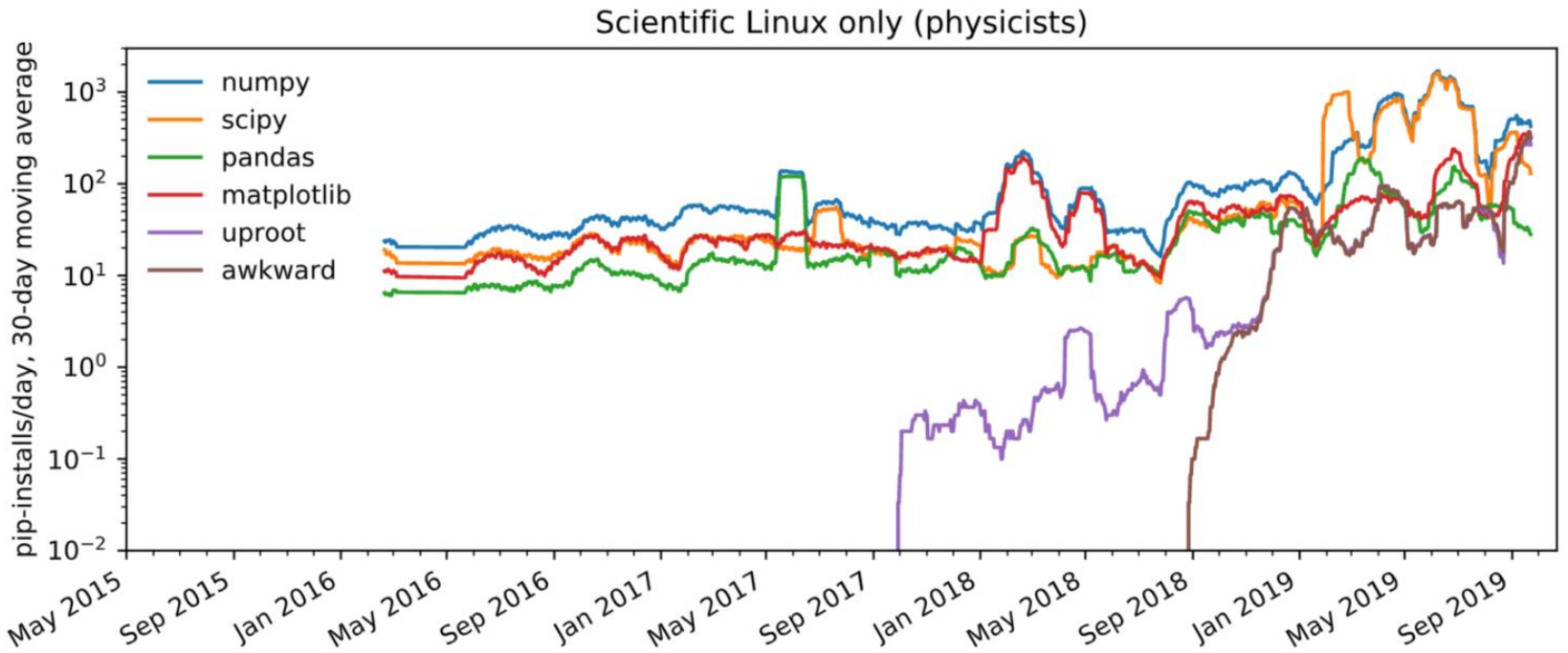
There exists an implementation of ROOT I/O in purely Python and Numpy - **uproot**.

Since it is written in python, it implicitly solves the first two issues.

uproot – Harbinger of Python in HEP?



- Really just Python.
- Very popular for a new package in HEP.



(I am one of the 2 core developers)

Python is not so slow

Simple arrays >> lists of custom objects

Object creation has a penalty even in C

Example: make a histogram of tau-tau jets in CMS

0.018 MHz	full framework (CMSSW, single-threaded C++)
0.029 MHz	load all 95 jet branches in ROOT
2.8 MHz	load jet p_T branch (and no others) in ROOT
12 MHz	allocate C++ objects on heap, fill, delete
31 MHz	allocate C++ objects on stack, fill histogram
250 MHz	minimal "for" loop in memory (single-threaded C)

Pivarski, J. et. al. "Toward real-time data query systems in HEP" ACAT 2017 proceedings, arXiv:1711.01229

In python, object-level code is particularly slow

... but numpy and numba allow array ops at native performance

- Christopher Tunnel, PyHEP 2018

Scikit-HEP project - welcome!

The *Scikit-HEP project* is a community-driven and community-oriented project with the aim of providing Particle Physics at large with an ecosystem for data analysis in Python. The project started in Autumn 2016 and is under active development.

It is not just about providing core and common tools for the community. It is also about improving the interoperability between HEP tools and the scientific ecosystem in Python, and about improving on discoverability of utility packages and projects.

For what concerns the project grand structure, it should be seen as a *toolset* rather than a *toolkit*. The project defines a set of five *pillars*, which are seen to embrace all major topics involved in a physicist's work. These are:

- **Datasets**: data in various sources, such as ROOT, Numpy/Pandas, databases, wrapped in a common interface.
- **Aggregations**: e.g. histograms that summarize or project a dataset.
- **Modeling**: data models and fitting utilities.
- **Simulation**: wrappers for Monte Carlo engines and other generators of simulated data.
- **Visualization**: interface to graphics engines, from ROOT and Matplotlib to even beyond.

Machine Learning

The recent surge in development of machine learning algorithms, particularly deep learning has played a major role in the shift from ROOT and C++ to Python in the form of PyROOT and uproot.

ROOT's machine Learning library TMVA cannot keep up with industry standard libraries such as PyTorch and Tensorflow.

Soumith Chintala Facebook AI Research

Speaker at Automatic Differentiation and Deep Learning

PYTORCH

Industry leaders in Machine Learning are invited to give talks at HEP conferences.

Hear from a Physicist

- 90%+ of what I write won't be used again
- I care about the time it takes to (write + execute)
- Designed to be readable
- Good libraries minimise boilerplate while remaining flexible
- Huge ecosystem
  - Tends to be well documented
  - StackOverflow answers for everything
  - 🔍 Code bases of packages tend to be understandable: no complex inheritance, templating, typedefs

- Chris Burr, PyHEP 2018

Two ways to approach adoption

Python bindings for existing projects using cppyy.

- It will be a few years till cppyy is mature enough to be used widely outside of ROOT.
- Once it is generalized, will be revolutionary.

Rewriting projects in Python using the existing python ecosystem.

- Uproot has proved that it is possible to rewrite integral parts of HEP software in Python.
- Impossible to rewrite everything in Python.

Concluding Remarks

- Python is a popular language, even in sciences where performance is critical.
- It has good features for readability and is easy to learn, especially by scientists whose primary interest is not programming.
- Python is the most natural bridge to machine learning and other statistical software written outside of HEP.
- Newcomers are familiar with Python libraries like Pandas and Tensorflow as compared to their HEP alternatives like TMVA.
- Growth of the python ecosphere outperforms growth of C++ ecosphere.
- Python is here to stay!

THANK YOU