NanoAOD with python

Colin Bernet

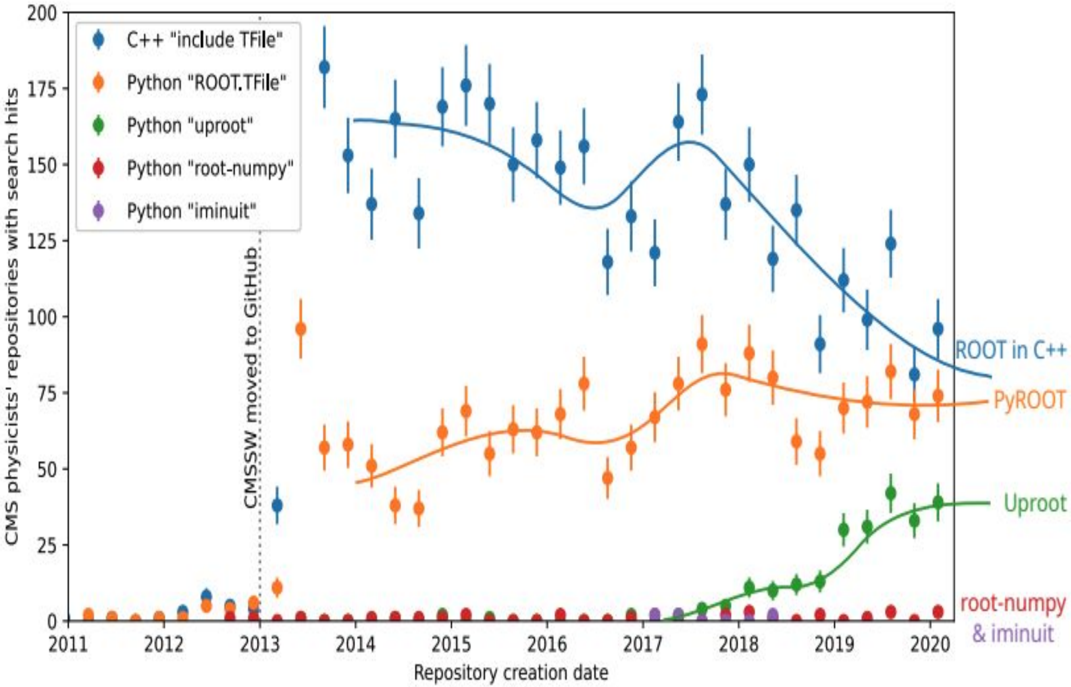
Disclaimer

- This is my view of
  - what is the global situation now
  - what will happen in the future for CMS data processing,
  - what I would do in your place

- You are of course free to choose your path

- Your current choice (RDataFrame) is ok
  - but: this is C++, performance is limited w/r to other solutions
  - just don't go back to PyROOT :-)

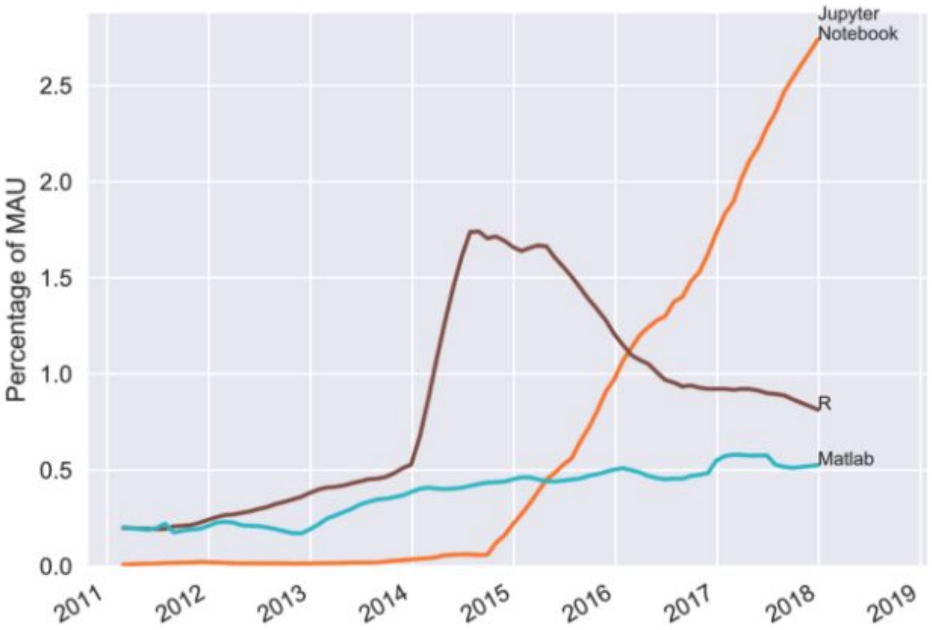
More and more people use python at CERN



- CERN :
  - 2007 : PyROOT
  - 2008 : python for CMSSW config files (B. Hegner, C. Jones)
  - 2009 : PyROOT + FWLite
  - 2011 : heppy (C. B.)

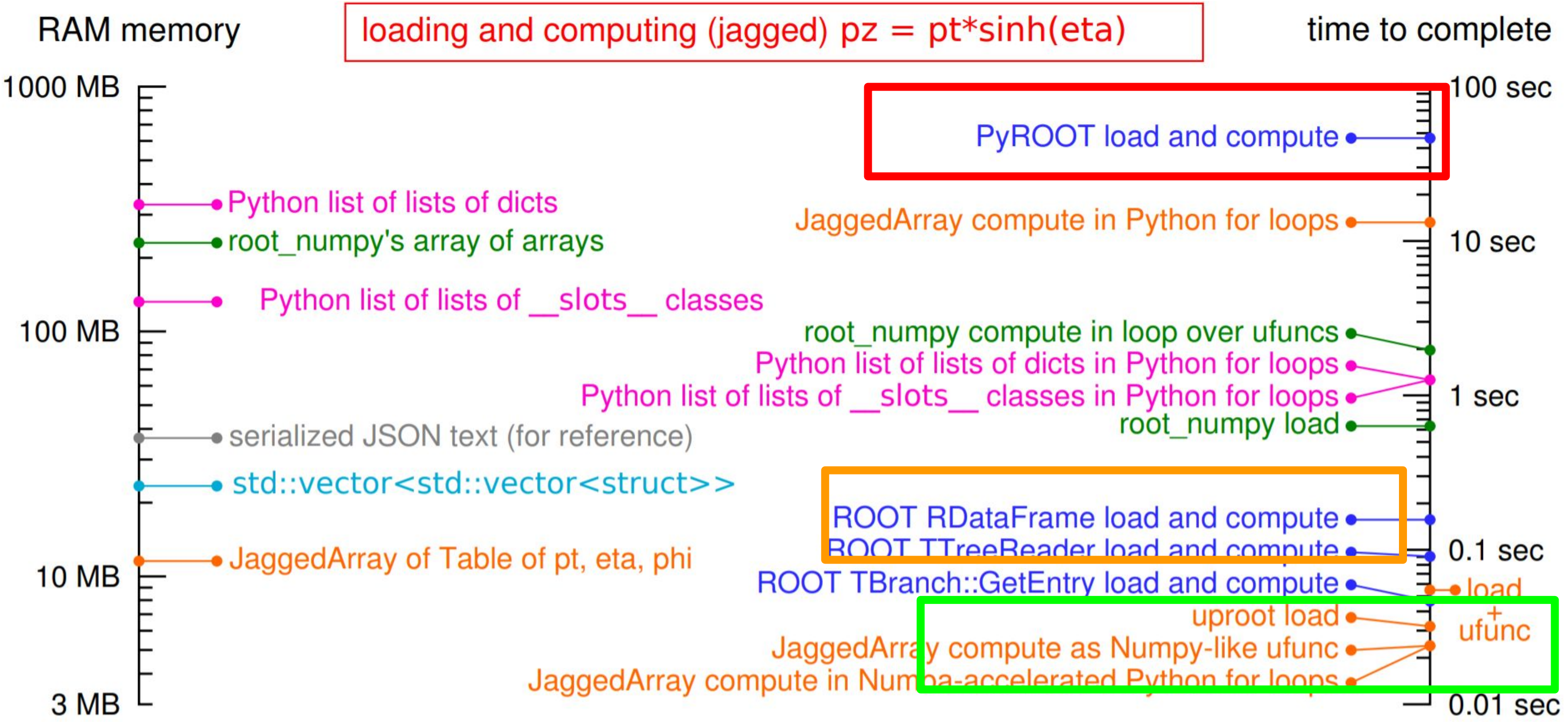
J. Pivarski https://indico.cern.ch/event/958432/contributions/4028932/attachments/2113570/3555538/main.pdf

Everywhere else :

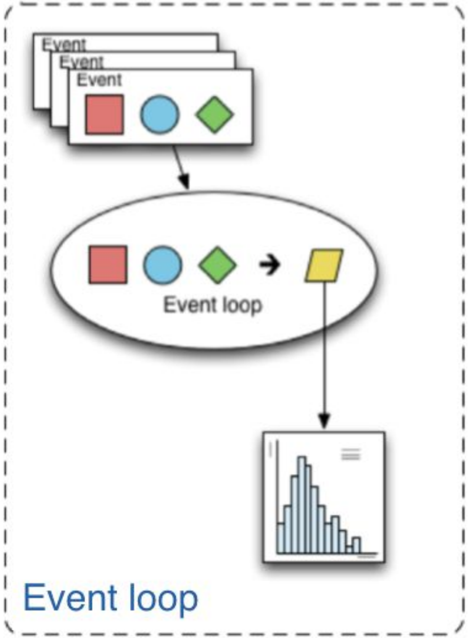


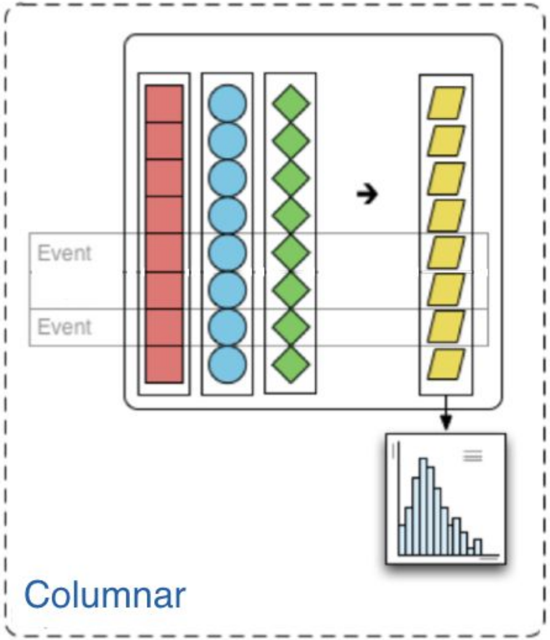
- python completely dominates the scientific and business ecosystem:
  - ML libraries (TF, pytorch)
  - big data libraries (dask)
  - data viz (matplotlib, bokeh, …)
  - and also :
    - APIs
    - Web servers
    - cloud development kits
    - Web clients (scraping)
    - …
- "The 2nd best language for everything"
- A must-have for transitioning to a career in the industry
  - data scientist, ML engineer

But python is slow …



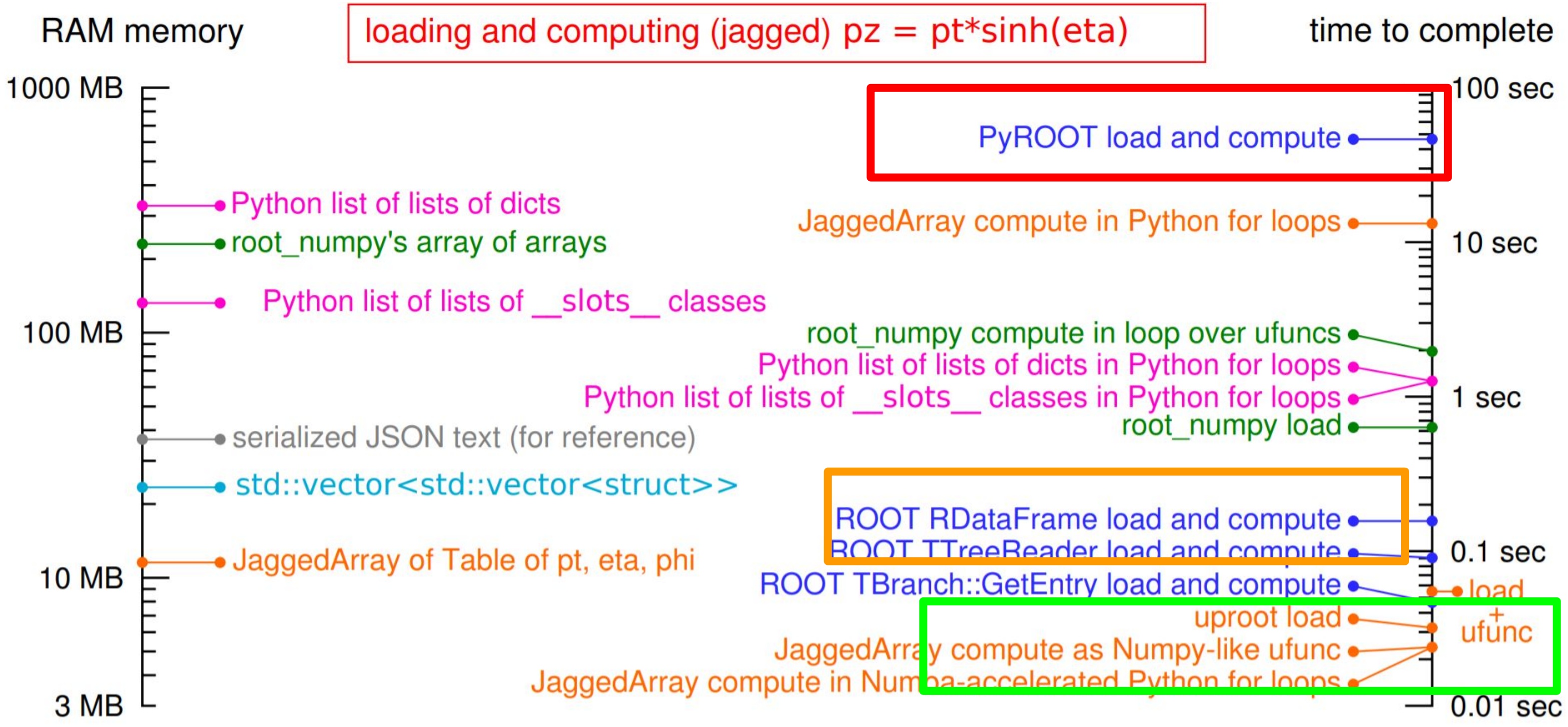
And also extremely fast when used correctly !





- Numpy columnar analysis: x500
  - python only used to drive C
  - same instruction multiple data (SIMD)
- Can also vectorize loops on the CPU with numba
- Can run
  - on GPUs with x500
    - numba + cuda
    - rapids
    - jax
  - on TPUs with
    - jax
  - on clusters with
    - dask

This talk : uproot & jagged arrays



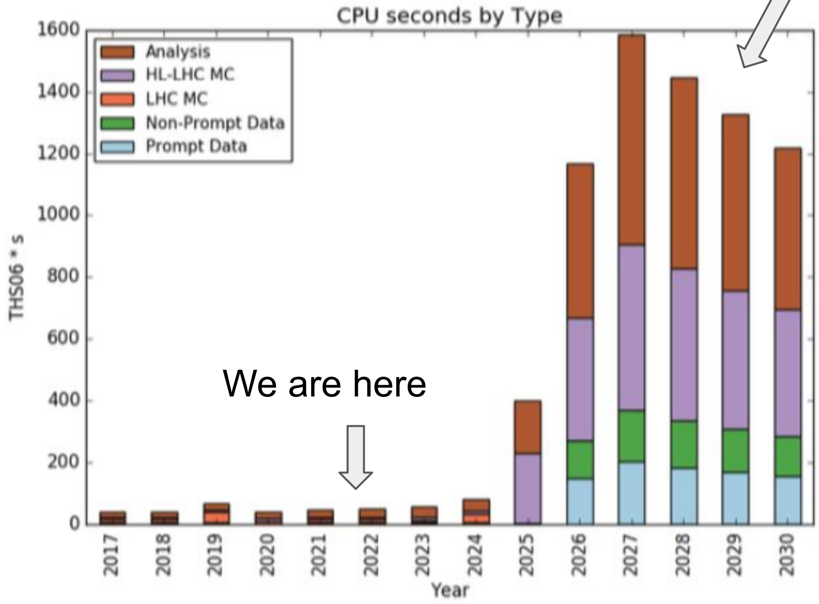
Could be x500 faster on GPUs

NanoAOD frameworks

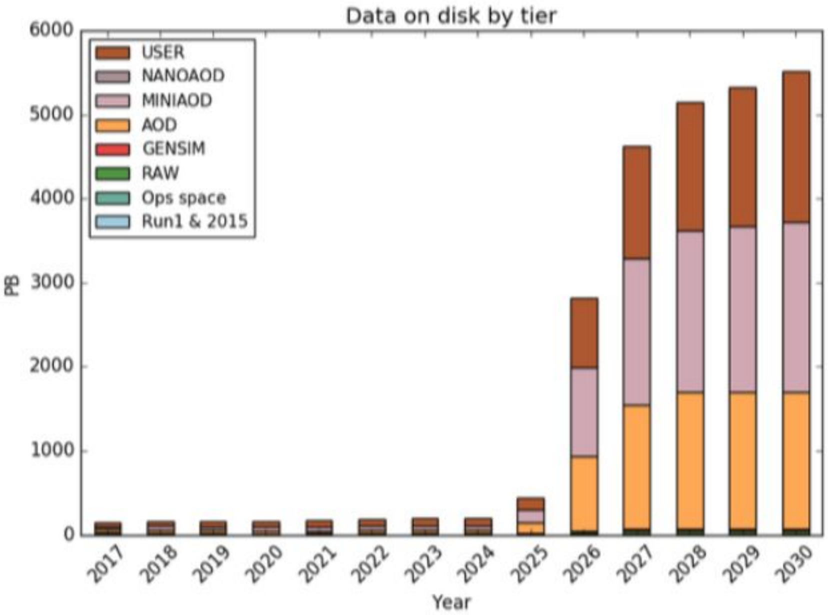
- two frameworks identified :
    - https://github.com/cms-nanoAOD/nanoAOD-tools
        - barely maintained and probably not used much
            - 45 stale issues, 10 open PRs (last merged : sept 2021)
            - no unittests, coverage, continuous integration
        - Based on PyROOT like heppy, but without its many features
    - https://gitlab.cern.ch/cdozen/nanoaodrdtool
        - where is the mother repo (not forked…) ?
            - not much activity on the Korean side
        - Barely maintained
            - no unittests, coverage, continuous integration
        - Based on RDataFrame

Why not PyROOT ?

Even RDataFrame could be too slow
x10 with uproot
x5000 with uproot on the GPU



We are here



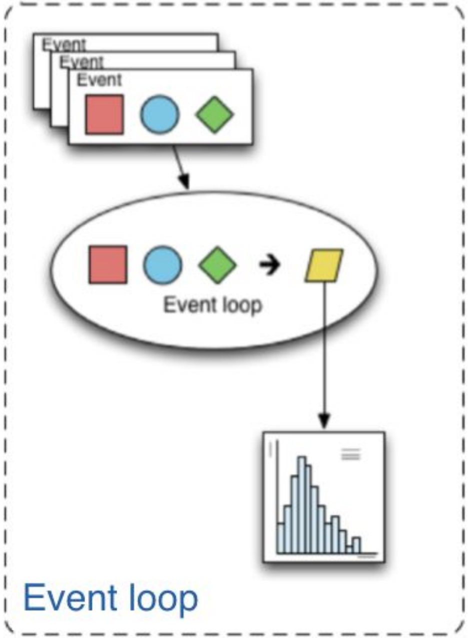
2021

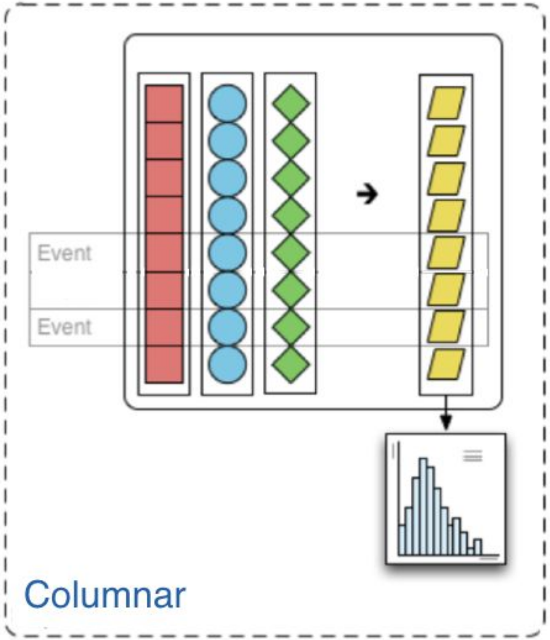
1k files, 1 billion events, **1 TB**

2027

30k files, 30 billion events, **30 TB**

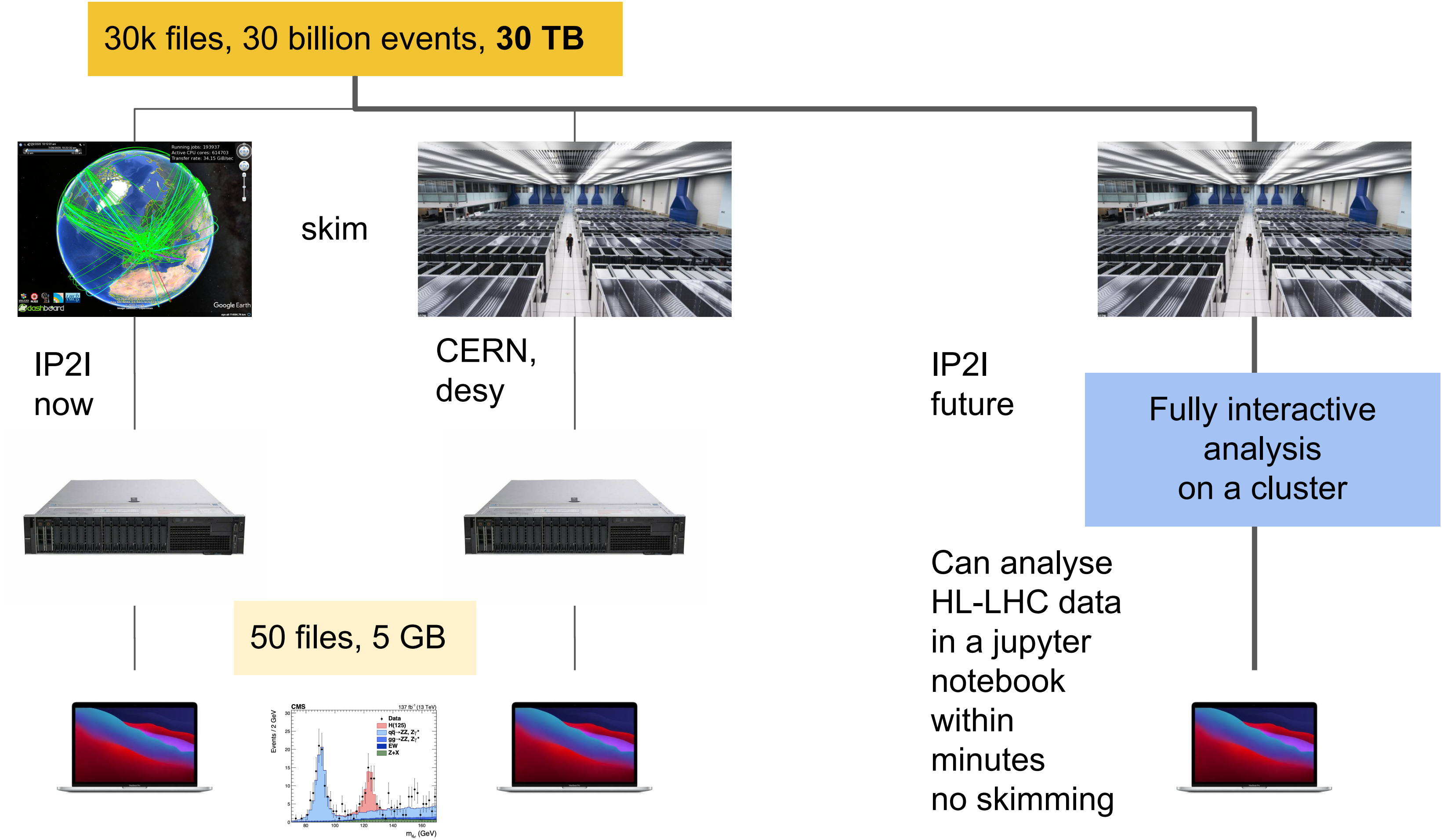
Solution : leverage python big-data ecosystem





- Numpy columnar analysis: x500
  - python only used to drive C
  - same instruction multiple data (SIMD)
- Can also vectorize loops on the CPU with numba
- Can run
  - on GPUs with x500
    - numba + cuda
    - rapids
    - jax
  - on TPUs with
    - jax
  - on clusters with
    - dask

My view of future analysis workflows



30k files, 30 billion events, **30 TB**

skim

IP2I
now

CERN,
desy

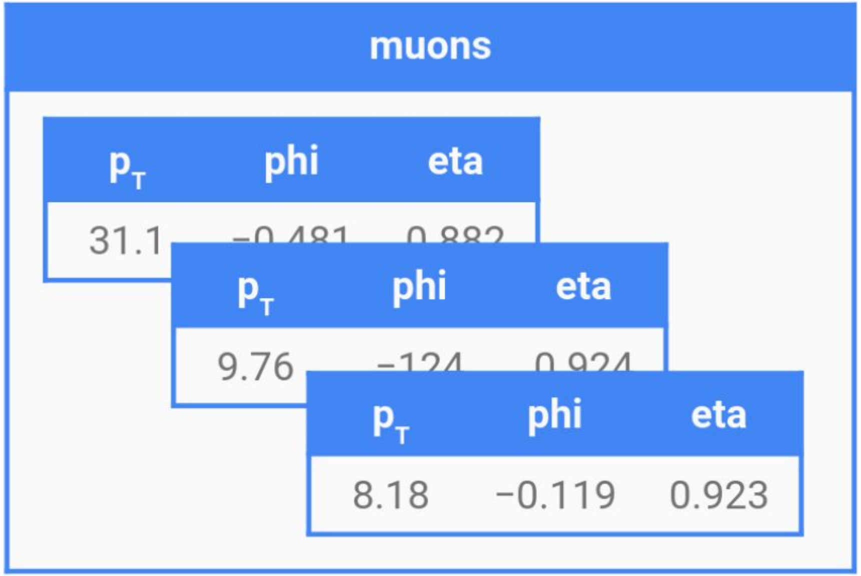
IP2I
future

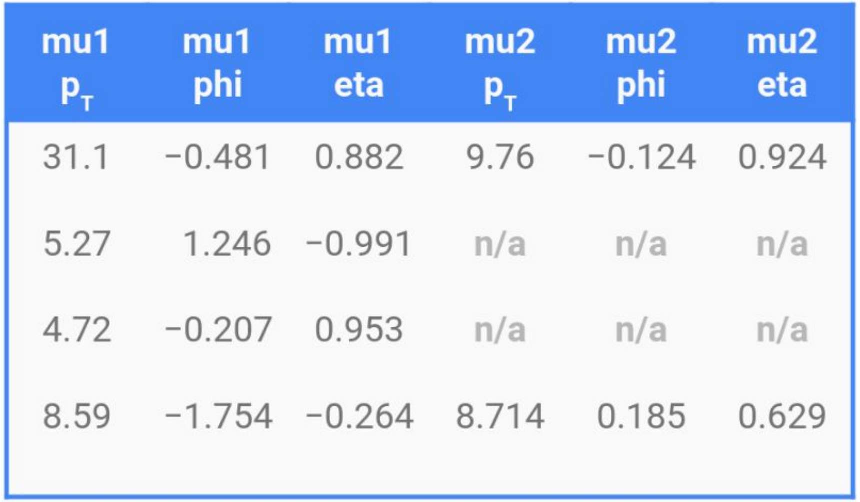
Fully interactive
analysis
on a cluster

50 files, 5 GB

Can analyse
HL-LHC data
in a jupyter
notebook
within
minutes
no skimming

But LHC data is not columnar

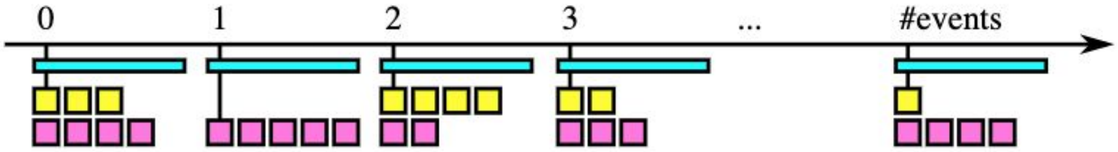




mu1 p_T	mu1 phi	mu1 eta	mu2 p_T	mu2 phi	mu2 eta
31.1	−0.481	0.882	9.76	−0.124	0.924
5.27	1.246	−0.991	n/a	n/a	n/a
4.72	−0.207	0.953	n/a	n/a	n/a
8.59	−1.754	−0.264	8.714	0.185	0.629

Columnar translation is lossy

Awkward arrays = Jagged Arrays



Jim Pivarski (Princeton)
Tutorial HEP (Z peak)
please watch later, this is nice :-)
https://www.youtube.com/watch?v=WInUF3LRBj4

```python
import awkward as ak

electrons = events.electrons
good_electrons = electrons[electrons.pt > 5]
ele1, ele2 = ak.unzip(
    ak.combinations(good_electrons, 2)
)
selected_events = events[ ele1.charge + ele2.charge == 0 ]
```

Ongoing:
CUDA kernels

numpy/pandas-like idioms

Tutorial : installation

- lyoui
- install miniconda for python 3.9, Linux 64 bit
- create a conda environment with python 3.9
- install coffea and uproot (no ROOT or CMSSW needed) :

```
conda install -c conda-forge coffea jupyter
```

coffea brings in its dependencies:
uproot, awkward, matplotlib, etc.