# Analysis tools

Pedro Ferreira da Silva, psilva@cern.ch - (CERN)

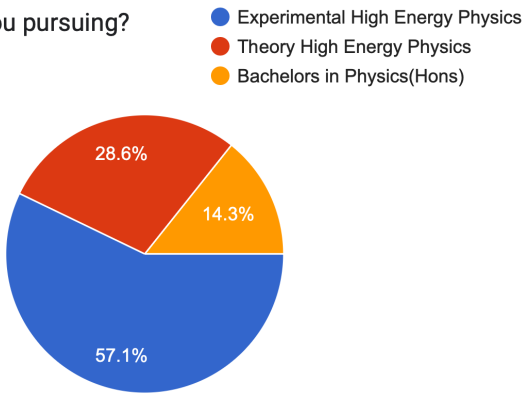
15th Course on Physics at the LHC

Monday 23rd March 2026

# First things first: your interests!

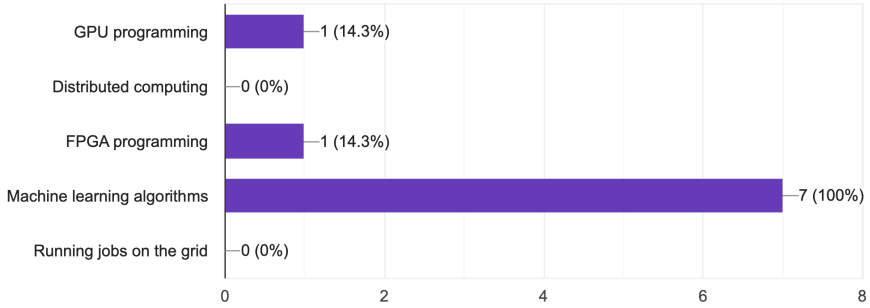
Which field are you pursuing?

7 responses



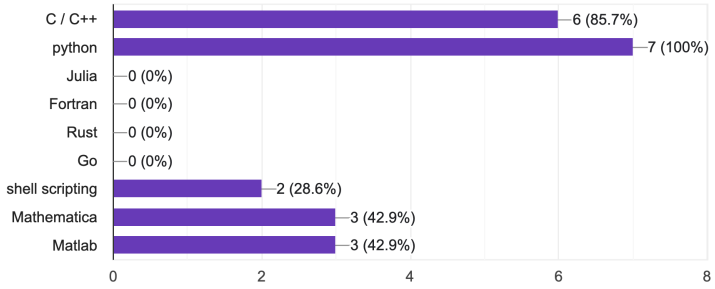

Are you familiar with any of these?

7 responses




Which programming languages are you familiar with?

7 responses




What would you be interested in learning about in an Analysis tools session?

4 responses

- Machine learning algorithms
- Rivet tool, machine learning tools, fitter tools like TRexFitter, wsmaker
- Yes
- COMSOL multiphysics and OriginLab

How are data created at the LHC?

The high level trigger

Offline analysis

Hands-on

## How are data created at the LHC?

The high level trigger

Offline analysis

Hands-on

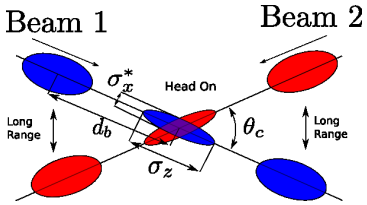
# How are data created at the LHC

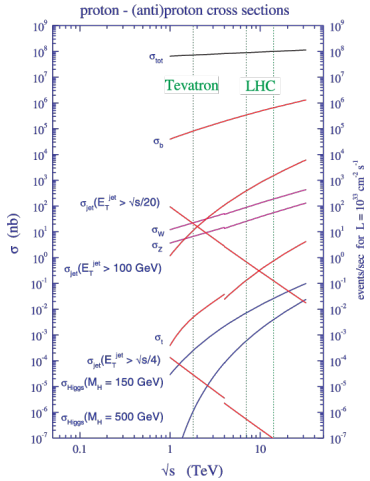
Bunches cross at the detector centre

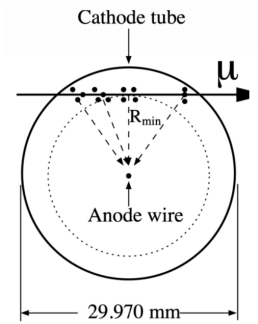
Beams interact at a rate dictated by the interaction cross sections

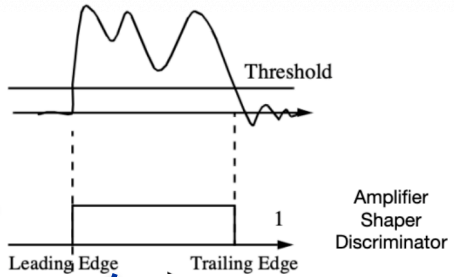
Particles cross the detector interacting with the material

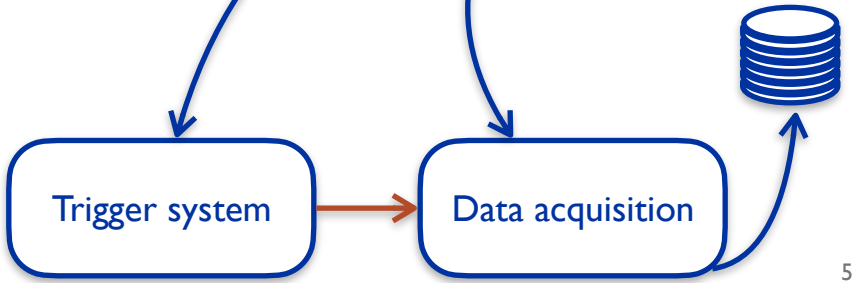
Readout electronics identifies regions of interest and measures amplitudes, regional sums, times



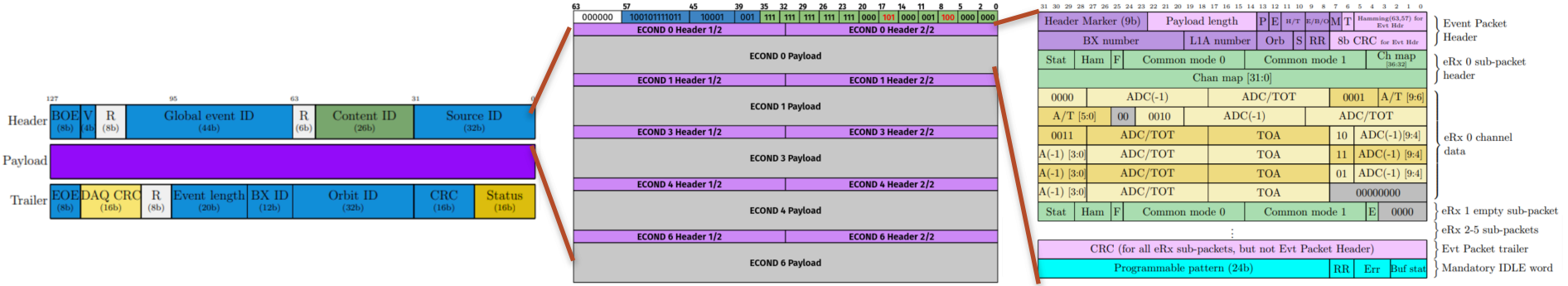



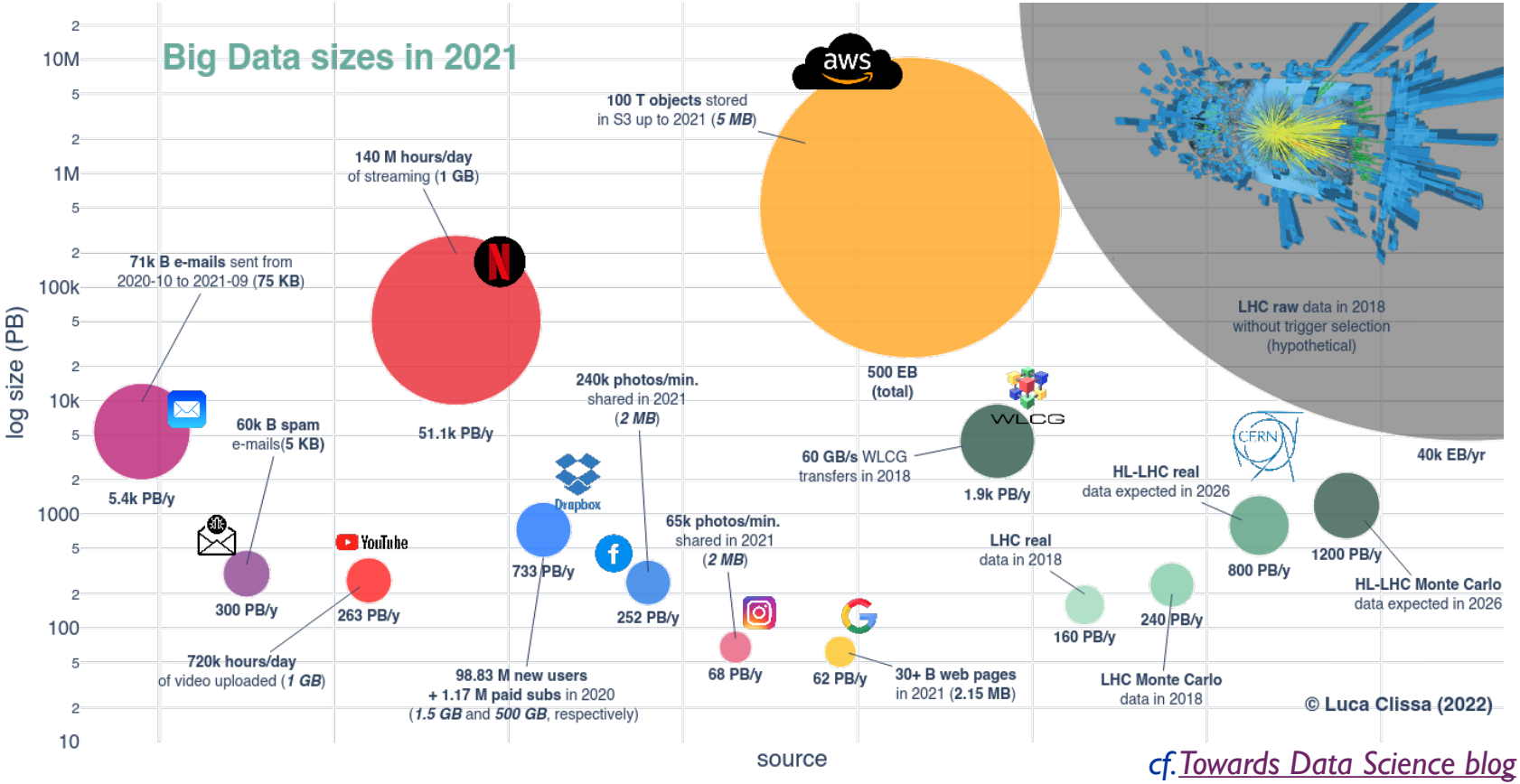
# What do data look like?

## A collection of event identifiers, quality flags, locators and amplitudes packed in a sardines' box

```
Begin processing the 1st record. Run 21000153, Event 4, LumiSection 1 on stream 0 at 20-Mar-2026 18:00:25.913 CET
%MSG-s DumpFEDRawDataProduct:  DumpFEDRawDataProduct:dump  20-Mar-2026 18:00:25 CET Run: 21000153 Event: 4
FED# 2102    21392 bytes  offset=       0 L1Id:      144 BXId:    0
0000  0000009000000836    0011  04510c000370dc00    0023  05b16c0006819c00    0034  0711bc0004d13c00
0001  5520000000000004    0012  0320dc000721bc00    0024  0691a40006318c00    0035  05514c0006f1bc00
0002  fe32820fffff8000    0013  06318c0005f17c00    0025  03d0ec0006518c00    0036  0711bc0004611c00
0003  aa3af0b41941004d    0014  05f17c0004f14800    0026  06f1b8000731cc00    0037  03d0ec0003e0f800
0004  e02f89ffffffffff    0015  04f1380003f0fc00    0027  0761d80005b16800    0038  03f0fc0005916400
0005  05a16c0006318c00    0016  05213c0005414c00    0028  05a1640005314c00    0039  07d1f80005f17c00
0006  04b1340002f0c800    0017  05e17c00e02f8cff    0029  0631880007e1fc00    003a  03e0f8000420fc00
0007  05f1880006f1bc00    0018  ffffffff06a1ac00    002a  05f17c0004311400    003b  05915c0005515800
0008  0350d40004311400    0019  0842180006f1bc00    002b  e02e88bfffffffff    003c  06d1b40004d12800
0009  05113c0006f1b400    001a  04c12c0006b1b400    002c  03b0ec0005b17400    003d  05715c0006319400
000a  05f17c0004211400    001b  06f1bc0007a1e000    002d  05f17c0006a1ac00    003e  04711c00e01b88ff
000b  05d16c0005f17c00    001c  0651940006619800    002e  04b130000821fc00    003f  ffffffff05f17800
000c  05a164000761dc00    001d  0661940006719c00    002f  04f1380004410c00    0040  05715c0004e13800
000d  06f1bc0006319400    001e  06f1b40006519400    0030  05b1700005b16400    0041  04b1280005715c00
000e  04d1340006e1b400    001f  06d1bc0004912400    0031  05d16c000390dc00    0042  06f1bc0003e0f800
000f  05f190000390ec00    0020  04f1380006d1b400    0032  0771d80004f13c00    0043  0711c40005314800
0010  055154000721c800    0021  07d1f4000791f400    0033  04f1480005f18800    0044  06719c0003a0d400
                          0022  0671a40006619c00                            0045  02f0b40002f0b400
```



# How large are data?




cf. Towards Data Science blog

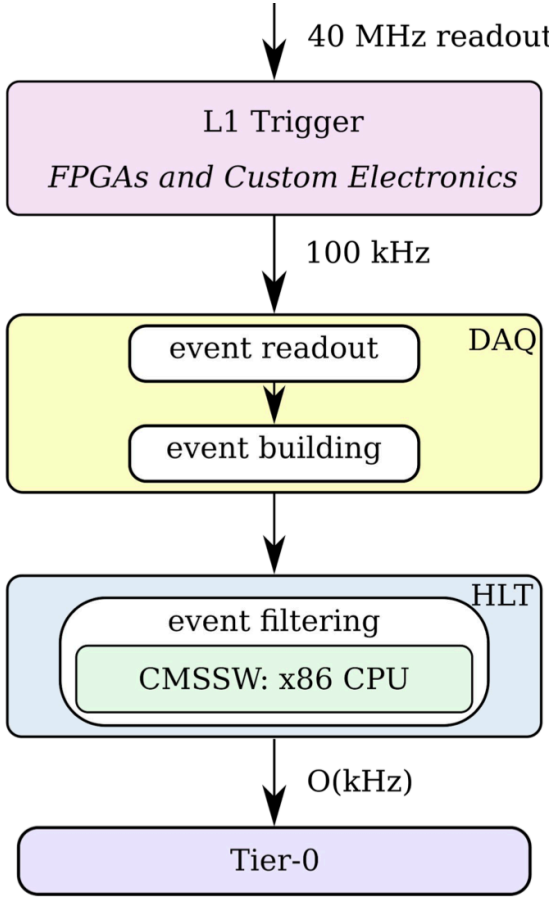
# Several transformations until a physics object is formed

## Data is produced at 40 MHz rate

- Undergoes stringent selections at Level-1 and later HLT
- Increased complexity in reconstruction and calibration requirements
- Permanent storage to disk of RAW data for later reconstruction

## With respect to Phase I of the LHC, Phase II foresees

- an increase of the L1 rate to 750 kHz
- more demanding reconstruction of more granular detectors in the presence of up to 200 pileup events
- quite a challenge from DAQ/Trigger point of view!



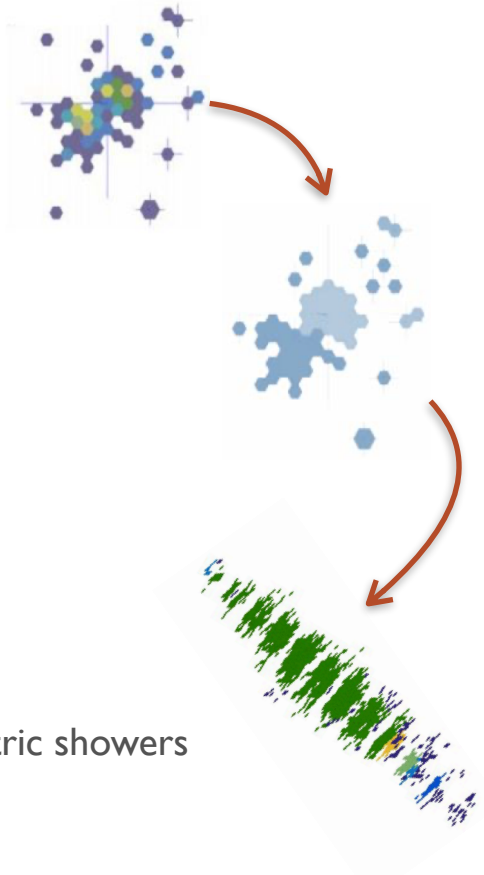
# Several transformations until a physics object is formed

## Local reconstruction

- Binary data (=DIGIs) conversion to floating-point precision (=Hits)
- Usually a complicated function to convert amplitudes to energy or time
- Hits are typically a 5D object (E,t,x,y,z) with some associated quality flag

## Clustering

- Reduction of complexity and noise
- Algorithms typically proceed by identifying seeds and aggregating neighbors
- Metric used to decide when to stop aggregation



## Building blocks for Particle Flow

- Clusters (sometimes hits) are used as input for tracking and building calorimetric showers
- A preliminary identification can be assigned in some cases

How are data created at the LHC?

**The high level trigger**

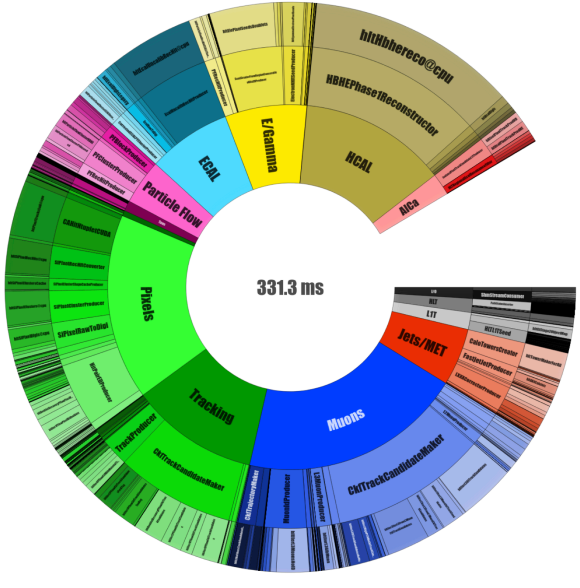
Offline analysis

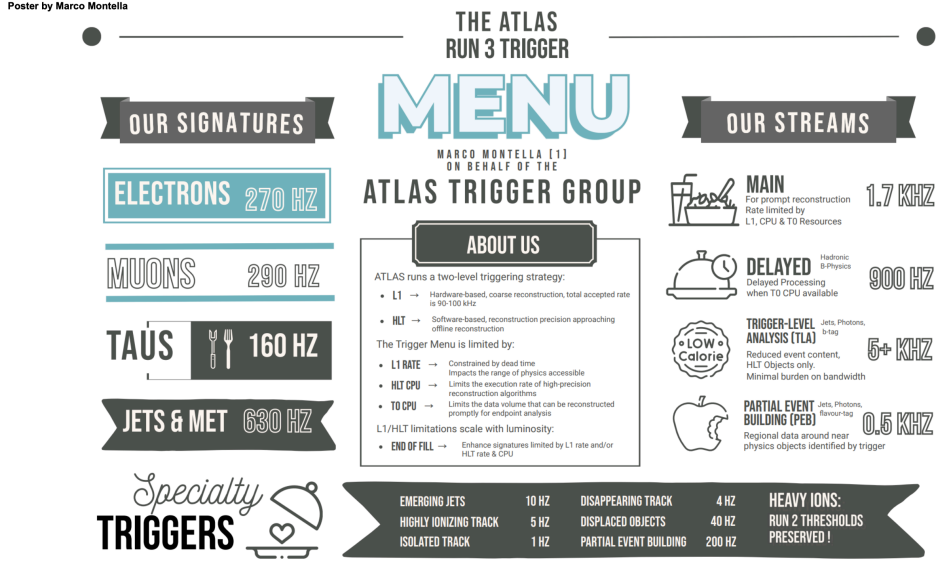
Hands-on

# The high level trigger

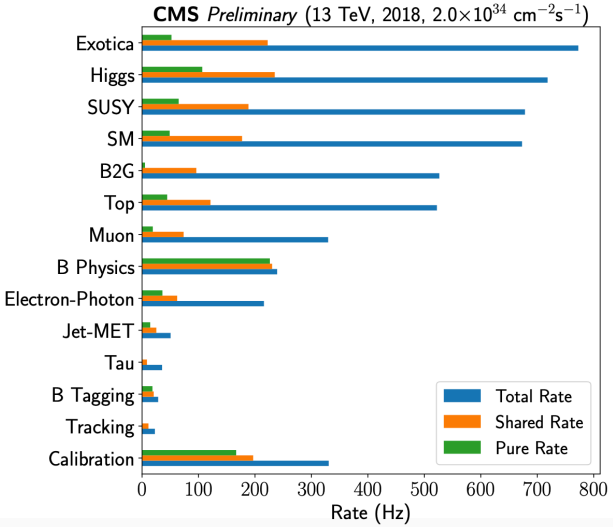
## Full event reconstruction is performed in the HLT farms

- Several filter modules share the output reconstructed objects
- Used to decide whether an event should be kept or not
- The collection of trigger bits is called the "Trigger Menu"



# The high level trigger - cont.

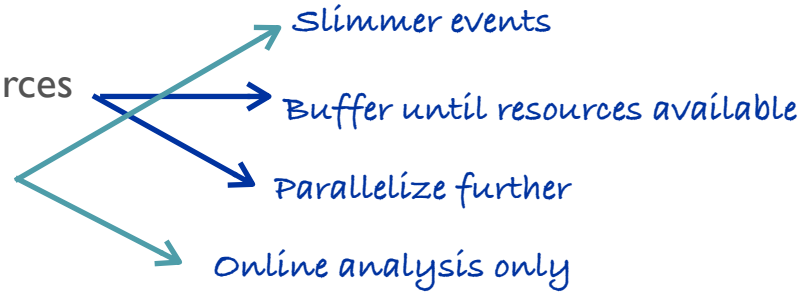


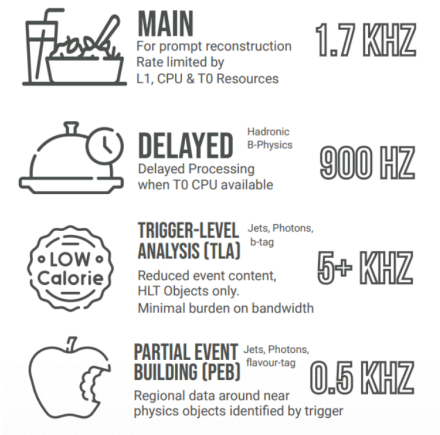

## HLT also ends up limiting the capacity of storing all that data

- not more than 1.5 kHz can be output by the HLT farms
- Every extra selection (HLT bit) needs to be negotiated between different physics and physics objects groups
- Often a trigger bit is prescaled (allow only "one out of N")

## Main bottlenecks

- Computing processing resources
- Limited storage

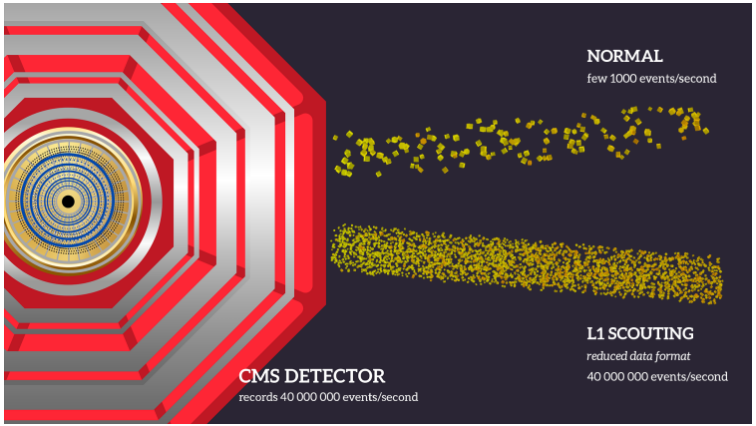


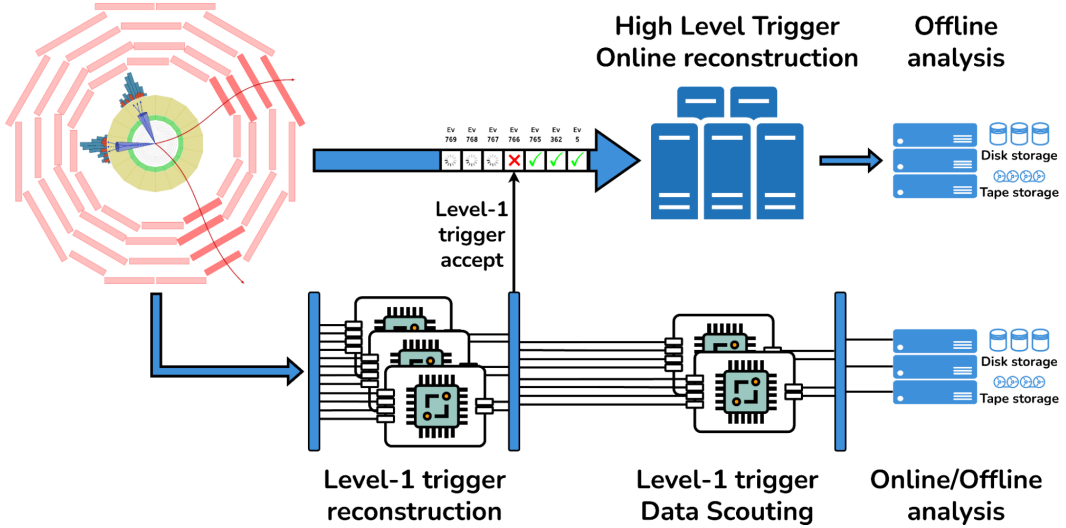



# Data scouting at source

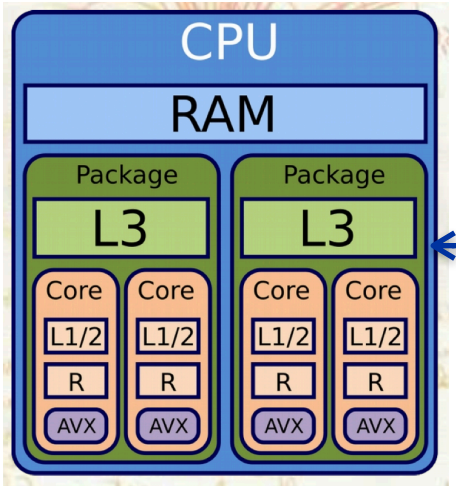
## Direct scouting at L1 gives access to full 40 MHz rate

- Quite feasible for some analysis with less stringent calibration / resolution degradation (e.g. using muons)
- Opens the door to analyses which become infective with traditional triggers
- Potentially gives access to the analysis of multiple bunch crossings simultaneously
- Recent example in [arxiv:2601.20063]

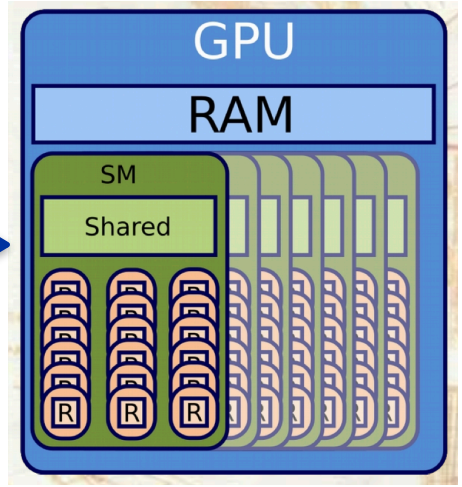


# Heterogenous computing approach









## Latency

- Large cache
- Average bandwidth between CPU and host memory
- Complex control unit
- Low core count / Powerful Arithmetic Logic Unit (ALU)

## Throughput

- Small cache
- High bandwidth between GPU cores and memory
- No complex control unit
- High core count

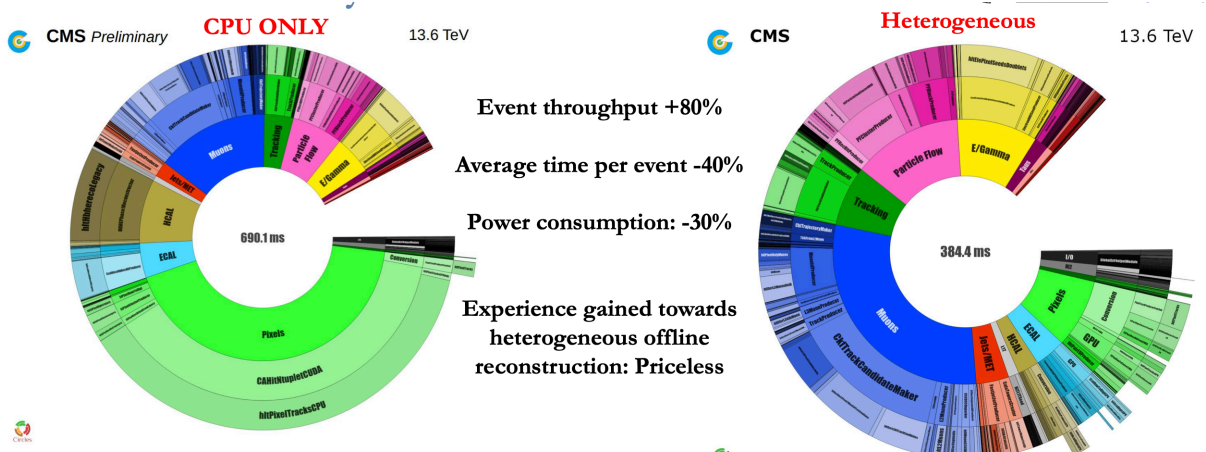
# Heterogenous computing approach - cont.

## Underlying objects need to be adapted for parallel computing

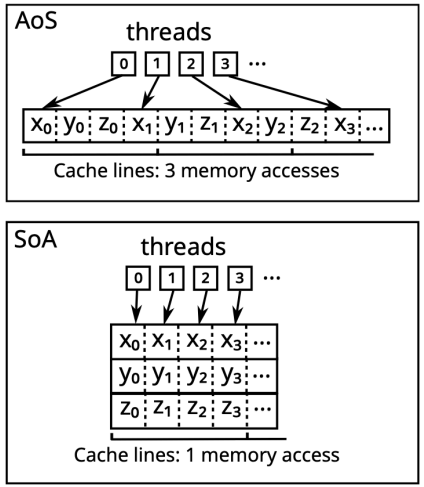
- Structure of arrays (SoA) divide a buffer of data into runtime sized columns
- Buffers host memory and are pinned to host or device
- Allocation and handling can be be made using

CUDA


ALPAKA


AoS and SoA




Currently ~40% of the CMS HLT uses this approach

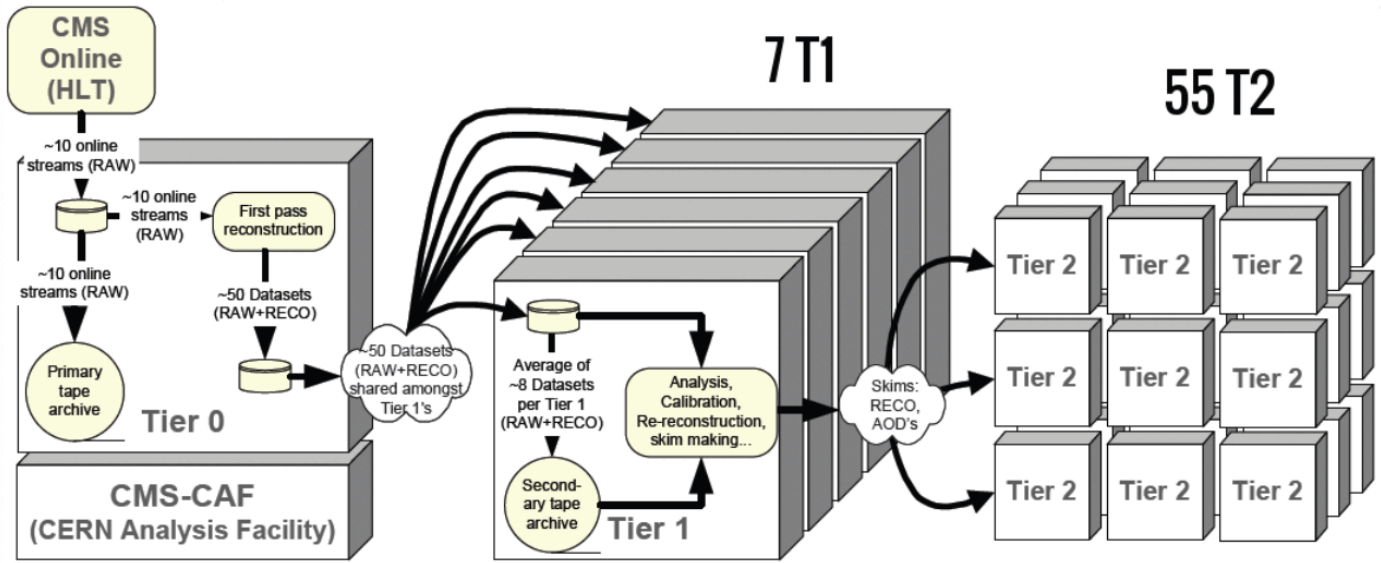
How are data created at the LHC?

The high level trigger

**Offline analysis**

Hands-on

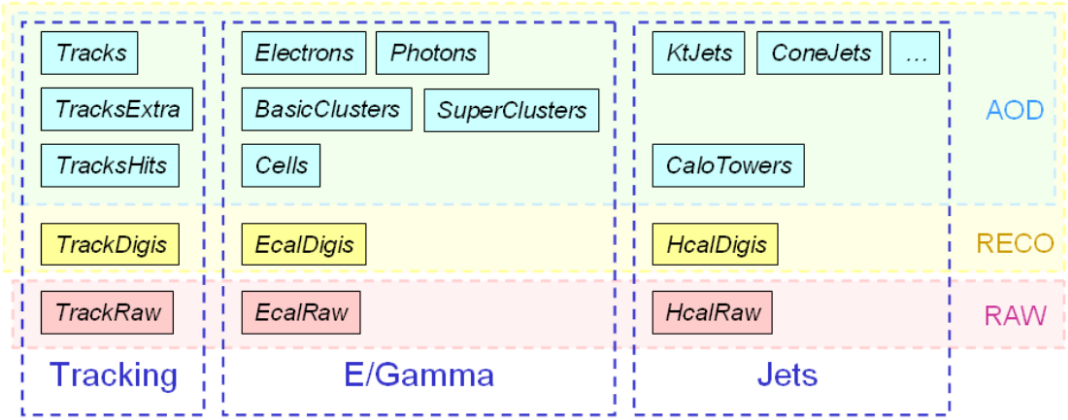
# From HLT onwards: the computing …

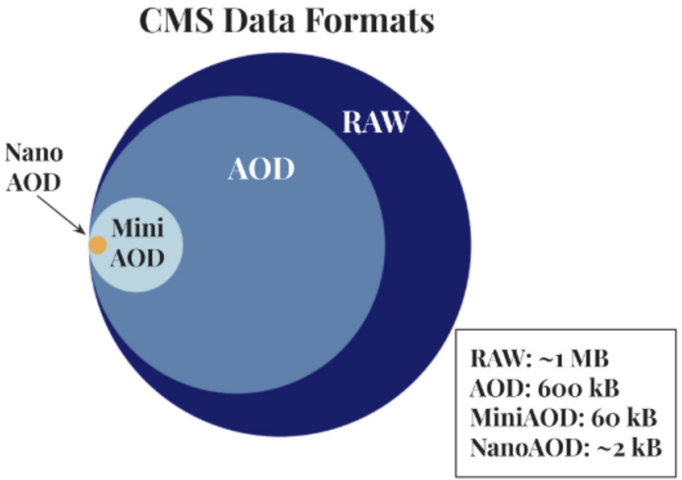



## Grid computing is structured in Tiers

- CERN always preserves a copy of all events (RAW) on tape and first pass reconstruction on disk
- Tier 1 are used to run re-reconstruction but also analysis, calibration and further selection
- Tier 2 are used mostly for analysis
- Usage relies on World LHC Computing Grid (WLCG)

# From HLT onwards: the computing … and data tiers

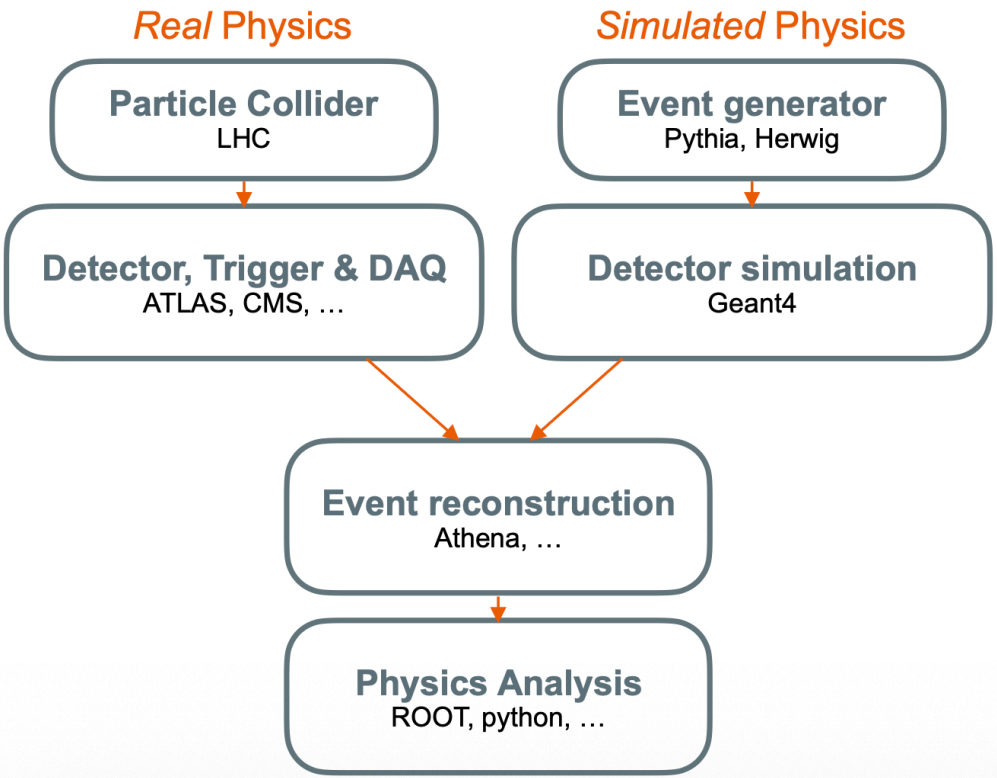


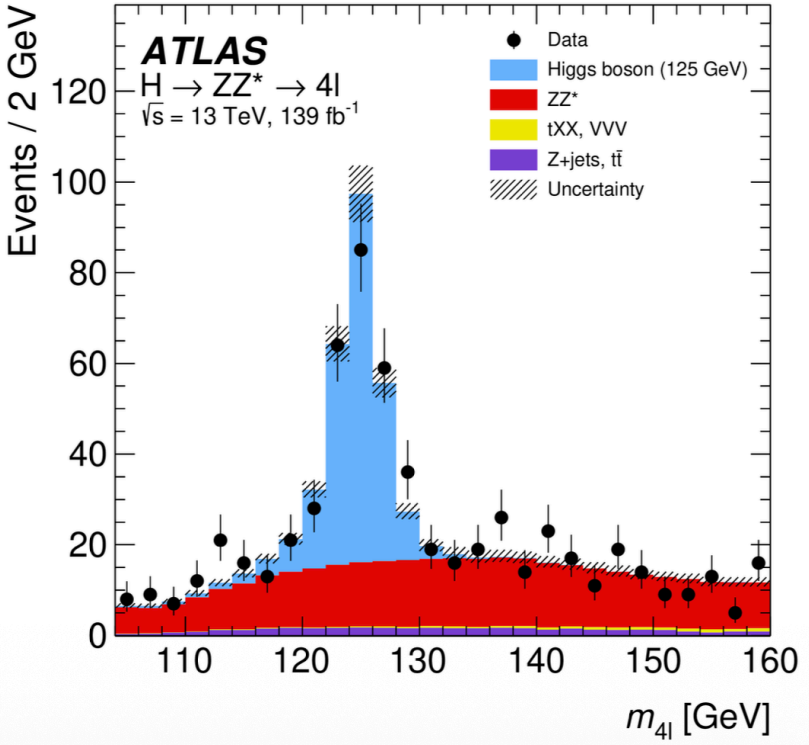




## LHC experiments use a variety of data tiers

- Depending on the analysis complexity it may or may not need some of the "core objects"
- It's fair to say that the vast majority only need the calibrated, final, physics objects

# What's in a paper plot?

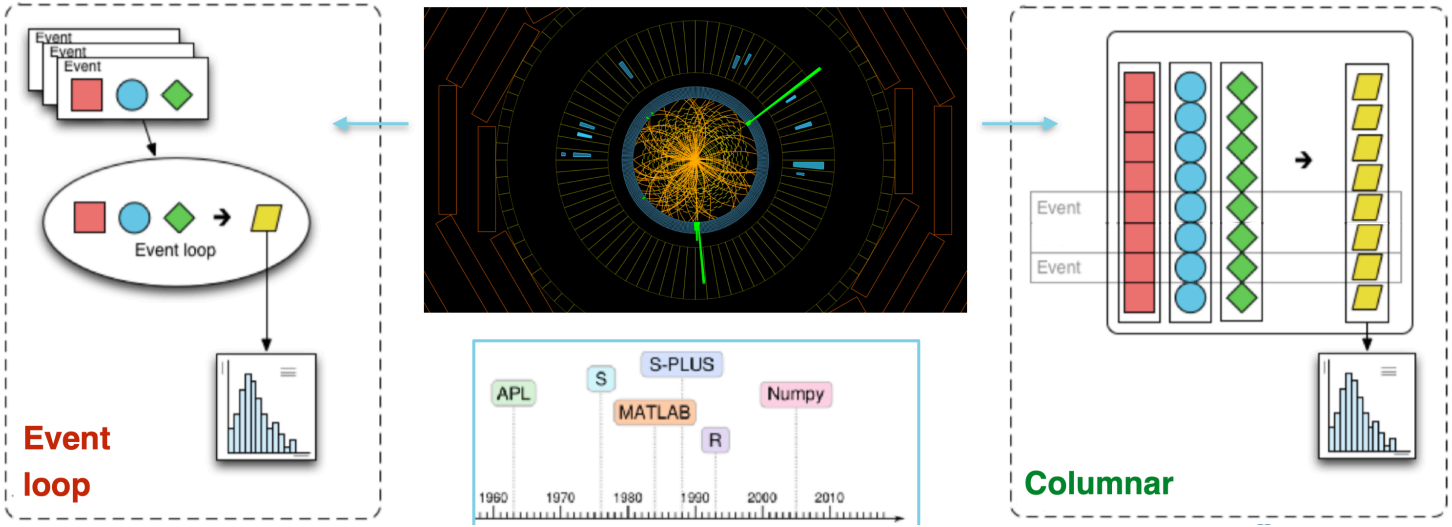


# High energy physics analysis: some challenges

## Data access is organized in an efficient way but it doesn't reflect their variety

- Events are highly structured and objects are complex with references to each other
- Complex objects can be inefficient on disk and memory (if not careful)



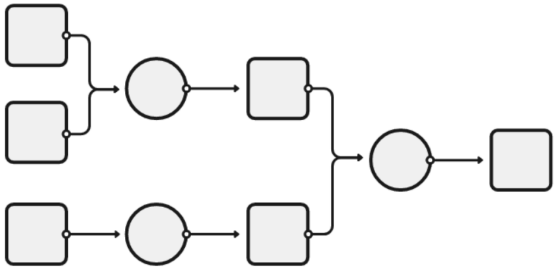

## When doing analysis: understand where data is and how can I analyze it efficiently

# Evolutions in abstraction

## Since some years HEP started moving to columnar analysis

- Parallelize as much as possible: multithreaded, GPU, distributed computing
- Actions are pre-defined and chained: execution is delayed

| | Electrons | Muons | Jets |
|---|---|---|---|
| Event #1 | ■■ | | ●●●●● |
| Event #2 | | ■ | ●●● |
| Event #3 | ■■■ | ■■ | ●●●●●● |
| … | | | |



```
for event in EVENTS:
    for e in ELECTRONS: do_something
    for m in MUONS: do_something
    …
    with selected_objects: do_something_else
    once finished, move to next event…
```

Scheduler: dask HTCondor Software Suite

# Evolutions in abstraction

## Since some years HEP started moving to columnar analysis

- Parallelize as much as possible: multithreaded, GPU, distributed computing
- Actions are pre-defined and chained: execution is delayed

```
void MyClass::Loop() {
  size_t nEvents;
  // load...

  for (Long64_t iEvent=0; iEvent<nEvents; iEvent++) {
    double MET_pt;
    int nElectron;
    double * Electron_pt;
    double * Electron_eta;
    // load...

    if ( MET_pt > 100. ) continue;

    for(size_t iEl=0; iEl<nElectron; ++iEl) {
      if ( Electron_pt[iEl] > 30. ) {
        hist->Fill(Electron_eta[iEl]);
      }
    }
  }
}
```

```
cut = (events.MET.pt < 100.) & (events.Electron.pt > 30.)
hist.fill(eta=events.Electron.eta[cut].flatten())
```

```
hist.compute()
```

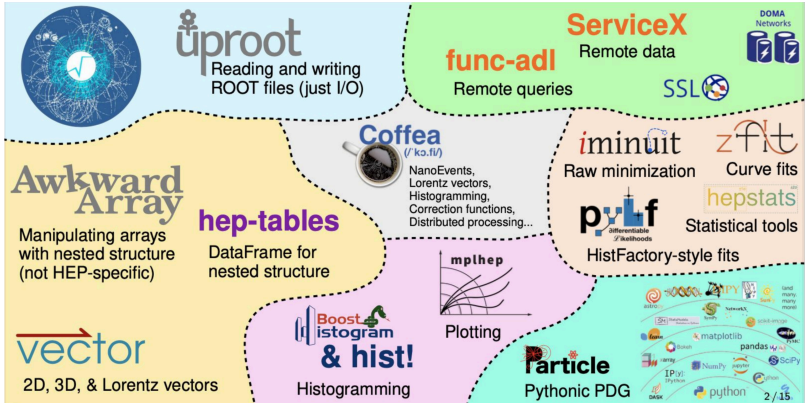
# Current paradigms

## Modern analysis software is factored and comprises a complex ecosystem

- each step should work well in isolation but needs to be chained in a system
- But require availability of many different tools and programming languages

## Managing all these tools as a user can be overwhelming

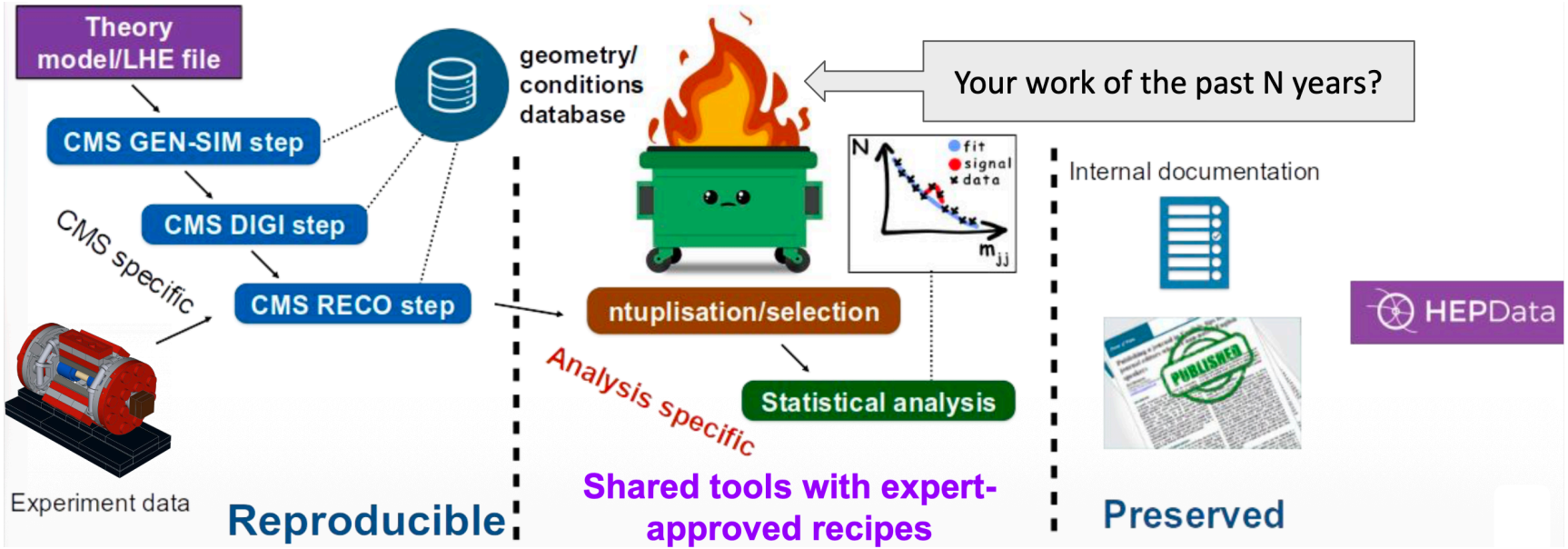
- Analysis facilities : CERN, FNAL, others can provide curation of software and hardware needed for the different experiments
- Web-based interface with access to juyterlab but also terminal
- Users can complement baseline software with additional software packages as needed



# Reproducibility rules

1) your closest collaborator is yourself 6 months ago (who doesn't reply to e-mails)

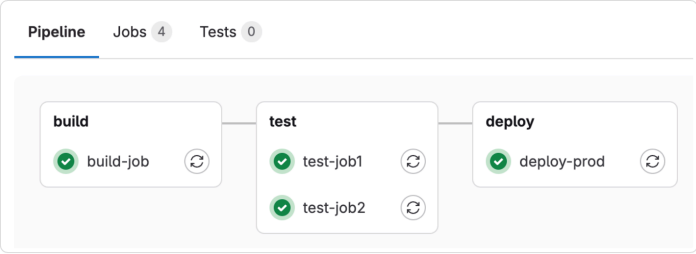
2) you shall re-run your analysis over and over again (until exhaustion)



# Reproducibility handles

## Continuous Integration/Deployment

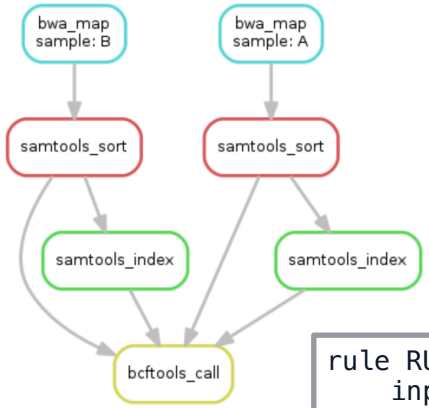
- All software scripts that you produce should be stored in git (or similar tool)
- New changes should not be merged blindly but rather tested
- Pipelines are there to help you out:
  - Describe the main steps that need to always work
  - Compare results with the previous
  - Track the changes
- A tutorial can be found in here




```
RULE_NAME:
    extends: BASE_RULE
    needs: [LIST_OF_RULES]
    artifacts:
      paths:
        - Value
    script:
        - myscript arg1 arg2 …
```

## Workflow management tools

- Several on the market: snakemake, luigi, law, ploomber, etc.
- Describe each step separately
  - Factorize chained steps from those that can run in parallel
  - Pick-up flow from the last step
  - Repeat as many times as needed




```
rule RULE_NAME:
    input:
        Key = Value
    output:
        Value
    shell:
        myscript arg1 arg2 …
```

How are data created at the LHC?

The high level trigger

Offline analysis

**Hands-on**

# Hands-on

We'll make use of the CERN analysis facility also know as 


We'll go through three notebooks illustrating usage of different tools

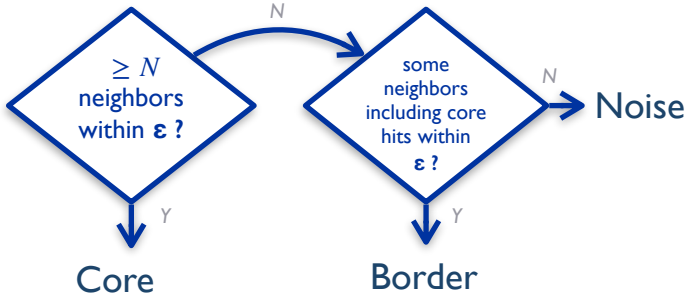
- Classic clustering examples
- A pseudo-analysis based on open data
- Running theory predictions and comparing to the result of your pseudo-analysis

First step: open https://gitlab.cern.ch/psilva/courselhcphysics and follow the instructions therein

# Example 1: two clustering algorithms

## DBSCAN

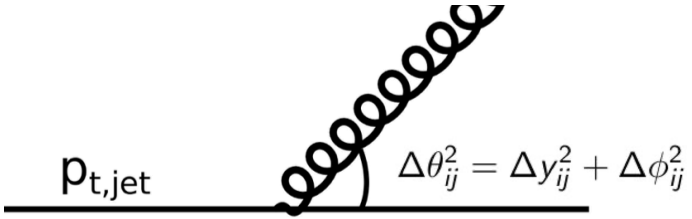
- Define reference radius (**ε**) and neighbors (**N**)
- Classify hits




- Connect core points into clusters if they are density connected
- Attach border points

## FastJet

- Inspired by angular ordering of QCD emissions
- recursive aggregation of neighbors in the η-ϕ plane (recall its invariance properties)
- For each pair compute: $d_{ij} = \min\left(p_{T,i}^{2k}, p_{T,j}^{2k}\right)\frac{\Delta\theta_{ij}^2}{R^2}$
- Stop if $d_{ij} \geq p_{T,i}^{2k}$



# Example 1: two clustering algorithms

**ClusteringAlgos.ipynb** 9.83 KiB

Code | Preview

## Crash course on LHC analysis tools - Part I : Clustering algos

In this notebook we inspect how two different clustering algorihms behave.

One (DBSCAN) is typically used in ML applications and uses a metric-based distance, while the other (FastJet) is typically used in HEP and adds an energy dimension.

You can learn more about these methods in the following references:

- https://en.wikipedia.org/wiki/DBSCAN
- https://fastjet.fr

The comparison is made using a toy MC which is used to demonstrate how tools such as numpy can be used for fast studies.

```
#you may need to install this plugin at start
#!python -m pip install fastjet --user
```

```
from sklearn.cluster import DBSCAN
import fastjet
import numpy as np
import matplotlib.pyplot as plt
import mplhep as mh
plt.style.use(mh.style.CMS)
```

# Example II: corrections

**ObjectCorrections.ipynb** 5.29 KiB

Code | Preview

## Exporting a correction to be used in an analysis

`correctionlib` provides a well-structured JSON data format for a wide variety of ad-hoc correction factors encountered in a typical HEP analysis.

It can be used in both python and C++. More details can be found in the documentation page

In generic terms a correction can be declared as a function of several variables.

One can use histograms, formulae, random number generators.

In this example we create a dummy corrector of the muon $p_T$ scale.

In collaborations therese are typically provided already by Physics Objects Groups.

However it's quite common that in an analysis additional corrections are needed.

Author: Pedro Ferreira da Silva (psilva@cern.ch)

```
#algebra tools
import numpy as np

#plotting tools
import matplotlib.pyplot as plt
import mplhep as mh
plt.style.use(mh.style.CMS)
```

# Example III: measuring the Z transverse momentum

A2ZAnalysis.ipynb 26.54 KiB

Code | Preview

## Crash course on LHC analysis tools - Part II : A-Z analysis

This is obviously not a real analysis, it's rather a collection of tools which you may find useful to include in a real analysis.

In this notebook we make use of a simulated sample of $Z \rightarrow \mu^+\mu^-$ events and perform the following operations:

- select events of interest applying corrections and saving a summary locally
- measure the efficiency of a cut using a "tag-and-probe" approach
- correct detector-level effects by unfolding
- save the final result in a format that can be shared with theoreticians

Author: P. Silva (psilva@cern.ch)

### Some useful tools

In the following we import tools which can be executed from python for simplicity. Most of them are widely used in high energy physics and it's good to get familiar with them. This is by no means an extensive list and it's mostly left as an appetizer

- ROOT - must-have in HEP: a library that spans efficient data storage and processing to math and physics tools
- numpy - offers a lot of useful tools to manipulate arrays efficiency
- matplotlib - a simple to use plotting tool
- mplhep - useful wrapper to beautify plots to publication standards
- scinum - scientific numbers with uncertainties and correlations, gaussian propagation and numpy support.
- correctionlib - provides a structured format and interface for correction factors
- zfit - one amongst several fitting tools, based on tensorflow
- hepdatalib - useful interface to export data measurements to HepData

# Example III: theory predictions

Rivet8Comparisons.ipynb 8.06 KiB

Code | Preview

## Crash course on LHC analysis tools - Part III - Theory predictions

### Comparing our results to other models using RIVET and PYTHIA8

The Rivet toolkit (Robust Independent Validation of Experiment and Theory) is a system for validation of Monte Carlo event generators. It's written in C++ and has a large set of experimental analyses implemented as well as links to HepMC. It is used to preserve analysis for comparison and development of theory models. It is used by phenomenologists, MC generator developers, and experimentalists on the LHC and other facilities. You can find more details in this page

PYTHIA is a general-purpose Monte Carlo event generator widely used in high-energy physics collision events. It is able to describe collisions at high energies between different beams and contains several theory models for a number of physics aspects, including hard and soft interactions, parton distributions, initial- and final-state parton showers, multiparton interactions, fragmentation and decay. It can be used to interface with matrix element generators and provide the dressing of a full event. You can find more details in this page

The following notebook gives an example of running RIVET and PYTHIA. Both are available through `cvmfs` (see details here).

Nota bene: this notebook is an example: in practice we'll mostly call shell commands for compilation and running.

- We start by compiling our implementation of the analysis in RIVET : `include/Z_RIVET_ANALYSIS.cc`
- Then we run two different PYTHIA8 predictions with to different underlying event tunes. To that aim a PYTHIA8 helper is compiled in : `include/pythia_run.cc`
- The events are then analyzed with the RIVET plugin.
- The plots obtained are compared