Muon tag-and-probe efficiencies with Apache Spark and Parquet

Andre Frankenthal (Princeton) for the **Muon Physics Object Group**

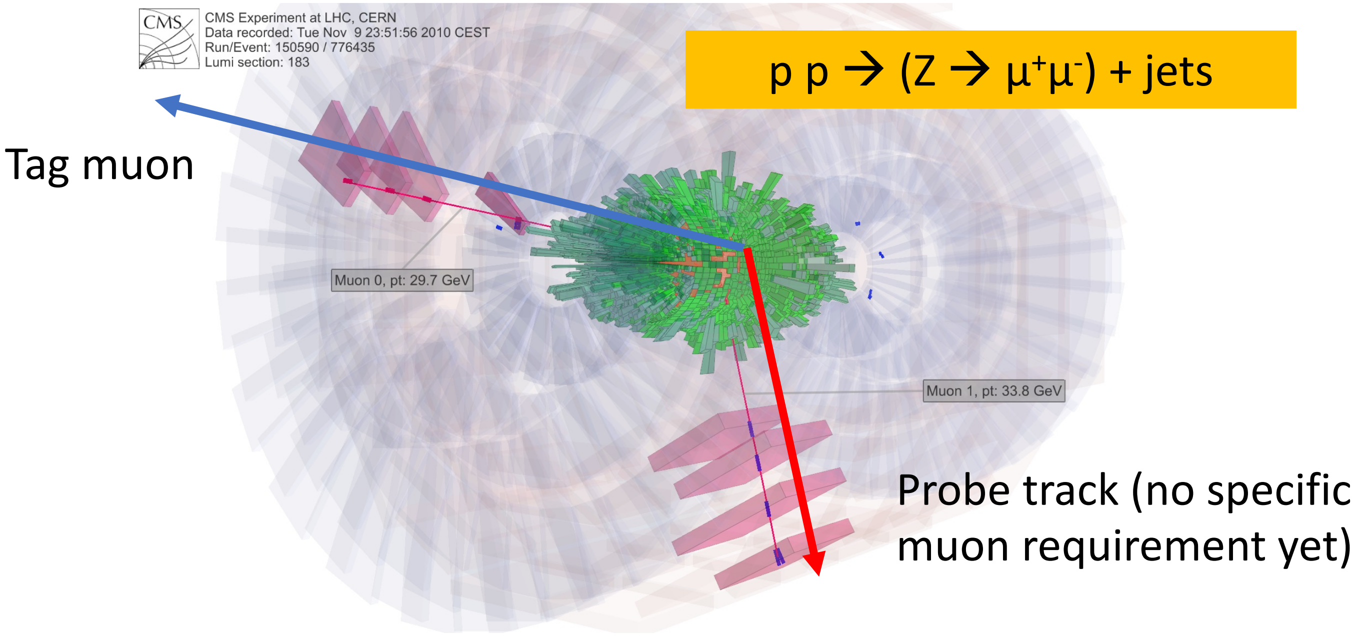
on behalf of the **CMS Collaboration**

Why efficiencies?

- Efficiencies are a key part of any experimental physics program
  - Object reconstruction *("Can we reconstruct this muon?")*
  - Identification *("How likely are we to identify this muon?")*
  - Trigger *("Does this muon trigger the event?")*
  - Isolation *("Is this muon isolated from other activity in the event?")*
- Many aspects of physics analyses rely predominantly on simulations, so it is crucial to ensure their validity and understand their limitations
  - Simulation can't capture every single detector misbehavior
  - Other physics activity in the event can unexpectedly degrade performance
  - Additional unaccounted phenomena can affect efficiencies
- Measuring discrepancy between efficiency in data and simulated efficiency is critical for obtaining correct representation of physics in play
  - These "**scale factors**" correct our expectation and improve the accuracy of our measurements
  - They are in essence a calibration between expected and observed performance

Scale factors with tag-and-probe in CMS

- In colliders, a common way of computing scale factors is via the **tag-and-probe (T&P)** method
- CMS mainly uses Z and J/Ψ resonances to compute efficiencies in data and in simulation, and derive scale factors from the discrepancy
- A role of the **Muon Physics Object Group (POG)** is to provide official and comprehensive efficiency recommendations for CMS analyses
  - Highest precision achievable
  - Covering broadest phase space possible
- Deriving corrections is fastidious work and without performant code it would take several days to produce baseline scale factors
  - Quick turnaround time is also critical for commissioning new data as it streams in



Tag & Probe

1. Ensure **robust tag muon and dimuon pair selection** to select signal
2. Apply **minimum pre-selections to probe track** (enough to ensure reliable sample of Z → $\mu^+\mu^-$ candidates)
3. Check **if probe satisfies selection under test** and compute efficiency

Sketch of a cut-and-count T&P computation

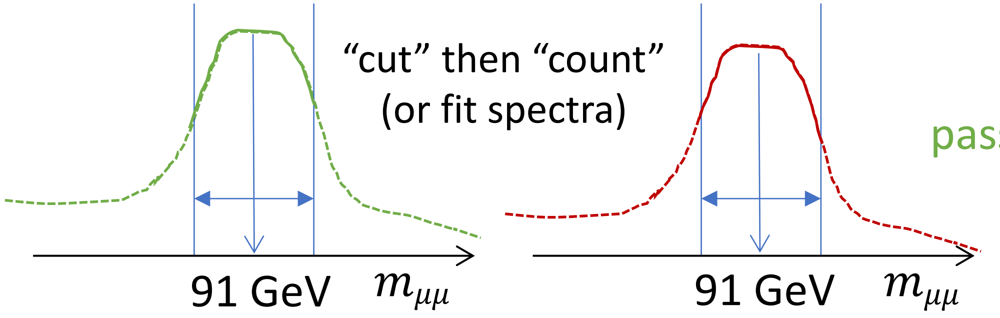
Sample with event content

Apply very loose selection to store flat ntuples with possible tag-probe pairs ("skimming")

flexibility *reusability*

Apply:
1. Pre-selection to tag, probe, and/or event
2. Baseline selection to probe ("*denominator*")
3. Test selection to probe (may pass or fail) ("*numerator*")

Bin passing and failing probes in bins of dimuon invariant mass



"cut" then "count" (or fit spectra)

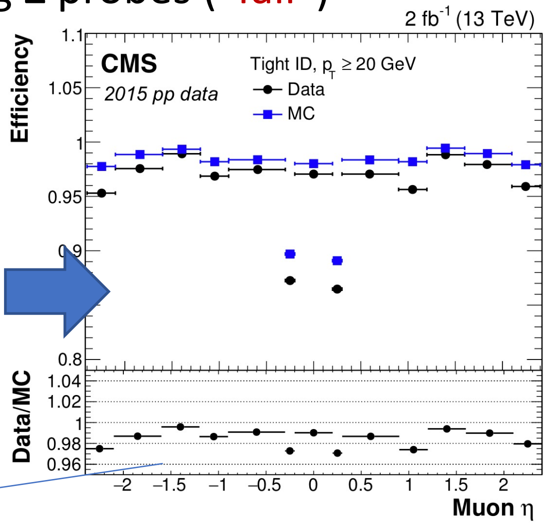
Passing Z probes ("pass") Failing Z probes ("fail")

Efficiency: pass / (pass + fail)

Sample in data collected with single-muon triggers → Passing Z probes in data, Failing Z probes in data → Efficiency in data

MC sample of Z → $\mu^+\mu^-$ events passing the triggers → Passing Z probes in MC, Failing Z probes in MC → Efficiency in MC

Bin in pT, rapidity, etc.



A.M. Sirunyan et al 2018 JINST 13 P06015

Scale factor!

Efficiency factorization and the need for speed

- In practice, the general muon efficiency of an analysis is factorized:

$$\epsilon = \epsilon_{\mathrm{trk}} \times \epsilon_{\mathrm{ID|trk}} \times \epsilon_{\mathrm{iso|ID}} \times \epsilon_{\mathrm{trig|iso}}$$

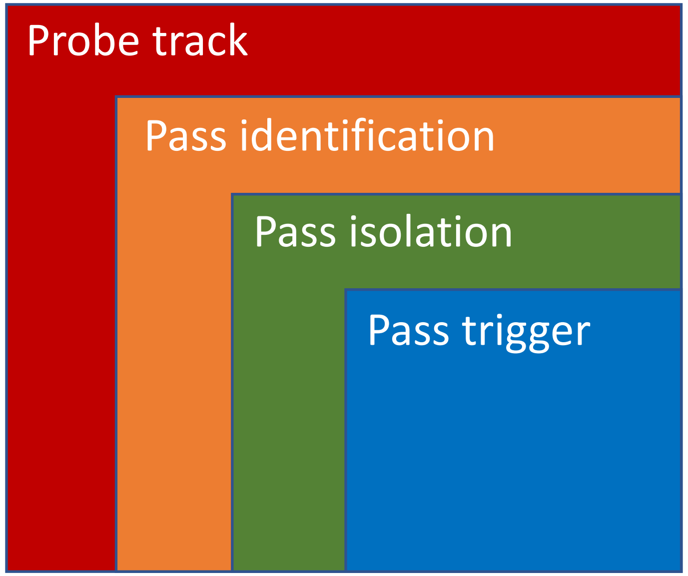
- Several combinations of efficiencies supported:

Identification
LooseID
MediumID
MediumPromptID
TightID
SoftID
HighPtID
TrkHighPtID
DisplacedID

Isolation
LooseIso
TightIso
LooseTrkIso
TightTrkIso

Trigger
Isolated muon triggers
Muon triggers
Displaced mu triggers

More than 20 possible combinations!



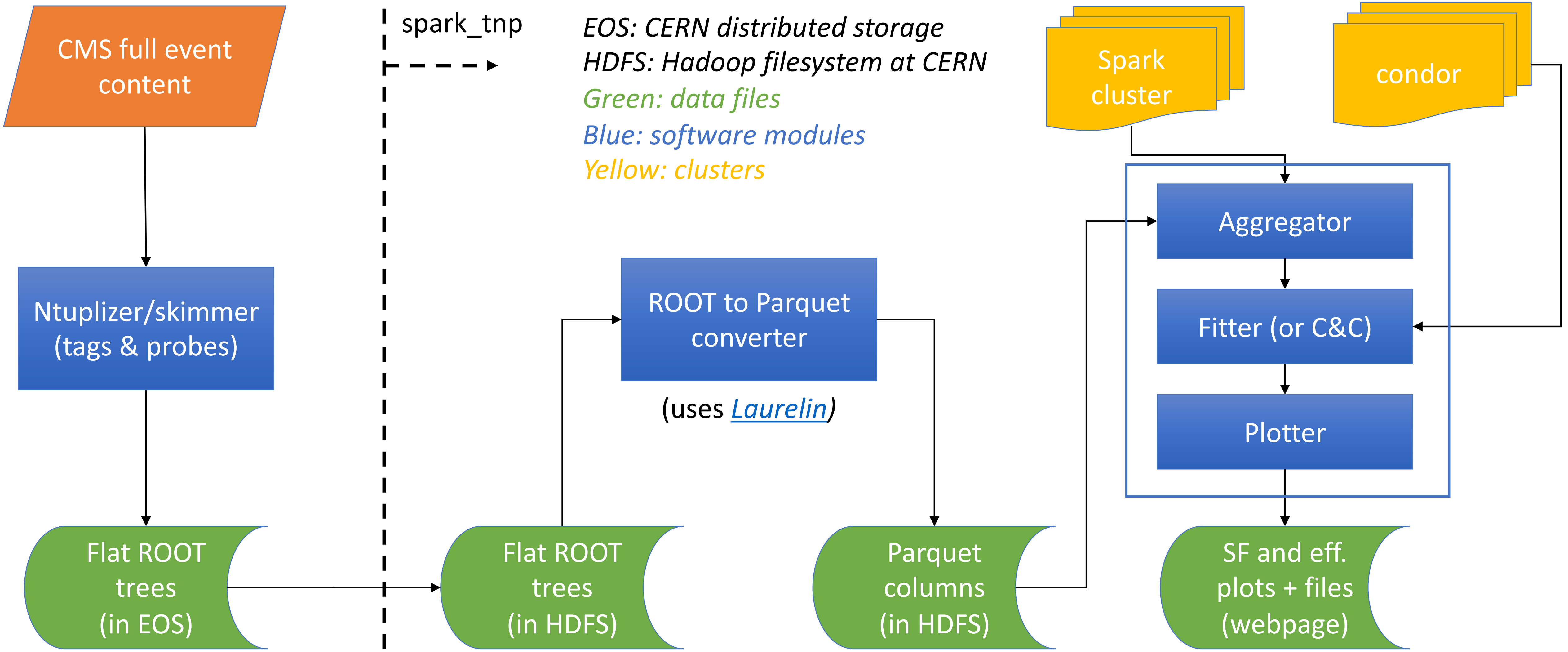
Estimated total number of fits:
About 10,000

- For each efficiency, several systematic variations are studied
  - An extra factor of 10 in the number of computations needed

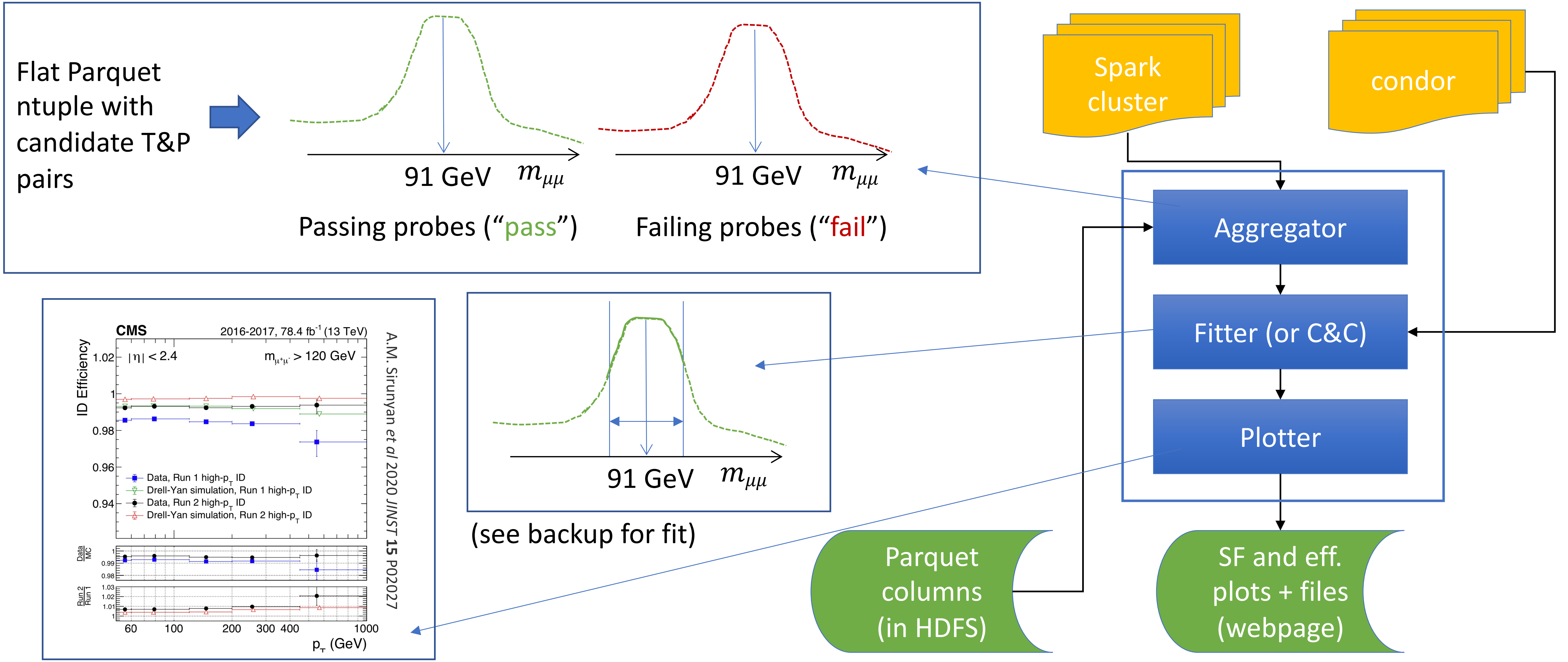
Our new package: spark_tnp

- Over the past year, a new T&P framework, ***spark_tnp***, has been developed by the Muon POG, originally written by Devin Taylor
  - Leverages CERN's *Apache Spark* cluster ("analytix") for computing efficiencies
  - Columnar data format (*Apache Parquet*) to efficiently interface with *Spark*
  - Managed to reduce scale factor computation time from **days to minutes**
  - Framework made available to CMS analyzers as well
    - Several groups have used *spark_tnp* to compute custom selection efficiencies
    - We have offered (and will continue to offer) training workshops for users
- Goal today is to display convenience and speed of scale factor calculation, and make framework/technique available to the wider HEP community
  - Codebase is now public at: https://gitlab.cern.ch/cms-muonPOG/spark_tnp

Scale factor software workflow

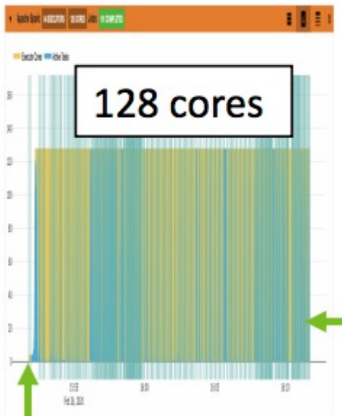


Matching the physics and the software

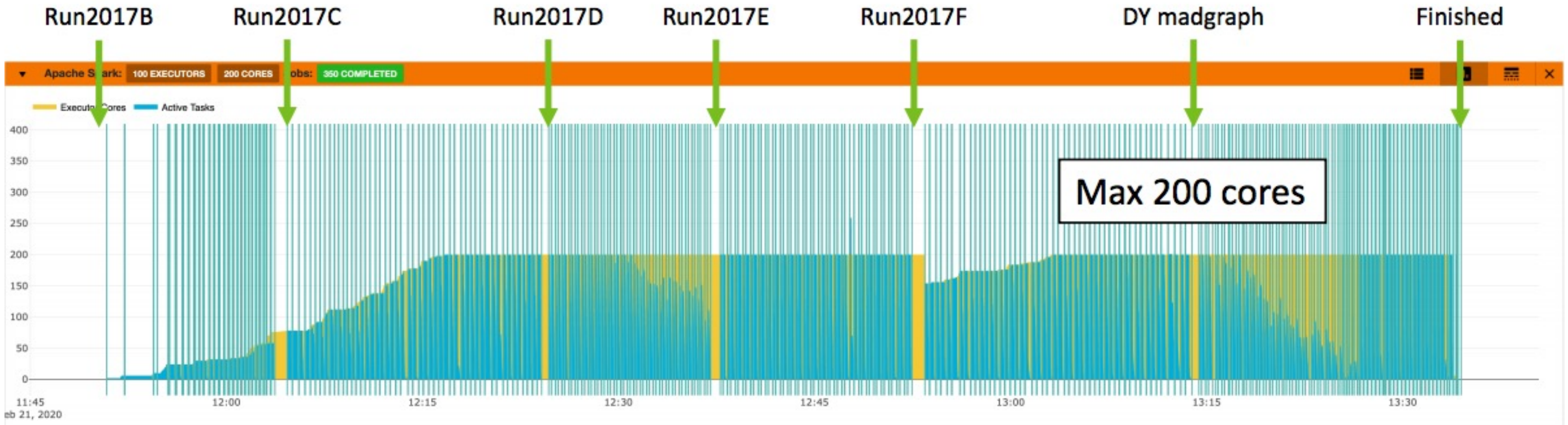


(see backup for fit)

spark_tnp speedup



- Change dataformat to parquet
- Same time scale axis (1.75 hours → ~20 mins to flatten all IDs)
- Different number of cores (200 → 128)
- Blue (yellow) means active task (no task)
  - Was not able to consistently saturate the executors when using parquet (could be even faster)
- Can do further optimizations in the way the query is built
- Majority of time spent on transferring data from executors



Devin Taylor

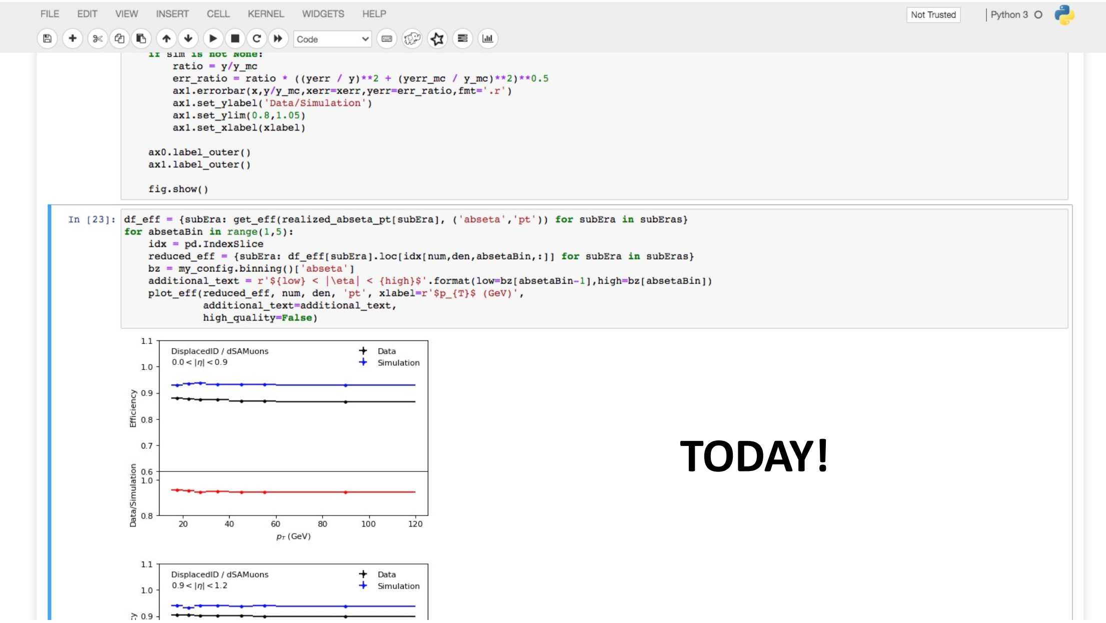
Pure *ROOT*-based T&P — **1 night for** single ID (1 lxplus core)

Spark with *ROOT* files via *Laurelin* — **12 hours** for all IDs, (16 *Spark* cores)

Spark with converted *Parquet* files — **1 hour** (~100 *Spark* cores)

User modes: script *vs.* notebook

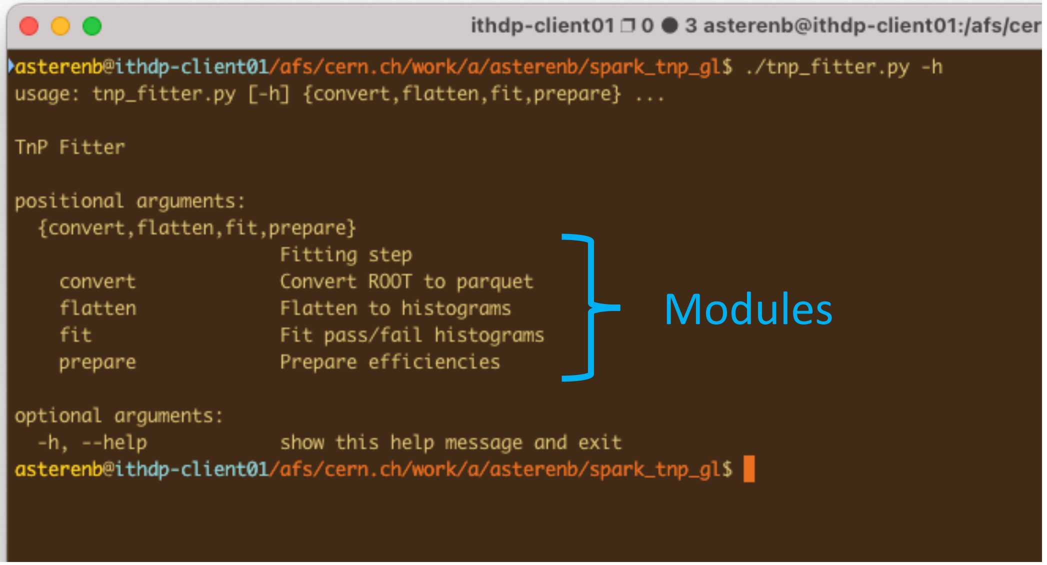
- Two modes available, with similar underlying backend
- They also use the same configuration file
- *Jupyter* & *Spark* integrated into CERN's *SWAN* environment



TODAY!

Jupyter notebooks (exploratory work)

[SWAN: Service for Web based ANalysis](#)



Command-line interface (official production)

The “Pivarski scale” of talk interactivity:

1. Pre-evaluated deck of slides
2. Watching cells being evaluated
3. Ask everyone to press shift+Enter with you
4. “What if I change something?”
5. Formal exercises in the talk

We are here today (CERN *SWAN/analytix* access needed)

Scale factor demo
Let’s move over to notebook!

Extra: sketch of fitting (official) computation

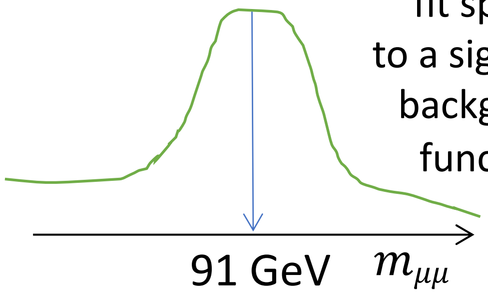
Sample with event content

Apply very loose selection to store flat ntuples with possible tag-probe pairs ("skimming")

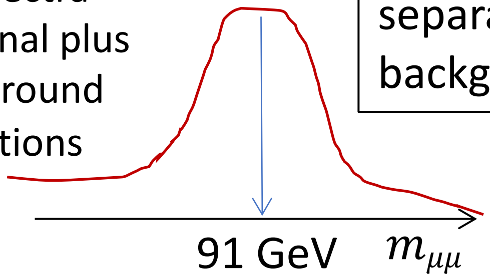
Apply:
1. Pre-selection to tag, probe, and/or event
2. Baseline selection to probe ("*denominator*")
3. Test selection to probe (may pass or fail) ("*numerator*")

Bin passing and failing probes in bins of dimuon invariant mass

fit spectra to a signal plus background functions



Passing Z probes ("pass")



Failing Z probes ("fail")

Fitting helps to separate signal and background

$$N_{\text{pass}} = f_s(\vec{p})\ N^s_{\text{pass}} + f_b(\vec{q})\ N^b_{\text{pass}}$$

$$N_{\text{fail}} = f_s(\vec{p})\ N^s_{\text{fail}} + f_b(\vec{q})\ N^b_{\text{fail}}$$

f_s: fitting function for signal shape

f_b: fitting function for background shape

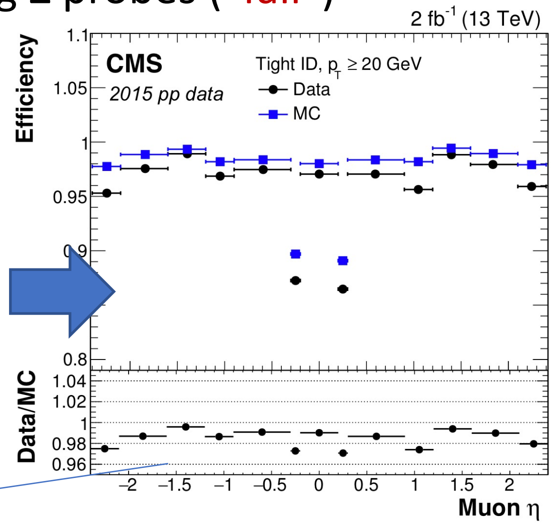
Efficiency:

$$\frac{N^s_{\text{pass}}}{N^s_{\text{pass}} + N^s_{\text{fail}}}$$

Efficiency in data

Efficiency in MC

Bin in pT, rapidity, etc.



A.M. Sirunyan et al 2018 JINST 13 P06015

Scale factor!