



# CMS experience at INFN with distRDF over HTCondor

D. Ciangottini
on behalf of the INFN/CMS AF team

# Outline

- INFN analysis infrastructure objectives
- Going interactive with RDF
  - evolving toward a declarative and interactive paradigm
- Seamlessly distribute tasks via DASK
- Real-life implementation of a CMS analysis
- Preliminary performance testing
- Plans


**Contributors**

- Diego Ciangottini
- Daniele Spiga
- Tommaso Tedeschi
- Mirco Tracolli
- Tommaso Boccali
- Massimo Biasotto
- Massimo Sgaravatto
- Stefano Nicotri
- Francesco Failla

**Also thanks to ROOT/SWAN developers for the support**

- Enrico Guiraud
- Enric Tejedor
- Vincenzo Padulano

# What are we talking about ( in a nutshell )

An **R&D project** started at the end 2021 to study if / how to **improve resources usage** for data analysis and (more challenging) how to enable the **exploitation of new approaches, new paradigms for analysing data** at CMS. Looking at Phase2 but targeting already Run3

The initially focus has been almost completely on computing and technological related matters:

- So far we did a **feasibility study** putting together several pieces and tested various capabilities for a resource integration model
- Actually also **early tests with real analysis** has been done

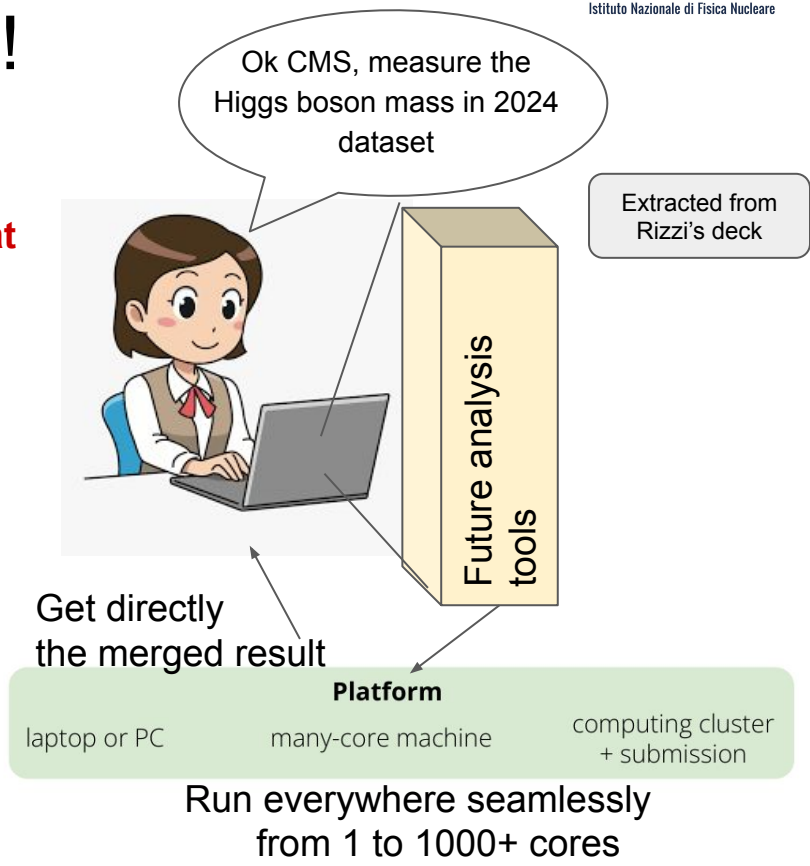
⇒ **Spoiling:** things are progressing nicely so we are looking for a step forwards now



ciangottini@pg.infn.it

# One objective: go declarative!

- Avoid wheels, do physics
  - Do not code event loops, but rather **declare only what you want to do** in the end
- Let the **framework to optimize things**
  - No configuration for data splitting or for explicit multi-threading
  - Get the best throughput out of the infrastructure
- Share and reuse **“code for humans”**
  - Easier to debug
  - Harder to get lost






ciangottini@pg.infn.it

# User's perspective

**Fast turnaround : how to allow the user to analize billions on NanoAOD within N hours**

- fast iteration time is essential for debugging experimental/theoretical/technical issues and for developing/improving the analysis
  - For O(billions) of events this means event throughput in the MHz

**Instead of..**

- **Submit** O(1000) single core jobs to condor batch system reading from mass storage, writing O(200MB) of histograms to afs
- **Resubmit** the  fraction of jobs which failed the first time
  - and the others which failed the second time..
- **Merge**



ciangottini@pg.infn.it

# What do we need?

A system that grants access to computing resources for analysis and enables a hybrid model: batch and interactive patterns –> Not a one size fits all solution

**Interactive** - “Read this as: I can get a Jupyter notebook as big as a Tier2”

- Transparently parallelize over a huge amount of cores allows implementing the interactivity
- I’m writing Jupyter, you can read it as distributed python
- And more in general mitigate/avoid user waiting idle for grid jobs

**Batch like processing** - “Read this as: I have a place where I can submit ( i.e condor_submit ) my analysis jobs”

- Yes, “yet another batch”... completely dedicated to analysis



ciangottini@pg.infn.it

# HW perspectives: the vision

Behind the scene: see this as a **continuum implementation via HTCondor + Dask**

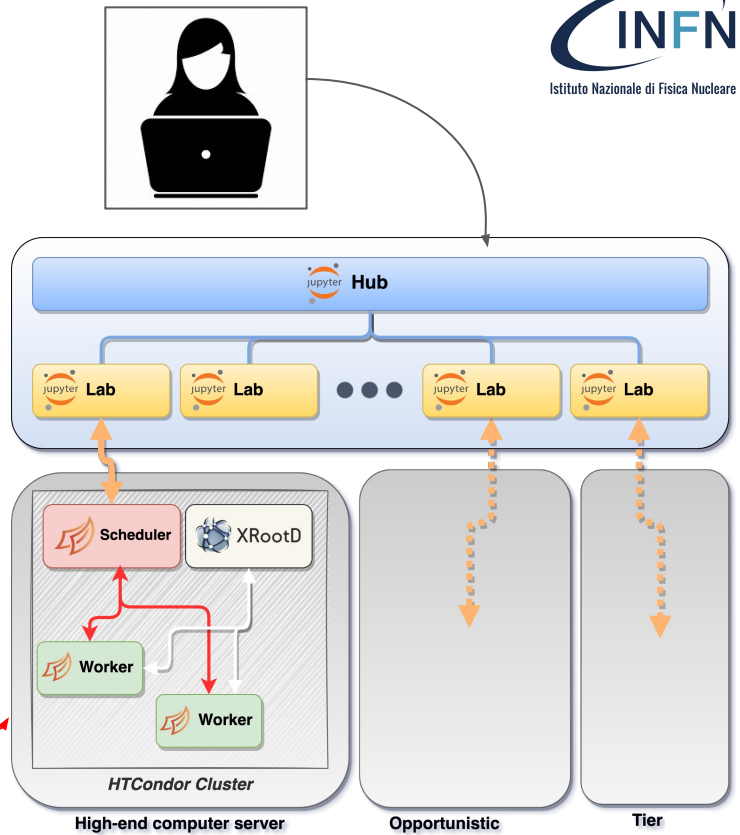
- "Grid vs Cloud vs HPC is not anymore a user issue". Everything is hidden behind a single Hub

We (CMS/INFN) have a distributed and pledged resources topology

- Integrate everything, possibly even opportunistic and "private" clusters

Not a single solution fits everything ( again )

- We need to test various configurations, to measure the costs/benefits and design future models




**A cluster/ a single fat node / a cluster of fat nodes…**



ciangottini@pg.infn.it

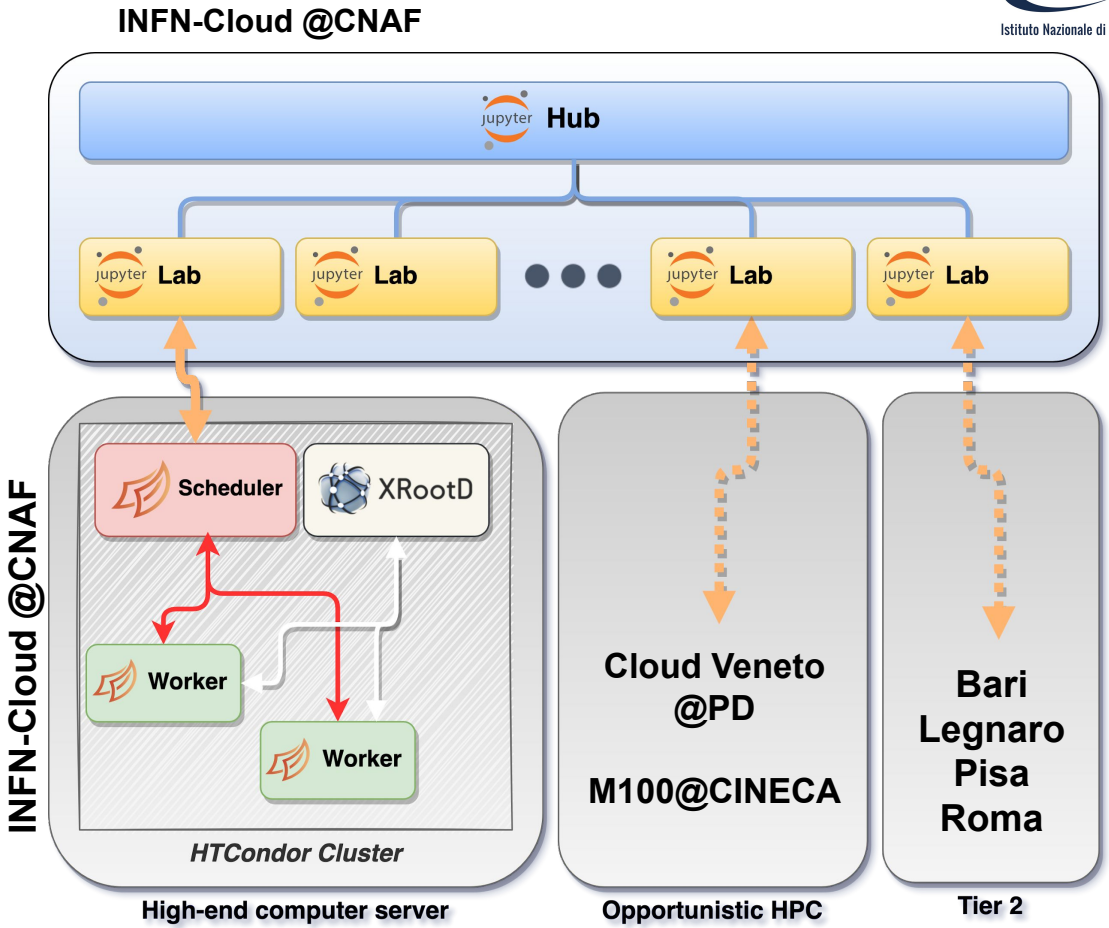
# Current Status

**We have developed an e2e testbed system which is now available**

A cluster hosted at CNAF (INFN-Cloud )
- Central services
- Fraction of local resources

**T2-Legnaro** is providing flat O(100) cores a seed that can be extended - ready to provide more on-demand

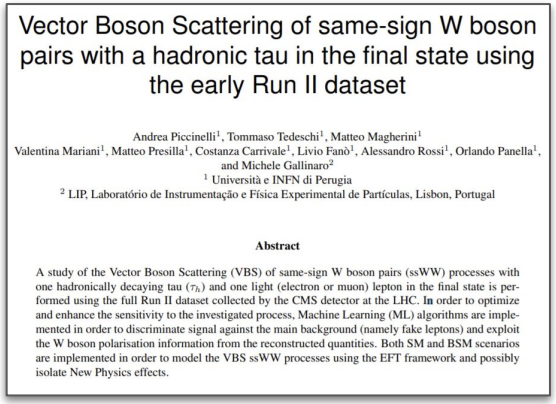
**Almost all italian Tier2 have been functionally tested**



# Interactive via distRDF: first use case

**A VBS SSWW** analysis based on NanoAOD inputs (~plain ROOT files) is **being ported** (credits to T. Tedeschi) from legacy approach (nanoAOD-tools/plain PyROOT-based) **to RDataFrame** in order to obtain:

- Enhanced **user experience** thanks to the modern high-level declarative interface
- Improved **efficiency** thanks to intrinsic parallelization
- **Optimized operations on data**
  - obtained by merging analysis steps
- **distribute workflows** on different back-ends with ~0 changes in code base



Vector Boson Scattering of same-sign W boson pairs with a hadronic tau in the final state using the early Run II dataset

Andrea Piccinelli[1], Tommaso Tedeschi[1], Matteo Magherini[1]
Valentina Mariani[1], Matteo Presilla[1], Costanza Carrivale[1], Livio Fanò[1], Alessandro Rossi[1], Orlando Panella[1], and Michele Gallinaro[2]
[1] Università e INFN di Perugia
[2] LIP, Laboratório de Instrumentação e Física Experimental de Partículas, Lisbon, Portugal

**Abstract**

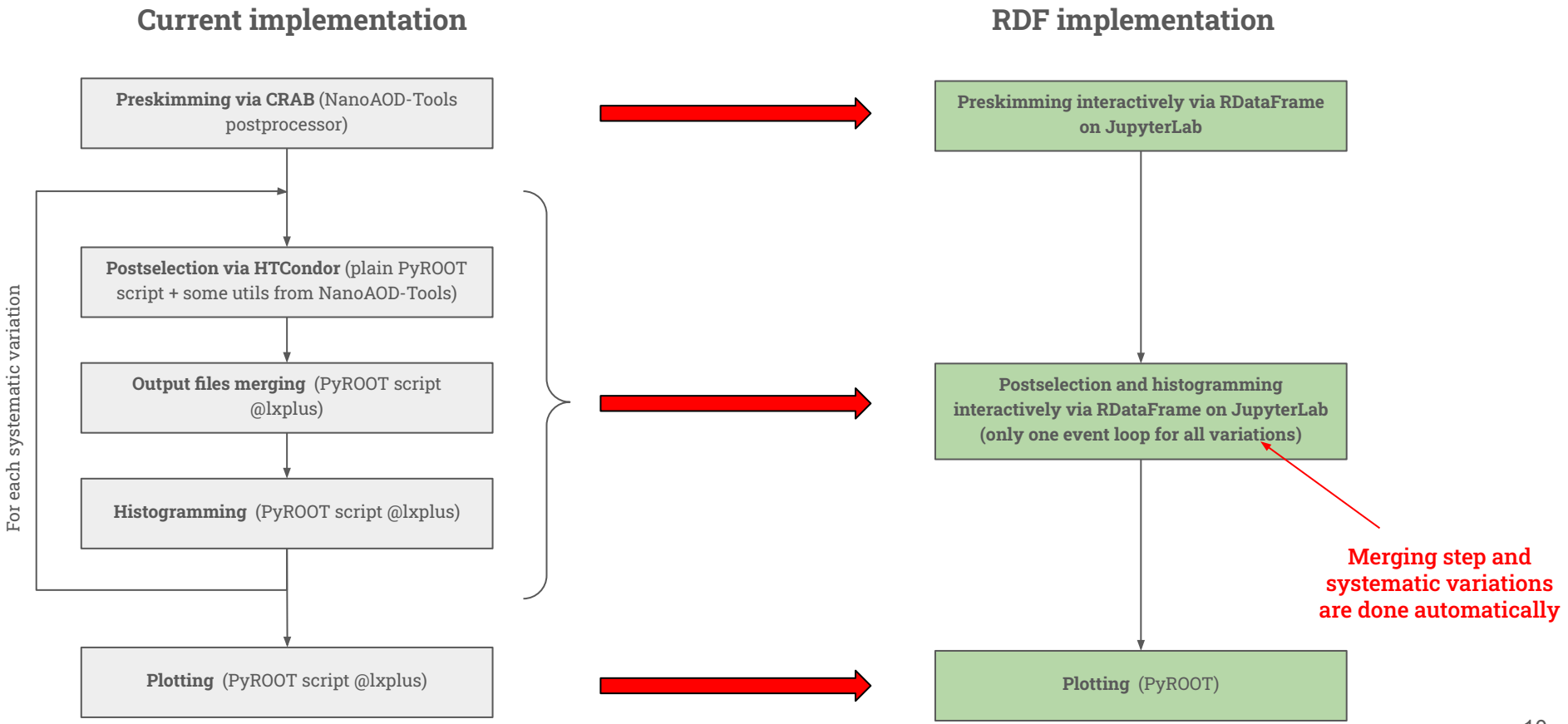
A study of the Vector Boson Scattering (VBS) of same-sign W boson pairs (ssWW) processes with one hadronically decaying tau ($\tau_h$) and one light (electron or muon) lepton in the final state is performed using the full Run II dataset collected by the CMS detector at the LHC. In order to optimize and enhance the sensitivity to the investigated process, Machine Learning (ML) algorithms are implemented in order to discriminate signal against the main background (namely fake leptons) and exploit the W boson polarisation information from the reconstructed quantities. Both SM and BSM scenarios are implemented in order to model the VBS ssWW processes using the EFT framework and possibly isolate New Physics effects.

**VBS SSWW with a light lepton and an hadronic tau in final state** on full Run2 (**NanoAOD**)



ciangottini@pg.infn.it

# Legacy →distRDF migration



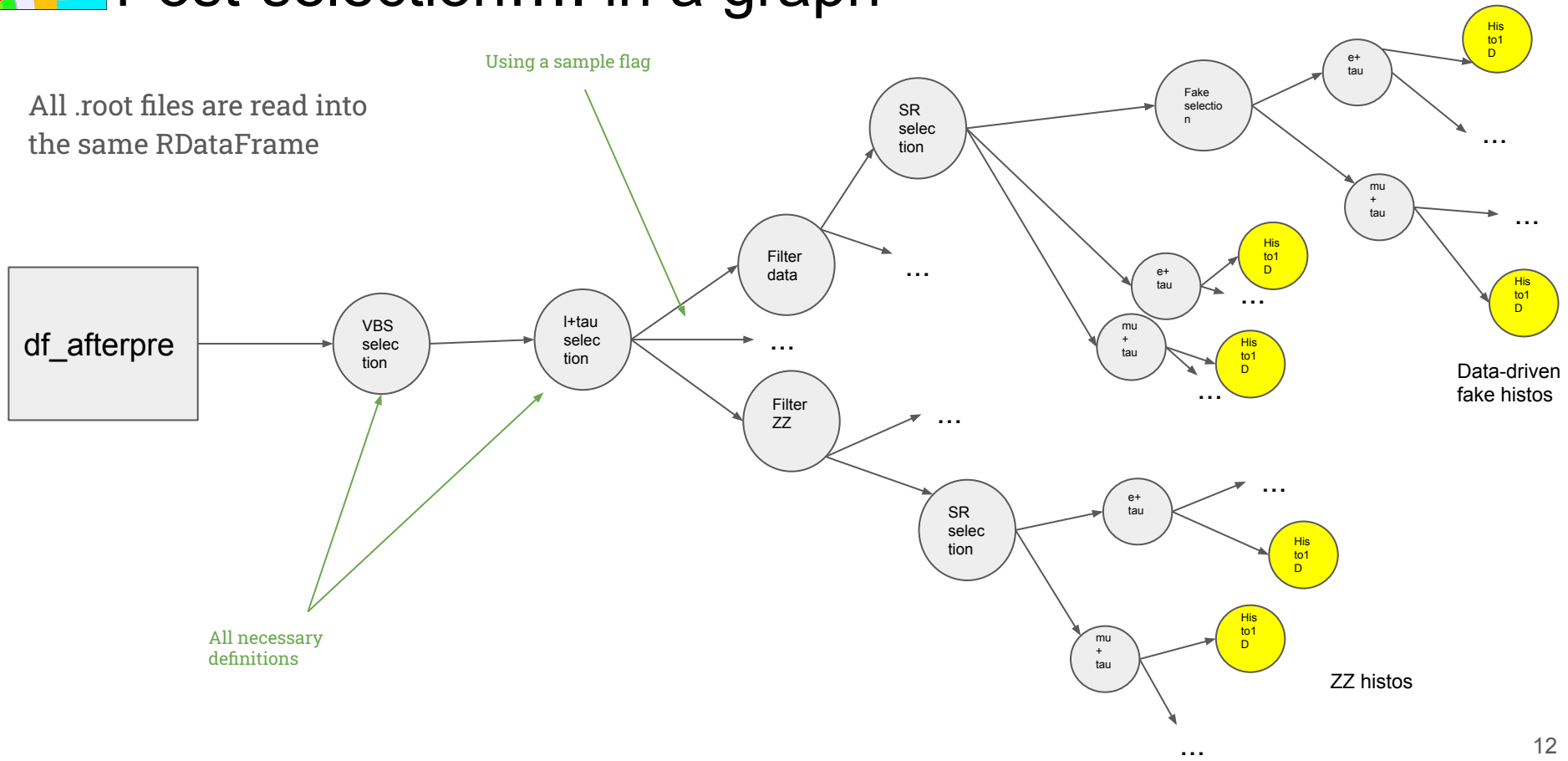
# How the code looks like, in a nutshell

```
def inizialization_function():
    ROOT.gInterpreter.Declare('#include "utils_functions.h"')

df = ROOT.RDF.Experimental.Distributed.Dask.RDataFrame("Events", chain, nPartitions = N, client = client)  #define the dataframe

df_processed = df.Define("column_c", "function(column_a, column_b)")\
                 .Filter("filtering_function(column_d)", "A filter")
                 ...

# book a snapshot (i.e. a saving)-> used in preselection
opts = ROOT.RDF.RSnapshotOptions()
opts.fLazy = True
df_lazy_snapshot = df_processed.Snapshot("treeName", "fileName.root", opts)

# book an histogram -> used in postselection
df_lazy_histo = df_lazy_snapshot.Histo1D("column_a", "weights_column")

# to trigger execution
histos = df_lazy_histo.GetValue()

# to inspect data
df_saved.Display(["column_a", "column_b", "column_c"], nRows = 1).Print()
+-----+----------+----------+----------+
| Row | column_a | column_b | column_c |
+-----+----------+----------+----------+
| 0   | -1       | -1       | -1       |
+-----+----------+----------+----------+
```

C++ functions that manipulates RVec objects
Target of the porting

# Post-selection…. in a graph

All .root files are read into the same RDataFrame



# Systematic variations

Moving from the nanoAOD-tools based analysis was reasonably **straightforward to implement in RDF via `.Vary` method:**

```
nominal_hx = df.Vary("pt", "ROOT::RVecD{pt*0.9, pt*1.1}", ["down", "up"])
                .Filter("pt > k")
                .Define("x", someFunc, ["pt"])
                .Histo1D("x")
hx = ROOT.RDF.Experimental.VariationsFor(nominal_hx)
hx["nominal"].Draw()
hx["pt:down"].Draw("SAME")
```



ciangottini@pg.infn.it

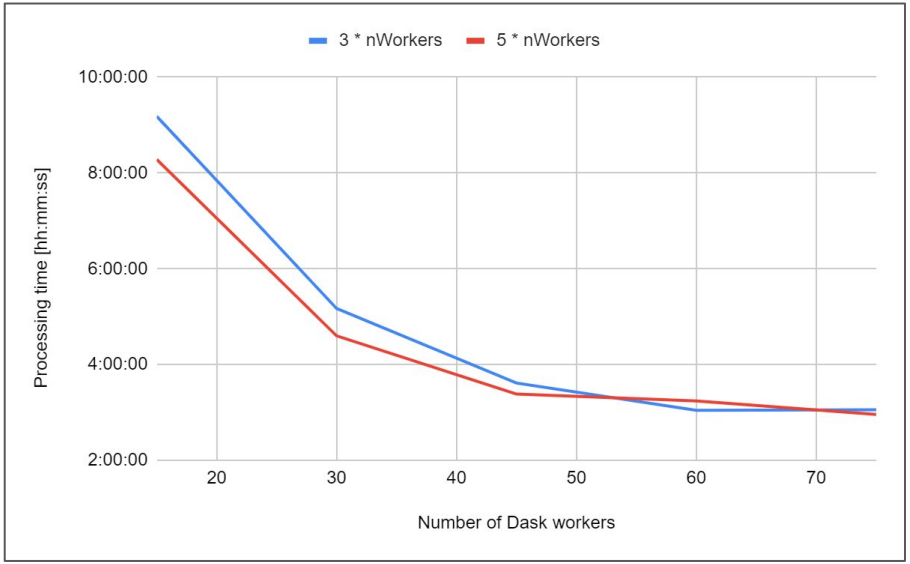
# Preliminary performance comparison

Overall **time-to-histogram for post-selection** step

- Legacy approach on AF (using 90 dedicated WNs):
  - **6h03m**, divided into:
    - HTCondor processing: **5h27m**
    - Data transfer: **0h17m**
    - Merging: **0h20m**
    - Histogram production: **0h03m**
- RDF + Dask approach on AF (using 90 Dask workers):
  - **3h05m** using 180 partitions, results obtained with no further processing step

The inefficiencies on the pure HTCondor ways can have multiple sources and might be optimized, still it is an **encouraging first outcome**.

Preselected ReReco 2017 data + MC (9605 files, 1.5 TB, 953 million events)

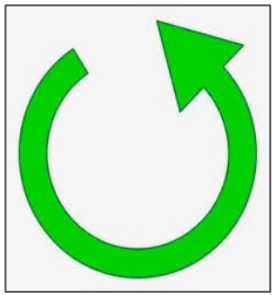



Performance results require further controlled tests and understanding, but still a good starting point

Similar measurements for pre-selection steps are planned!

# How we see the next steps

The mantra should be “Starting little and expand step by step, co-design and iterate”



1. Starting little: **Use the current proof of concept**
   - Explore new ideas…
   - Identify problems, bottlenecks
   - Define user requirements
2. **Co-design: facilitate discussions between users and computing people**
   - Provide feedback and discuss how to solve problems
   - New features requests and needs
3. **Extend the proof of concept by meaning**
   - Identify specific development
   - Implement technical solutions and deploy new features
   - Add computing resources, eventually configure the underlying setup to adapt to …



ciangottini@pg.infn.it

# Workplan

- Additional **volunteer users have been involved**
  - Two use cases are closely working to enter the test phase
  - Both of them are **already using RDF locally in the current analysis** implementation
- Scale
  - Ready **to get O(1k) cores for scale tests**
    - In particular we want to make measures on pre-selection step
  - Verify **multi-user scenario**
- Performance measurements on **pre-selection** step
  - Heavier from I/O perspective
- Debugging some occasional Dask **task failure with no clear error pattern**
- Improving **automatic retry of failed tasks**

# Conclusions

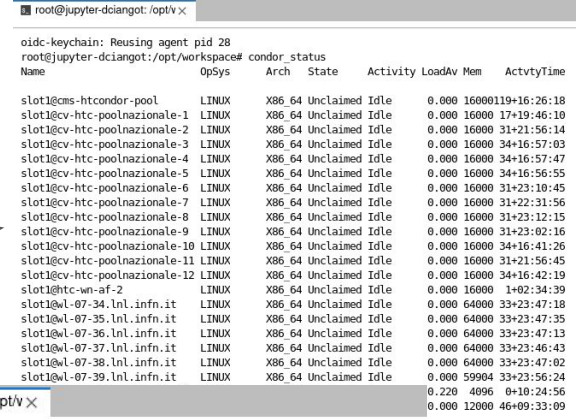
- Overall **the transition from a “legacy” analysis code looks reasonably easy**
  - At least for analyses based on NTuple-like data source
  - Good time to provide **early feedback and to interact with developers**
- The **capability to scale seamlessly** what I currently run locally is a great **added value**
  - The preliminary performance test is also showing pretty good results
- Still to **improve on distributed logging and debugging procedure**
  - Sort of DASK/HTCondor native problem to be honest
- More **users joining means different pattern** and probably different requirements
  - Really looking forward to it!

# BACKUP

# What can I do in there?

- Batch/Legacy
  - Run your analysis on a batch system with resources collected over sites
  - Run your analysis on a batch system targeting specifically HPC resources
- Quasi-interactive python scripting
  - On-demand leverage big-notebooks on big machines (hpc node, dedicate hw) and run locally with a portable and ready-to-use environment
- Interactive python notebooks
  - Effortlessly scale local code with distributed mode RDataFrame workflows over a T2 site or over dedicated/specialized resources
  - Edit an reproduce plot interactively

Demo tomorrow





Demo tomorrow