Seamless transition from **TTree** to **RNTuple** analysis with RDataFrame

Marta Czurylo[1], Andrii Falko[3], Danilo Piparo[1], Enric Tejedor Saavedra[1], Enrico Guiraud[1,2], Jakob Blomer[1], Philippe Canal[4], Vincenzo Eduardo Padulano[1]

(1) CERN
(2) Princeton University
(3) Taras Shevchenko National University of Kyiv
(4) Fermi National Accelerator Laboratory

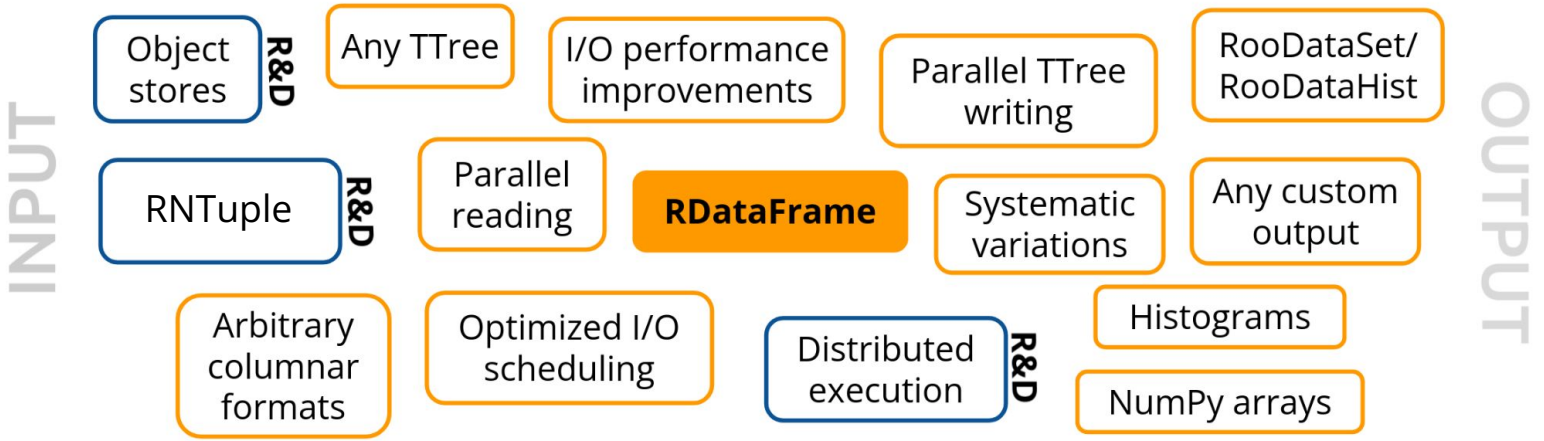
ACAT 2024
Stony Brook University

RDataFrame

```
# Create an RDF
df = ROOT.RDataFrame(dataset)
# Create observable
df = df.Define("my_observable", "x+y")
# Perform cuts
df = df.Filter("my_observable > z")
# Add systematic variations
df = df.Vary(
            "my_observable",
            "my_observable*my_observable_scale_up()",
            ["my_observable_scale_up"],
    )
# Fill in a histogram
h1 = df.Histo1D("my_observable")
```

ROOT analysis interface since 6.14 (2018):

- Intuitive
- Declarative and fast
- Flexible



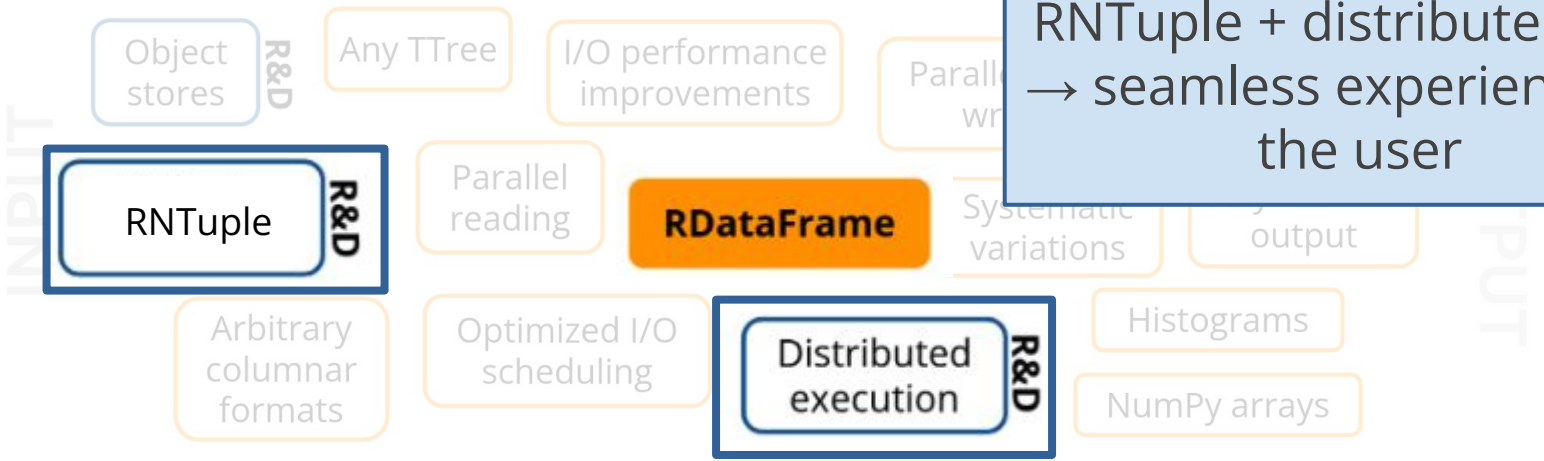
RDataFrame

```python
# Create an RDF
df = ROOT.RDataFrame(dataset)
# Create observable
df = df.Define("my_observable", "x+y")
# Perform cuts
df = df.Filter("my_observable > z")
# Add systematic variations
df = df.Vary(
            "my_observable",
            "my_observable*my_observable_scale_up()",
            ["my_observable_scale_up"],
    )
# Fill in a histogram
h1 = df.Histo1D("my_observable")
```

ROOT analysis interface since 6.14 (2018):

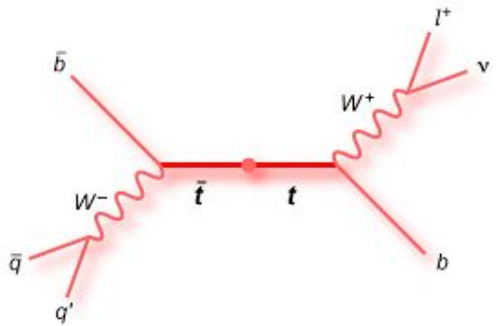
- Intuitive
- Declarative and fast
- Flexible

Today's focus:
RNTuple + distributed RDF
→ seamless experience for the user



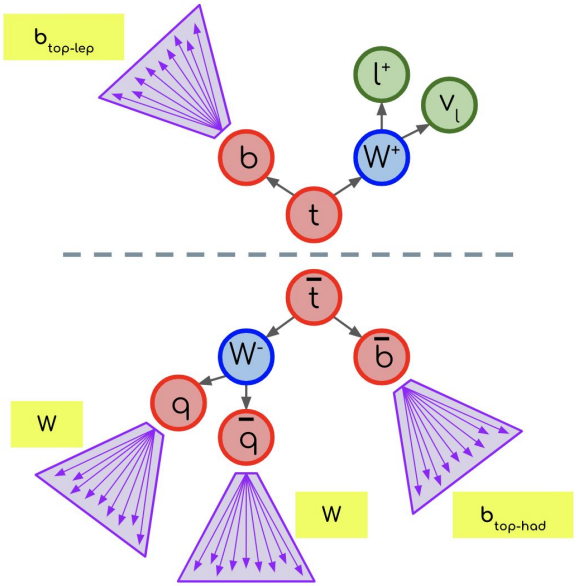
Analysis Grand Challenge

- AGC – HEP analysis benchmarks
  - In various implementations, including with RDataFrame
  - In particular: $t\bar{t}$ analysis based on CMS Open Data



Current status of AGC with RDF

- Talk at CHEP last year
  - AGC v.0.1.0
- Since then:
  - RDF implementation: new data format – NanoAOD (AGC v.1)
  - RDF implementation: Machine Learning inference for **jet-parton assignment** (AGC v.2)
- In this talk:
  - Replicating AGC benchmark with RNTuple, including distributed execution via condor



Details of the analysis

Distributed analysis environment

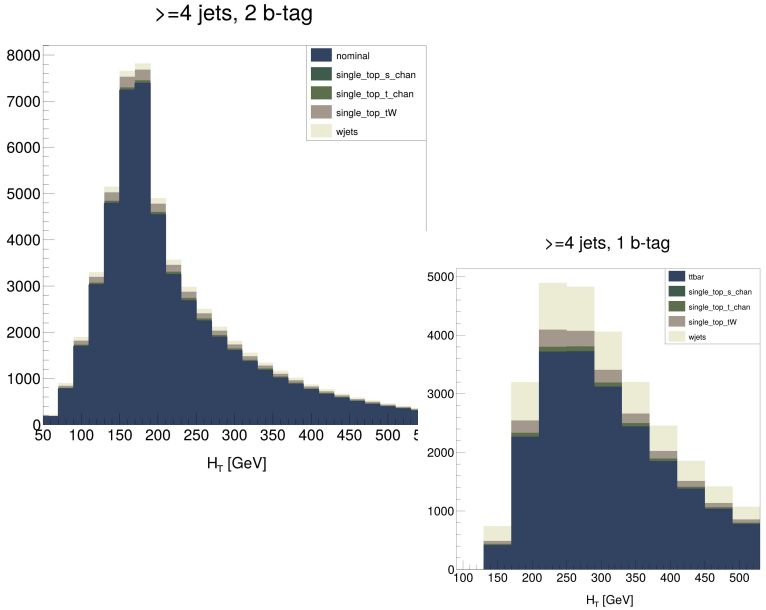
- Number of ways to run distributed RDF
- Focus here - rediscover **existing** infrastructures and services in a modern way
  - SWAN
  - HTCondor pools
  - Schedule via Dask

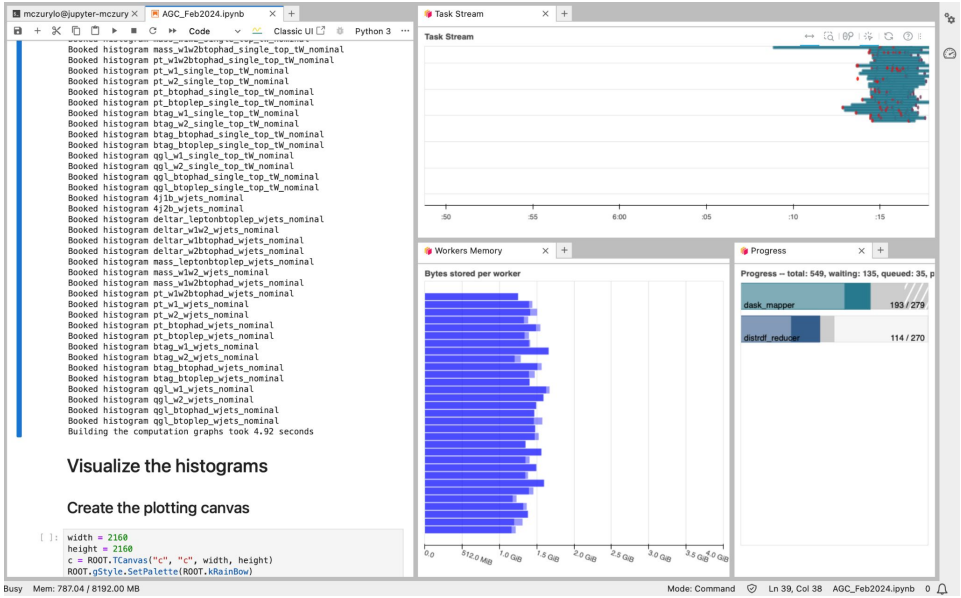
cvmfs + EOS + CERN batch + ROOT → CERN Analysis Facility (?)

Distributed AGC with TTree and RNTuple – user side

The only change for the user - the ROOT input file!

```
REMOTE_DATA_PREFIX: str = "root://eospublic.cern.ch//eos/root-eos/AGC/"
REMOTE_DATA_PREFIX: str = "root://eospublic.cern.ch//eos/root-eos/AGC/rntuple/"
```

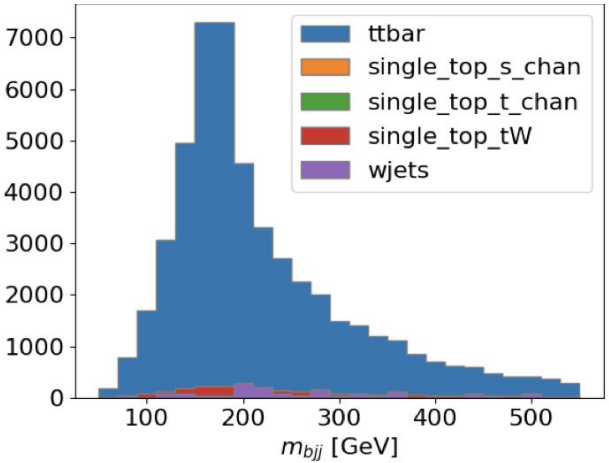




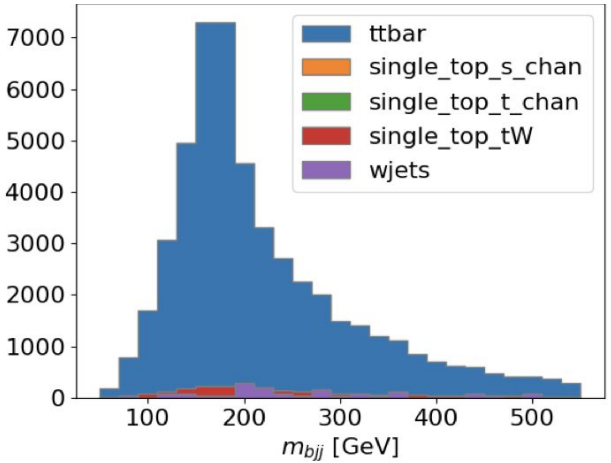
Validation of histograms

- Distributed analysis with RNTuple, **it just works!**
- Satisfactory agreement with equivalent histograms from other execution policies
  - 100% bin-by-bin agreement for 120 histograms
  - 2 histograms with <1% disagreement because of the bin migrations

RDF



IRIS-HEP

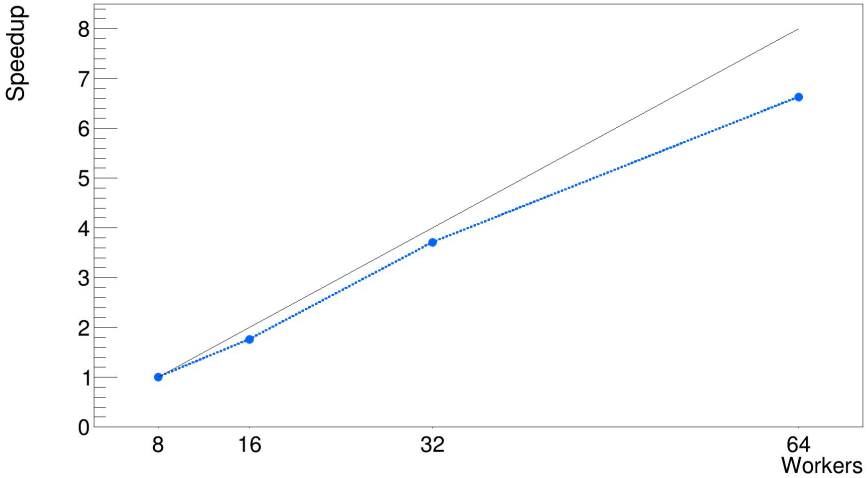


AGC v.1 performance – TTree and RNTuple

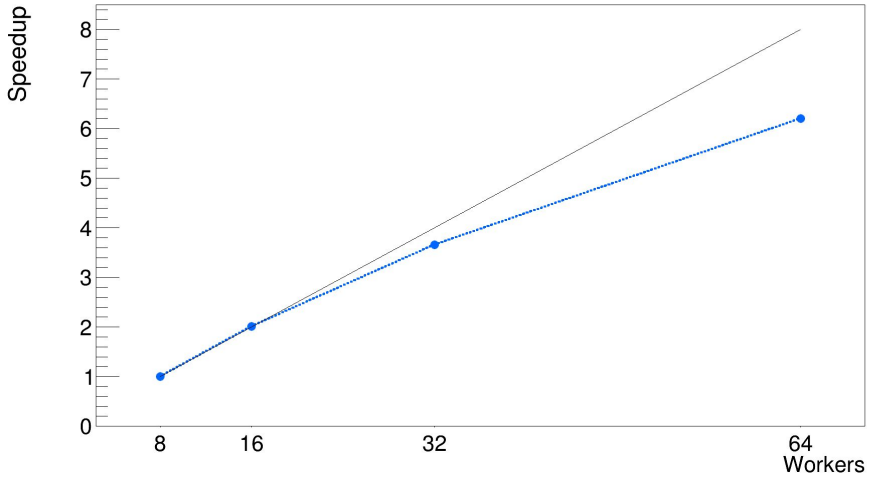
- Scaling tests on SWAN, AGC v.1 → Up to 64 workers

Speedup vs number of workers

TTree



RNTuple



Summary and next steps

- Running AGC on SWAN with the HTCondor pools via Dask with both TTrees and **RNTuples** is **smooth**
  - **With zero code change for the user**
  - Achieving (almost) **perfect agreement** with available histograms
  - Sanity check: distributed execution up to **64 workers with RNTuple**
- Making RDataFrame ready for HL-LHC analyses
- Next steps
  - Keep track of latest AGC benchmark specification
  - Include different benchmarks with existing or new TTree open data converted to RNTuple