Improving the RDataFrame interface for typical cases in physics analysis

Enrico Guiraud, Stefan Wunsch

ROOT

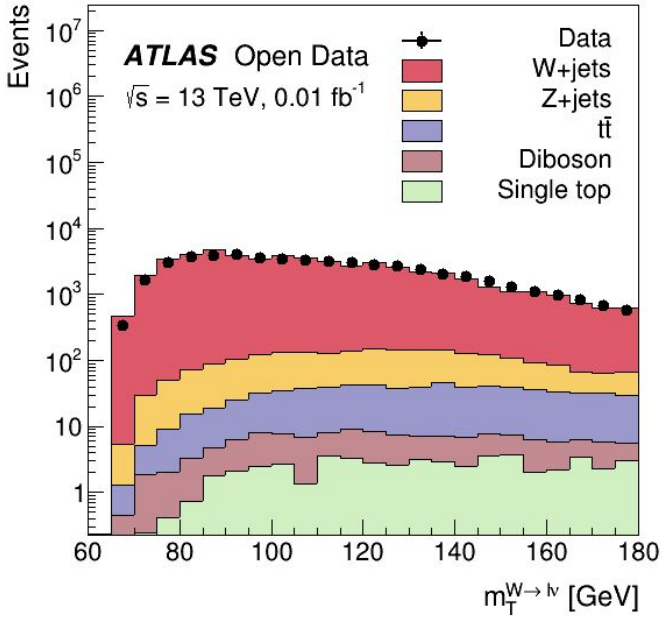
Data Analysis Framework

https://root.cern

The story

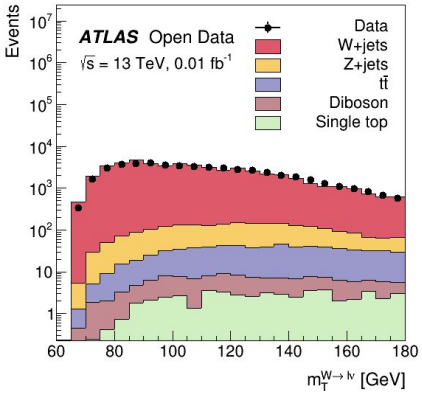
- **The problem**
  RDataFrame does not cover some typical use cases in HEP analysis in the most efficient way

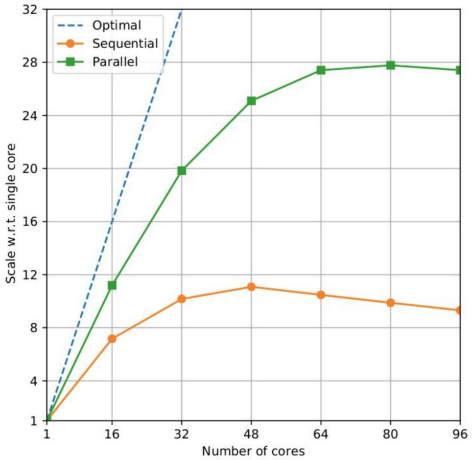
- **Current status**
  As soon as you cannot express your analysis **with a single RDataFrame,** the scaling will suffer with many threads or workers.



Do we have a reproducer?

- RDF tutorial with reconstruction of the W boson based on ATLAS Open Data: https://root.cern/doc/v622/df105_WBosonAnalysis_8py.html
- 65 files with a size of about 60 GB in total
- Requires multiple RDFs because of a different weight column for each dataset
- Already shown in the last PPP: Parallel RDF event loops make the analysis run significantly faster on many threads





Does it matter for HEP analyses?

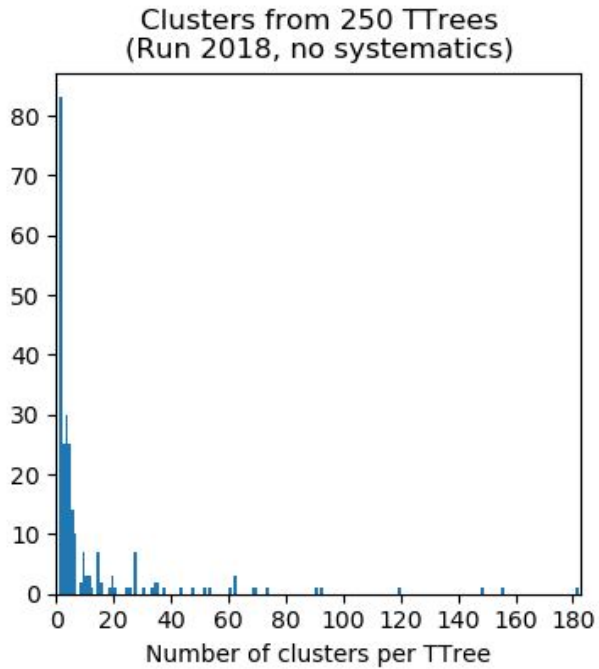
Typical structure for the analysis of (ROOT) datasets, e.g. for a full Run 2 analysis of SM Higgs boson to two tau leptons:

- **Each physical process** is described by multiple simulations, which are stitched together. Typically each simulation is kept in a separate ROOT file.
- **Each decay channel** requires an additional TTree in the ROOT file, e.g., 4 (μτ, eτ, ττ, eμ) in a typical Higgs to two taus analysis. Systematics require many additional copies of the Trees, about 40 in this analysis.
- The full procedure is repeated for **each year of data taking**, e.g., 3 (2016, 2017, 2018) for full Run 2.
- **Approximate numbers of TChains for CMS Higgs to two tau leptons processing full Run2**
  - 3 (years) × 90 (files) × 4 (channels) × 40 (systematics) = **43200 TTrees**
  - 43200 TTrees / 7 processes ≈ **6200 TChains**
  - About 4 TB in total

```
$ # Drell-Yan process
$ ls DY*
DY1JetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
DY2JetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
DY3JetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
DY4JetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v1
DYJetsToLLM10to50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
DYJetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_amcatnlo-pythia8_ext2-v1
DYJetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_amcatnlo-pythia8_v1
DYJetsToLLM50_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v1

$ # W+jets process
$ ls W*Jets*
W1JetsToLNu_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
W2JetsToLNu_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
W3JetsToLNu_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
W4JetsToLNu_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2
WJetsToLNu_RunIIAutumn18MiniAOD_102X_13TeV_MINIAOD_madgraph-pythia8_v2

$ # Data
$ ls Tau_Run2018*
Tau_Run2018A_17Sep2018v1_13TeV_MINIAOD
Tau_Run2018B_17Sep2018v1_13TeV_MINIAOD
Tau_Run2018C_17Sep2018v1_13TeV_MINIAOD
Tau_Run2018D_PromptRecov2_13TeV_MINIAOD

$ # ...
```

Does it matter for HEP analyses?

- Analyses typically require a large amount of RDFs to process all datasets
- A good amount of clusters per TChain and thread/worker is crucial to achieve good scaling
- On the right: Distribution of clusters per TTree (multiply by 7 to get the approximate number of clusters per TChain)
- **Scaling with many threads or in a distributed environment (aka PyRDF) is strongly limited in realistic scenarios**



The problem

- Realistic analysis in HEP cannot be expressed as a single RDF computation graph
  - Analysis has to differ between datasets, e.g., different weights for different samples
  - Analysis performs different operations on different datasets, e.g., data vs simulation
- If many RDFs have to be used and run one after each other, each RDF will not process enough clusters to allow scaling with many threads or workers
- This is what happens in actual analyses on full Run 2 scale!

Proposals for new RDF features

Transformations per dataset

- **Feature**
  - Offer the possibility to run transformations once per dataset
- **Reasons to add this feature**
  - In line with the current RDF programming and computing model
  - Computational efficient solution to typical HEP cases such as sample weights
- **Implementation details**
  - Introduce magic column `rdfdataset_` and transformations such as `DefineForDataset`
- **Potential problems**
  - Requires new features in the RDF data source
  - Does not solve all potential cases, e.g., using different branches per dataset

```
// Construct RDF
RDataFrame df(tree, files);

// Declare computations
auto get_scale = [](std::string& dataset)
        {
            // dataset = filename.root/treename
            if (dataset.contains("Data")) return 1.0;
            else if (dataset.contains("DY")) return 0.9;
            else if (dataset.contains("WJets")) return 1.1;
            else throw std::runtime_error("Unknown dataset");
        };

auto h = df.DefineForDataset("weight", get_scale)
           .Histo1D("nMuon", "weight");

// Access result
h->Draw();
```

Facility to run RDF loops in parallel

- **Feature**
  - Offer a facility to run multiple RDFs in parallel
- **Reasons to add this feature**
  - Simple to use
  - Adds an efficient computing model to the existing programming model
  - PyRDF could use the same API call to distribute work efficiently to the backend, e.g., Spark
  - Orthogonal to the transformations per dataset
- **Potential problems**
  - Shared state between different RDFs is evil
  - Advanced feature, requires excellent documentation
- **Prerequisites already implemented**
  - Make RDF thread safe (PR 6266)
  - Fix interpreter mutex (PR 6301)

```
// Construct RDFs
RDataFrame df_1(tree, files_1);
RDataFrame df_2(tree, files_2);
...

// Declare results
auto r_1 = df_1.DeclareSomeComputation();
auto r_2 = df_1.DeclareAnotherComputation();
auto r_3 = df_2.DeclareAnotherComputation();
...

// Compute results all together with
// parallel RDF event loops
ROOT::RDF::Run({r_1, r_2, r_3});
```

Summary

- We propose two new features promising large performance gains for typical cases in physics analysis which require multiple RDFs
- The goal is to recover scaling on many workers/threads, which is strongly impaired by the highly imbalanced distribution of clusters in physics datasets
- **Coming soon: Dealing with systematics in RDF computation graphs**
- **Comments, ideas, objections?**