

Wmass analysis framework

Valerio Bertacchi

Lorenzo Bianchini

Elisabetta Manca

Gigi Rolandi

Suvankar Roy Chowdhury

7th February 2020



Introduction

- A brief overview of the offline analysis framework developed for the analysis.
- Challenges : complexity and volume.
- Discussion of concepts, not much technical details.

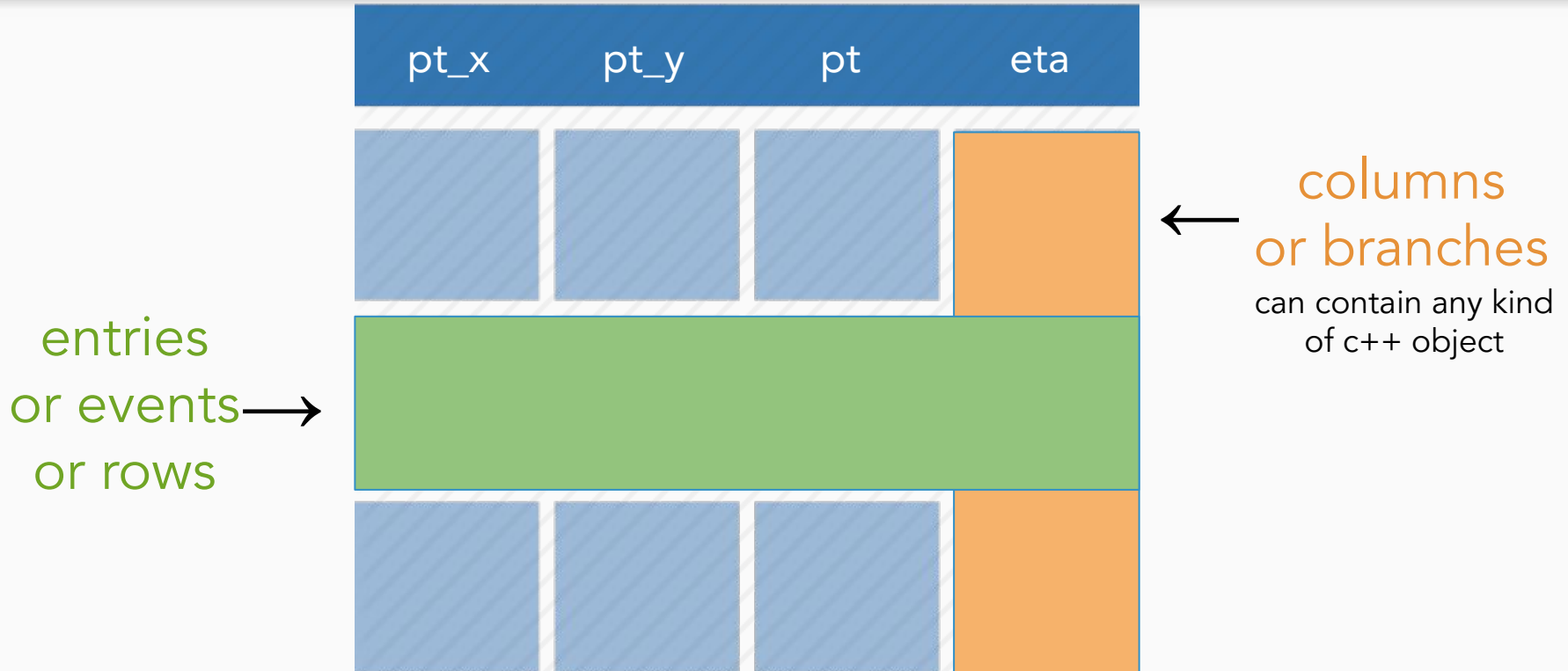
Requirements of offline software

- Our wishlist - **fast, flexible and robust framework** w.r.t the complexity of the analysis.
- What we are dealing with?
 - Even in the most simple case of studying the control plots, we have
 - About 100 histograms(~ 20 histos X 5 regions)
 - 130 variations of each histogram.
 - Complexity increases when we study the production modes in 2D spectrum(p_T , rapidity) or background estimation(3D histos in p_T , η , sign).
 - ~ 1000 2D histos with 1000 variations for each.
- Must remember that the final aim is to analyze the full Run 2 data($\sim 140 \text{ fb}^{-1}$).

Requirements of offline software

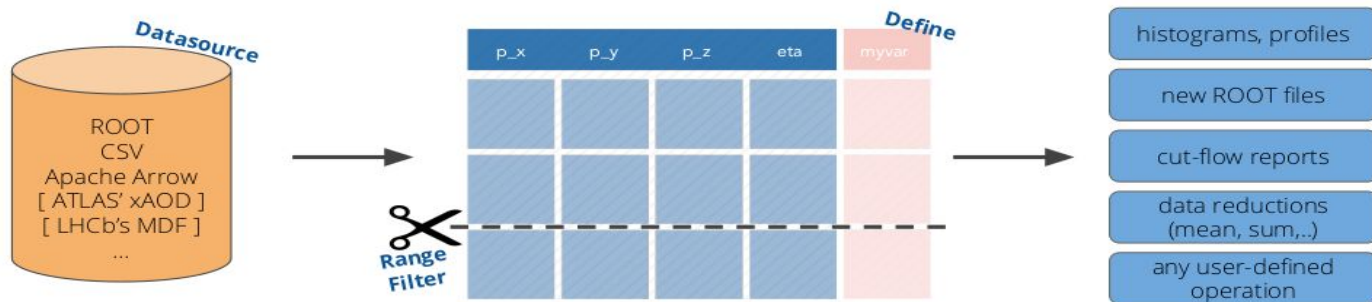
- In the view of the increasing complexity and volume of data analysis, ROOT team came up with a solution - **RDataFrame**.
 - Simple but powerful tool to analyse data with modern C++, python.
 - Parallelizable - transparent MT and supported for multi-core machines.
 - Easy to express dependencies on different objects.
 - Graph style approach, optimized event loop.
 - User writes analysis - ROOT takes care of optimization.
- E. Manca started collaborating with the ROOT development team from about ~1.5 years ago and presently a large section of the HEP community is moving towards usage of RDataFrame.

Data representation in RDataFrame



RDataFrame overview

- Three simple steps for an user.
- **Build** a DataFrame object from a source
- Transformations on the input.
 - **Define**: new columns from existing columns.
 - **Filter**: create ranges of events by applying cuts
- **Apply actions** on the transformed data.
 - e.g., Histogramming
- Event loop is triggered only when the full analysis has been set up and **an first access** to a result has been made.
 - User should be careful here.



Read tree "t" in file "f.root". For events for which "v2 > 2", fill histogram "h" with "v1"

```
TFile f("f.root");  
TTree *t = nullptr;  
f.GetObject("t", t);  
t->Draw("v1 >> h", "v2 > 2");  
TH1 *h = (TH1F*)(gDirectory->Get("h"));  
h->Draw();
```

```
TDataFrame d("t", "f.root");  
auto h = d.Filter("v2 > 2")  
          .Histo1D("v1");  
h->Draw();
```

Traditional approach


```
import ROOT

fIn = ROOT.TFile.Open("file.root")
tree = fIn.tree

for event in tree:
    if len(event.muons)<1: continue
    if not event.MET>20: continue

    for muon in event.muons:
        if muon.pt > 25 and abs(muon.eta)<2.4 \
            and muon.dz<0.1 and muon_dxy<0.01 \
            and muon.relIso<0.5:

            selmuon_pt = muon_pt
```



RDataFrame approach

```
import ROOT

ROOT.ROOT.EnableImplicitMT()

RDF = ROOT.ROOT.RDataFrame
d = RDF(treeName, inputFile)

d = d.Filter("nMuon>1 && MET>20")\
    .Define("SelMuon_pt"
        , "Muon_pt[Muon_pt>25 \
            && abs(Muon_eta)<2.4 \
            && Muon_dz<0.1 && Muon_dxy<0.01 \
            && Muon_relIso<0.5]")
```

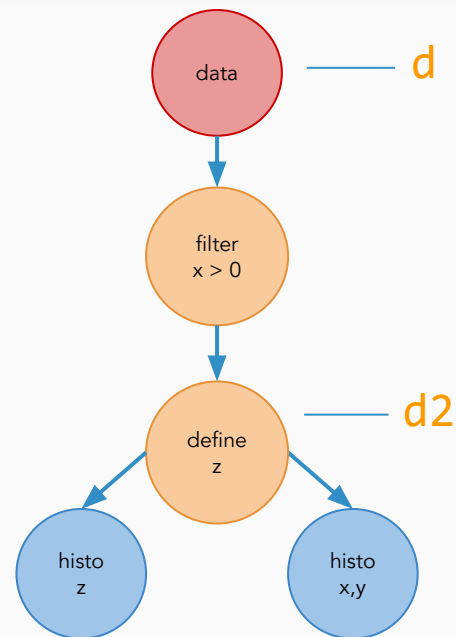

Thinking in terms of Graphs

```
// d2 is a new data-frame, a transformed version of d
```

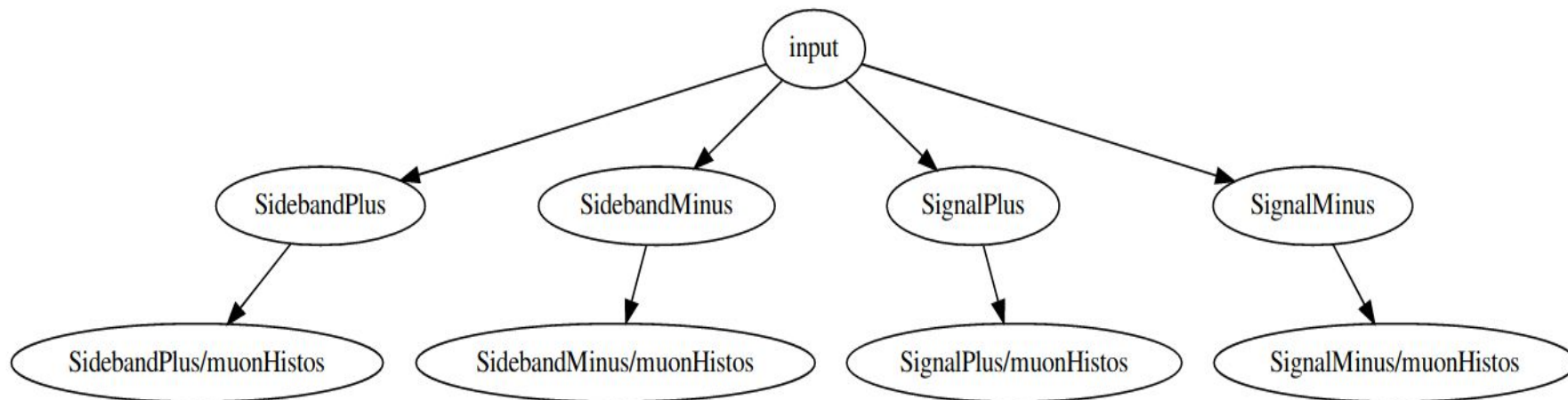
```
auto d2 = d.Filter("x > 0")  
        .Define("z", "x*x + y*y");
```

```
// make multiple histograms out of it
```

```
auto hz = d2.Histo1D("z");  
auto hxy = d2.Histo2D("x","y");
```



- More intuitive way of visualizing an analysis workflow.

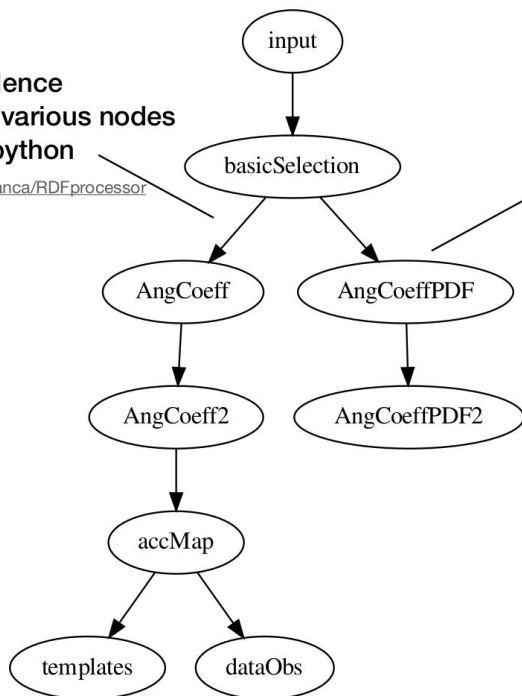


- Simplest example from our analysis
- Input split for different regions(filters)
- Single module(muonHistos) with histogram definitions called for each region.

Thinking in terms of Graphs

the dependence among the various nodes is done in python

<https://github.com/emanca/RDFprocessor>



each node contains a list of modules to be executed

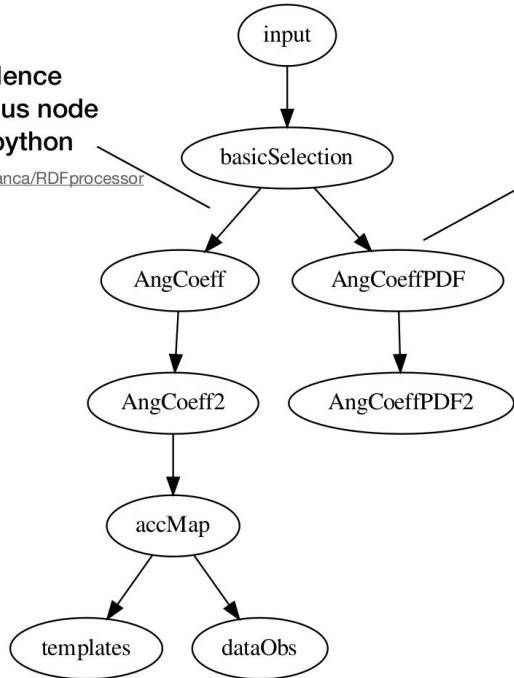
each module is a python or C++ class that transforms a RDataFrame object



Thinking in terms of Graphs

the dependence of the various node is done in python

<https://github.com/emanca/RDFprocessor>



each node contains a list of modules to be executed

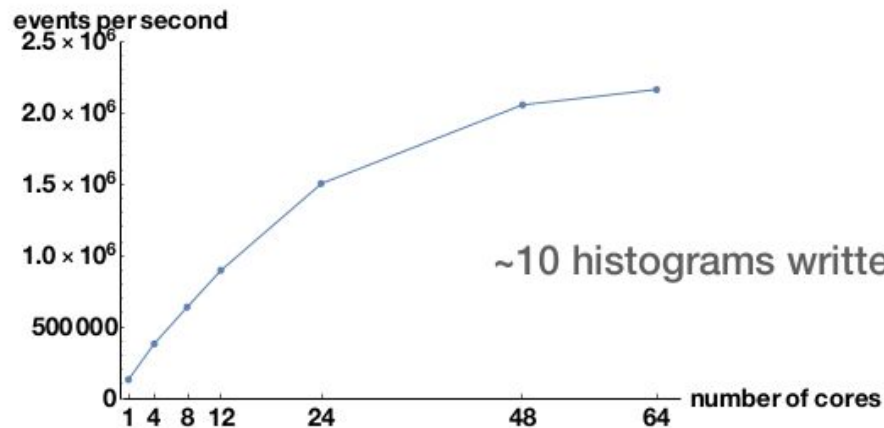
RDataFrame provides optimised execution in a single parallelised event loop

each module is a python or C++ class that transforms a RDataFrame object

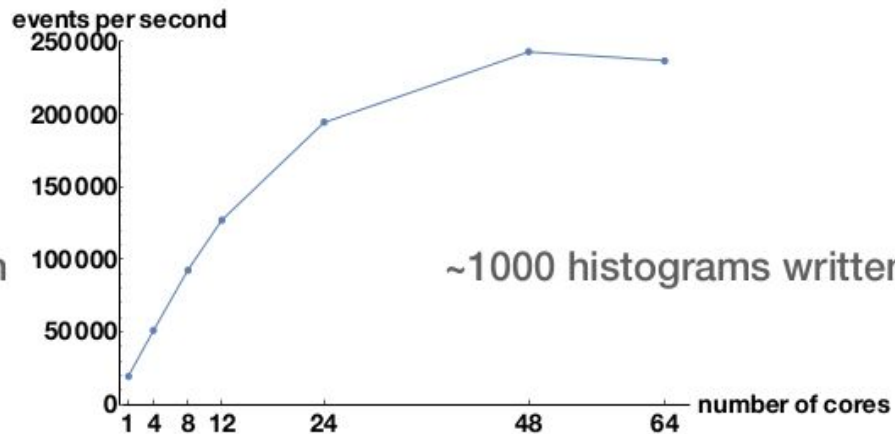


Performance scaling

64 cores AMD machine, SSD, warm cache



~10 histograms written



~1000 histograms written

Roughly same scaling

Outlook

- We have developed an analysis framework based on ROOT DataFrame.
 - Lot of brainstorming from our side in the last year.
- A flexible framework to handle the large complexity of the analysis.
- Encouraging performance results already obtained.
- We must thank the ROOT team who have incorporated our feedback and provided new features when we needed them.
- Many experiments - ATLAS, ALICE, LHCb have started using RDF.
- Some materials in this presentation have been taken from tutorials from ROOT team, talks by E. Manca and E. Guiraud. Many thanks to them.

- Seminar by E. Manca and E. Guiraud.
 - <https://indico.cern.ch/event/849610/>
- Reference manual
 - https://root.cern/doc/master/classROOT_1_1RDataFrame.html
- Tutorials
 - https://root.cern.ch/doc/master/group__tutorial__dataframe.html