Event Loop Analysis for Higgs to 4 Leptons on POET Output

Marcelo Anda

CMS DPOA
marcelo.antonio.anda.chavarria@alumni.cern
Supervisors: Clemens Lange, Edgar Carrera.

Abstract

This project consists of translating the RDF Analysis code to an Event Loop Analysis code for Higgs to 4 Leptons as an example of particle physics analysis that can be performed using POET output. In order to do this, a filter will be developed such that it allows us to isolate the events of interest. Finally, using Event Loop Analysis we will reconstruct the Higgs mass and obtain the respective histograms.

1 Introduction

We want to develop a code to analyze the events of some datasets. We must be familiar with the CMSSW environment, which is used to run the Physics Object Extractor Tool (POET). This tool, as its name suggests, extracts information (properties of physics objects: electrons, muons, etc) from a certain event. POET is basically an infrastructure that transforms AOD data into ROOT N-tuples that are easy to handle.

Our motivation is to create simple examples, which can be used pedagogically for external or legacy users of CMS data. In particular, the example developed in this project (Higgs to Four Leptons) shows how to make an analysis in a simple way. The Event Loop Analysis is an alternative to RDF analysis.

2 RDF Analysis

RDataFrame Class offers a high level interface for analyze stored data. It consists of multi-threading and other low-level optimisations. RDF is built with a modular and flexible workflow that allows users to build a data-frame object by specifying a data-set; apply a series of transformations to the data, like a filter (e.g. apply some cuts) or define a new column (e.g. the result of an expensive computation on columns); and apply actions to the transformed data to produce results (e.g. fill a histogram).

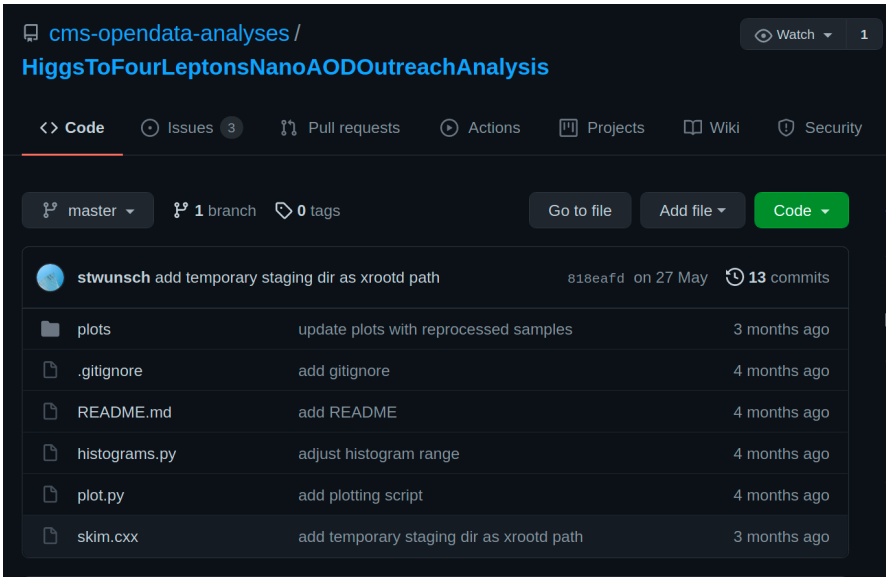

Figure 1: Higgs To Four Leptons Nano AOD Outreach Analysis.

In the repository "Higgs To Four Leptons Nano AOD Outreach Analysis" we will find the file "skim.cxx", which is basically written using RDF Analysis. Inside this file the functions of our interest will be "MinimalSelection()" and "ReconstructHiggs()". The Minimal Selection function filters the events that are of our interest, i.e. those that meet the minimum cuts of certain physical properties such as transverse momentum, pseudorapidity, isolation, etc. On the other hand, the Reconstruct Higgs function takes the filtered events and analyzes

the final state (4 Electrons, 4 Muons and 2 Electrons - 2 Muons) of each one in order to reconstruct the Higgs mass. This is because the Higgs boson decays into 2 Z bosons and these, in turn, decay into 4 leptons in any of the final states mentioned before.

```
312 int main() {
313     ROOT::EnableImplicitMT();
314     const auto poolSize = ROOT::GetThreadPoolSize();
315     std::cout << ">>> Thread pool size for parallel processing: " << poolSize << std::endl;
316 
317     for (const auto &sample : samples) {
318         const auto name = sample.first;
319         const auto finalStates = sample.second;
320         std::cout << ">>> Process sample " << name << ":" << std::endl;
321         ROOT::RDataFrame df("Events", samplesBasePath + name + ".root");
322 
323         for (const auto &finalState : finalStates) {
324             std::cout << ">>> Process final state " << finalState << " for sample " << name << ":" << std::endl;
325             TStopwatch time;
326             time.Start();
327 
328             auto df2 = MinimalSelection(df, finalState);
329             auto df3 = ReconstructHiggs(df2, finalState);
330             auto df4 = DeclareVariables(df3);
331             auto df5 = AddEventWeight(df4, name);
332 
333             auto dfFinal = df5;
334             auto report = dfFinal.Report();
335             dfFinal.Snapshot("Events", name + finalState + "Skim.root", finalVariables);
336             report->Print();
337             time.Stop();
338             time.Print();
339         }
340     }
341 }
```

Figure 2: Main Function.

Once we run the file “skim.cxx” we will obtain the file “skim.root”, which will contain the variables of our interest for each event (i.e. the mass of the 2 Z bosons and the Higgs). Now we run the file “histograms.py” on the root file of the previous step, which will proceed to create histograms of all the variables. Thus we will obtain the file “histograms.root” as output. Finally, we will run the file “plot.py” on “histograms.root” and we will obtain the histograms of the variables in png or pdf format.

We will translate the code of the “MinimalSelection()” function into an event filter, called Simple 4 Lepton filter, which can be executed together with the rest

of the POET files and as an output we will obtain a root file that will contain the events of our interest. We will also combine the code from the "ReconstructHiggs()" function with the code from the file "histograms.py" into a single file called "EventLoopAnalysis_HT4L.cxx", which will run on the output of the filter and will give us a root file with the histograms of our variables. Similarly, to get the histograms in png or pdf format, we will run the file "plot.py" on the root file that contain the histograms.

3 Simple 4 Lepton Filter

The code of this filter basically contains the structure of an EDAnalyzer, the only difference is that the output of this file is a boolean. If the output is true, the event that is currently being processed will be saved into the POET output file, and the event will be discarded if the output is false. This filter counts the number of particles that meet certain minimum cuts for the cases 4 Electrons, 4 Muons and 2 Electrons - 2 Muons, this can be observed in the code of the following figure:

```
//4 Muons in the Event
if(nummuons >= 4){ //Good Number of Muons
    if(goodmuon_pfrelIso04all == nummuons){ //All Muons Must Have a Good Isolation
      if(goodmuonpt == nummuons && goodmuoneta == nummuons){ //Good Muon Kinematics
        if(goodmuon_sip3d == nummuons && goodmuon_dxy == nummuons && goodmuon_dz == nummuons){ //Muon Track close to primary vertex with small uncertainty
          if(nummuons == 4 && muon_postch == 2 && muon_negch == 2){ //At least Two positive and two negative Muons
            return isGood4Lepton = true;
          }
        }
      }
    }
  }

//4 Electrons in the Event
if(numElec >= 4){ //Good Number of Electrons
    if(goodElec_pfrelIso03all == numElec){ //All Electrons Must Have a Good Isolation
      if(goodElecpt == numElec && goodEleceta == numElec){ //Good Electron Kinematics
        if(goodElec_sip3d == numElec && goodElec_dxy == numElec && goodElec_dz == numElec){ //Electron Track close to primary vertex with small uncertainty
          if(numElec == 4 && Elec_postch == 2 && Elec_negch == 2){ //At least Two positive and two negative Electrons
            return isGood4Lepton = true;
          }
        }
      }
    }
  }

//2 Electrons and 2 Muons in the Event
if(numElec >= 2 && nummuons >= 2){ //Good Number of Leptons
  if(goodEleceta == numElec && goodmuoneta == nummuons){ //Eta Cuts
    if(goodElec_pfrelIso03all == numElec && goodmuon_pfrelIso04all == nummuons){ //All Leptons Must Have a Good Isolation
      if(goodElec_sip3d == numElec && goodElec_dxy == numElec && goodElec_dz == numElec){ //Electron Track close to primary vertex with small uncertainty
        if(goodmuon_sip3d == nummuons && goodmuon_dxy == nummuons && goodmuon_dz == nummuons){ //Muon Track close to primary vertex with small uncertainty
          if(Elec_postch == 1 && Elec_negch == 1 && muon_postch == 1 && muon_negch == 1 ){ //Two opposite charged electron and muon pairs
            if(goodElecpair_pt == true && goodmuonpair_pt == true){ //Pt cuts
              if(goodDR == true){ //DeltaR Cuts
                return isGood4Lepton = true;
                }
              }
            }
          }
        }
      }
    }
  }
```

Figure 3: Filter.

To run the filter together with the rest of the POET files, we must declare the process in the configuration file "poet_cfg.py" as shown in the figure below:

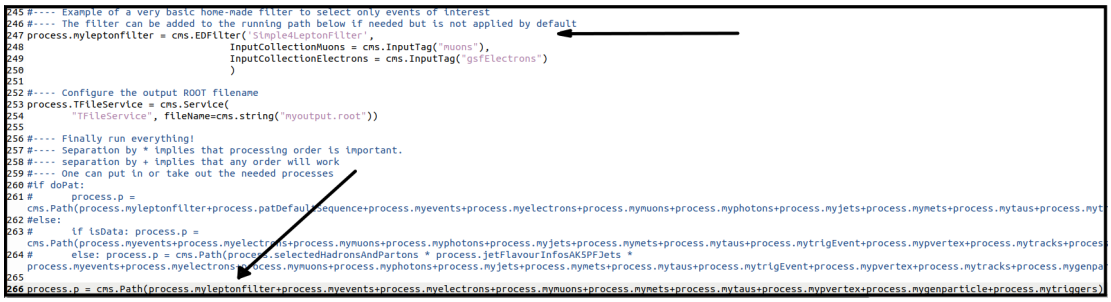

Figure 4: Adding the Filter to the configuration file.

4 HT Condor

Our goal is to run the filter on all events of the dataset. To do this, we must write -1 in the maximum number of events to process in the configuration file, as shown in the figure below:

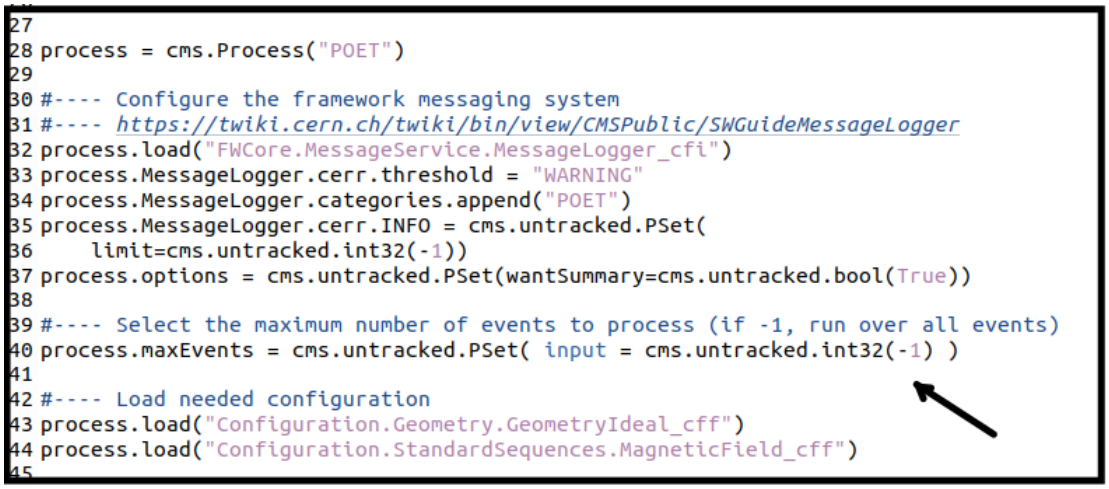

Figure 5: Run over all events.

A desktop computer would take much time to process the Terabytes of information in a dataset, so we must use tools like HT Condor that allows us to process heavy data in a very short time. In the POET repository we will find a folder called "condor", which contains all the files necessary to send the jobs of a dataset. Copy and paste those files into the directory "YourWorkSpace/PhysObjectExtractorTool/PhysObjectExtractor". We will have to make certain changes such as set our working directory (e.g. the directory of an lxplus account) and the output directory (e.g. the directory of an EOS account). Once our directories are correctly configured, using a text editor we open the file "submit_jobs.sh" and select the dataset that we want to process:

```
#!/bin/bash

# Define path for job directories
 BASE_PATH=/afs/cern.ch/user/m/mandacha/workspace/CMSSW_5_3_32/src/PhysObjectExtractorTool/PhysObjectExtractor
#BASE_PATH=/path/to/job/directory
mkdir -p $BASE_PATH

# Set processes
PROCESSES=( \
    #SMHiggsToZZTo4L \
    #ZZTo2e2mu \
    #ZZTo4mu \
    #ZZTo4e \
    #GluGluToHToTauTau \
    #VBF_HToTauTau \
    #TTbar \
    #W1JetsToLNu \
    #W2JetsToLNu \
    #W3JetsToLNu \
    #DYJetsToLL \
    #Run2012B_TauPlusX\
    #Run2012C_TauPlusX\
    #DY2JetsToLL \
    #DY3JetsToLL \
    #DY4JetsToLL \
    #Run2012B_SingleMu\
    #Run2012C_SingleMu\
    Run2012B_DoubleMuParked \
    #Run2012C_DoubleMuParked \
    #Run2012B_DoubleElectron \
    #Run2012C_DoubleElectron \
    )
```

Figure 6: Select a dataset.

To send the jobs we simply write the command "./submit_jobs.sh" in the directory "YourWorkSpace/PhysObjectExtractorTool/PhysObjectExtractor" as shown in the figure below:

```
[mandacha@lxplus796 PhysObjectExtractor]$ ./submit_jobs.sh
Process: Run2012B_DoubleMuParked
Filelist:
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_10000_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_20000_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_20001_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_20002_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_210000_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_30000_file_index.txt.
    CMS_Run2012B_DoubleMuParked_AOD_22Jan2013-v1_310000_file_index.txt.
Number of jobs: 2279
Jobdir: /afs/cern.ch/user/m/mandacha/workspace/CMSSW_5_3_32/src/PhysObjectExtractorTool/PhysObjectExtractor/Run2012B_DoubleMuParked
Submitting job(s).....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
2279 job(s) submitted to cluster 260223.
[mandacha@lxplus796 PhysObjectExtractor]$
```

Figure 7: Jobs submitted with HT Condor.

We repeat this process with all the datasets of our interest and in the end we will obtain the following folders that contain the files with the filtered events:

Figure 8: Folders of each dataset.

5 Event Loop Analysis

The most important function in this code is "ReconstructHiggs(std::string fs)", which basically returns a vector with 3 float type numbers that contains the masses of the 2 Z bosons and the Higgs. This function has as input a string with the final state of the event, which is used to rebuild the system of particles in which the Higgs boson has decayed. This function is called in each loop to save each component of the output vector in a histogram. A complete view of the code can be seen in the image below:

```
//------------------------------------------------------------------

vector<float> EventLoopAnalysisTemplate::ReconstructHiggs(std::string fs){

    bool fil = false;
    ROOT::RVec<ROOT::Math::PtEtaPhiMVector> Z_fourvecs(2);
    vector<float> vecmass(3,0); //Vector of 3 components with 0 value.

    // Reconstruct the ZZ system for all final states

    if (fs.compare("FourMuons") == 0) {
        auto Z_idx = reconstruct_samekind_muon();
        fil = fillter_deltar_muon(Z_idx); //Delta R separation of particles building the Z systems
        if(fil == true){
          Z_fourvecs = Z_fourvectors_samekind_muon(Z_idx);
        }

    } else if (fs.compare("FourElectrons") == 0) {
        auto Z_idx = reconstruct_samekind_electron();
        fil = fillter_deltar_electron(Z_idx); //Delta R separation of particles building the Z systems
        if(fil == true){
          Z_fourvecs = Z_fourvectors_samekind_electron(Z_idx);
        }

    } else if (fs.compare("TwoMuonsTwoElectrons") == 0) {
        // With exactly two muons and two electrons, the reconstruction is trivial (each Z is built from two of the same kind).
        Z_fourvecs = z_fourvectors_2el2mu();

    } else {
        throw std::runtime_error("Unknown final state " + fs);

    }

    // Apply cut on the reconstructed Z masses
    if(Z_fourvecs[0].mass() > 40 && Z_fourvecs[0].mass() < 120){ //Mass of first Z candidate in [40, 120]
      if(Z_fourvecs[1].mass() > 12 && Z_fourvecs[1].mass() < 120){ //Mass of second Z candidate in [12, 120]
        // Combine the fourvectors of the two Z bosons to the fourvector of the reconstructed Higgs boson
        auto Higgs_fourvec = Z_fourvecs[0] + Z_fourvecs[1];
        vecmass[0] = Z_fourvecs[0].mass(); //1st Z Boson mass
        vecmass[1] = Z_fourvecs[1].mass(); //2nd Z Boson mass
        vecmass[2] = Higgs_fourvec.mass(); //Higgs mass
        return vecmass;

        }
    }

    return vecmass;

}
```

Figure 9: Reconstruct Higgs.

To run the file "EventLoopAnalysis_HT4L.cxx" we have used root 6.24, and we have obtained as output the file "histograms.root" that can be used with the file "plot.py" to produce the final histograms.

6 Results

The following images are some examples of the histograms for the Z1 and Z2 bosons, whose final state is given by 2 Electrons and 2 Muons. On the left side we can see the histogram obtained through RDF analysis and on the right side we can see the histogram obtained through Event Loop Analysis.

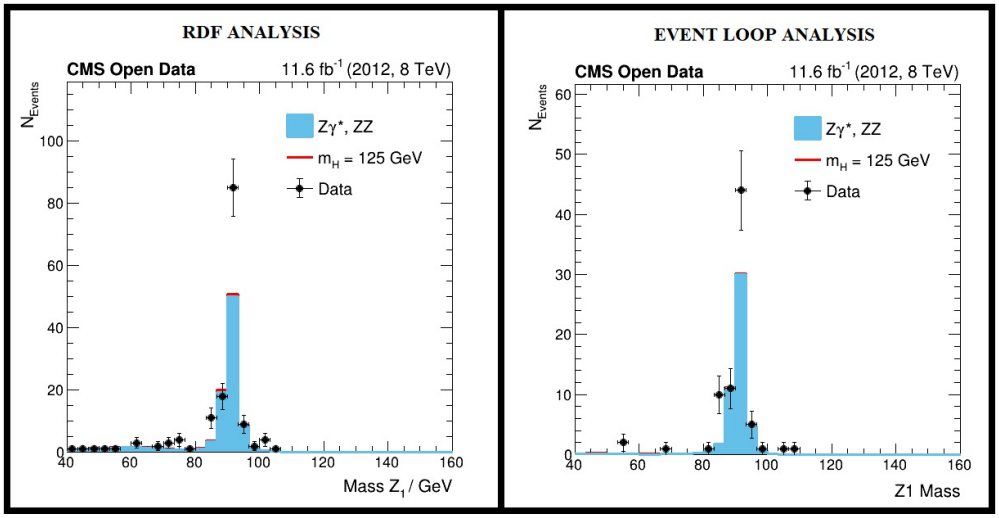

Figure 10: Histogram of the first z boson.

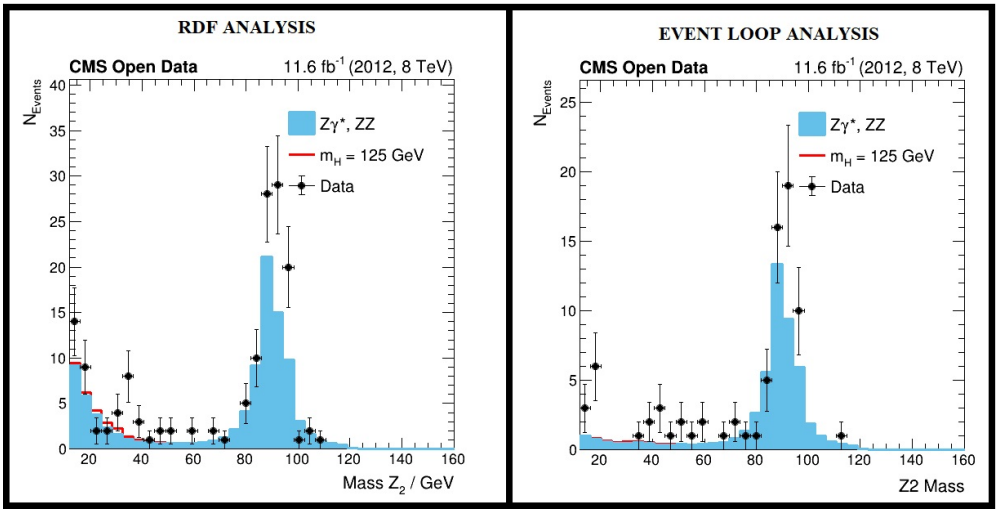

Figure 11: Histogram of the second z boson.

7 Conclusion

Although the histograms obtained with Event Loop Analysis are very similar to those obtained with RDF Analysis, there are some discrepancies between them. For example, normalization is not the same in the two types of analysis and this will be investigated in the future. However, the skeleton code is done and this will be useful for future development.

References

[1] Higgs To Four Leptons Nano AOD Outreach Analysis. (2021). GitHub. `https://github.com/cms-opendata-analyses/HiggsToFourLeptonsNanoAODOutreachAnalysis`

[2] Jomhari, Nur Zulaiha; Geiser, Achim; Bin Anuar, Afiq Aizuddin; (2017). Higgs-to-four-lepton analysis example using 2011-2012 data. CERN Open Data Portal. DOI:10.7483/OPENDATA.CMS.JKB8.RR42

[3] Physics Object Extractor Tool: This repository has working code examples (snippets) on how to access different physics objects in the context of CMSSW software. (2021). GitHub. `https://github.com/cms-legacydata-analyses/PhysObjectExtractorTool/tree/2012`

[4] S. Chatrchyan *et al.* [CMS], Phys. Lett. B **716** (2012), 30-61 doi:10.1016/j.physletb.2012.08.021 [arXiv:1207.7235 [hep-ex]].

[5] Wunsch, S. (2018). ROOT tutorials: df103_NanoAODHiggsAnalysis.C File Reference. ROOT Reference Guide. `https://root.cern/doc/master/df103__NanoAODHiggsAnalysis_8C.html`