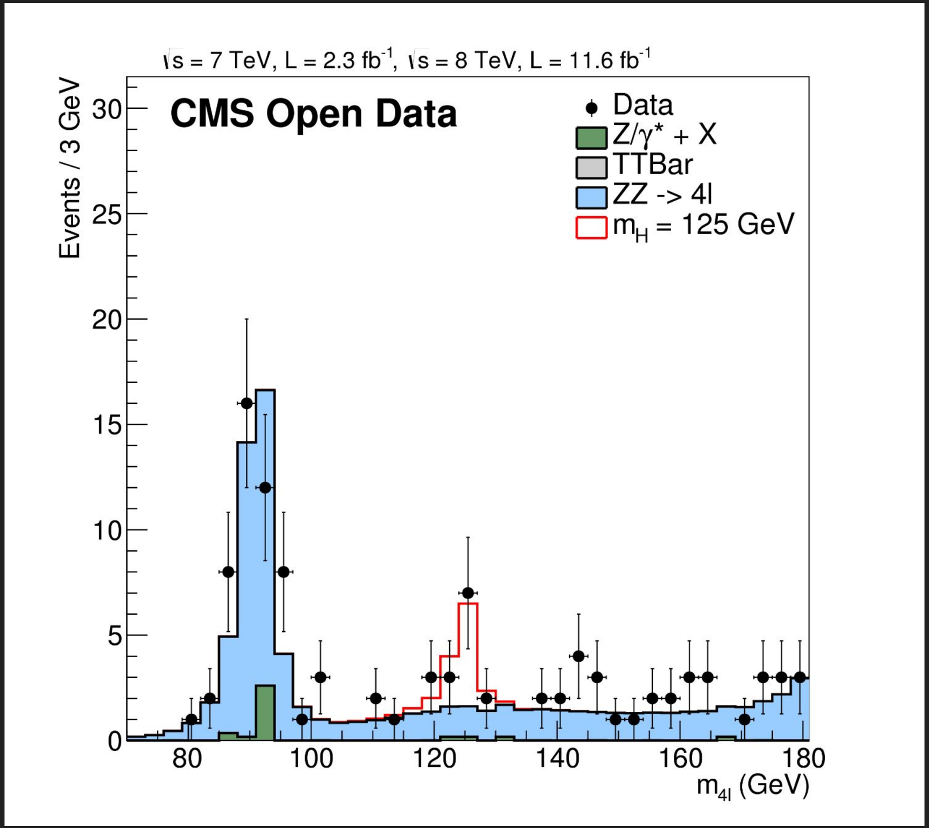
Translating Analyses Into Prototype Analysis Systems

Brian O. Cruz Rodríguez
Jim Pivarski

Objective

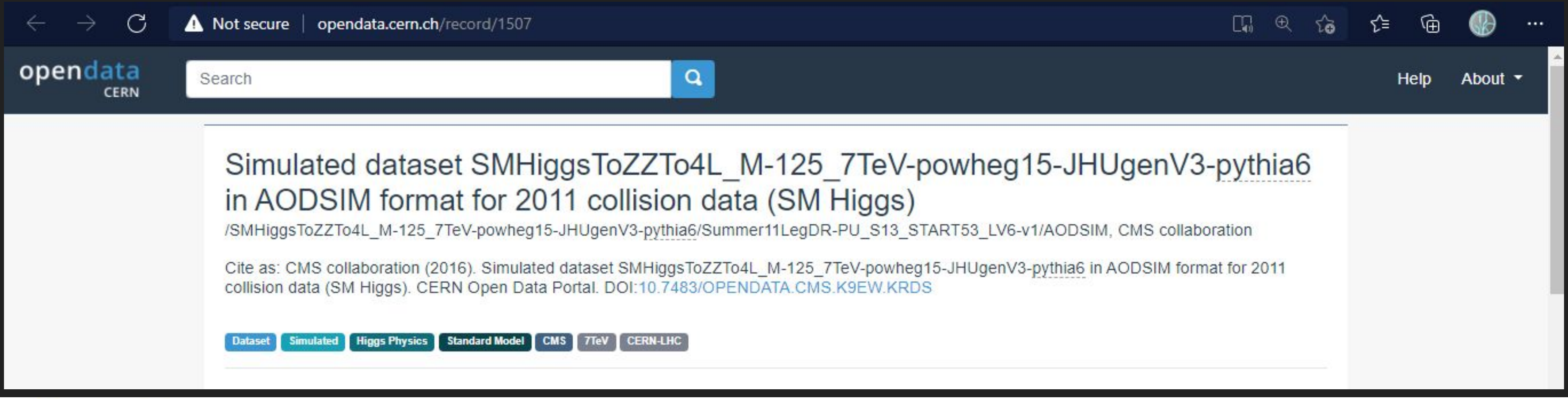
- Translate the Higgs to four leptons, CMS Open Data analysis example into a prototype analysis system that uses Coffea and Awkward-array
    - The four-leptons decay include: 4muons, 4electrons, and 2muons2electrons
- Compare the prototype and original's time-to-insight, functionality and reusability

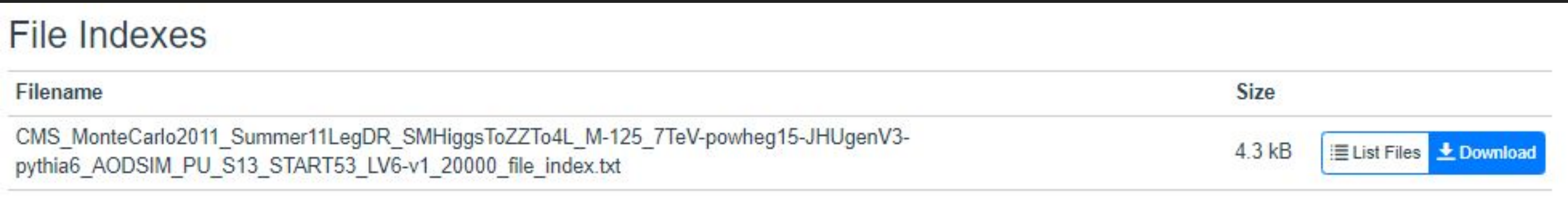
- To show that the translation is working, I’ll show results from getting the Higgs mass histogram (shown in red)



Downloading the Dataset

- Download the simulated Higgs to 4 leptons AODSIM sample from the [CERN Open Data Portal] into a Docker container





- Creating a Docker container from the CMSSW_5_3_32 image

```
PS C:\Users\bocr9> docker run -it --privileged --name iheppproject --net=host --env="DISPLAY" --volume C:\Users\bocr9\shared-folder\:/home/cmsusr/shared
-folder cmsopendata/cmssw_5_3_32 /bin/bash
```

- Download the [cms-opendata-analyses/HiggsExample20112012]() Github repository in the Docker container and compile the codes with ***scram b***

```
[17:44:42] cmsusr@docker-desktop ~/CMSSW_5_3_32/src $ git clone git://github.com/cms-opendata-analyses/HiggsExample20112012.git
Cloning into 'HiggsExample20112012'...
```

- Downloading the [cms-opendata-analyses/AOD2NanoAODOutreachTool]() Github repository too, and compile the code as well

```
[17:46:46] cmsusr@docker-desktop ~/CMSSW_5_3_32/src $ cd workspace/
eachTool -b v1.2 AOD2NanoAODktop ~/CMSSW_5_3_32/src/workspace $ git clone git://github.com/cms-opendata-analyses/AOD2NanoAODOutre
```

- Download the AODSIM index file

```
[17:57:06] cmsusr@docker-desktop ~/CMSSW_5_3_32/src/MCdatasets $ ls
CMS_MonteCarlo2011_Summer11LegDR_SMHiggsToZZTo4L_M-125_7TeV-powheg15-JHUgenV3-pythia6_AODSIM_PU_S13_START53_LV6-v1_20000_file_index.txt
```

- Add the index file as input to the proper Outreach Tool python configuration file, *simulation_cfg.py*

```
# HiggsToZZTo4L_M-125
files = FileUtils.loadListFromFile("/home/cmsusr/CMSSW_5_3_32/src/samples/AODSIM/AODSIM_2011/CMS_MonteCarlo2011_Summer11LegDR_SMHiggsToZZTo4L_M-125_7TeV-powheg15-JHUgenV3-pythia6_AODSIM_PU_S13
_START53_LV6-v1_20000_file_index.txt")
```

- Add the original EDAnalyzer, *HiggsDemoAnalyzerGit*, to the config

```
# Register fileservice for output file
process.aod2nanoaod = cms.EDAnalyzer("AOD2NanoAOD", isData = cms.bool(False))
process.giteda = cms.EDAnalyzer("HiggsDemoAnalyzerGit")
process.TFileService = cms.Service(
    "TFileService", fileName=cms.string("2011MCNtuples.root"))

process.p = cms.Path(process.aod2nanoaod*process.giteda)
```

- Run the config file to produce the NanoAOD

```
[19:23:47] cmsusr@docker-desktop ~/CMSSW_5_3_32/src/workspace/AOD2NanoAOD/configs $ cmsRun simulation_cfg.py
```

  - Move the produced file to the shared folder

```
[05:26:49] cmsusr@docker-desktop ~/CMSSW_5_3_32/src/workspace/AOD2NanoAOD/configs $ mv 2011MCNtuples.root /home/cmsusr/shared-folder/
```

JupyterLab: Translating the analysis

- Compare the original, C++ code (left) to the translated Python code (right)
- First, the Higgs decay to 4 muons
- Then, Higgs decay to 4 electrons
- Finally, Higgs decay to 2 muon and 2 electrons

- Selecting the Outreach Tool's EDAnalyzer Events TTree

```
aod2naod = NanoEventsFactory.from_root(
        "\\Users\\bocr9\\shared-folder\\2011MCNtuples.root",
        "aod2nanoaod/Events",
        schemaclass=NanoAODSchema
).events()
```

- Selecting the Muon branch

```
Muons = aod2naod.Muon
```

- Selecting the good muons from the branch

```
#good muons
good_mu = Muons[ (np.abs(np.sqrt(Muons.dxy**2+Muons.dz**2)/np.sqrt(Muons.dxyErr**2+Muons.dzErr**2))<4) &
                 (np.abs(Muons.dxy)<0.5) &
                 (np.abs(Muons.dz)<1) &
                 (np.abs(Muons.pfRelIso04_all)<0.4) &
                 (Muons.pt>5) &
                 (np.abs(Muons.eta)<2.4) ]
```

```
#entries with 4 muons
mu4 = good_mu[ak.num(good_mu) >= 4]
```

```
# sort the events from highest to lowest transverse momentum
mu4 = mu4[ak.argsort(mu4.pt, axis=-1, ascending=False)]
```

- 4 muon selection and pair combinations for Z bosons

```cpp
//======================== ZZ/ZZ*To4Muon start ================================//

// Now, for these goodmuons, pair up and calculate mass
if (nGoodRecoMuon >= 4)
  {
    const reco::Muon &muon1 = (*muons)[vIdPtmu.at(0).first];
    const reco::Muon &muon2 = (*muons)[vIdPtmu.at(1).first];
    const reco::Muon &muon3 = (*muons)[vIdPtmu.at(2).first];
    const reco::Muon &muon4 = (*muons)[vIdPtmu.at(3).first];

    if (muon1.charge() + muon2.charge() + muon3.charge() + muon4.charge() == 0)
      {
        // First combination: Combine muon 1234
        if (muon1.charge() + muon2.charge() == 0) // each lepton pair cas = 0
          {
            eZ12 = (sqrt(muon1.p() * muon1.p() + sqm1)) +
                   (sqrt(muon2.p() * muon2.p() + sqm1));

            pxZ12 = muon1.px() + muon2.px();
            pyZ12 = muon1.py() + muon2.py();
            pzZ12 = muon1.pz() + muon2.pz();
```

```python
# select the first 4 muons from the sorted mu4
f4_mu4 = mu4[:,0:4]

# select the 4 muons with a zero net charge
f4c0_mu4 = f4_mu4[ak.sum(f4_mu4[:,:,"charge"], axis=-1) == 0]


# creating muon pairs
muon_pair = ak.combinations(f4c0_mu4, 2)


# combinations with zero net charge
muon_pair_c0 = muon_pair[(muon_pair["0"][:,"charge"]+muon_pair["1"][:,"charge"])==0]
```

- Determining the closest and farthest Z boson

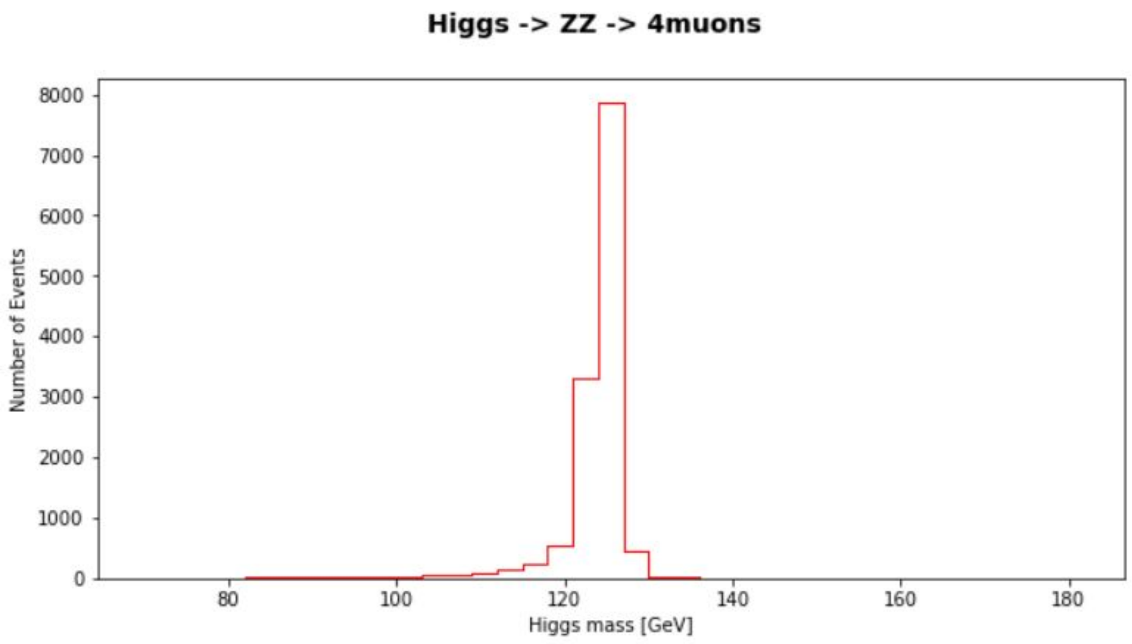
```python
# function that finds the best combination of muon pairs with a mass closest to the real Z mass
def closest(pair):
    delta = abs(91.1876 - pair.mass[:])
    closest_masses = np.min(delta, axis=-1)
    the_closest = (delta == closest_masses)
    return the_closest
```

```cpp
if (ptZadaug) {
 if (mZa > 40. && mZa < 120.) {
  if (mZb > 12. && mZb < 120.) {

      h_mZa_4mu->Fill(mZa);
      h_mZb_4mu->Fill(mZb);

      // 4 vector
      p4Za.SetPxPyPzE(pxZa, pyZa, pzZa, eZa);
      p4Zb.SetPxPyPzE(pxZb, pyZb, pzZb, eZb);

      p4H = p4Za + p4Zb;

      mass4mu = p4H.M();
```

```python
# get the closest Z boson and farthest Z boson
Z = (muon_pair_c0["0"] + muon_pair_c0["1"])
Za = closest(Z)
Zb = Za[:,::-1]

# best combination of muons
Za_mu = muon_pair_c0[Za]
Zb_mu = muon_pair_c0[Zb]

# flatten the arrays
Za_mu = ak.flatten(Za_mu)
Zb_mu = ak.flatten(Zb_mu)

# Mass requirement
Za_M = (((Za_mu["0"]+Za_mu["1"]).mass>40)&((Za_mu["0"]+Za_mu["1"]).mass<120))
Zb_M = (((Zb_mu["0"]+Zb_mu["1"]).mass>12)&((Zb_mu["0"]+Zb_mu["1"]).mass<120))

# Transverse momentum requirement
Z_pt = (Za_mu["0"].pt>20)&(Za_mu["1"].pt>10)

# applying the cuts to get the good Z bosons
good_Za = Za_mu[Z_pt&Za_M&Zb_M]
good_Zb = Zb_mu[Z_pt&Za_M&Zb_M]

# adding the Z boson pairs (Higgs decay to 4 muons)
higgs = good_Za['0']+good_Za['1'] + good_Zb['0']+good_Zb['1']

# applying a mass range
higgs = higgs[higgs.mass>70]
```

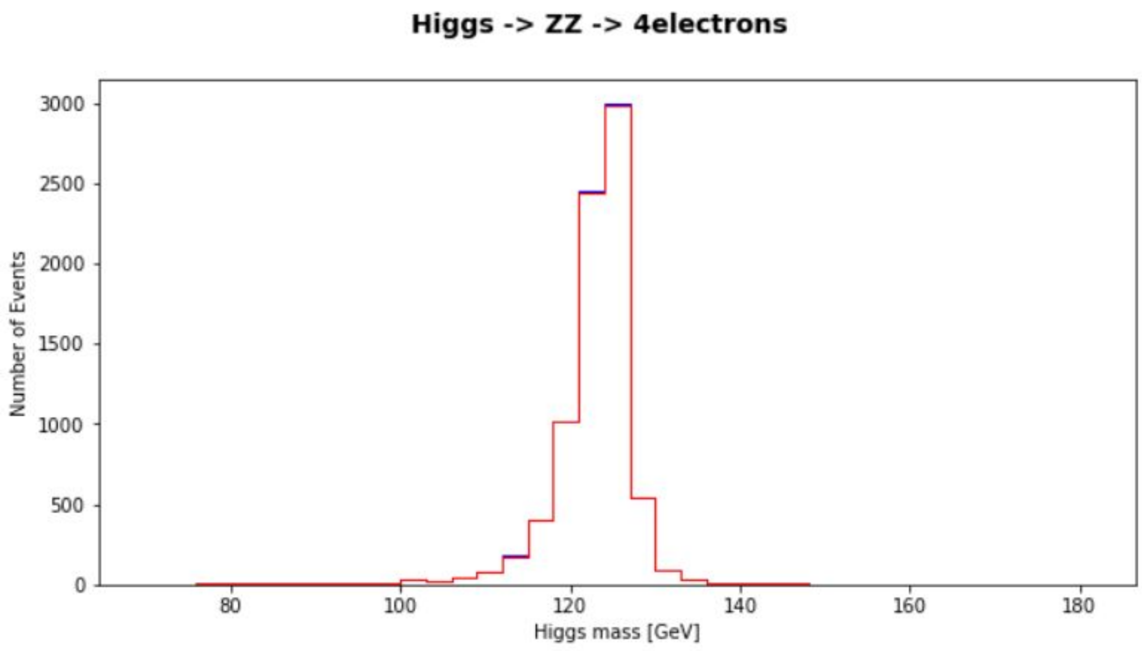
- Plotting the Higgs decay to 4 muon mass histogram

```python
import matplotlib.pyplot as plt

bins = np.arange(70, 184, 3)

fig, ax = plt.subplots(1,1, figsize=(10,5))

# outreach tool
ax.hist(higgs.mass, histtype="step", bins=bins, color = "blue")
# original
ax.hist(giteda_Higgs, histtype="step", bins=bins, color = "red")

fig.suptitle(r'Higgs -> ZZ -> 4muons', fontsize=14, fontweight='bold')

ax.set_xlabel('Higgs mass [GeV]')

ax.set_ylabel('Number of Events')
```



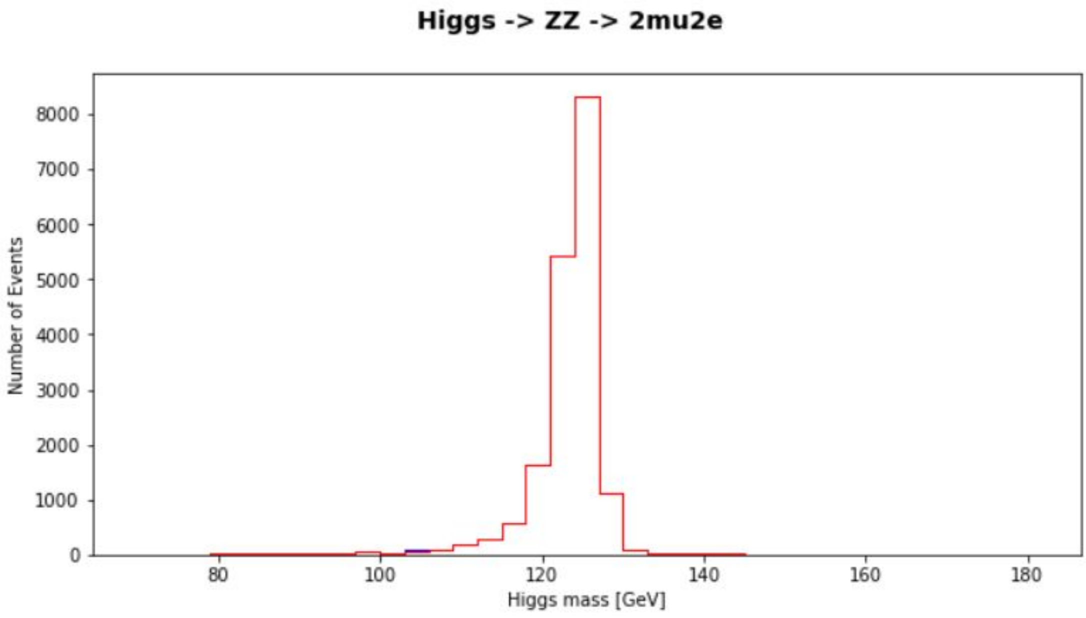
- Plotting the Higgs decay to 4 electron mass histogram

```
import matplotlib.pyplot as plt

bins = np.arange(70, 184, 3)

fig, ax = plt.subplots(1,1, figsize=(10,5))

# outreach tool
ax.hist(higgs.mass, histtype="step", bins=bins, color = "blue")
# original
ax.hist(giteda_Higgs, histtype="step", bins=bins, color = "red")

fig.suptitle(r'Higgs -> ZZ -> 4electrons', fontsize=14, fontweight='bold')

ax.set_xlabel('Higgs mass [GeV]')

ax.set_ylabel('Number of Events')
```



- Plotting the Higgs decay to 2 muons and 2 electrons mass histogram

```
import matplotlib.pyplot as plt

bins = np.arange(70, 184, 3)

fig, ax = plt.subplots(1,1, figsize=(10,5))

# outreach tool
ax.hist(higgs, histtype="step", bins=bins, color = "blue")
# original
ax.hist(giteda_Higgs, histtype="step", bins=bins, color = "red")

fig.suptitle(r'Higgs -> ZZ -> 2mu2e', fontsize=14, fontweight='bold')

ax.set_xlabel('Higgs mass [GeV]')

ax.set_ylabel('Number of Events')
```



Scale-up

- Make the python code scriptable
- Use a Kubernetes cluster in Google Cloud to scale up NanoAOD production.
  - Produce the NanoAODs of all the 21 Higgs analysis samples (in the Open Data Portal) with all their indexes and root files
  - Produce the Higgs plot shown earlier