AthenaROOTAccess Analysis on D3PD

Marcello Barisonzi

DESY

Background

- In 2005, ATLAS Physics Workshop in Rome took place
- Talks had to present results of AOD analyses
- Users 'revolted' against slow Athena-based analyses, results based on ROOT flat ntuples were presented instead
- After the fiasco, ATLAS management pushed to make POOL files accessible from ROOT
- Finally, with release 13, AthenaROOTAccess available

Plan for today

- We will learn how to use AthenaROOTAccess to write an analysis for D2PD data
- Ideally, you should analyze D2PD files you produced this morning
- If you don't have files ready, you can test files in this directory:
  /scratch/current/atlas/FDR-1

ROOT vs. PyROOT

- AthenaROOTAccess can be used from ROOT/Cint or from an interactive Athena session with PyROOT
- In this tutorial, I will give examples in PyROOT, however feel free to use ROOT for your exercises if you wish, conversion is straightforward
- Reference page for help:
  https://twiki.cern.ch/twiki/bin/view/Atlas/AthenaROOTAccess

Setting up the environment

- After you have set up the Athena environment for 13.0.40 and got a kerberos ticket for CERN, check out the following packages:

```
cmt co -r AthenaROOTAccess-00-00-38-09
PhysicsAnalysis/AthenaROOTAccess
cmt co -r AthenaROOTAccessExamples-00-00-11
PhysicsAnalysis/AthenaROOTAccessExamples
```

- AthenaROOTAccess is the core package we will work with, while AthenaROOTAccessExamples contains examples you can look at
- Now please compile these packages

ARA File access

- Create a run directory in AthenaROOTAccess and use the following command: get_files -jo test.py
- Edit test.py, changing this line to the location of your D2PD file: aodFile = 'AOD.pool.root'
- Now start athena in interactive mode by typing: athena -i test.py
- Your file is read, and athena waits for your inputs
- A ROOT tree is created, with the variable name 'tt'

ROOT Tree

- All ROOT functions can be accessed by ROOT.<function> for example try ROOT.TBrowser()
- Open the CollectionTree folder and you will find a list of containers of the format <container class>_p1_<name> e.g. ElectronContainer_p1_ElectronAODCollection
- All the containers can be accessed from the tree using the syntax <tree name>.<container name>
- Try to type tt.ElectronAODCollection ; what is the result?

Self-documented objects

- In Python, objects and classes can be inspected using the `dir()` function
- Try using `dir(tt.ElectronAODCollection)` to get a list of attributes and methods of the class: you will find among others, `'size', 'push_back', 'begin'`. What type of object does this remind you of?
- Assign `ele = tt.ElectronAODCollection[0]` What type of object do you expect `ele` to be?
- Try `dir(ele)` Do you recognize the methods?

Toy Analysis

- Prepare a file called myAnalysis.py like this:

```
include('test.py')

hist_1 = ROOT.TH1F("hist_1", "electron Pt spectrum", 100, 0., 1e6)

def loop():
   for i in xrange(0, tt.GetEntries()):
      tt.GetEntry(i)
      for ele in tt.ElectronAODCollection:
         hist_1.Fill(ele.pt(), 1.)
```

- Now run athena -i myAnalysis.py
  Then type `loop()` and `hist_1.Draw()` when the loop has finished. This is how a (toy) analysis is made

Open Exercise

- Now you should have enough input to start writing your own analysis:
  - Get container names by using TBrowser on an open file
  - Get container and object methods by using `dir()` on classes and objects
- When in doubt, check the wiki page, or browse the examples in AthenaROOTAccessExamples
- Try things out and don't be afraid to ask
- Use ROOT instead of PyROOT if you wish