AOD ROOt Access

my first attempt...

Personal notes and impressions
Comments, advice more than welcome

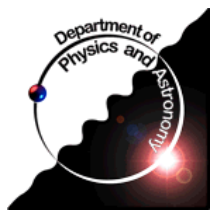
LAPP, 22 June 2007

Michel Lefebvre

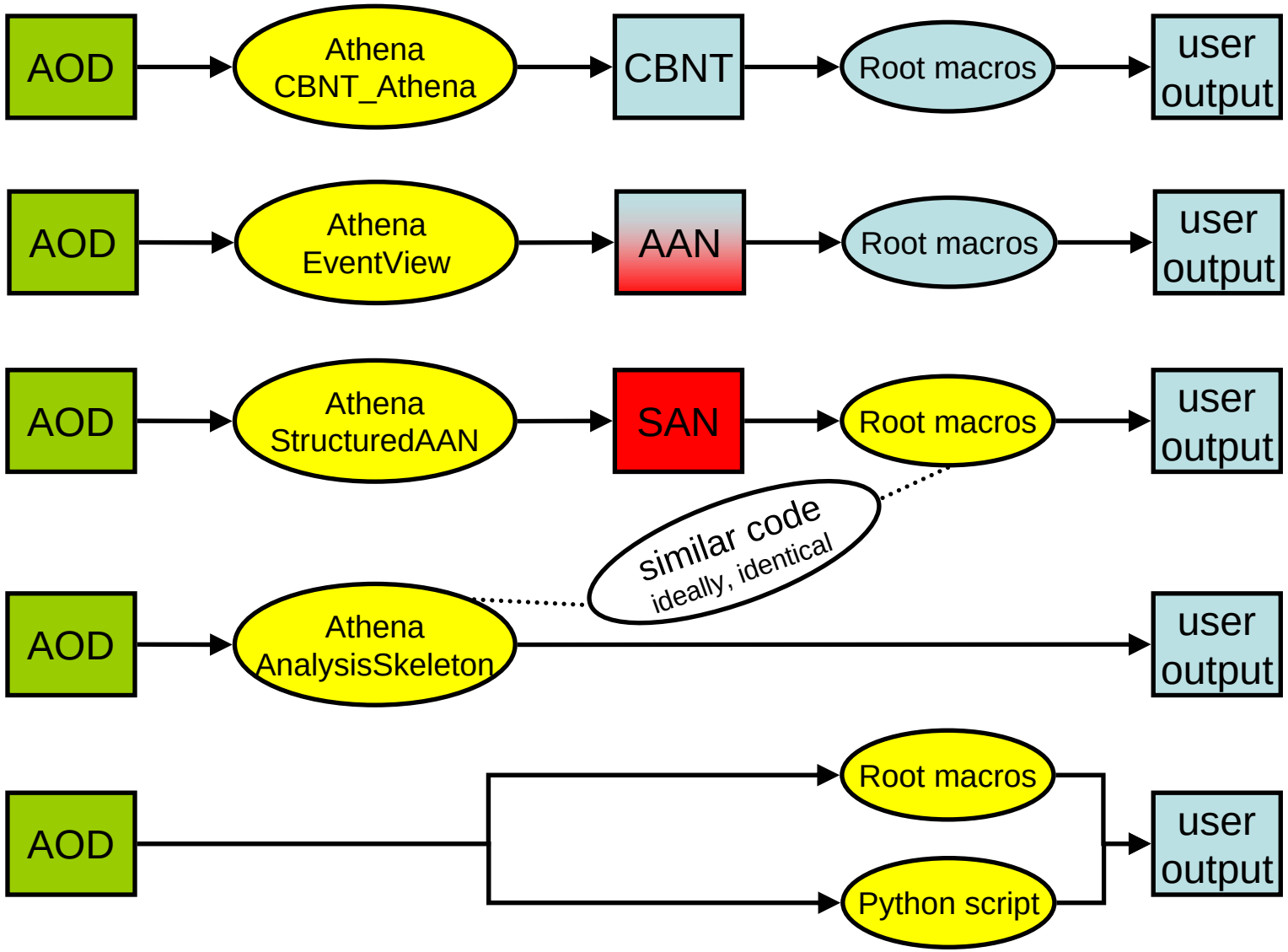
Physics and Astronomy
University of Victoria
British Columbia, Canada

Laboratoire d'Annecy-le-vieux de physique des particules, France

Disclaimer:
I have not tried everything!
I do not understand everything!

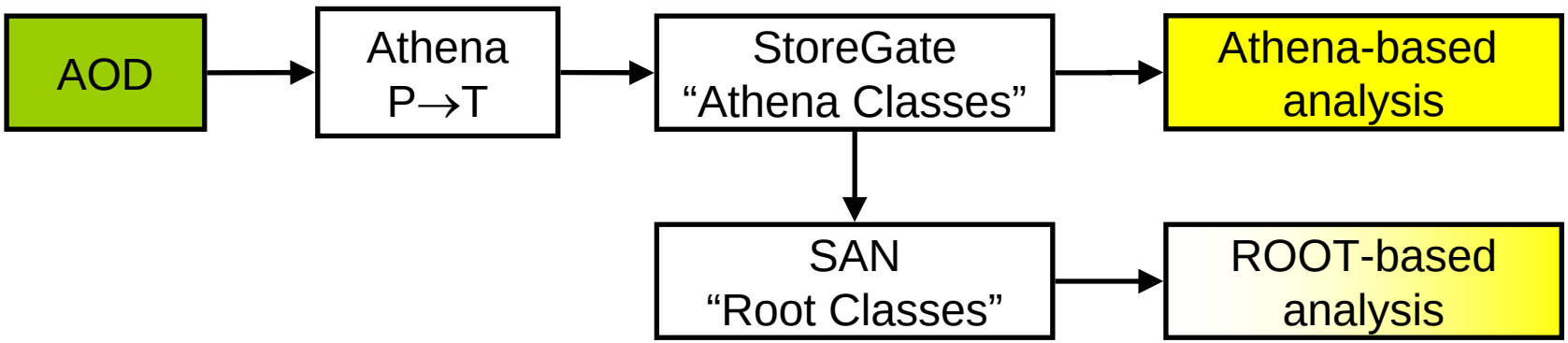


Analysis models



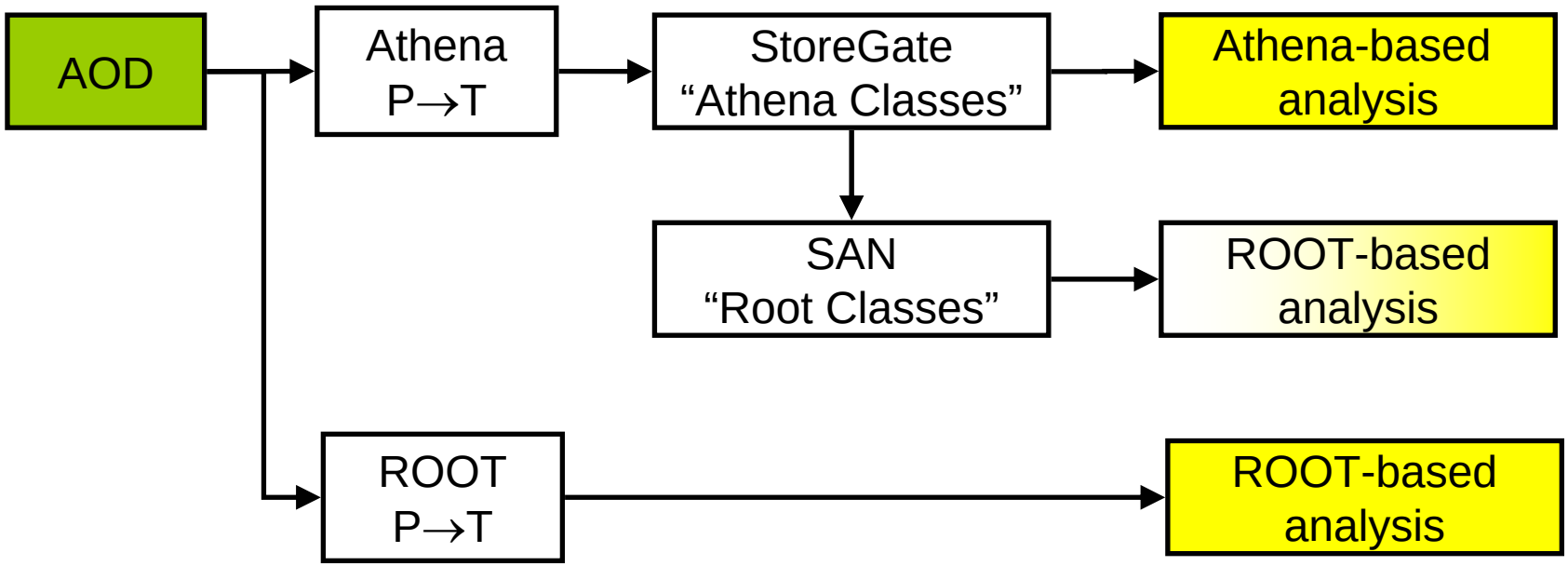
SAN

- add the transient AOD objects to a structured Root tree to provide Root access to the AOD
  - transient AOD objects is what you access in an Athena analysis
  - need to provide “Root version” of these objects
    - User:: namespace classes
    - ....extra maintenance!
  - same functionalities and same interface in Root and in Athena
  - my naive schematic view:



ROOT Access to AOD

- The AOD persistent format has changed in Athena 13
  - Athena 13 AOD is a pAOD!
  - new format allows Persistent→Transient conversion outside Athena
  - ROOT accesses the **same transient objects** as Athena
    - through the use of T/P converters
  - my naive schematic view:



Producing the AOD

- Athena 13.0.10
- “Out of the box”, but...
  - doTrigger=False
  - doTauRec=False
- Produced AOD with 50 events from RDO
  - misal1_valid1.005144.PythiaZee.digit.RDO.v12000605_tid009160._00100.pool.root.1

Reading the AOD in ROOT

- Start from Python script example
  - PhysicsAnalysis/AthenaROOTAccess/share/test.py
  - Required minor modifications
- Just run it
  - python –i test.py

test.py [1]

```
import ROOT
import PyCintex
import AthenaROOTAccess.transientTree

from ROOT import TCanvas,TH1D,TLorentzVector,gROOT,gStyle

aodFile = 'AOD.pool.root'
f = ROOT.TFile (aodFile)
assert f.IsOpen()

# Fill this in if you want to change the names of the transient branches.
branchNames = {}
branchNames['ElectronAODCollection'] = 'ele'
branchNames['PhotonAODCollection'] = 'gam'

tt = AthenaROOTAccess.transientTree.makeTree(f, branchNames = branchNames)
# tt is the transient tree "CollectionTree_trans" containing the (proxies) to
# all available transient data object in the file f
# "CollectionTree_trans" is declared as a friend of the original, persistent
# "CollectionTree". "CollectionTree" will hence provide access to both
# transient data objects and to their persistent counterparts

gROOT.SetStyle("Plain")
gStyle.SetOptStat(111110)

c1 = TCanvas('c1', 'pAOD test', 50, 50, 850, 550)
c2 = TCanvas('c2', 'pAOD test', 150, 150, 950, 650)
h = TH1D('h', 'electron pair mass (MeV)', 20, 0, 200000)
```

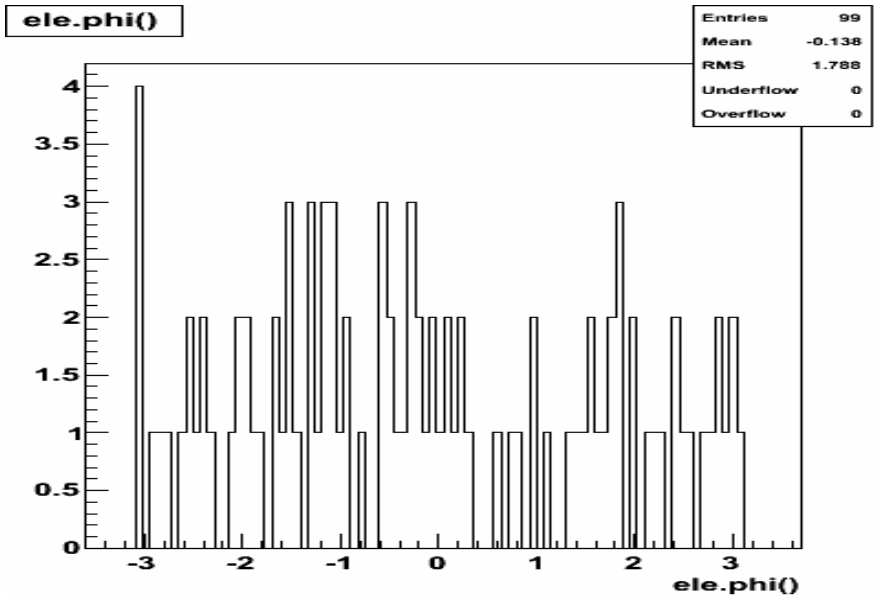
test.py [2]

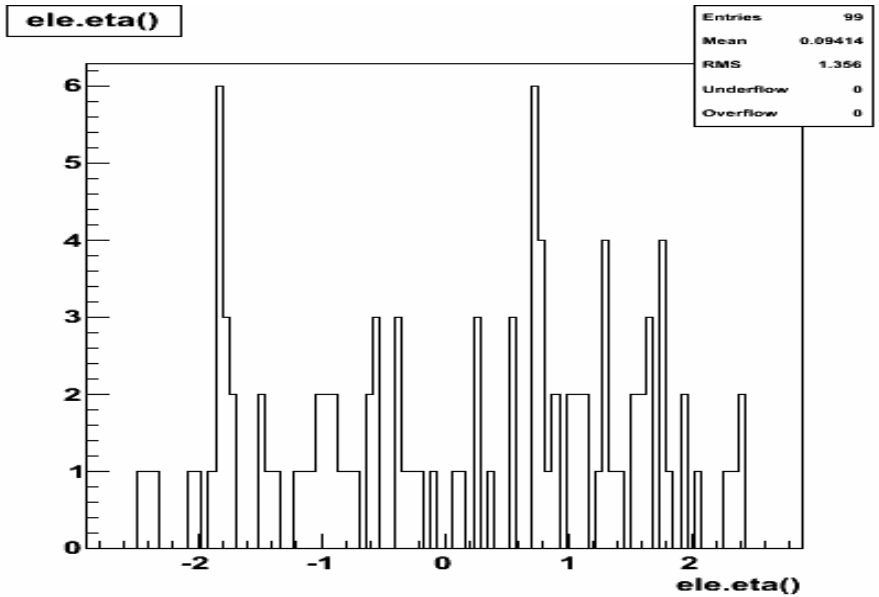
```
def tryThingsOut() :

   # tree contains both the transient and the persistent objects
   tree = AthenaROOTAccess.transientTree.getFullTree(f)

   # scan a few variables
   tree.Scan('ele.e():ele.cluster().e():ele.eta():ele.phi()')
   # NOT YET tree.Show(0)
   # NOI YET tree.StartViewer()

   # prepare canvas for phi and eta distributions
   c1.Divide(2, 1)
   c1.cd(1)
   tree.Draw('ele.phi()') # one entry per electron per event
   c1.cd(2)
   tree.Draw('ele.eta()') # one entry per electron per event

   # get the number of events
   nEvent = tree.GetEntries()
   print 'number of events = ', nEvent
```

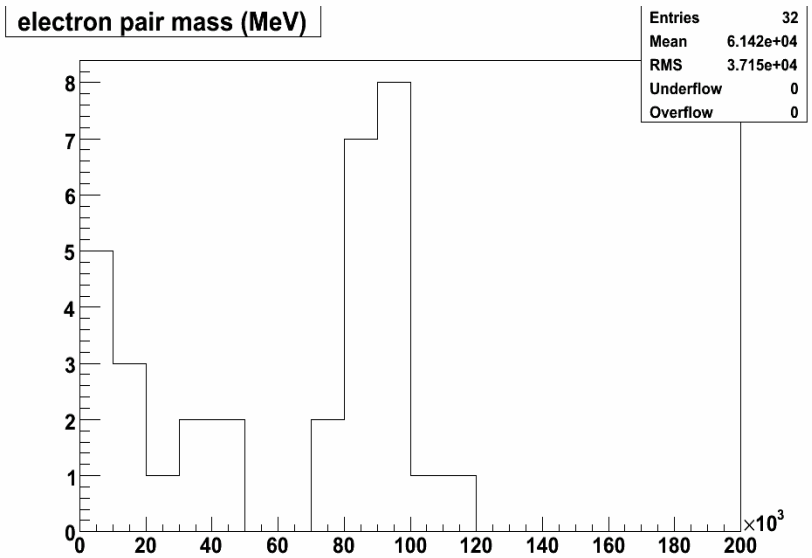
test.py [3]

```
    # look over events
    for iEvent in range(nEvent):
        # loading the whole event
        # tree.GetEntry(iEvent)

        # loading only selected branches
        tree.GetBranch('ele').GetEntry(iEvent)
        # get number of electrons in this event
        n = tree.ele.size()
        if n > 0:
            print iEvent, n, tree.ele.at(0).e() # will crash if n = 0
        else:
            print iEvent, n

        # take first two electrons an compute invariant mass
        if n > 1:
            e0 = TLorentzVector(tree.ele.at(0).px(), tree.ele.at(0).py(), tree.ele.at(0).pz(), tree.ele.at(0).e())
            e1 = TLorentzVector(tree.ele.at(1).px(), tree.ele.at(1).py(), tree.ele.at(1).pz(), tree.ele.at(1).e())
            m = (e0 + e1).M()
            h.Fill(m)

    # plot histo
    c2.cd()
    h.Draw()

#
tryThingsOut()
#
```

test.py : output







Conclusions

- My very first attempt at using AOD ROOT access
  - it works!
  - thanks to RD, Scott for comments
- Many things to explore
  - C++ based analysis
    - with / without CINT
  - produce trimmed, slimmed, skimmed AOD
    - from production AOD
- Hypernews
  - hn-atlas-PATDevelopment@cern.ch