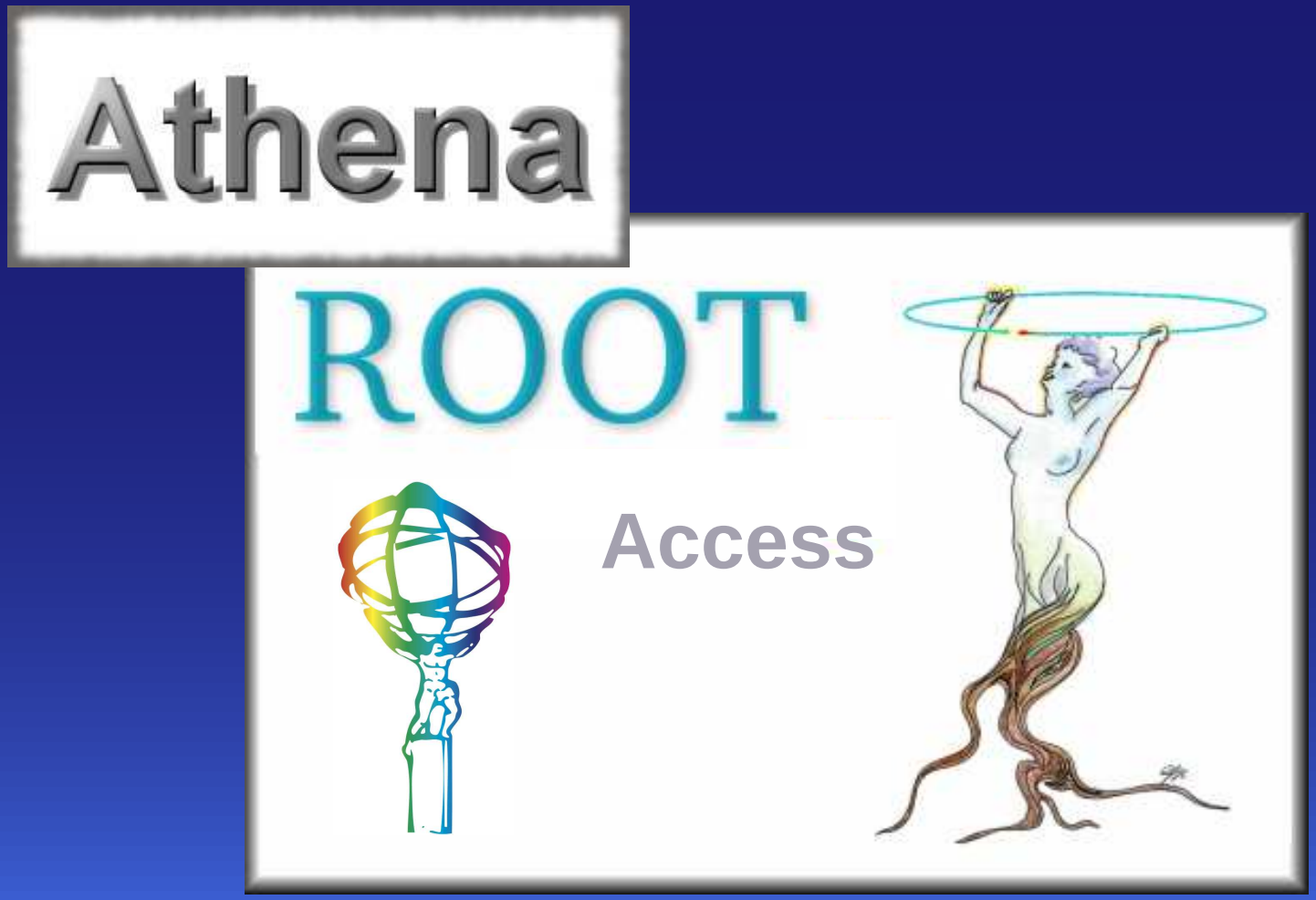
AthenaROOTAccess Tutorial

1st Artemis Annual Meeting

S. Binet, LBL
Jyothsna Komaragiri, Simon Fraser University
Sven Menke, MPI München
Stahes Paganis, University of Sheffield
R.D. Schaffer, LAL

28. September 2007, Chalkidiki, Greece

- Motivation
- Scripts
- Tutorial
- Discussion



Motivation

- Scott's `AthenaROOTAccess` should make the use of AOD's as easy as analyzing a CBNT or SAN
- aim of this tutorial is to get started with the 3 different ways of doing analysis this way:
  - CINT macros
  - C++ compiled code
  - Python scripts
- new package `AthenaROOTAccessExamples` has been created to collect simple (skeleton) examples
- the tutorial should enable and encourage you to write your own
- feedback is very welcome

Scripts ► tags to checkout on top of 13.0.20

- Preparation for the tutorial:
  - working setup of athena-13.0.20 plus some tags from Scott's AthenaROOTAccess wiki page

```
cmt co -r RecExCommon-00-08-08 Reconstruction/RecExample/RecExCommon
cmt co -r AnalysisTriggerEvent-00-02-05 PhysicsAnalysis/AnalysisTrigger/AnalysisTriggerEvent
cmt co -r AnalysisTriggerEventTPCnv-00-00-04 PhysicsAnalysis/AnalysisTrigger/AnalysisTriggerEventTPCnv
cmt co -r AthenaROOTAccess-00-00-29 PhysicsAnalysis/AthenaROOTAccess
cmt co -r AthenaROOTAccessExamples-00-00-04 PhysicsAnalysis/AthenaROOTAccessExamples
cmt co -r EventCommonTPCnv-00-02-03 Event/EventCommonTPCnv
cmt co -r RecTPCnv-00-00-21 Reconstruction/RecTPCnv
cmt co -r TrigConfigSvc-00-00-73-01 Trigger/TrigConfiguration/TrigConfigSvc
cmt co -r TrigDecision-00-01-14 Trigger/TrigEvent/TrigDecision
cmt co -r TrigEventAthenaPool-00-01-08 Trigger/TrigEvent/TrigEventAthenaPool
cmt co -r DetDescrConditions-00-01-17 DetectorDescription/DetDescrCond/DetDescrConditions
cmt co -r NavFourMom-00-01-08 Event/NavFourMom
cmt co -r DataModel-00-12-05-01 Control/DataModel
cmt co -r GeneratorObjects-01-02-37 Generators/GeneratorObjects
cmt co -r TrkTrackSummary-00-18-03 Tracking/TrkEvent/TrkTrackSummary
cmt co -r Particle-02-04-03 Reconstruction/Particle
cmt co -r TrigSteeringEvent-01-00-22 Trigger/TrigEvent/TrigSteeringEvent
cmt co -r TrigConfHLTData-00-00-33 Trigger/TrigConfiguration/TrigConfHLTData
cmt co -r TrigConfL1Data-00-00-36 Trigger/TrigConfiguration/TrigConfL1Data
cmt co -r TrigInDetTruthEvent-00-00-18 Trigger/TrigTruthEvent/TrigInDetTruthEvent
cmt co -r JetTagInfo-00-02-25 PhysicsAnalysis/JetTagging/JetTagInfo
cmt co -r PyParticleTools-00-00-25 PhysicsAnalysis/PyAnalysis/PyParticleTools
cmt co -r PyAnalysisCore-00-00-21 PhysicsAnalysis/PyAnalysis/PyAnalysisCore
cmt co -r PyTriggerTools-00-00-06 PhysicsAnalysis/PyAnalysis/PyTriggerTools
cmt co -r TrkTrack-04-32-05 Tracking/TrkEvent/TrkTrack
cmt co -r TrkTrackLink-00-00-03 Tracking/TrkEvent/TrkTrackLink
cmt co -r TrkParticleBase-00-06-04 Tracking/TrkEvent/TrkParticleBase
cmt co -r TrkGeometry-00-10-03 Tracking/TrkDetDescr/TrkGeometry
cmt co -r TrkSurfaces-00-10-06-01 Tracking/TrkDetDescr/TrkSurfaces
cmt co -r VxMultiVertex-00-01-09-01 Tracking/TrkEvent/VxMultiVertex
```

Scripts ► compiling the tags

- insert the following lines in the requirements file in `Reconstruction/RecExample/RecExCommon/cmt`
- and do a `cmt broadcast gmake` in `Reconstruction/RecExample/RecExCommon/cmt` (takes about 1 hour on average laptop)

```
use AthenaROOTAccessExamples AthenaROOTAccessExamples-* PhysicsAnalysis
use AthenaROOTAccess AthenaROOTAccess-* PhysicsAnalysis
use EventCommonTPCnv EventCommonTPCnv-* Event
use RecTPCnv RecTPCnv-* Reconstruction
use TrigConfigSvc TrigConfigSvc-* Trigger/TrigConfiguration
use TrigDecision TrigDecision-* Trigger/TrigEvent
use TrigEventAthenaPool TrigEventAthenaPool-* Trigger/TrigEvent
use DetDescrConditions DetDescrConditions-* DetectorDescription/DetDescrCond
use NavFourMom NavFourMom-* Event
use DataModel DataModel-* Control
use GeneratorObjects GeneratorObjects-* Generators
use TrkTrackSummary TrkTrackSummary-* Tracking/TrkEvent
use Particle Particle-* Reconstruction
use TrigSteeringEvent TrigSteeringEvent-* Trigger/TrigEvent
use TrigConfHLTData TrigConfHLTData-* Trigger/TrigConfiguration
use TrigConfL1Data TrigConfL1Data-* Trigger/TrigConfiguration
use TrigInDetTruthEvent TrigInDetTruthEvent-* Trigger/TrigTruthEvent
use JetTagInfo JetTagInfo-* PhysicsAnalysis/JetTagging
use PyParticleTools PyParticleTools-* PhysicsAnalysis/PyAnalysis
use PyAnalysisCore PyAnalysisCore-* PhysicsAnalysis/PyAnalysis
use PyTriggerTools PyTriggerTools-* PhysicsAnalysis/PyAnalysis
use TrkTrack TrkTrack-* Tracking/TrkEvent
use TrkTrackLink TrkTrackLink-* Tracking/TrkEvent
use TrkParticleBase TrkParticleBase-* Tracking/TrkEvent
use TrkGeometry TrkGeometry-* Tracking/TrkDetDescr
use TrkSurfaces TrkSurfaces-* Tracking/TrkDetDescr
use VxMultiVertex VxMultiVertex-* Tracking/TrkEvent
```

- once `athena-13.0.20` and the additional tags or `athena-13.0.30` and `AthenaROOTAccessExamples-00-00-04` are setup you are ready to go
  - start with getting one AOD suitable for `13.0.20`
  - `dq2_get -r valid1_misal1_mc12.005200.T1_McAtNlo_Jimmy.recon.AOD.v13002002_tid013611 AOD.013611._00065.pool.root.14`
  - or link to my copy on lxplus scratch space: `ln -s /afs/cern.ch/user/m/menke/scratch0/athena/AtlasOffline-13.0.30/AOD.013611._00065.pool.root.14 .`
  - copy the file `AthenaROOTAccess/share/test.py` to your current directoy and edit the line
    `aodFile = 'AOD.pool.root'`
    to
    `aodFile = 'AOD.013611._00065.pool.root.14'`

- start a root session

```
laptop:~>   root
root [0]  TPython::ExecScript("test.py");
root [1]  CollectionTree_trans = (TTree *)gROOT->FindObjectAny("CollectionTree_trans");
root [2]  ClusterExample ce;
root [3]  ce.plot(CollectionTree_trans);
```

  - this should create two canvases and plot $\log_{10}(E/\mathrm{MeV})$ for positive cluster energies E and $-\log_{10}(-E/\mathrm{MeV})$ for negative E and the η distribution for the `CaloCalTopoCluster` collection

- A CINT macro for the cluster example can be executed like this:

```
laptop:~>   root
root [0]  TPython::ExecScript("test.py");
root [1]  CollectionTree_trans = (TTree *)gROOT->FindObjectAny("CollectionTree_trans");
root [2]  gROOT->LoadMacro("AthenaROOTAccessExamples/macros/cluster_example.C")
root [3]  plot(CollectionTree_trans);
```

  - this should also create two canvases and plot $\log_{10}(E/\mathrm{MeV})$ for positive cluster energies E and $-\log_{10}(-E/\mathrm{MeV})$ for negative E and the η distribution for the `CaloCalTopoCluster` collection

- but be aware of differences to C++ example
  - CINT is slower
  - iterators over `ClusterMoments` don't work in CINT (yet)

- A python script for the cluster example can be executed like this:

```
laptop:~>   python -i test.py
>>> import AthenaROOTAccessExamples.cluster_example
>>> AthenaROOTAccessExamples.cluster_example.plot(tt)
```

  - this should again create two canvases and plot $\log_{10}(E/\mathrm{MeV})$ for positive cluster energies E and $-\log_{10}(-E/\mathrm{MeV})$ for negative E and the η distribution for the `CaloCalTopoCluster` collection

- python is (on a log scale) in between CINT and compiled C++ code for this example
  - will become faster once compiled python works
  - iterators over `ClusterMoments` don't work in python either – this is due to missing dictionaries
  - the ROOT version shipping with 13.0.20 forgets about enums and thus the script has to be modified for this release (see http://www.cern.ch/menke/cluster_example.py)

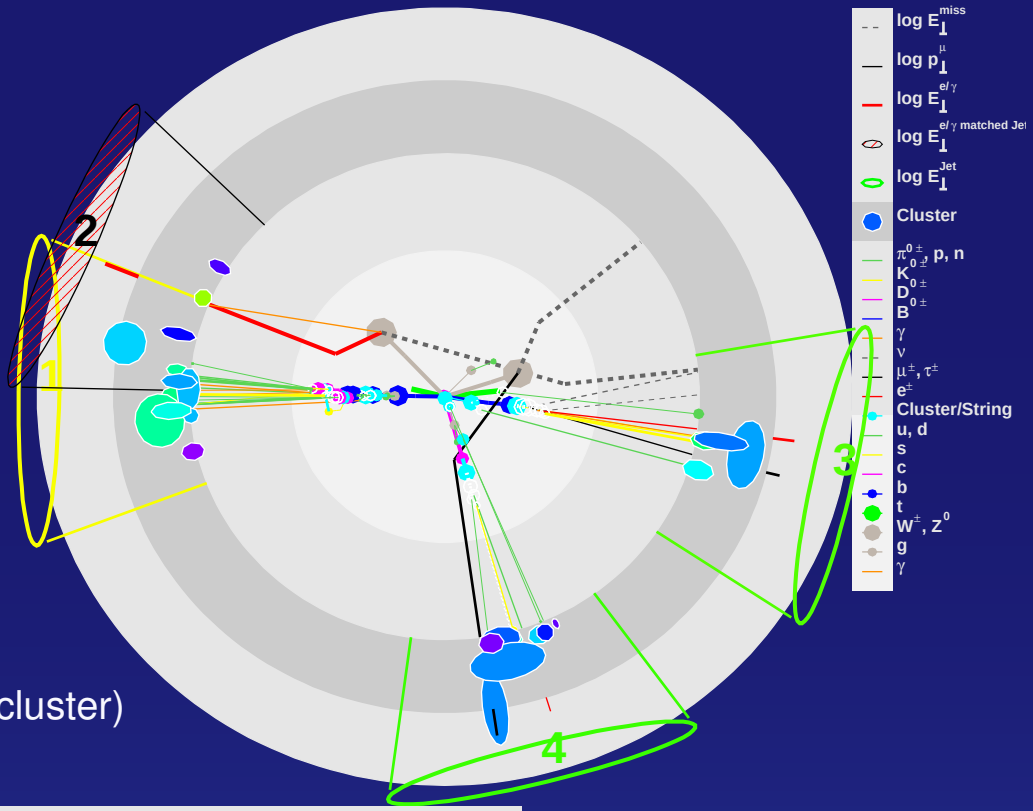
- Other C++ examples are all the classes mentioned in `AthenaROOTAccessExamples/AthenaROOTAccessExamples/selection.xml`:
  - `TruthInfo`
  - `ClusterExample`
  - `TauSelEff`
  - `tauEffTest`
- the can all be instantiated from `root` but differ in the name of the maim method to be called:
  - `void TruthInfo::truth_info(TTree * trans);`
  - `void ClusterExample::plot(TTree * trans);`
  - `void TauSelEff::tau_seleff(TTree * trans);`
  - `void tauEffTest::execute(TTree *trans);`

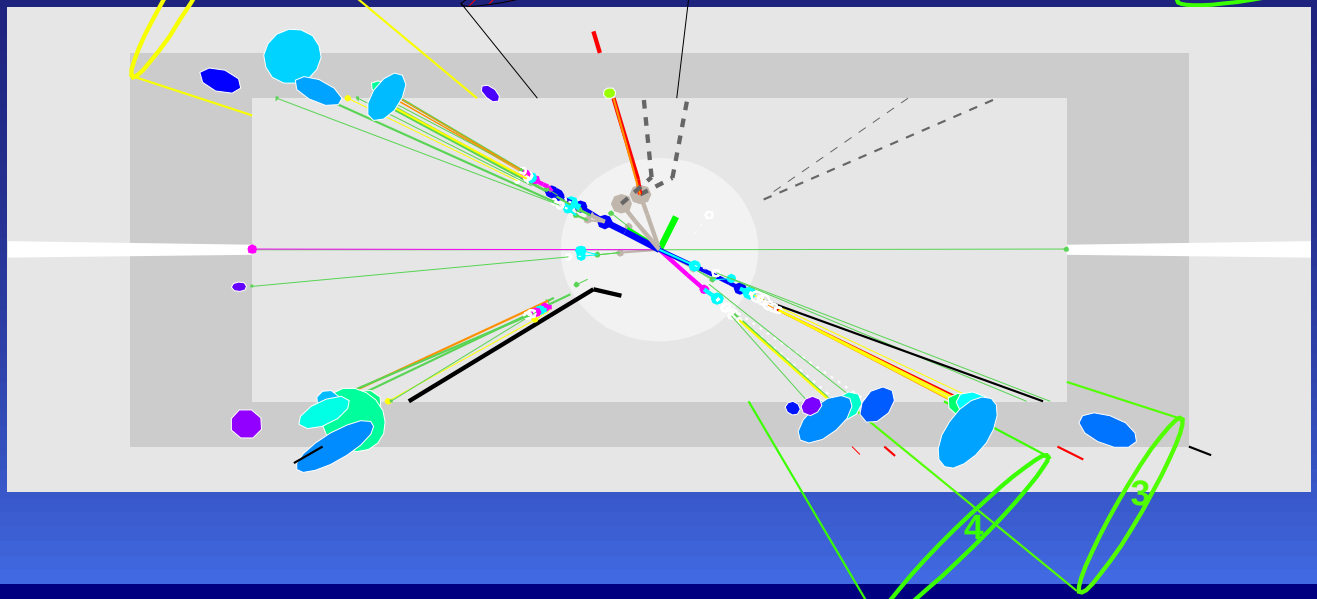
- visualization example for AOD based on the following containers:
  - MC truth on the AOD (`SpclMC`)
  - topo clusters (`CaloCalTopoCluster`)
  - Kt jets from topo clusters (`Kt4H1TopoParticleJets`)
  - Muons (`StacoMuonCollection`)
  - Electrons (`ElectronAODCollection`)
  - MET and tracks to be added in `13.0.30`
- http://www.cern.ch/menke/plotVerticesARA.C creates an $r - \phi$-view and an $r - z$-view of the objects in the above containers
  - for MC truth the displayed lines represent particles connecting production and decay vertex with η, ϕ at each vertex corresponding to the particle decaying there. Color indicates the pdgID, line width the energy and vertex size the mass
  - clusters are drawn as colored ellipses indicating the position, depth and width in the calorimeter (shape) and the energy (color)
  - jets are indicated by fixed size cones (so for Kt the outline is not exactly correct); numbers and color indicate transverse energy order; overlaps with electrons (with more than 50% of the jets energy carried by the matched electron) indicated by black cone with red strips
  - electrons (red) and muons (black) drawn as short lines (length indicates transverse energy) beyond cluster layer

Tutorial ► Pseudo Event Display

```
laptop:~>   root
root [0]  TPython::ExecScript("test.py");
root [1]  CollectionTree_trans = (TTree *)
          gROOT->FindObjectAny("CollectionTree_trans");
root [2]  gROOT->LoadMacro("plotVerticesARA.C")
root [3]  plotVerticesARA(CollectionTree_trans,7);
          // plot Event 7
```

- this event is a $t\bar{t}$ di-lepton example
  - 2 b-jets (1 and 3)
  - one jet (2) coinciding with 1 high energetic electron
  - one jet (4) from the UE
  - μ and e reconstructed (the μ even makes a cluster)
  - one b-jet with 2 softer leptons





Discussion

- Feedback on the Examples
  - are the examples useful?
  - what is missing, needed?
- Feedback on development of own examples
  - did you manage to run your own analysis with `AthenaROOTAccess`
  - if not, what was missing?
- Questions, Answers, Wishes ...