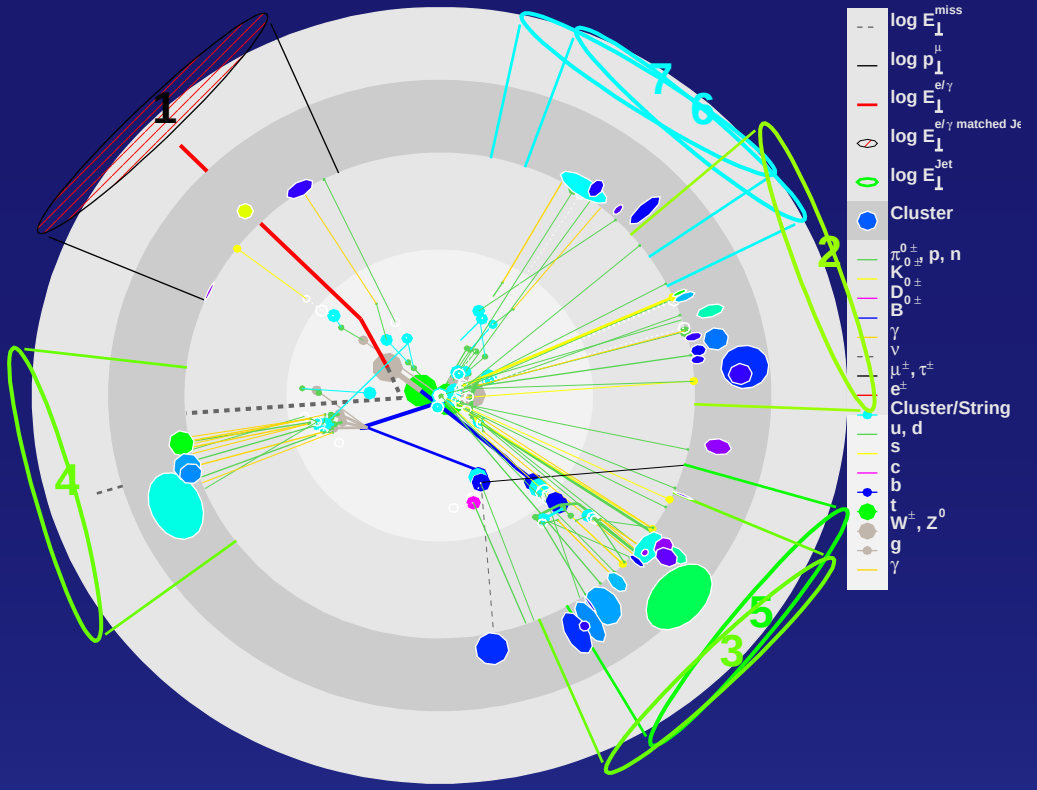
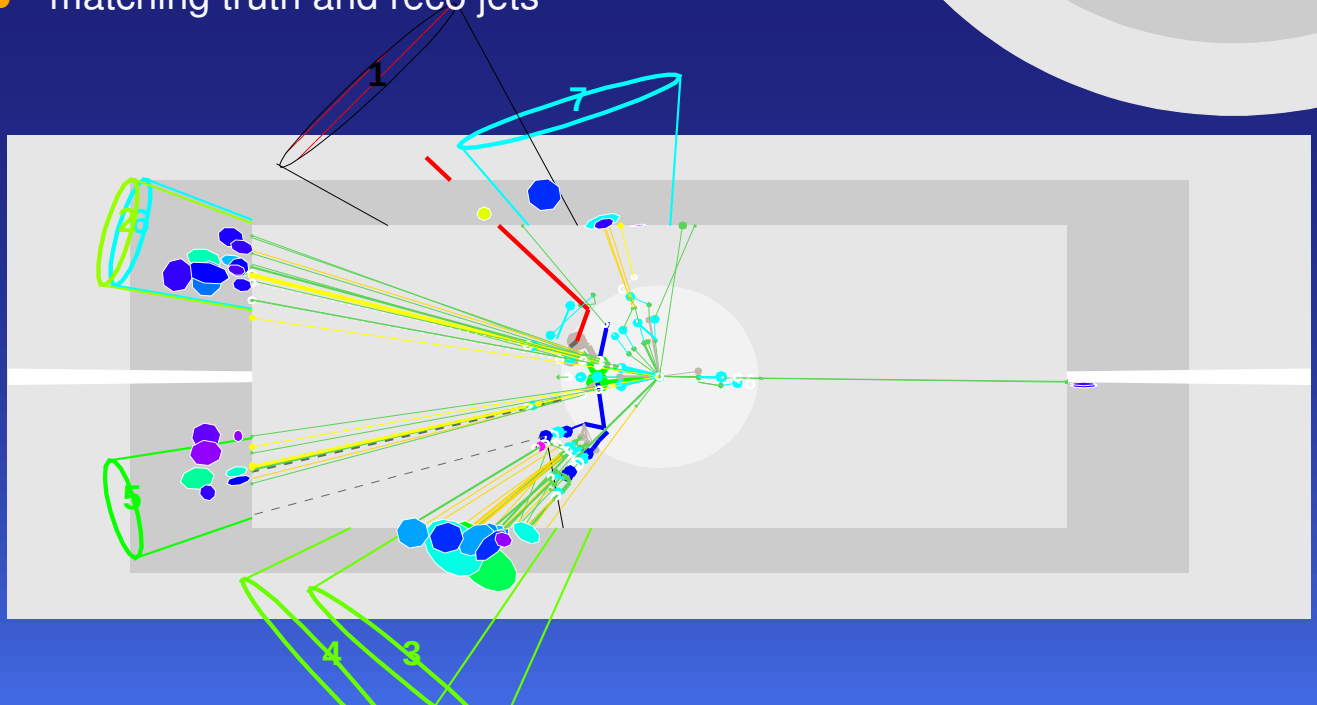
Jets on AOD-level

1st Artemis Annual Meeting Sven Menke, MPI München 28. September 2007, Chalkidiki, Greece

- AOD contents
  - `ParticleJetContainer` in `13.0.X`
- Jet reconstruction on AOD
  - based on calibrated topo clusters
- Jet access examples with `AthenaROOTAccess`
  - matching truth and reco jets





AOD Jet Contents in 13.0.X

- `checkFile.py AOD.pool.root |grep ParticleJetContainer`

```
...  0.278 kb ...  ParticleJetContainer_p1_AtlfastParticleJetContainer
...  3.432 kb ...  ParticleJetContainer_p1_Cone7H1TopoParticleJets
...  3.540 kb ...  ParticleJetContainer_p1_Cone7TruthParticleJets
...  3.550 kb ...  ParticleJetContainer_p1_Cone4H1TopoParticleJets
...  3.596 kb ...  ParticleJetContainer_p1_Kt4TruthParticleJets
...  3.721 kb ...  ParticleJetContainer_p1_Kt4H1TopoParticleJets
...  3.788 kb ...  ParticleJetContainer_p1_Kt6TruthParticleJets
...  3.789 kb ...  ParticleJetContainer_p1_Kt6H1TopoParticleJets
...  4.344 kb ...  ParticleJetContainer_p1_Cone7H1TowerParticleJets
...  5.477 kb ...  ParticleJetContainer_p1_Cone4TruthParticleJets
...  5.673 kb ...  ParticleJetContainer_p1_Kt4H1TowerParticleJets
...  6.327 kb ...  ParticleJetContainer_p1_Kt6H1TowerParticleJets
...  6.594 kb ...  ParticleJetContainer_p1_Cone4H1TowerParticleJets
```

- 3 different choices of input
  - towers
  - truth particles
  - topo clusters
- 4 different choices of jet algorithms
  - Cone with $R = 0.4$
  - Cone with $R = 0.7$
  - kT with $R = 0.4$
  - kT with $R = 0.6$

- both towers and topo clusters are un-calibrated and receive cell-level calibrations during jet making
- the 4 corresponding jets from calibrated topo clusters (local hadron calibration) can be made on the fly from the clusters on the AOD
- since neither towers nor un-calibrated topo clusters are available on the AOD the 8 corresponding jet collections can not be re-made from AOD
- the truth particles are not complete (η and $p_\perp$ cuts) ► the 4 truth jet collections can not be re-made from AOD

AOD Jet Contents in 13.0.X ► ParticleJet contents

- ParticleJets from towers and topo clusters contain the following
  - the usual 4-vector
  - 4 types of jet constituents (constituent) (calorimeter Jet, tracks, soft electrons, soft muons)
  - a set of tag info (b-tag weights like "IP3D")
- calorimeter jets: Analysis::JetConstituent *
  - contains pointer to the original Jet (not on AOD)
  - contains energyInSample and energyInCryostat
- tracks: Analysis::TrackConstituents *
  - contains vector of associated tracks
  - provides access to $p_\perp$, track paramters etc.
- electrons: Analysis::ElectronConstituent *
  - contains vectors of associated (soft) electrons
- muons: Analysis::MuonConstituent *
  - contains vector of associated (soft) muons
- all constituents point to their underlying objetcs via Navigable
- in case the objects are not present only the directly stored info (number of objects, energy per sampling, etc.) is available

Jet Reconstruction on AOD

- jets based on local hadron calibrated topo clusters (`CaloCalTopoCluster`) can be made directly from AOD
  - with `13.0.30` or `13.0.20` + `JetMissingEtUtils-00-02-10`
- example demonstrates how to make new AOD with one additional `ParticleJetContainer` based on the topo clusters
- run with `athena.py aodtoaod.py`

Jet Reconstruction on AOD ► aodtoaod.py

```
get_files -jo aodtoaod.py
```

- insert Kt jet maker and remove crashing containers

```
cat aodtoaod.py
# steering file for AOD->AOD step
# see myTopOptions.py for more info
doCBNT=False
doWriteRDO=False
doWriteESD=False
doWriteAOD=True
doAOD=False # uncomment if do not run AOD making algorithms
doWriteTAG=False # uncomment if do not write TAG
readAOD=True
PoolAODInput=["AOD.pool.root"]
PoolAODOutput="copy_AOD.pool.root"

UserAlgs=[ "Kt4TopoJet_jobOptions.py" ]

# main jobOption
include ("RecExCommon/RecExCommon_topOptions.py")

StreamAOD.ItemList.remove('egammaContainer#ElectronAODCollection')
StreamAOD.ItemList.remove('egammaContainer#PhotonAODCollection')
StreamAOD.ItemList.remove('egDetailContainer#egDetailAOD')
StreamAOD.ItemList.remove('Analysis::TauJetContainer#Tau1P3PContainer')
StreamAOD.ItemList.remove('Analysis::TauJetContainer#TauRecContainer')
StreamAOD.ItemList.remove('JetCollection#HLTAutoKey*')
```

- steering jobO file to create a new AOD

Jet Reconstruction on AOD ► Kt4TopoJet_jobOptions.py

```
get_files -jo Kt4TopoJet_jobOptions.py
```

- change collection names, add JetCaloClusterAdaptorTool, disable the cell calibrator and append ParticleJetBuilder

```
cat Kt4TopoJet_jobOptions.py
#--------------------------------------------------------------
#      jobOptions for JetRec package
#      Kt 0.4 jets from TopoClusters
#--------------------------------------------------------------
#--------------------------------------------------------------
# DLL Libraries
#--------------------------------------------------------------
theApp.Dlls += [ "JetRec", "JetSimTools", "JetRecTools" ]
theApp.Dlls += [ "CaloUtils"]
#--------------------------------------------------------------
# Algorithm steering
#--------------------------------------------------------------
theApp.topAlg += [ "JetAlgorithm/Kt4TopoJets" ]
# -- input container
Kt4TopoJets = Algorithm( "Kt4TopoJets" )

# Kt4TopoJets.InputCollectionNames = [ "CaloTopoCluster" ]
Kt4TopoJets.JetCollectionName = "Kt4CalTopoJets"


if not 'doJetMonitoring' in dir():
        doJetMonitoring = False

# continued next slide
```

- this fragment adds the jetmaker to the main application

Jet Reconstruction on AOD ► Kt4TopoJet_jobOptions.py, cont'd

```
# -- setup with jet monitoring
if doJetMonitoring:
    theApp.Dlls += [ "JetMonitoring" ]
    Kt4TopoJets.AlgTools = [

      "JetCaloClusterAdaptorTool/FetchClusters",

      "JetSignalSelectorTool/InitialEtCut",
      "JetDisplayTool/InputMonitor",
      "JetFastKtFinderTool/KtFinder",
      "JetDisplayTool/JetFinderMonitor",

#       "JetCellCalibratorTool/CellCalibrator",

      "JetDisplayTool/CalibMonitor",
      "JetSignalSelectorTool/FinalEtCut",
      "JetSorterTool/Sorter",
      "JetDisplayTool/OutputMonitor" ]
# -- setup without jet monitoring
else:
    Kt4TopoJets.AlgTools = [

      "JetCaloClusterAdaptorTool/FetchClusters",

      "JetSignalSelectorTool/InitialEtCut",
      "JetFastKtFinderTool/KtFinder",

#       "JetCellCalibratorTool/CellCalibrator",

      "JetSignalSelectorTool/FinalEtCut",
      "JetSorterTool/Sorter" ]
# continued next slide
```

- jet tools like input fetching, actual jet making, cuts and sorting are added to the jet maker

Jet Reconstruction on AOD ► Kt4TopoJet_jobOptions.py, cont'd

```
#
#-------------------------------------------------------------------
# AlgTool steering
#-------------------------------------------------------------------
#

Kt4TopoJets.FetchClusters.InputCollectionKey="CaloCalTopoCluster"

# -- JetKtFinderTool (defaults)
## Kt4TopoJets.KtFinder.BeamType = "PP"
## Kt4TopoJets.KtFinder.DistScheme = "DeltaR"
## Kt4TopoJets.KtFinder.RecomScheme = "E"
Kt4TopoJets.KtFinder.AlgoType = "Standard"
Kt4TopoJets.KtFinder.RParameter = 0.4
#
Kt4TopoJets.InitialEtCut.UseTransverseEnergy = True
# FIXME the following cut may need tuning!
Kt4TopoJets.InitialEtCut.MinimumSignal       = 10*MeV
#
# -- Final signal selection
Kt4TopoJets.FinalEtCut.UseTransverseEnergy = True
# FIXME the following cut may need tuning!
Kt4TopoJets.FinalEtCut.MinimumSignal       = 7.*GeV
#
# -- make sure jets are sorted
Kt4TopoJets.Sorter.SortOrder="ByEtDown"
#
# -- Monitoring
if doJetMonitoring:
    ...

# continued next slide
```

- the jet tools are configured

Jet Reconstruction on AOD ► Kt4TopoJet_jobOptions.py, cont'd

```
if doWriteAOD:
   theApp.Dlls += ["JetMissingEtUtils"]
   theApp.Dlls += ["JetMissingEtAlgs"]
   import EventKernel.ParticleDataType
   from JetMissingEtUtils.JetMissingEtUtilsConf import ParticleJetBuilderTool
   from JetMissingEtAlgs.JetMissingEtAlgsConf import ParticleJetBuilder
   thisBuilderTool = ParticleJetBuilderTool(
          name               = "KtTopoParticleJetBuilderTool",
          dataType           = EventKernel.ParticleDataType.Full,

          CellCalibratorName = "", # ignore related ToolSvc ERROR messages
          DoCellsLoop = False)

   print thisBuilderTool
   ToolSvc += thisBuilderTool
   KtTopoParticleJetBuilder = ParticleJetBuilder(
          name                  = "KtTopoParticleJetBuilder",
          JetCollection         = "Kt4CalTopoJets",
          ParticleJetContainer  = "Kt4CalTopoParticleJets",
          ParticleJetBuilderTool = thisBuilderTool
          )
   print KtTopoParticleJetBuilder
   topSequence += KtTopoParticleJetBuilder
```

- finally the jets are converted to particle jets to be written to AOD
- `athena.py aodtoaod.py` creates a new AOD file (copy_AOD.pool.root) with one additional `ParticleJetContainer` (`Kt4CalTopoParticleJets`)
- since for the new particle jets the underlying jets are present in the athena job above you can also access more detailed `Jet` info during the AOD making
- look also at Rolf Seuster's twiki page https://twiki.cern.ch/twiki/bin/view/Atlas/JetsReRunning

AthenaROOTAccess to Jets on AOD

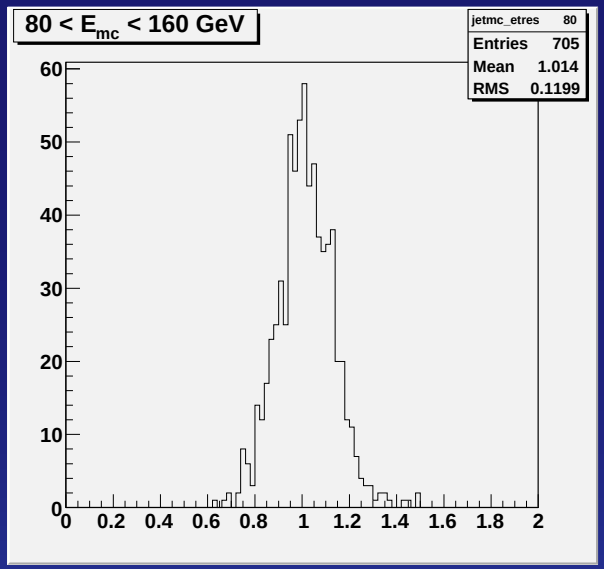
- AthenROOTAccess from Scot Snyder et al. makes working with AOD's as easy as with CBNT or SAN
  - load AOD.pool.root file in root session
  - get a transient TTree * with converted objects from persistent AOD
  - plot or run macros on them from
    - CINT
    - python
    - C++
- AthenROOTAccessExamples collects examples for all three access modes (S. Binet, J. Komaragiri, S.M., R.D. Schaffer)
- look at jet matching example for truth and reco jets (the C++ version from Jyothsna is in CVS – I am showing a slightly modified CINT version since some iterators don't work in CINT)
- look at https://twiki.cern.ch/twiki/bin/view/Atlas/AthenaROOTAccess for more info, examples, tutorials, etc.

- loop over all pairs of reco and truth jets, find matches in ΔR and plot ratio of reconstructed over true $p_\perp$
- JetExample.C CINT macro:

```
void Jet_Example(TTree *trans) {

  double GeV = 1000.;
  const int nbin = 8;
  const double ptbin[9] = {20., 40., 80., 160., 320.,
                           640., 1280., 2560., 1000000.};
  /*
     histo booking etc.
     ...
     end of histo booking etc.
  */

  Long64_t nentries = trans->GetEntriesFast();

  ParticleJetContainer *Rjet = new ParticleJetContainer;
  TBranch *b = trans->GetBranch("Cone4H1TowerParticleJets");
  b->SetAddress(&Rjet);

  ParticleJetContainer *Tjet = new ParticleJetContainer;
  TBranch *br1 = trans->GetBranch("Cone4TruthParticleJets");
  br1->SetAddress(&Tjet);

  for(Long64_t jentry=0; jentry < nentries; jentry++) {
  //event loop

    b->GetEntry(jentry);
    br1->GetEntry(jentry);

    for(int ijet = 0; ijet < Rjet->size(); ijet ++) {
      ParticleJet * rj = Rjet->at(ijet);

      double pt = rj->et();
      double eta = rj->eta();

      double drmc = 0.999;
      double ptmc = 7000*GeV;
      double emc = 7000*GeV;


      for(int jjet = 0; jjet < Tjet->size(); jjet ++) {
        ParticleJet * tj = Tjet->at(jjet);

        double r = rj->hlv().deltaR(tj->hlv());
        if(r < drmc) {
          drmc = r;
          ptmc = tj->et();
          emc = tj->e();
        }
      }//truth jet loop

      // Good match
      if(drmc < 0.1) {//drcut
        for(int i=0; i<nbin; ++i) {
          if(emc > ptbin[i]*GeV && emc < ptbin[i+1]*GeV
             && fabs(eta)<3)
            jetmc_etres[i]->Fill(pt/ptmc);
        }
      }//drcut


    }//reco jet loop

  }//event loop

  /*
     histo drawing etc.
     ...
     end of histo drawing etc.
  */
}
```

- running the example:
- have a working 13.0.20 plus tags from the wiki setup (or just a recent nightly or 13.0.30)
- get the file PhysicsAnalysis/AthenaROOTAccess/share/test.py and edit the name of the AOD file to analyze

```
laptop:~> root
root [0]  TPython::ExecScript("test.py");
root [1]  trans = (TTree *)gROOT->FindObjectAny("CollectionTree_trans");
root [2]  .L JetExample.C
root [3]  Jet_Example(trans);
```



- you can also examine the transient TTree with TBrowser
- or plot directly with TTree::Draw()
- ...

Final Remarks

- Jets on AOD
  - still to many collections to choose from
  - merger of `Jet` and `ParticleJet` for rel 14
  - need full stable particle truth to get rid of truth jets
    - currently under investigation
  - jet constituents provide info about underlying and associated objects for `ParticleJet`s
- re-making Jets from AOD
  - possible without code changes for the first time in `13.0.30`
  - not for tower or un-calibrated topo cluster jets
    - still useful to compare corresponding calibrated topo jet in terms of jet moments, actual cluster constituents etc.
    - need final jet-level corrections for jets from local hadron calibrated topo clusters for wider adoption of those
- `AthenaROOTAccess` to Jets and other containers
  - works and is fun
  - performance with CINT only o.k. for single loops or few events
    - development cycle should be to try on few events with CINT then compiling (almost) same code inside `AthenaROOTAccessExamples`
    - python in the middle (on a log scale)