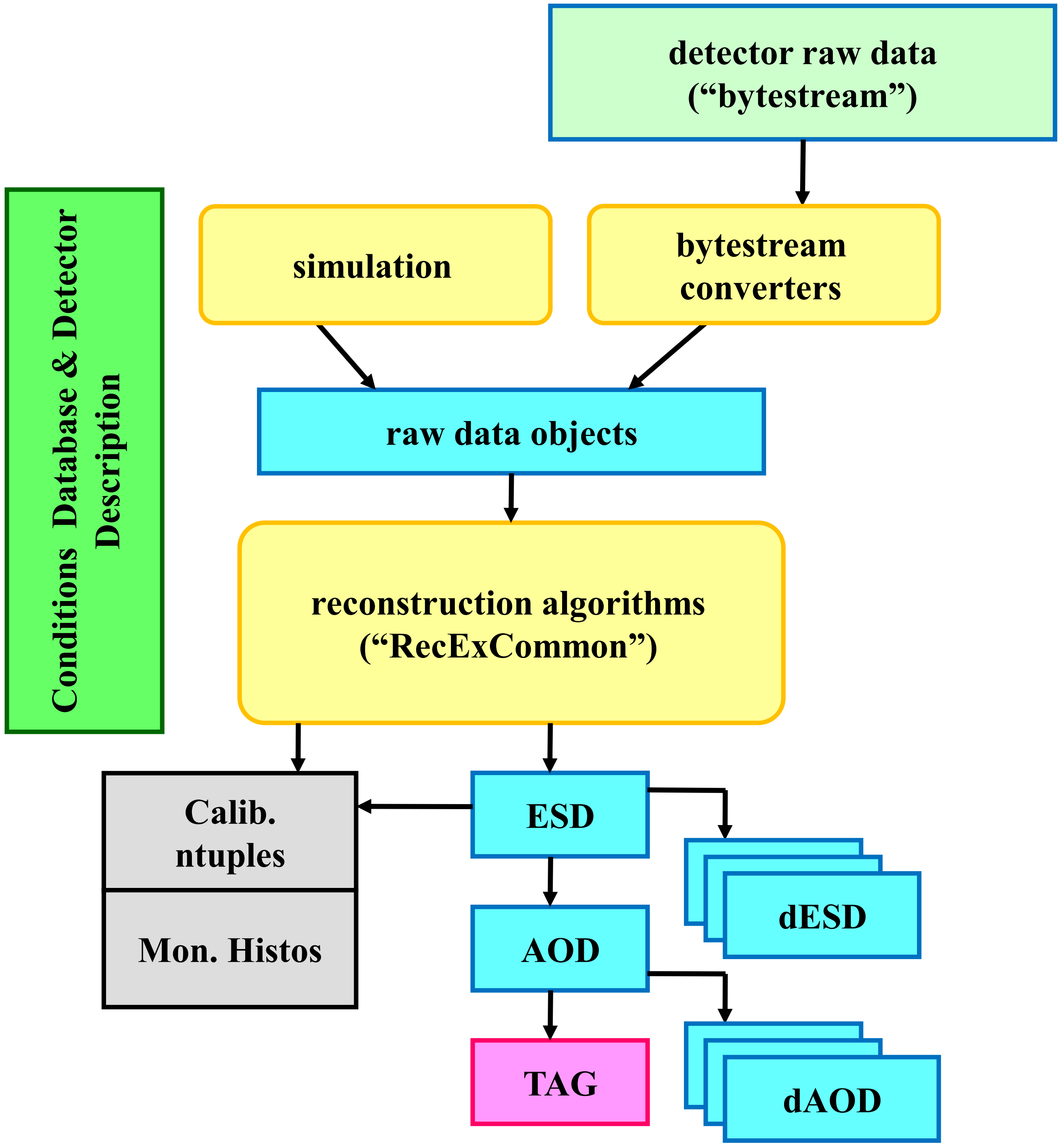
Physics Analysis Examples

James Walder
Lancaster University

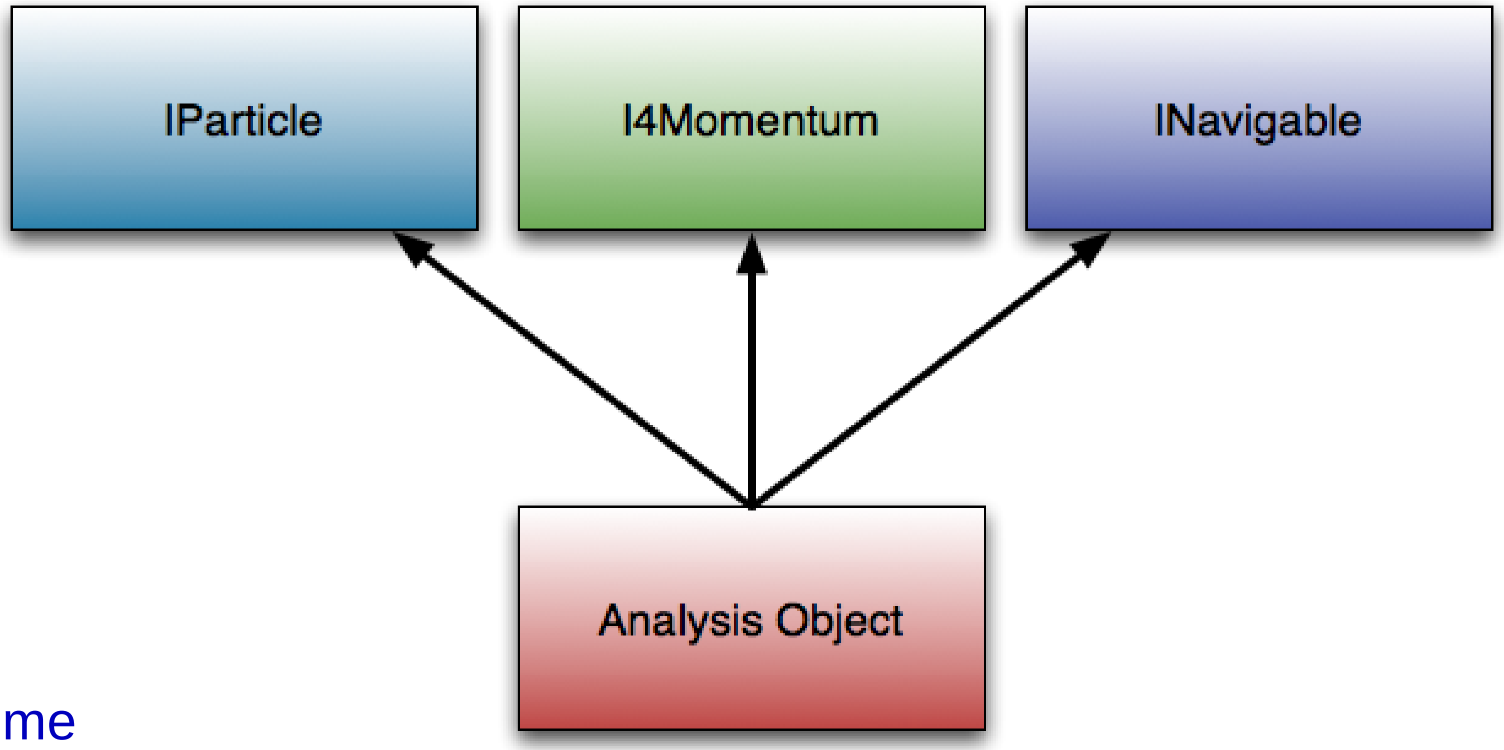
Overview

- Physics Analysis on ATLAS data can be performed at many stages along the data processing chain
  - For optimal efficiency should understand the requirements of you analysis
  - I,e, if all interesting objects are in AOD do not use the ESD for your study
- Physics and performance groups also are providing skimmed datasets.
- Next choice is to understand if you need
  - Athena-based analysis
    - Interactive ARA
  - ROOT-based analysis
    - C++ or PyROOT
- Always need to use Athena at some stage to generate output ROOT-ntuple in any case
  - E.g D3PDs
- Will illustrate some examples from the different EDM objects for analysis
- More information on D3PDs later in the week,
- With thanks to the many contributors on the following slides.



ATLAS EDM Objects for User Analysis

- For User analysis the main objects you will be interacting with are:
  - Trigger
  - EventInfo
  - TrackParticle (also Trk::Track)
  - Muon
  - Jet
  - Egamma
  - Tau
  - MissingET
  - Monte Carlo Truth information



- Many of these objects share some common interfaces:
  - 4-momentum
  - Charge, pdgID, vertex type information
  - Possible links to constituent objects
- Will next present some information for each of the main EDM physics objects.
  - In each example, the list is **not** exhaustive.
- ESD / AOD summary for release 15:
  - https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AODClassSummary15
- With thanks to the many contributors on the following slides.

Accessing Objects within Athena

- **Storegate** allows a common interface to access and retrieve all the EDM objects.
  - Exists as a transient store
  - removes the need for users to directly interact with the format of the persistified data written to data files (pool.root),
  - and the converters to perform the translation from the disk-resident object to the fully implemented physics object.

- Example method to extract a list of objects from a container:

```
const Analysis::MuonContainer* muons;
m_storeGate->retrieve(muons, "StacoMuonCollection");
MuonContainer::const_iterator muon, muonE = muons->end();
for(muon = muons->begin(); muon != muonE; muon++){
      //do stuff
}
```

- The *XXX*Container is a DataVector<*XXX*>
  - Treat similar to the STL std::vector, but with some more advanced features.

Inner Detector EDM

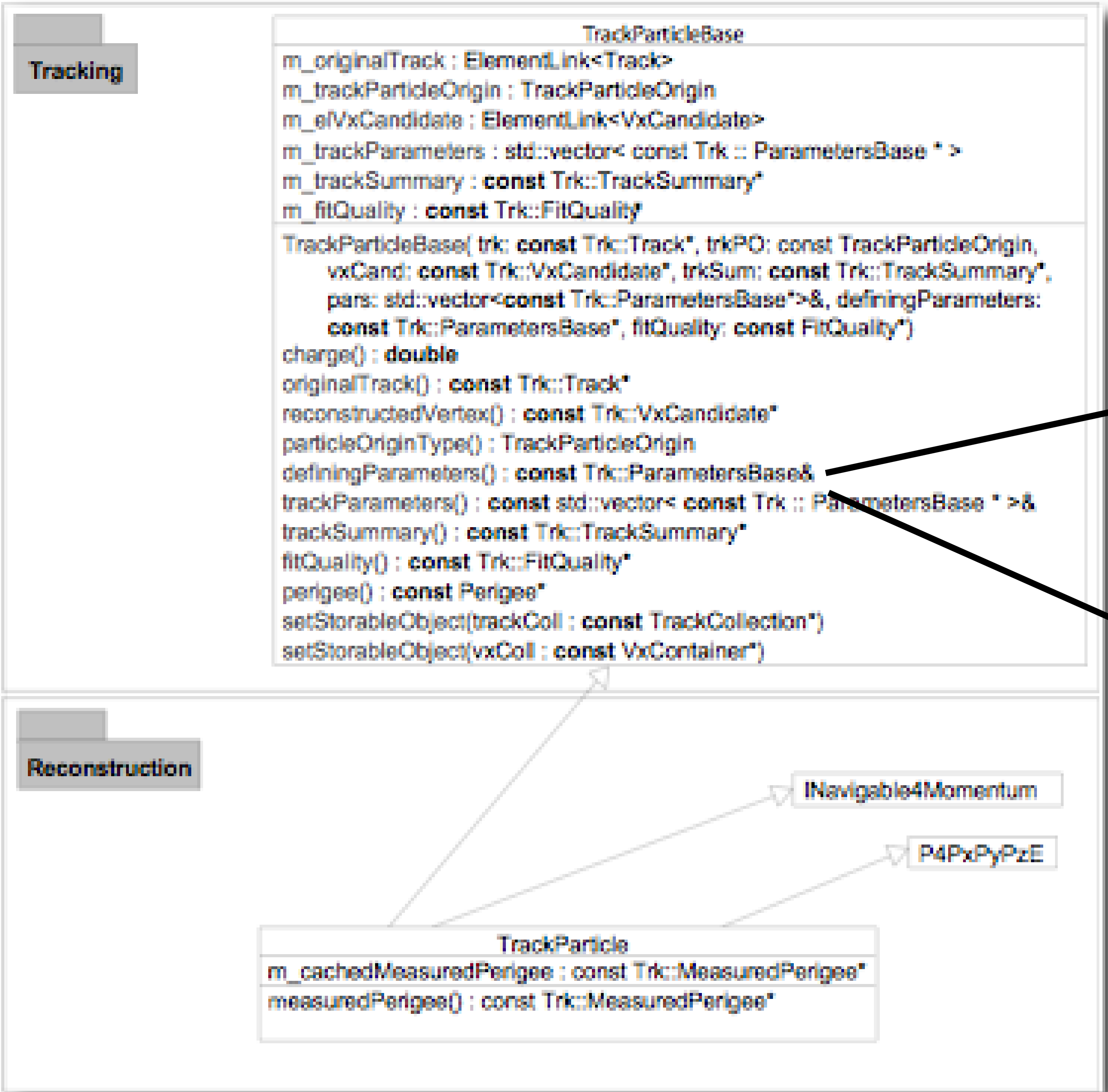
- For the Inner detector, the common analysis objects are given by:

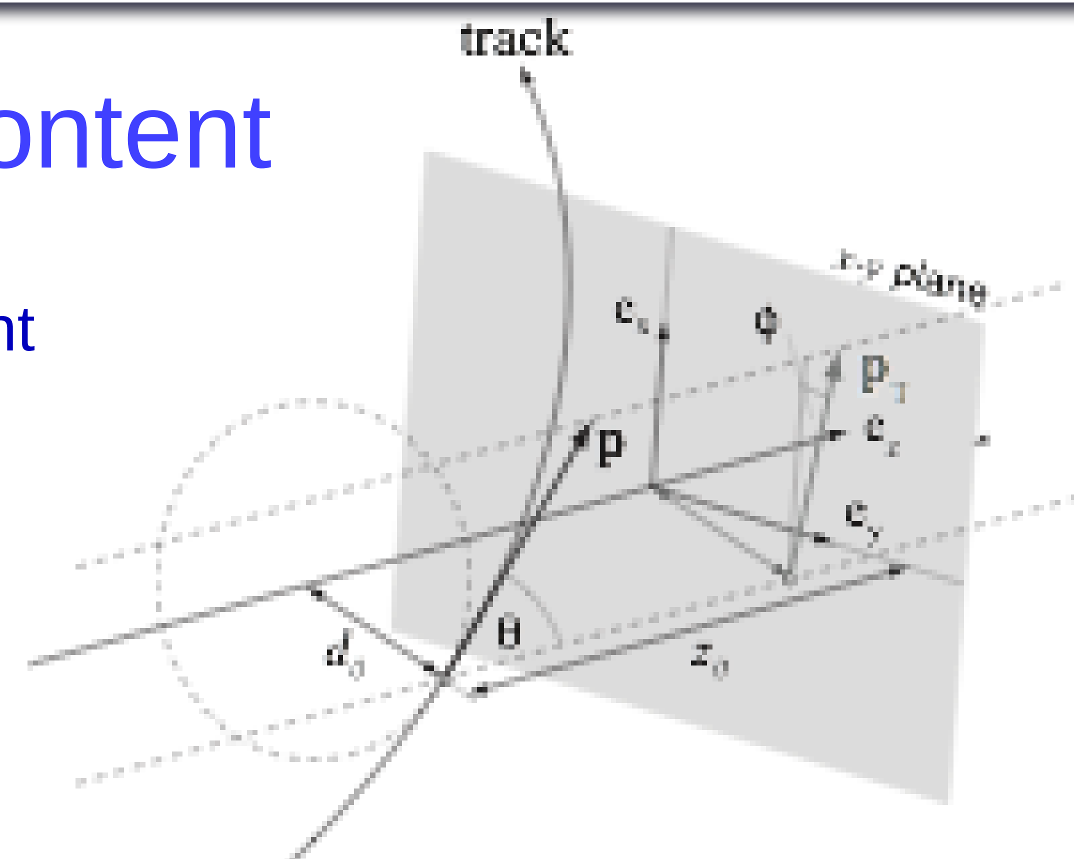
C++ Type	Name (C++ String)	Description
TrackParticleContainer	TrackParticleCandidate	Tracks from inner detector
VxContainer	VxPrimaryCandidate	List of primary and pile up vertices
VxContainer	ConversionCandidate	Vertices from conversions and from V0's
V0Container	V0Candidates	
TrackParticleTruthCollection	TrackParticleTruthCollection	Truth association for ID track particles

- VxCandidate represents a reconstructed vertex;
  - In Primary Vertex Collection, contains vertex type:
    - 0 = Dummy Vertex
    - 1 = Primary Vertex (as defined by reconstruction algorithm)
    - 3 = Pile-up vertex
  - Dummy vertex is technical device,
    - Responds with the beamspot position.

TrackParticle Content

- Simplified UML of TrackParticleBase Content





These are the track parameters which define the kinematics of this *TrackParticle*.

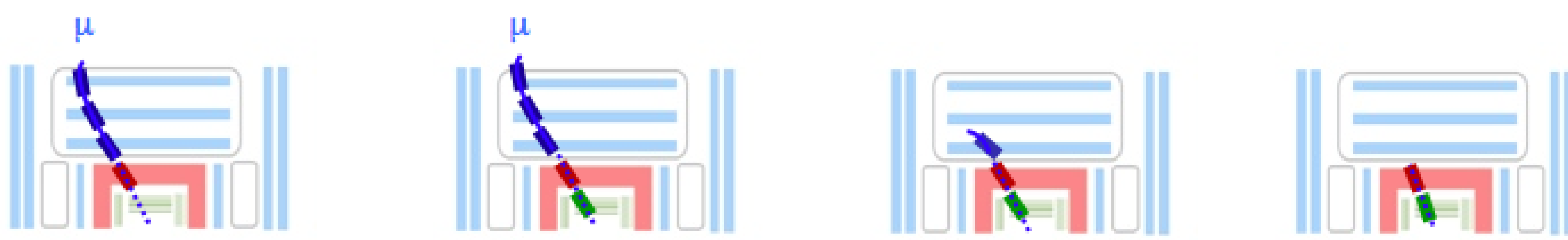
These are by default the perigee parameters defined at the transverse point of closest approach to z the axis:
$(d_0, z_0, \vartheta, \varphi, q/p)$

But the Data Model allows different references as well (for ex. a perigee w.r.t. primary vertex).

Muon reconstruction

- Several Muon reconstruction algorithms are run to identify and classify different classes of interactions of muons with the detector

Standalone Muons	Combined Muons	Segment-Tagged Muons, Inside-Out Reconstruction	Calo-Tagged Muons



Standalone Muons	Combined Muons	Segment-Tagged Muons, Inside-Out Reconstruction	Calo-Tagged Muons
Measurement of muon momentum in the muon spectrometer (MS)	Measurement of muon momentum in the MS	Measurement of momentum in the ID	Measurement of momentum in the ID
Corrected for the energy loss in the calorimeters	Corrected for the energy loss in the calorimeters	Track propagated through the calorimeters	Track propagated through the calorimeters and energy depositions around the track analyzed to assign or not a muon tag
Back-tracked to the perigee	And combined with the measurement in the inner detector (ID)	And matched to one or more segments in the MS (may be used to improve momentum measurement)	

- Type of muon algorithms used (or Author) is analysis dependent.

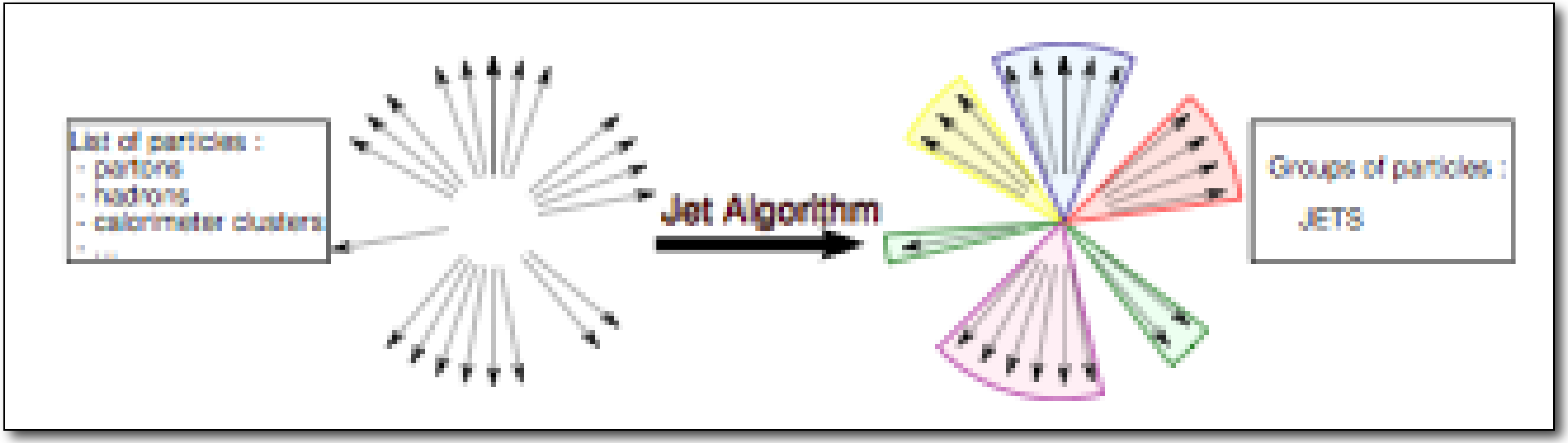
Analysis::Muon Object

- Inherits from I4Momentum
  - Has access to the pt(), e(), px(), eta(), phi(), etc... accessor methods
- Links to combined TrackParticle (if available)
  - or Muon Spectrometer extrapolated / Inner Detector / MS only tracks
- Summary of reconstruction:
  - Author (i.e. the muon reconstruction algorithm).
  - Fit quality,
  - numbers of hits (and holes) in each technology.
  - Loose, medium and tight definitions
- Additional information
  - Calorimeter Isolation Energy in the calorimeter in some cone around the muon
    - double Muon::parameter(MuonParameters::etcone20)
  - Tracking Isolation Number of tracks and pT of those tracks
    - double Muon::parameter(MuonParameters::nucone10)
    - double Muon::parameter(MuonParameters::ptcone10)
  - Associated Vertex Might or might not have been used for the fit
    - Trk::RecVertex* Muon::origin()

Egamma

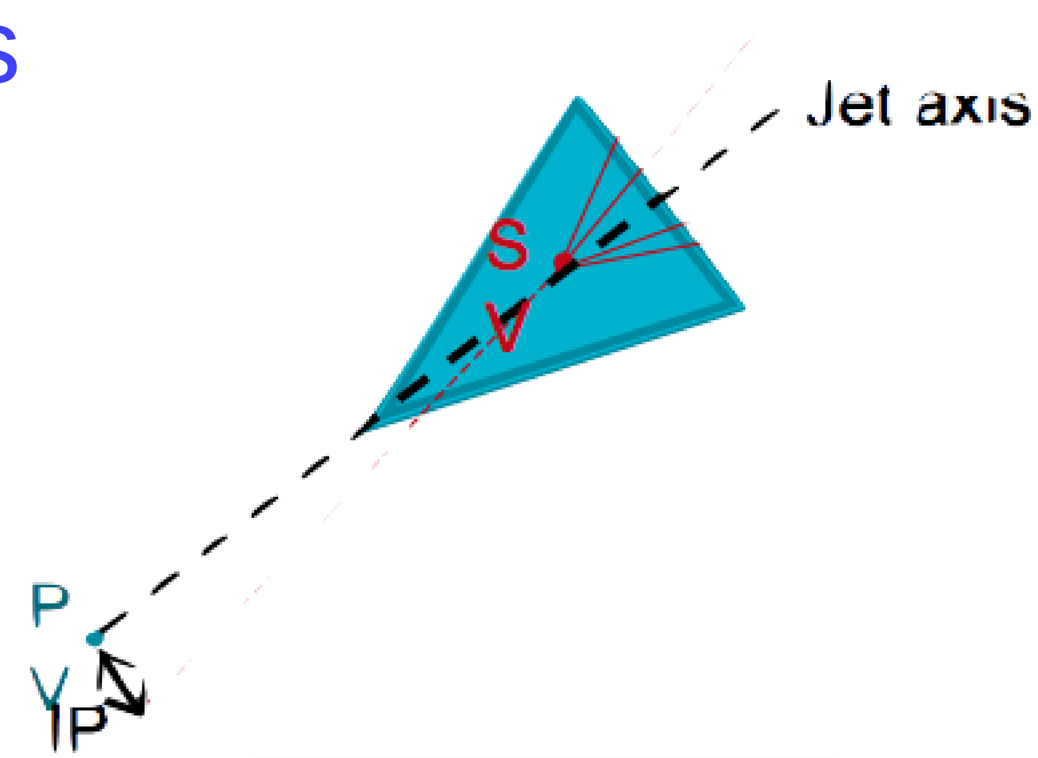
- Detectors involved
  - Inner detector : track reconstruction, particle-id in TRT
  - LAr EM calorimeter : cell, clusters + shower shapes for particle-id
  - hadronic calorimeter : for leakage and isolation requirements
- Three reconstruction algorithms
  - standard egamma algorithm – cluster-based for $|\eta|<2.5$
  - "softe" - track-based for $|\eta|<2.5$
  - forward electrons for $|\eta|>2.5$ – no inner detector information
- Electron and Photon objects inherit from egamma object
  - Author for reconstruction algorithms and 4-momentum
- Pid for Electron Photon:
  - For early data, simple cut-based quality cuts.
- eg->isElectron(egammaPID::ElectronMedium), eg->isPhoton(egammaPID::PhotonTight)
- Links back to calorimetry, TrackParticle and VxCandidate information
- General information:
  - https://twiki.cern.ch/twiki/bin/view/AtlasProtected/ElectronGamma

Jet



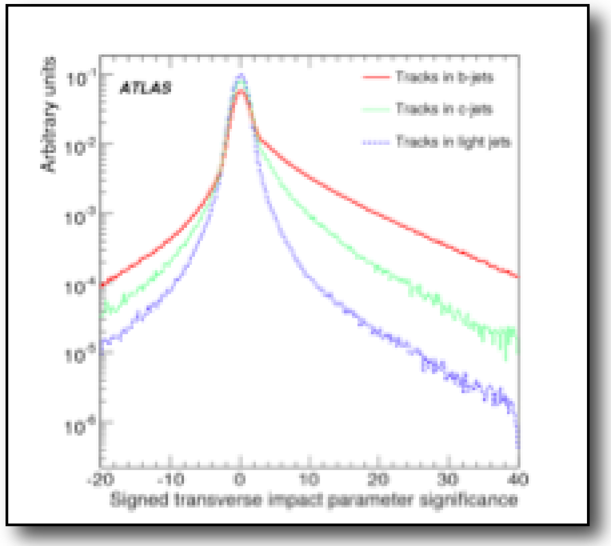
- Jet finding algorithms designed with flexibility to run on: Tracks, clusters, towers, truth
  - Accepts INavigable4Momentum input
- Different types of Jet algorithms (Kt, Cone, etc.)
- Jet has:
  - 4-momentum of the jet, links to constituent objects
  - *moments* - b-tagging, jet width, Energy per calorimeter sampling
  - "Calibration states":
    - EMSCALE , CONSTITUENTSCALE, FINALSCALE(default, calibrated scale)
  - Switches in code to access momentum for the different sates.
- https://twiki.cern.ch/twiki/bin/view/AtlasProtected/JetEtMiss

b-tagging Jets



- Distinguishing jets with a heavy-flavour component is significant in many analyses (e.g. top-physics).
  - Lifetime and lepton-tagging algorithms to assign weight of b-jet likelihood.
- Different tagging methods developed:

  - **Simple taggers:** →**Early data taggers**
    - **TrackCounting:** Counts tracks with high IP
    - **JetProb:** Track compatibility with the primary vertex
    - **SV0:** flight length significance of the SV
  - **Advanced taggers :** →**Need more commissioning**
    - **IP*n*D** (*n*=1,2,3) : IP based likelihood tagger
    - **SV*n*** (*n*=1,2): SV based likelihood tagger
    - **JetFitter*X*** (*X*=Tag,TagNN,COMB,COMBNN)
  - **Soft lepton taggers :** →**Limited efficiency**
    - SoftMuonTag
    - SoftElectronTag



- Access b-tagging weight information from each jet

```
Jet *myJet = myJetContainer[i];
```

- Access favourite tagger

```
double w = myJet->getFlavourTagWeight("Tagger");
```

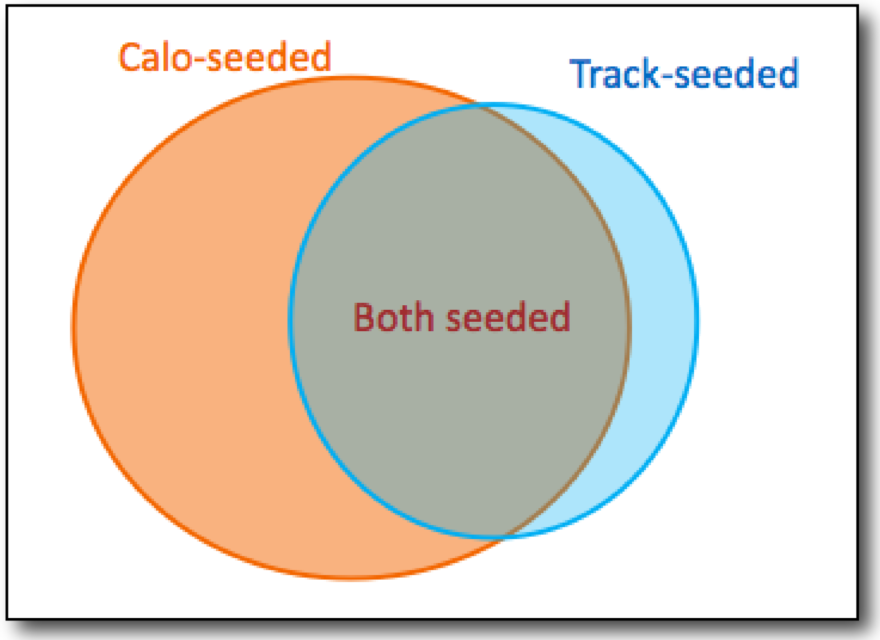
- B-tagging can be re-run 'on-the-fly' with different tunings, or on different jet collections.
- Truth matching (in MC) also available.

Tau (hadronically decaying)

- As with many EDM objects, more detailed information available only in ESD:

Information	Container	Base class	Availability
Basic	TauRecContainer	TauJet	ESD/AOD
Details	TauRecDetailsContainer	TauCommonDetails	ESD/AOD
ExtraDetails	TauRecExtraDetailsContainer	TauCommonExtraDetails	ESD only

- Two reconstruction algorithms
- Again, has a common 4-momentum interface.
- Many Tau identification discriminant algorithms available.



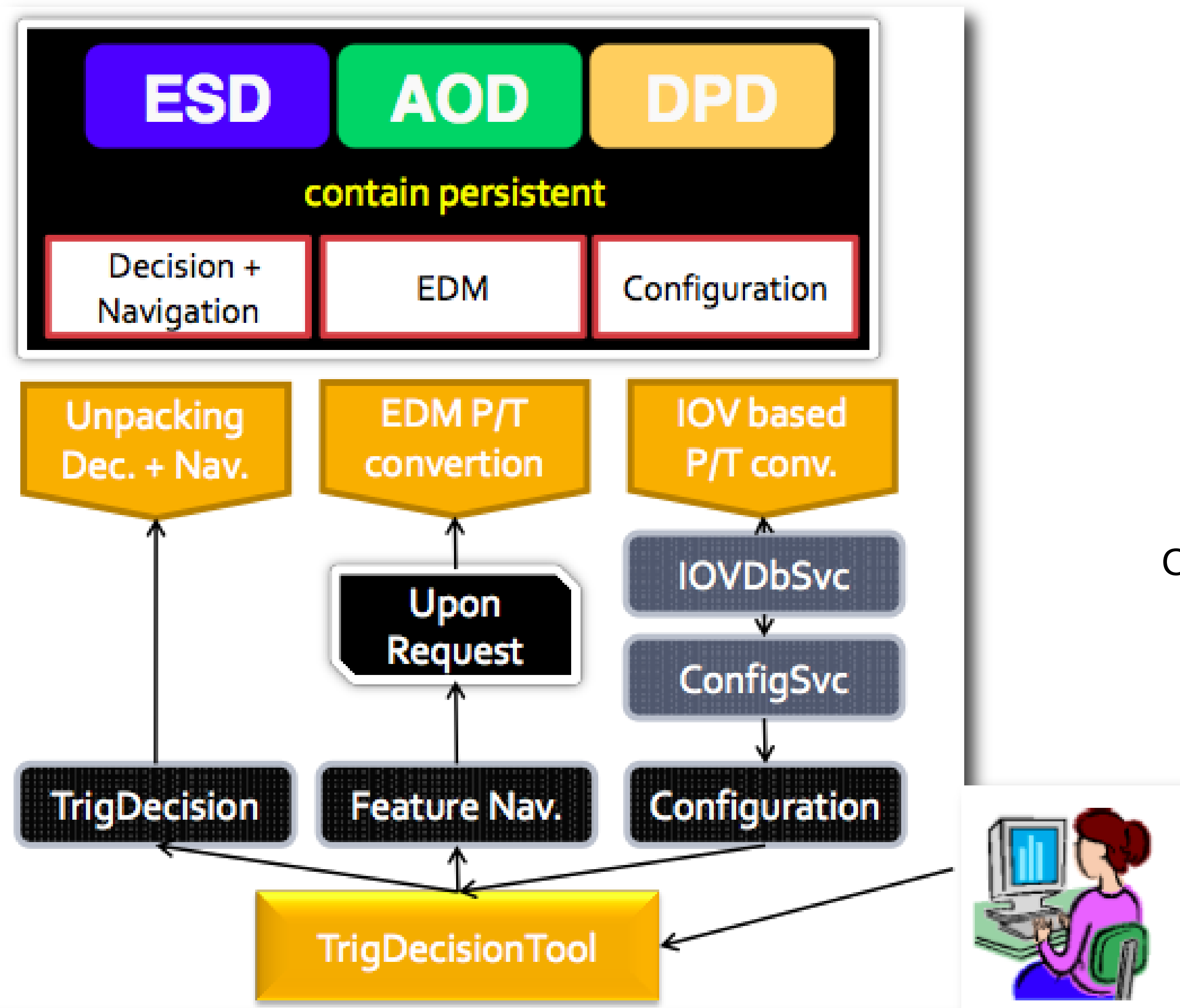
Algorithms

- Likelihood - newest likelihood (wrapper, tool in tauRec)
- DiscCut - baseline human-optimized cuts (wrapper, tool in tauRec)
- Likelihood - older likelihood version
- DiscCutTMVA - cuts optimized using TMVA
- DiscLL - projected likelihood ratio
- DiscPDERS - PDE_RS algorithm
- DiscNN - Neural Network
- EfficNN - compensated Neural Network (flat efficiency in ET)
- BDTEleScore - Boosted Decision Tree using same vars as Likelihood
- BDTJetScore - Boosted Decision Tree algorithm originally based on the D0 BDT software

- Is Tau method to allow access to baseline set of discriminants
  - **bool pass = myTauJet->tauID()->isTau(TauJetParameters::XXXX);**
- Where XXXX= TauCutSafeLoose, TauCutSafeMedium or TauCutSafeTight for cut based approach TauLlhTight, TauLlhMedium or TauLlhLoose for likelihood
  - https://twiki.cern.ch/twiki/bin/view/AtlasProtected/TauEDM

Trigger

- Trigger has complex implementation within the ATLAS Software framework.
  - Possible to directly access trigger items, however a simple tool exists:
- TriggerDecisionTool – called from your Algorithm.




```
private:
    ToolHandle<Trig::TrigDecisionTool>  tdt;
```

```
MyAlgo::MyAlgo(const std::string &name, …)
tdt("Trig::TrigDecisionTool/TrigDecisionTool")
{…}
```

```
StatusCode sc = tdt.retrieve();
```

```
if (tdt->isPassed ("L2_e15i")) {
    log << MSG::INFO << "I'm happy!" << endreq;
}
```

Can defined more advanced ChainGroups

```
mMyTrigger =

tdt.createChainGroup("EF_e10_loose","EF_mu10",...);
```

```
bool myEvent = mMyTrigger.isPassed();
```

And access the trigger objects

```
const Trig::FeatureContainer fc = mMyTrigger.features();
const std::vector< Trig::Feature< TrigTau > > taus =
    fc.get();
```

Tools for trigger matching to reconstructed objects available

Example of Athena Algorithm

- Standard Athena Algorithm contains *Initialize, Execute,* and *Finalize* methods.
- In Initialize (and the constructor):
  - Configure inputs from jobOptions,
  - Configure services and histograms
- Execute:
  - Runs once per event, Reads in objects from the Transient Store
  - ‘Do Analysis’
  - Write histograms / ntuples
- Finalize:
  - Clean up any memory, finalize plots,

Example of Athena Algorithm - Constructor

- Actual example from the the AthExRegTutorial Package example that we will use in the practical session:

```
SimpleAnalysisSkeleton::SimpleAnalysisSkeleton(const std::string& name,
  ISvcLocator* pSvcLocator) : AthAlgorithm(name, pSvcLocator)
{

  // Declare user-defined properties from jobOpts - cuts and vertexing methods etc
  declareProperty("muonPtCut", m_muonPtCut);
  declareProperty("muonEtaCut", m_muonEtaCut);
  declareProperty("muonMass", m_muonMass);
  declareProperty("muonContainerName", m_muonContainerName);
}
```

- declareProperty connects the variables in C++ to the Python jobOptions which are used to steer the job.

Example of Athena Algorithm - Initialize

```
StatusCode SimpleAnalysisSkeleton::initialize() {

  ATH_MSG_INFO("in initialize()");

  //locate the StoreGateSvc and initialize our local ptr
  StatusCode sc = service("StoreGateSvc", m_storeGate);
  if (!sc.isSuccess() || 0 == m_storeGate)
    ATH_MSG_ERROR("Could not find StoreGateSvc");

 // create a histogram
  m_dimuonMass = new TH1D("h_diMuonMass","M(#mu#mu); M(#mu#mu)
[MeV/c^{2}]; Di-muon candidates / (20 MeV/c^{2})",100,2e3,4e3);

  if (StatusCode::SUCCESS != m_thistSvc->regHist
("/ATHEXREGTUTORIAL/h_diMuonMass",m_dimuonMass)) {
    msg(MSG::ERROR) << "Unable to register module histogram " <<endreq;
  }

 return StatusCode::SUCCESS;
```

Example of Athena Algorithm - Execute

```
const Analysis::MuonContainer* importedMuonColl;
StatusCode sc = m_storeGate->retrieve(importedMuonColl,m_muonContainerName);
if (sc.isFailure() ) {
  ATH_MSG_WARNING("No Muon Collection of type " << m_muonContainerName << " found in
StoreGate");
  importedMuonColl=0;
}else{
  ATH_MSG_DEBUG("You have " << importedMuonColl->size() << " muons in this event");
}

Analysis::MuonContainer::const_iterator muonIter;
for (muonIter=importedMuonColl->begin(); muonIter!=importedMuonColl->end(); ++muonIter) {
      bool isGoodMuon = false;
      double pt = fabs((*muonIter)->pt());
      double eta = fabs((*muonIter)->eta());

if ( (*muonIter)->isCombinedMuon() || (*muonIter)->isLowPtReconstructedMuon() ) isGoodMuon = true;
      if ( (pt > m_muonPtCut) && (eta < m_muonEtaCut) && isGoodMuon ) {
            // Fill histogram
      }
}
```


LANCASTER UNIVERSITY

Example of Athena Algorithm - Finalize

```
StatusCode SimpleAnalysisSkeleton::finalize() {

  ATH_MSG_DEBUG("in finalize()");

  // Summary of analysis can go here

  // print a summary of the tree
  m_theTree->Print();

  return StatusCode::SUCCESS;
}
```

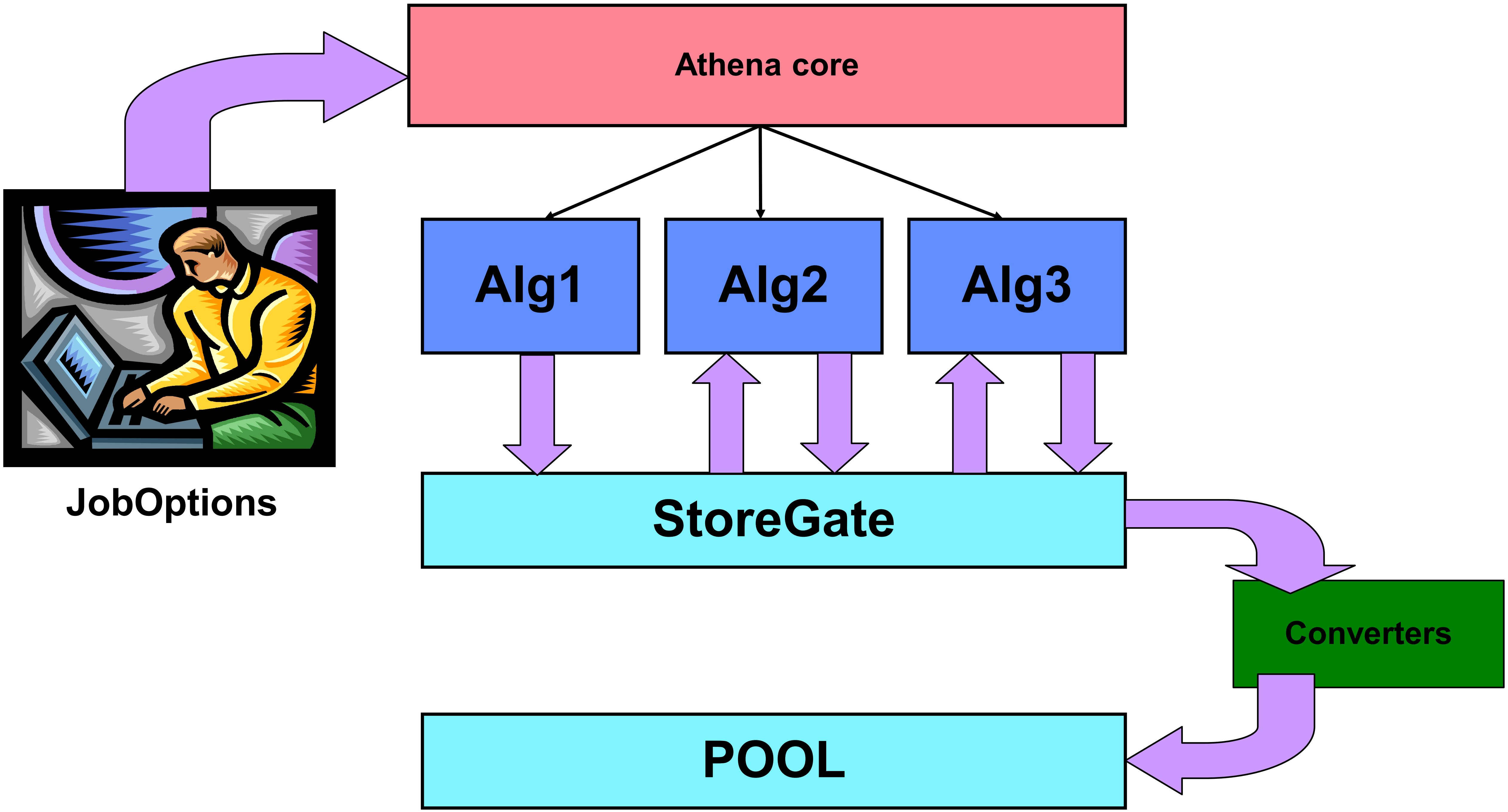
- In this example, the finalize method has nothing to do.

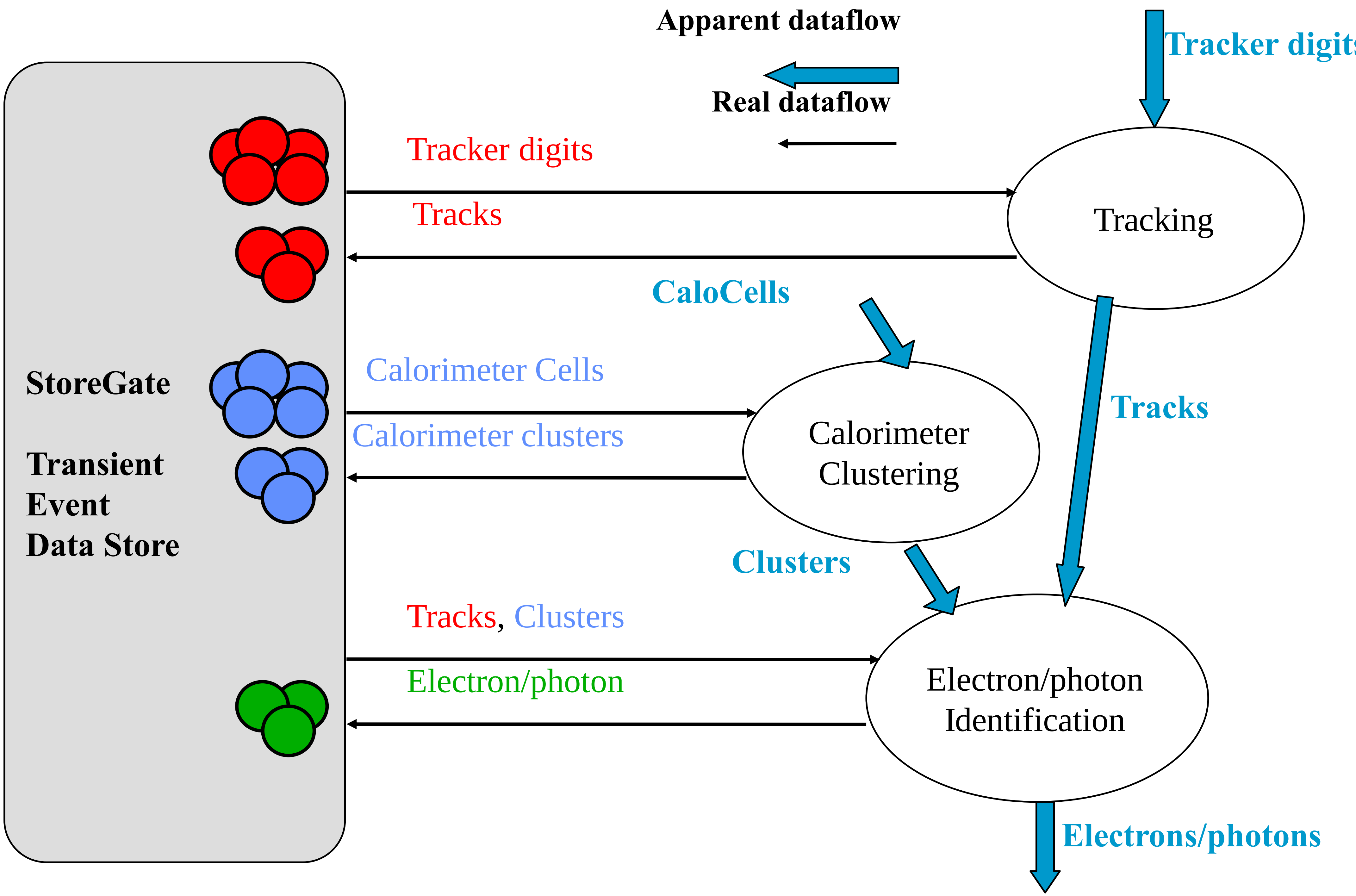
- This is a simple example,
  - But demonstrates the common methods and tools available to users.
- Even the D3PD Making Algorithm runs from this basic setup.

Summary

- Physics Analysis can be performed at:
  - ESD, AOD stage in athena, (even some RAW studies)
  - Many groups implementing Standard ntuple-making software
    - The D3PD set of tools is being adopted amongst many groups
- For Final analysis:
  - For example, histogram filling, fits etc..
  - Almost always in the simple ROOT format
- Even for ROOT-based analysis you should still know the origin of the Objects and Tools available in the EDM.
- The Analysis-object Classes provides:
  - Common interfaces to access 'main' physics quantities (e.g. 4-mom.)
  - Quantities specific to each object type are also contained (e.g Author)

Backup


Athena core
Alg1
Alg2
Alg3
JobOptions
StoreGate
Converters
POOL



Apparent dataflow
Real dataflow
Tracker digits
StoreGate
Transient
Event
Data Store
Tracker digits
Tracks
Tracking
CaloCells
Calorimeter Cells
Calorimeter clusters
Calorimeter
Clustering
Tracks
Clusters
Tracks, Clusters
Electron/photon
Electron/photon
Identification
Electrons/photons