中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

Tutorial on CEPCSW calorimeter reconstruction

Wenxing Fang(IHEP)

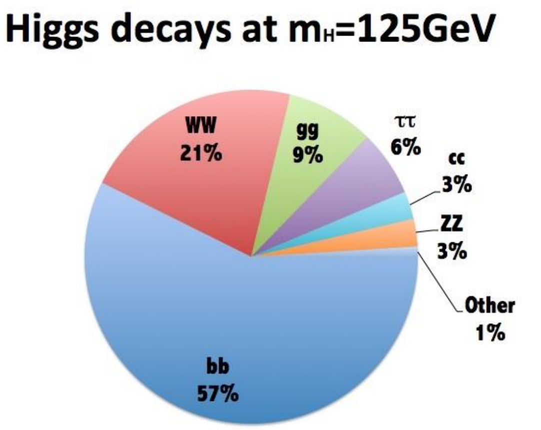
New CEPCSW Tutorial and detector study (17th-18th September 2020)

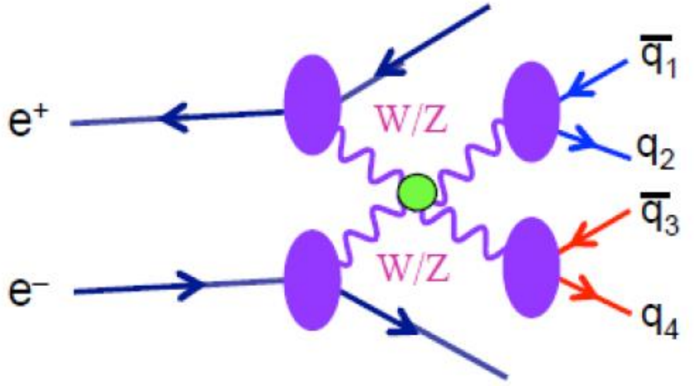
Outline

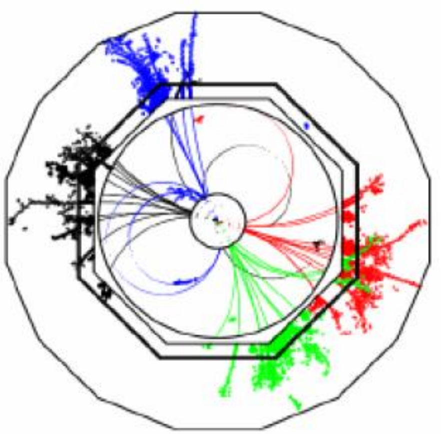
- Why particle flow calorimetry
- How pandora PFA works
- An example in CEPCSW

Motivation

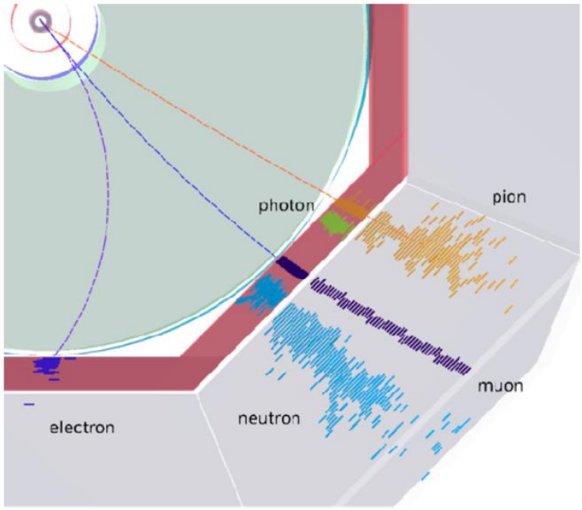
- The majority of the Higgs, W, and Z bosons decay into quarks or gluons which fragment into hadronic final states. Usually these final states are reconstructed jets.
- Therefore, the jet energy resolution (JER) σ_E/E is very important for CEPC experiment.
- For example, in order to have 2σ separation between Z and W using di-jet invariant mass, JER ~ 4% is needed.





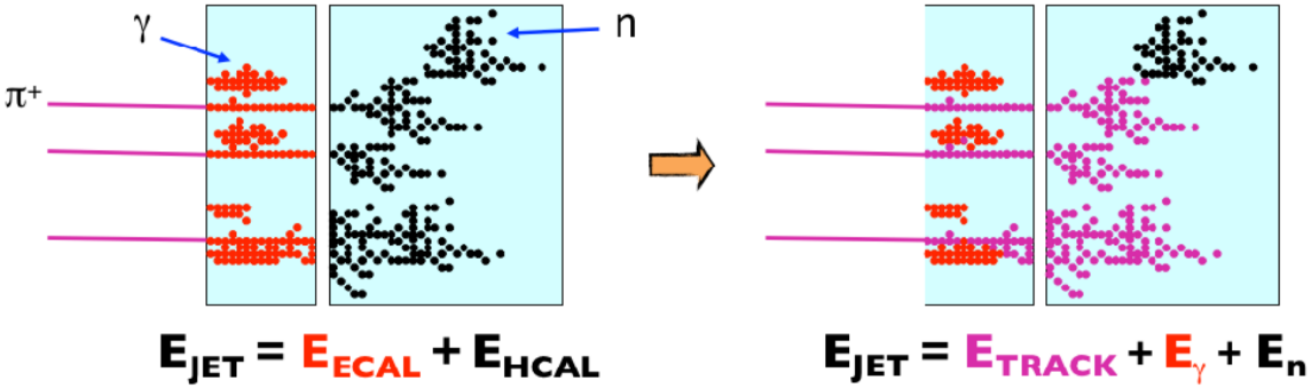


Particle flow calorimetry



- Energy component of a typical jet:
  - 60% in charged hadrons
  - 30% in photons (mainly from $\pi^0 \rightarrow \gamma\gamma$)
  - 10% in neutral hadrons (mainly η and K_L)

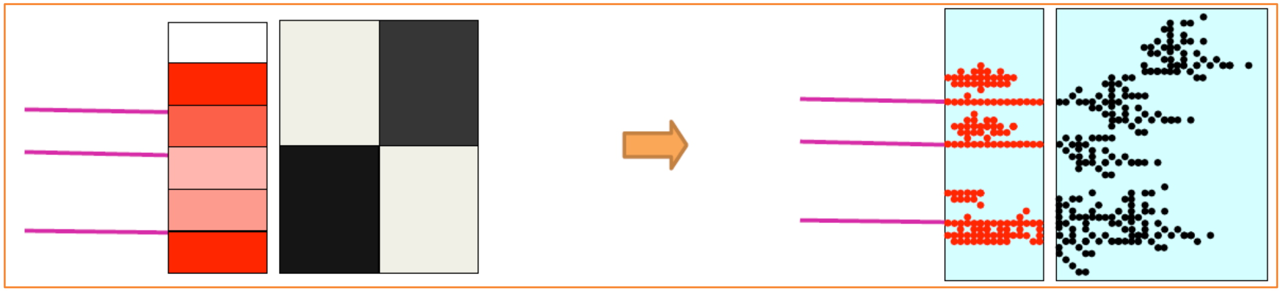
- Traditional calorimetric approach (energy flow):
  - Measure all components of jet energy in ECAL and HCAL
  - Approximately 70% of energy measured in HCAL: $\frac{\sigma_E}{E} \approx 60\%/\sqrt{E}(GeV)$



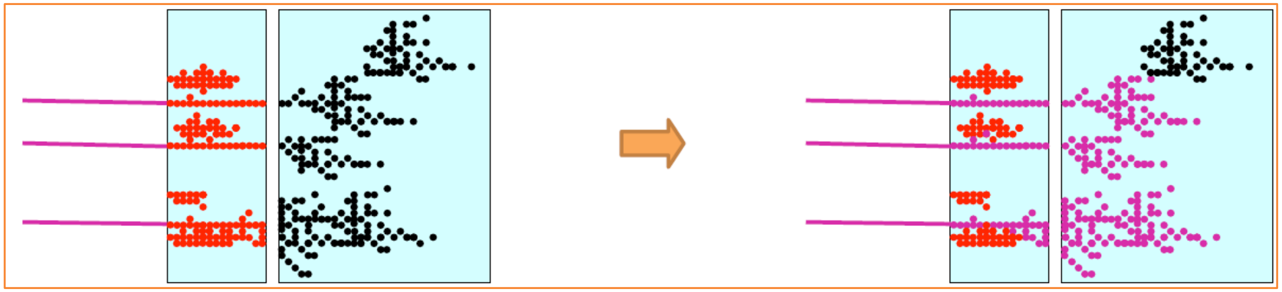
- Particle Flow Calorimetry: reconstruct individual particles
  - Charged particle momentum measured in tracker (essentially perfectly)
  - Photon energies measured in ECAL: $\frac{\sigma_E}{E} < 20\%/\sqrt{E}(GeV)$
  - Only neutral hadron energies (10% of jet energy) measured in HCAL
  - Much improved resolution

Requirement

- Hardware: need to be able to resolve energy deposits from different particles.
  - Require highly granular detectors (as the one in CEPC).



- Software: need to be able to identify energy deposits from each individual particle.
  - Require sophisticated reconstruction software to deal with complex events, containing many hits.



Particle Flow Calorimetry = HARDWARE + SOFTWARE

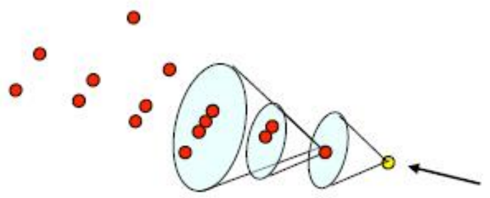
PFAs

- There are many Particle Flow algorithms for e^+e^- collider experiments:
  - **PandoraPFA**: the standard PFA in ILCSoft. It derived a general pattern recognition package, **PandoraSDK**. NIMA 611 (2009) 25–40
  - **ARBOR**, originated from LEP ear; now used for the CEPC CDR studies. arXiv:1403.4784, Eur. Phys. J. C78 (2018) no. 5, 426
  - **APRIL** (Algorithm of Particle Reconstruction for the ILC, Lyon): use PandoraSDK as the framework, and the basic idea of ARBOR for clustering. arXiv:2002.09678
  - **GARLIC** (GAmma Reconstruction at LInear Collider experiment): MVA for PID. JINST 7 (2012) P06003

- Currently only the PandoraPFA is in the CEPCSW, so following part will focus on PandoraPFA.

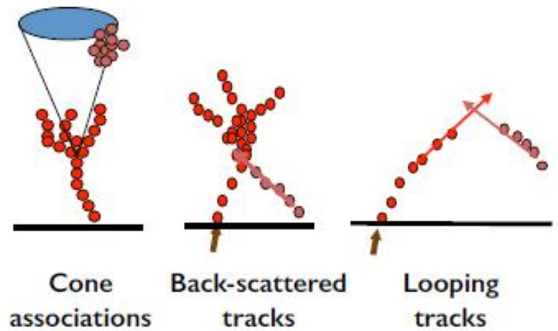
Pandora Algorithms

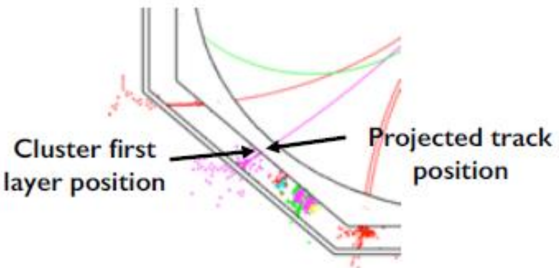
60+ algorithms for fine-granularity detectors



ConeClustering Algorithm

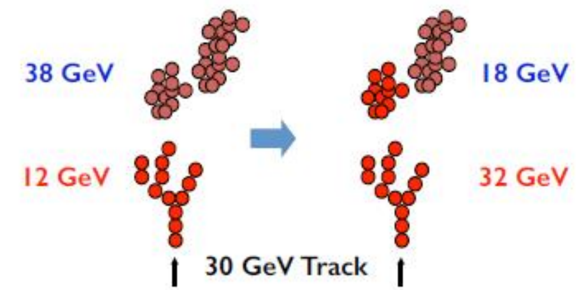
Topological Association Algorithms

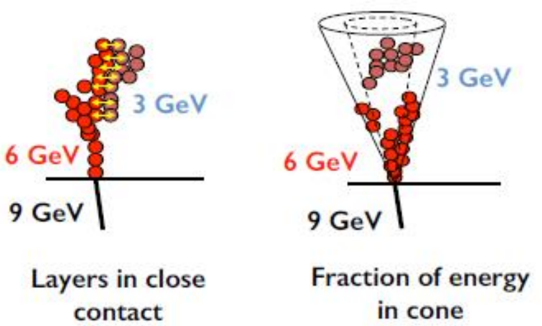




Track-Cluster Association Algorithms

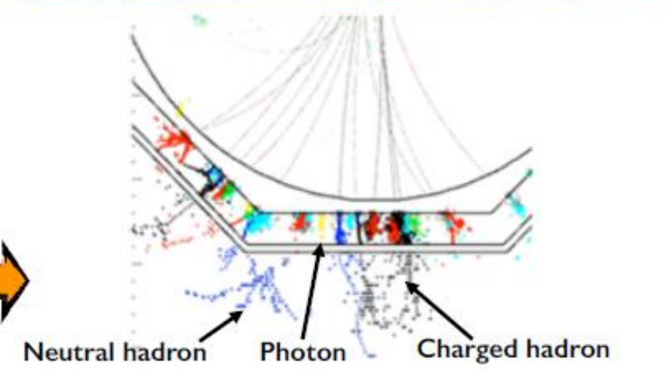
Reclustering Algorithms





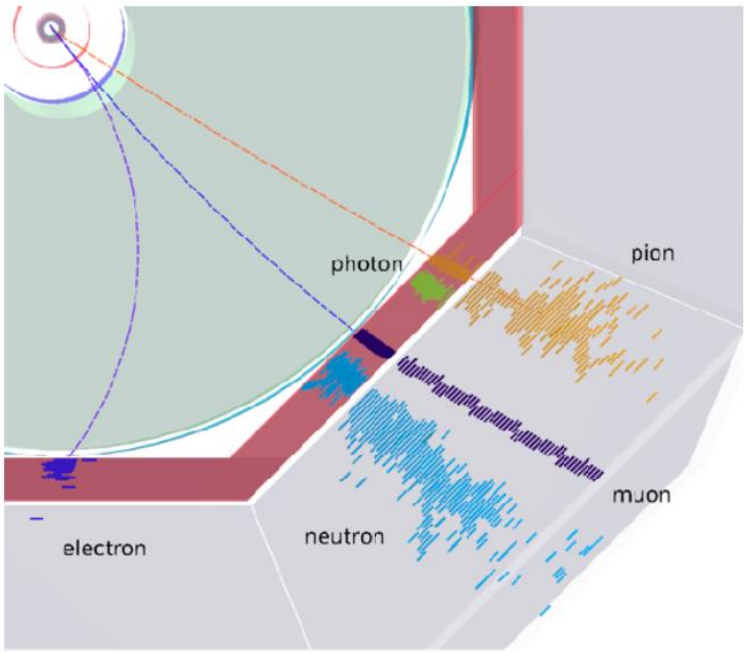
Fragment Removal Algorithms

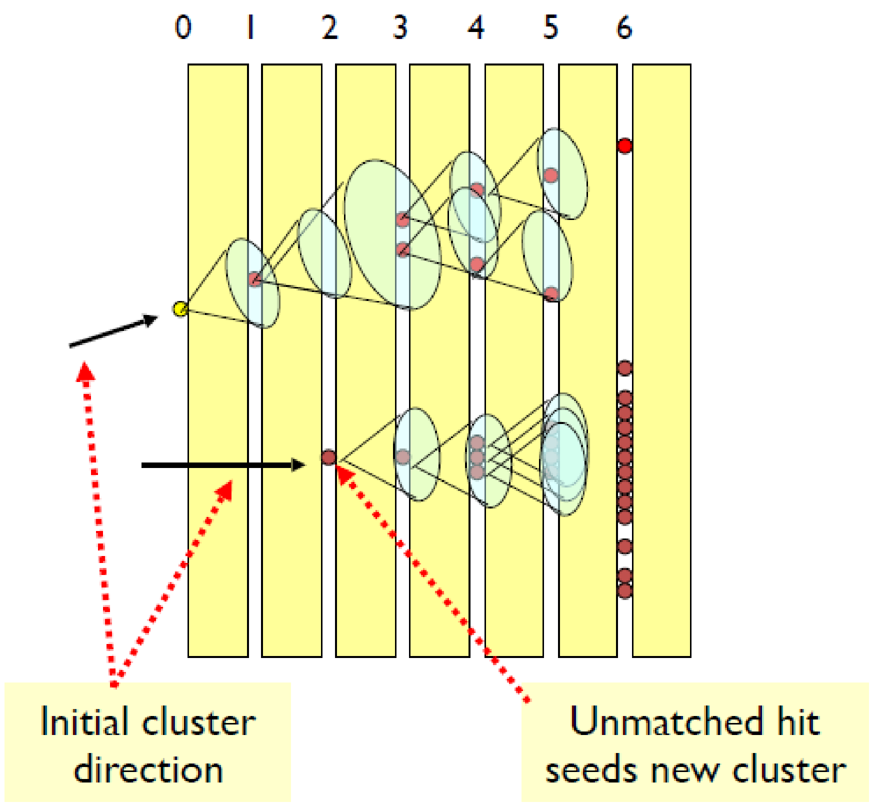
PFO Construction Algorithms



Cone Clustering

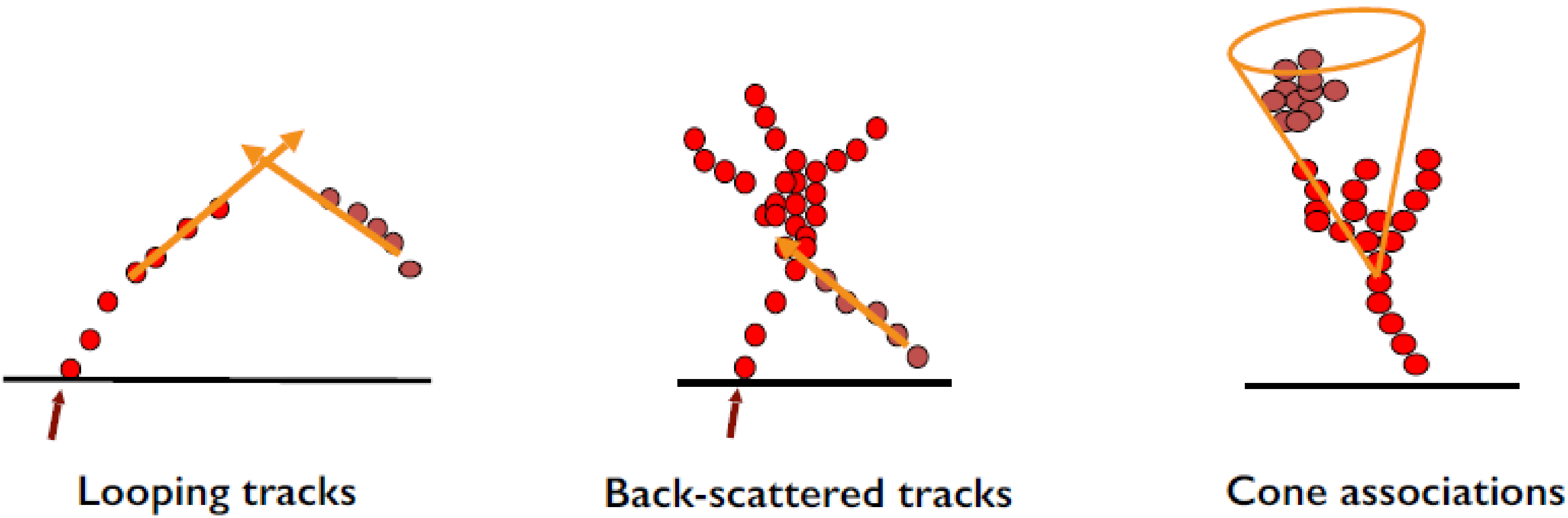
- Clusters seeded by projections of inner detector tracks to surface of calorimeter.
- Start at innermost layers and work outward, considering each calorimeter hit in turn.
  - If hit lies within cone defined by existing cluster, and is suitably close, add hit to cluster.
  - If hit is unmatched, use it to form a new cluster.





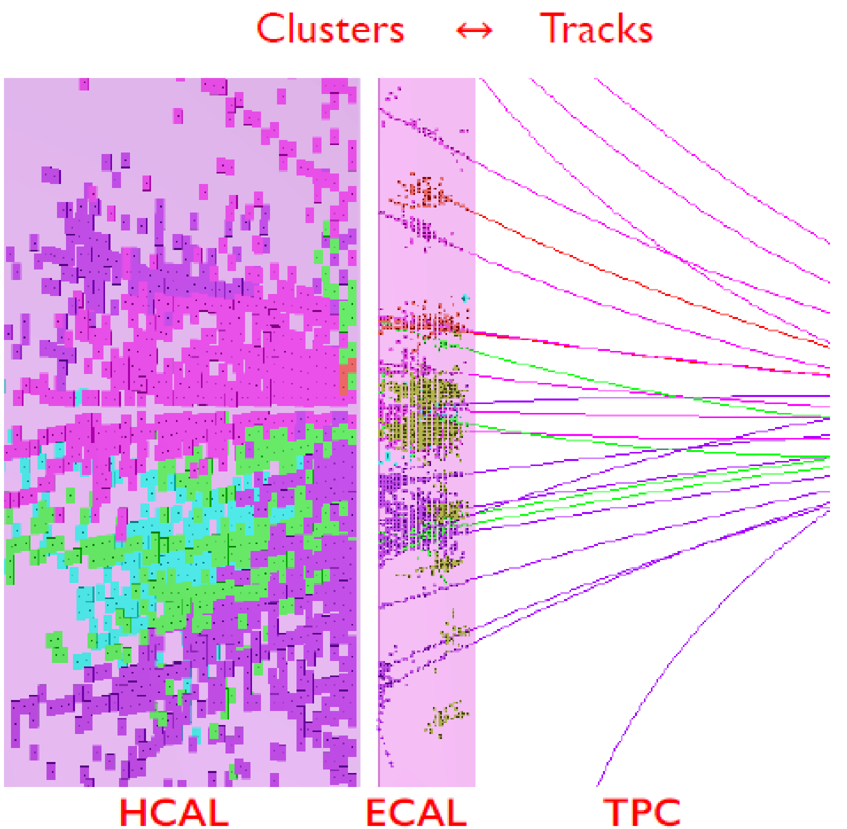
Topological Association

- ❑ Cone clustering algorithm may creates clusters that are fragments of single particles, rather than risk merging deposits from separate particles.
- ❑ Cluster fragments are then merged together by a series of algorithms, each of which follows well-defined topological rules.
- ❑ Fine granularity of the calorimeters exploited to merge cluster fragments that are clearly associated. Very few mistakes!



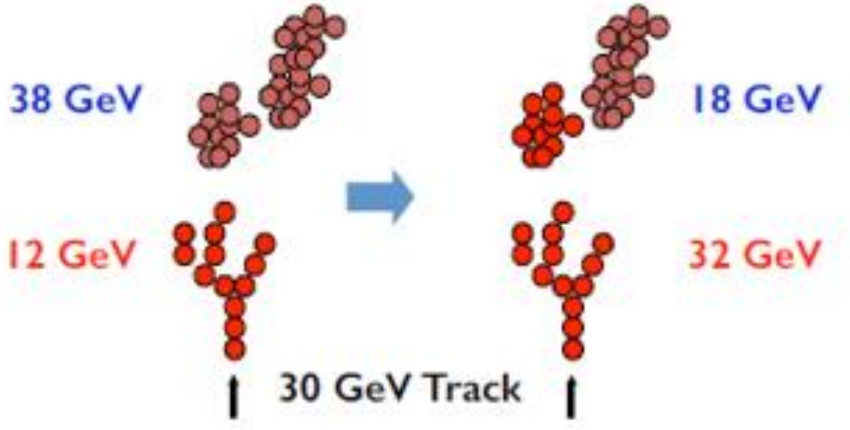
Track-Cluster Associations

- The Pandora track-cluster association algorithms look for consistency between cluster properties and the helix-projected track state at the front face of the calorimeter:
    - Close proximity between cluster and track positions.
    - Consistent track and initial cluster directions.
    - Consistent track momentum and cluster energy.



Reclustering

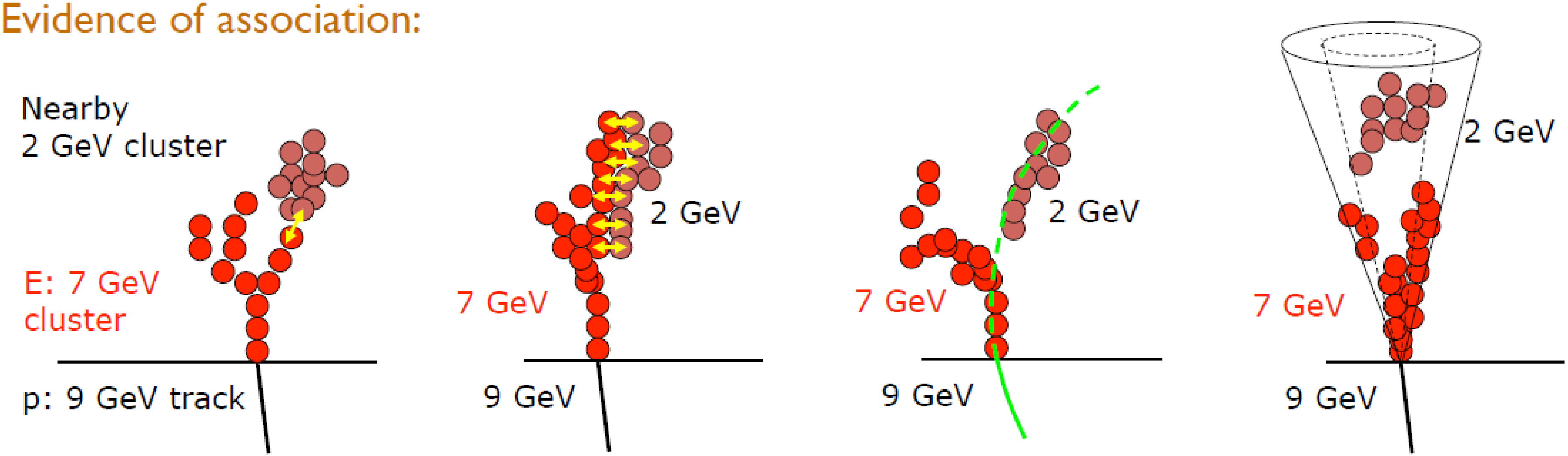
- If there are significant discrepancy between energy of a cluster and momentum of its associated track, choose to recluster.
- Alter clustering parameters, or change clustering algorithm entirely, until cluster splits in such a way that we obtain sensible track-cluster associations.



Fragment Removal

- Fragment removal algs aim to remove neutral clusters (those without track-associations) that are really fragments of charged (track-associated) clusters.
- Algs look for evidence of association between nearby clusters, merging the clusters together. In order to merge clusters, the change must bring about a satisfactory change in E/p $\chi 2$.

Evidence of association:



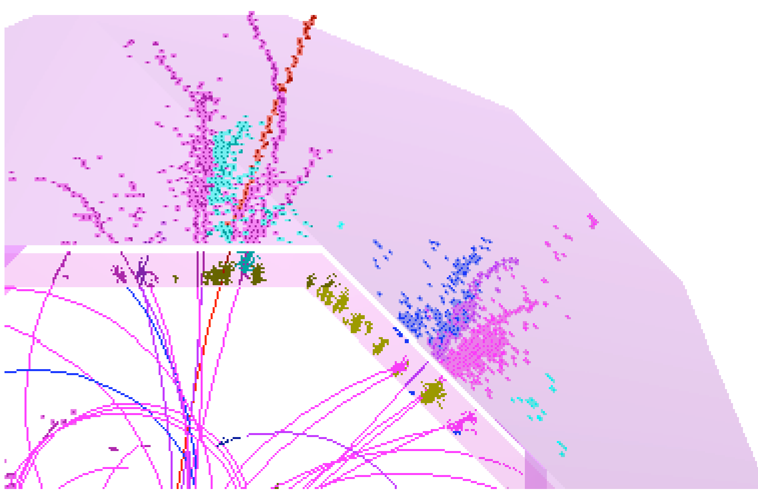
Small distance of closest approach | Multiple layers in close contact | Small distance to track extrapolation | Large fraction of energy in cone

Particle Identification

- Particle ID is crucial for many physics analyses. Currently available: charged lepton and photon ID

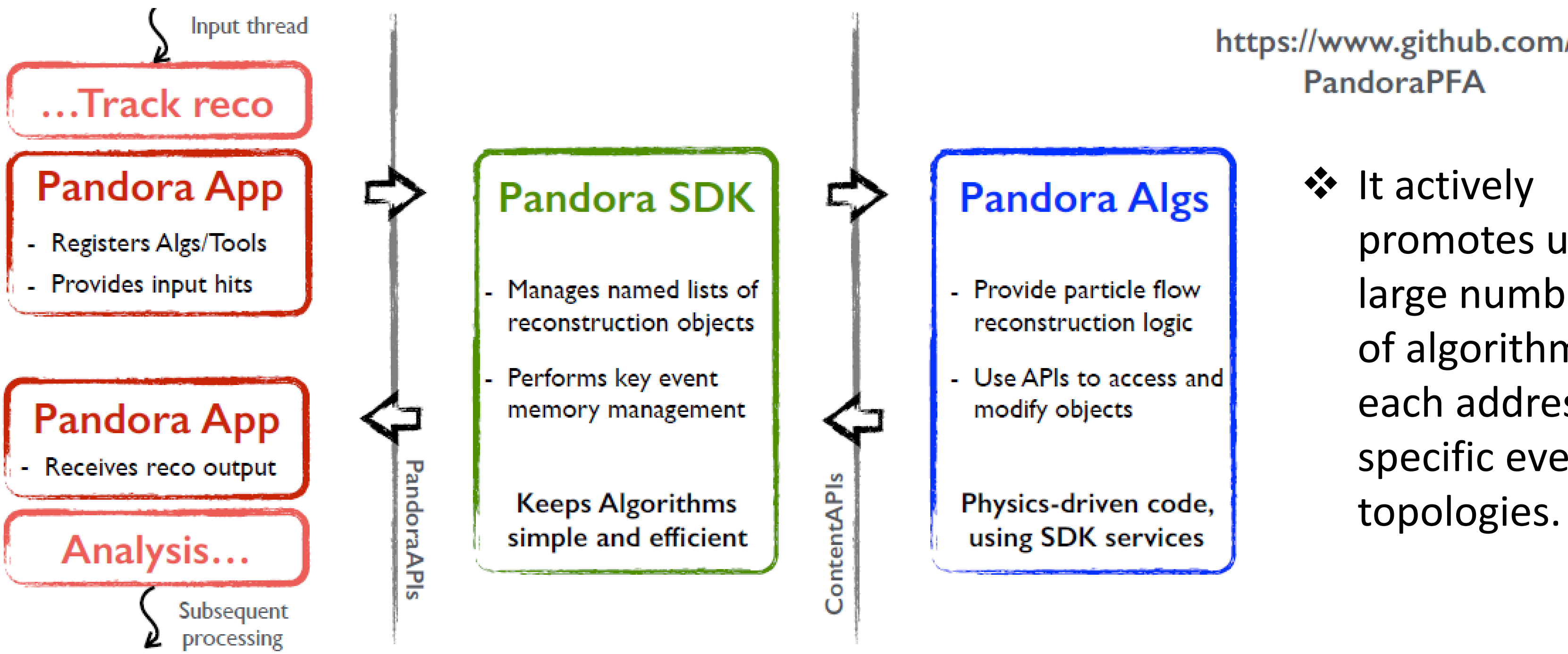
e.g. dedicated muon alg.

1. Cluster hits in muon yoke
2. Associate to inner detector track
3. "Swim" through calorimeter



PandoraSDK

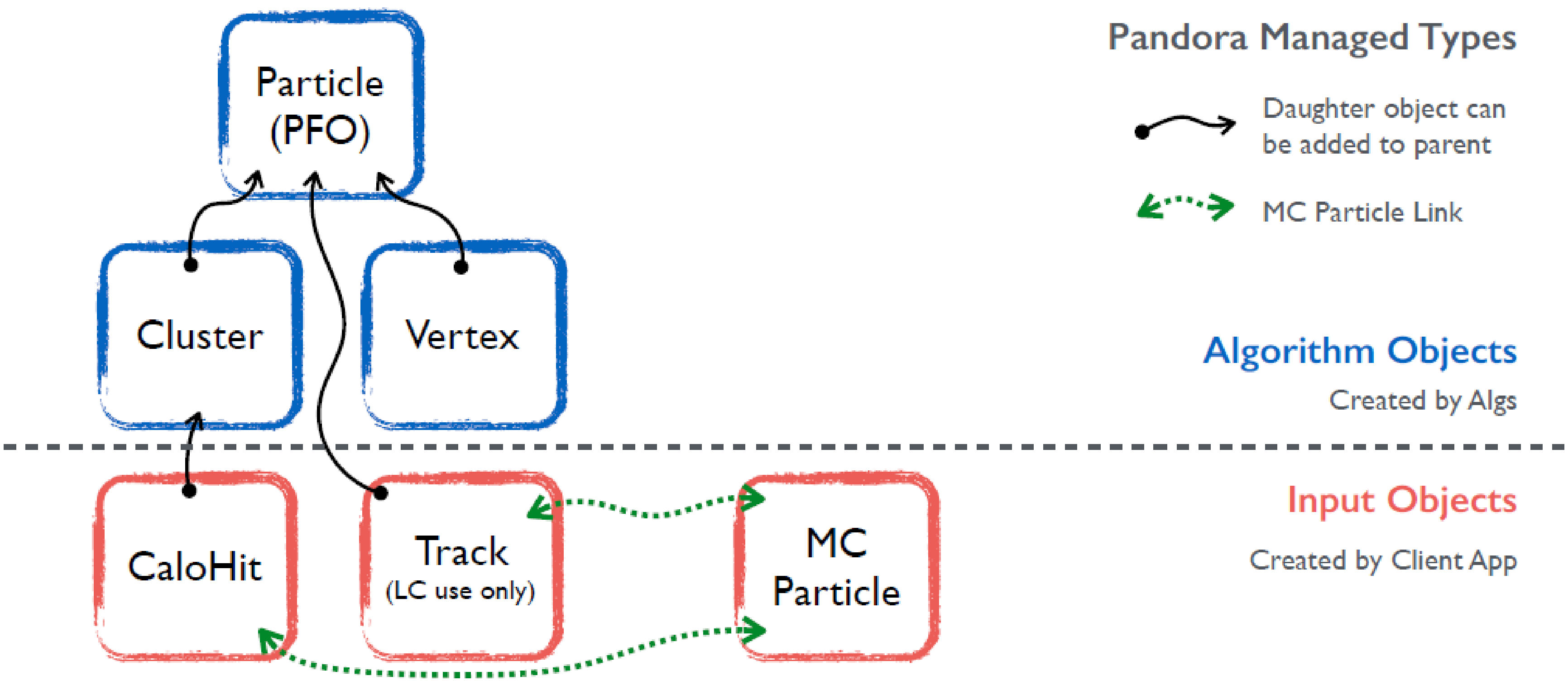
- The **Pandora Software Development Kit** is engineered to provide an environment in which:
  1. It is easy for users to provide the building-blocks that define a pattern recognition problem.
  2. Logic required to solve pattern recognition problems is cleanly implemented in algorithms.
  3. Operations to access or modify building-blocks, or to create new structures, are requested by algorithms and performed by the Pandora framework.



- It actively promotes use of large numbers of algorithms, each addressing specific event topologies.

Event Data Model

- **EDM consists of classes to represent the input building-blocks for pattern-recognition problems and the structures that can be created using these building-blocks.**
- Provides well-defined development environment for managing pattern-recognition problems and allows for independence of algorithms, which can only communicate via the EDM.
- EDM aims to be self-describing, with each object providing all the information required to allow investigation and processing by the pattern-recognition algorithms.



Input Objects

- **Input Objects are the building-blocks for pattern recognition, typically created by the client app before algorithm operations begin.**
- Their properties are defined at creation and cannot be changed. They are instead used to build new constructs, termed "Algorithm Objects".
- The usage of all Input Objects is monitored to ensure that no double-counting/usage occurs.

CaloHit	Primary building-block, defining a position and extent in space (or time), with an associated intensity or energy measurement and detector location details.
Track (LC use only)	Represents a continuous trajectory of well-defined space-points, with helix parameterisation. Track parent-daughter and sibling relationships supported.
MC Particle	For development purposes, provide details of true pattern-recognition solution. Support parent-daughter links and can be associated to CaloHits and Tracks.

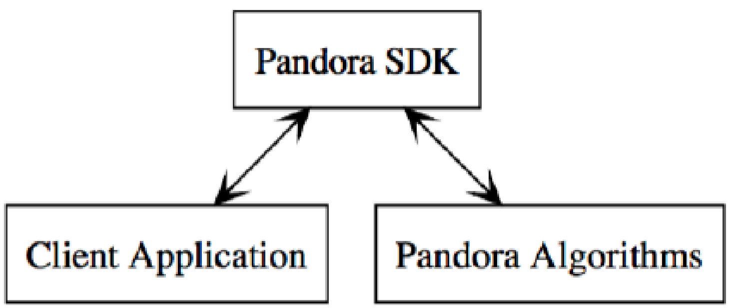
Algorithm Objects

- **Algorithm Objects represent the higher-level structures created in order to solve pattern-recognition problems.**
- Pandora carefully manages the allocation and manipulation of these objects and all non-const operations can only be requested by algorithms via the Pandora Content APIs.
- Pandora is then able to perform the memory-management for these objects.

Cluster	Collection of CaloHits and main working-horse for algorithms (which create, merge, split Clusters). Provides some derived properties of CaloHit collection.
Vertex	The identification and classification of a specific point in space, typically used to flag positions of particle creation or decay.
Particle	Container of Clusters, Tracks and Vertices, together with metadata describing e.g. particle type. Ultimate Pandora output and can represent a hierarchy.

Client Application

- **Client app is responsible for providing Input Objects that define the pattern-recognition problem and for persisting the output Particles. Experiment dependent.**
  - Responsible for creating Pandora instances and for configuring the reconstruction algorithms via the Pandora Settings XML file.



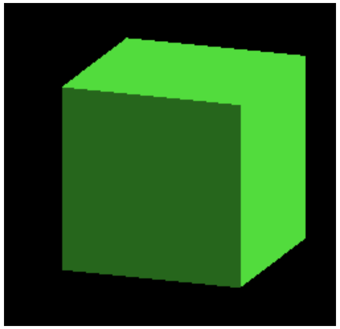
Built with client app, or in libraries of Pandora content

Algorithm Pseudocode description of a client application for LAr TPC event reconstruction in a single drift volume

```
procedure MAIN
    Create a Pandora instance
    Register Algorithms and Plugins
    Ask Pandora to parse XML settings file
    for all Events do
        Create CaloHit instances
        Create MCParticle instances
        Specify MCParticle-CaloHit relationships
        Ask Pandora to process the event
        Get output PFOs and write to file
        Reset Pandora before next event
```

Example

git clone git@github.com:cepc/CEPCSW.git (as a user)
cd CEPCSW
git checkout master
vim Examples/options/tut_detsim_pan_matrix.py



```python
from Configurables import GeoSvc
geosvc = GeoSvc("GeoSvc")
#geosvc.compact = geometry_path
geosvc.compact = "./Detector/DetEcalMatrix/compact/det.xml"

##############################################################
# Physics Generator
##############################################################
from Configurables import GenAlgo
from Configurables import GtGunTool
from Configurables import StdHepRdr
from Configurables import SLCIORdr
from Configurables import HepMCRdr
from Configurables import GenPrinter

gun = GtGunTool("GtGunTool")
gun.Particles = ["gamma","gamma"]
gun.EnergyMins= [5 , 10] # GeV
gun.EnergyMaxs= [5 , 10] # GeV
gun.ThetaMins = [90, 90] # degree
gun.ThetaMaxs = [90, 90] # degree
gun.PhiMins   = [0,  4 ] # degree
gun.PhiMaxs   = [0,  4 ] # degree
```

```xml
<detectors>
  <detector id="1" name="CaloDetector" type="EcalMatrix" readout="CaloHitsCollection"
    <!-- Use cm as unit if you want to use Pandora for reconstruction -->
    <position x="200*cm"  y="0"  z="0"/>
    <dimensions dx="30*cm"  dy="30*cm"  dz="30*cm"/>
  </detector>
</detectors>

<readouts>
  <readout name="CaloHitsCollection">
    <!-- <segmentation type="NoSegmentation"/> -->

    <segmentation type="CartesianGridXYZ"
                  grid_size_x="1*cm"
                  grid_size_y="1*cm"
                  grid_size_z="1*cm"/>
    <id>system:8,x:32:-6,y:-6,z:-6</id>
  </readout>
</readouts>
```

Detector/DetEcalMatrix/src/calorimeter/EcalMatrix.cpp
Construct geometry and save Extension data (e.g. layer thickness, cellSize, …) for reconstruction.

Example

```
##############################################################################
# Detector Simulation
##############################################################################
from Configurables import DetSimSvc
detsimsvc = DetSimSvc("DetSimSvc")
from Configurables import DetSimAlg
detsimalg = DetSimAlg("DetSimAlg")
# detsimalg.VisMacs = ["vis.mac"]
detsimalg.RunCmds = [
#    "/tracking/verbose 1",
]
detsimalg.AnaElems = [
    "Edm4hepWriterAnaElemTool"
]
detsimalg.RootDetElem = "WorldDetElemTool"
from Configurables import AnExampleDetElemTool
example_dettool = AnExampleDetElemTool("AnExampleDetElemTool")
##############################################################################
# Detector digitization
##############################################################################
from Configurables import CaloDigiAlg
example_CaloDigiAlg = CaloDigiAlg("CaloDigiAlg")
example_CaloDigiAlg.Scale = 1
example_CaloDigiAlg.SimCaloHitCollection = "SimCalorimeterCol"
example_CaloDigiAlg.CaloHitCollection     = "ECALBarrel"
example_CaloDigiAlg.CaloAssociationCollection     = "RecoCaloAssociation_ECALBarrel"
##############################################################################
```

```
##############################################################################
# Pandora
##############################################################################
from Configurables import PandoraMatrixAlg

pandoralg = PandoraMatrixAlg("PandoraMatrixAlg")
pandoralg.collections = [
        "MCParticle:MCParticle",
        "CalorimeterHit:ECALBarrel",
        "MCRecoCaloAssociation:RecoCaloAssociation_ECALBarrel"
        ]
pandoralg.WriteClusterCollection               = "PandoraClusters"
pandoralg.WriteReconstructedParticleCollection = "PandoraPFOs"
pandoralg.WriteVertexCollection                = "PandoraPFANewStartVertices"
pandoralg.AnaOutput = "AnaMatrix.root"
pandoralg.PandoraSettingsDefault_xml = "./Reconstruction/PFA/Pandora/PandoraSettingsDefault.xml"
pandoralg.TrackCollections      =  ["MarlinTrkTracks"]
pandoralg.ECalCaloHitCollections=  ["ECALBarrel", "ECALEndcap", "ECALOther"]
pandoralg.HCalCaloHitCollections=  ["HCALBarrel", "HCALEndcap", "HCALOther"]
pandoralg.LCalCaloHitCollections=  ["LCAL"]
pandoralg.LHCalCaloHitCollections= ["LHCAL"]
pandoralg.MuonCaloHitCollections=  ["MUON"]
pandoralg.MCParticleCollections =  ["MCParticle"]
pandoralg.RelCaloHitCollections =  ["RecoCaloAssociation_ECALBarrel"]
pandoralg.RelTrackCollections   =  ["MarlinTrkTracksMCTruthLink"]
pandoralg.KinkVertexCollections =  ["KinkVertices"]
pandoralg.ProngVertexCollections=  ["ProngVertices"]
pandoralg.SplitVertexCollections=  ["SplitVertices"]
pandoralg.V0VertexCollections   =  ["V0Vertices"]
pandoralg.ECalToMipCalibration  = 112 #1000MeV/8.918
pandoralg.HCalToMipCalibration  = 34.8
pandoralg.ECalMipThreshold      = 0.225# 8.918*0.225=2.00655
pandoralg.HCalMipThreshold      = 0.3
pandoralg.ECalToEMGeVCalibration= 1.# BGO, to be tuned
pandoralg.HCalToEMGeVCalibration= 1.007
pandoralg.ECalToHadGeVCalibrationBarrel= 1.12
pandoralg.ECalToHadGeVCalibrationEndCap= 1.12
```

```
-bash-4.2$ ls Reconstruction/PFA/Pandora/MatrixPandora/src/
CaloHitCreator.cpp   GeometryCreator.cpp   MCParticleCreator.cpp   PandoraMatrixAlg.cpp   PfoCreator.cpp   TrackCreator.cpp
```

PandoraMatrixAlg (Gaudi Alg)

input/output edm4hep data

Geometry service (Gear/DD4HEP)

Example

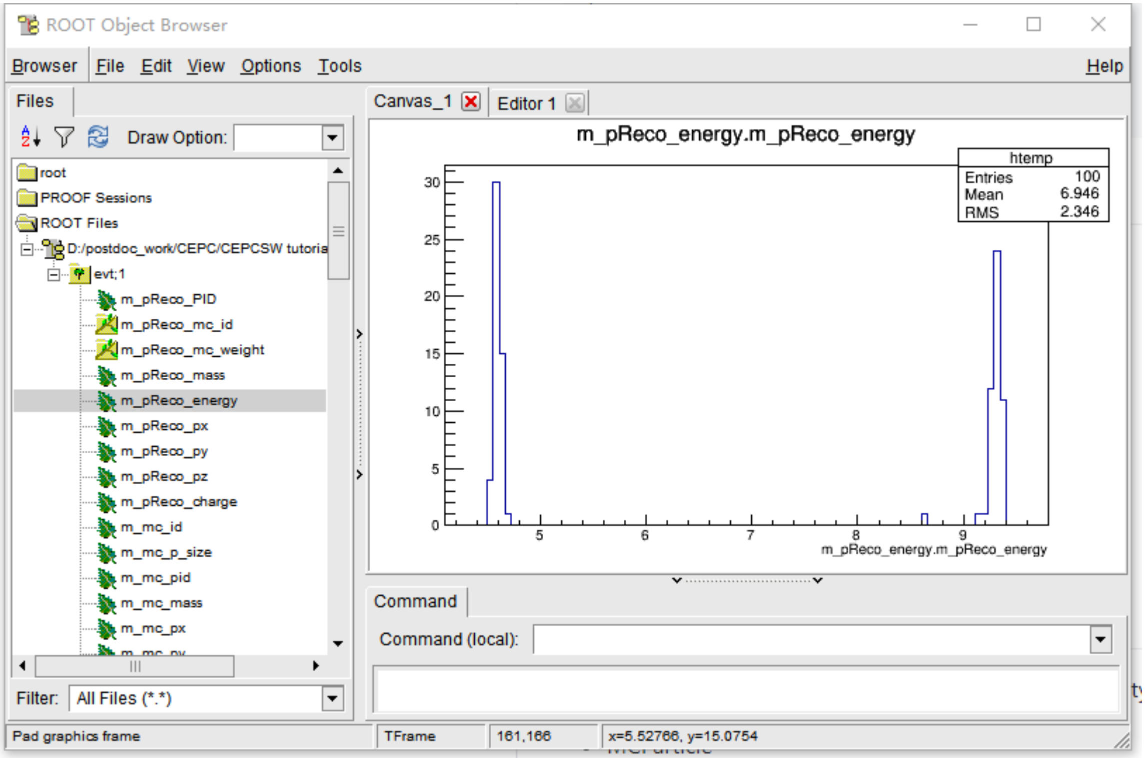

```
-bash-4.2$ ls Reconstruction/PFA/Pandora/MatrixPandora/src/
CaloHitCreator.cpp  GeometryCreator.cpp  MCParticleCreator.cpp  PandoraMatrixAlg.cpp  PfoCreator.cpp  TrackCreator.cpp
```

```
m_pPandora = new pandora::Pandora();
m_pMCParticleCreator = new MCParticleCreator(m_mcParticleCreatorSettings, m_pPandora);
m_pGeometryCreator = new GeometryCreator(m_geometryCreatorSettings, m_pPandora);
PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pGeometryCreator->CreateGeometry(svcloc));
m_pCaloHitCreator = new CaloHitCreator(m_caloHitCreatorSettings, m_pPandora, svcloc, 0);
m_pTrackCreator = new TrackCreator(m_trackCreatorSettings, m_pPandora, svcloc);
m_pPfoCreator = new PfoCreator(m_pfoCreatorSettings, m_pPandora);
PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, this->RegisterUserComponents());
PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::ReadSettings(*m_pPandora, m_settings.m_pandoraSettingsXmlFile)
```

```
StatusCode PandoraMatrixAlg::execute()
{

    try
    {
        std::cout<<"execute PandoraMatrixAlg"<<std::endl;

        updateMap();
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateMCParticles(*m_CollectionMaps));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pCaloHitCreator->CreateCaloHits(*m_CollectionMaps));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateCaloHitToMCParticleRelationships(*m_Collecti
onMaps, m_pCaloHitCreator->GetCalorimeterHitVector() ));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pTrackCreator->CreateTrackAssociations(*m_CollectionMaps));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pTrackCreator->CreateTracks(*m_CollectionMaps));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateTrackToMCParticleRelationships(*m_Collection
Maps, m_pTrackCreator->GetTrackVector() ));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::ProcessEvent(*m_pPandora));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pPfoCreator->CreateParticleFlowObjects(*m_CollectionMaps, m_ClusterCol
lection_w, m_ReconstructedParticleCollection_w, m_VertexCollection_w));

        StatusCode sc0 = CreateMCRecoParticleAssociation();
        StatusCode sc = Ana();

        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::Reset(*m_pPandora));
        this->Reset();
```

Example


```
git clone git@github.com:cepc/CEPCSW.git
cd CEPCSW
git checkout master
/cvmfs/container.ihep.ac.cn/bin/hep_container shell SL6
source setup.sh
./build.sh
./run.sh Examples/options/tut_detsim_pan_matrix.py
root -l AnaMatrix.root
```

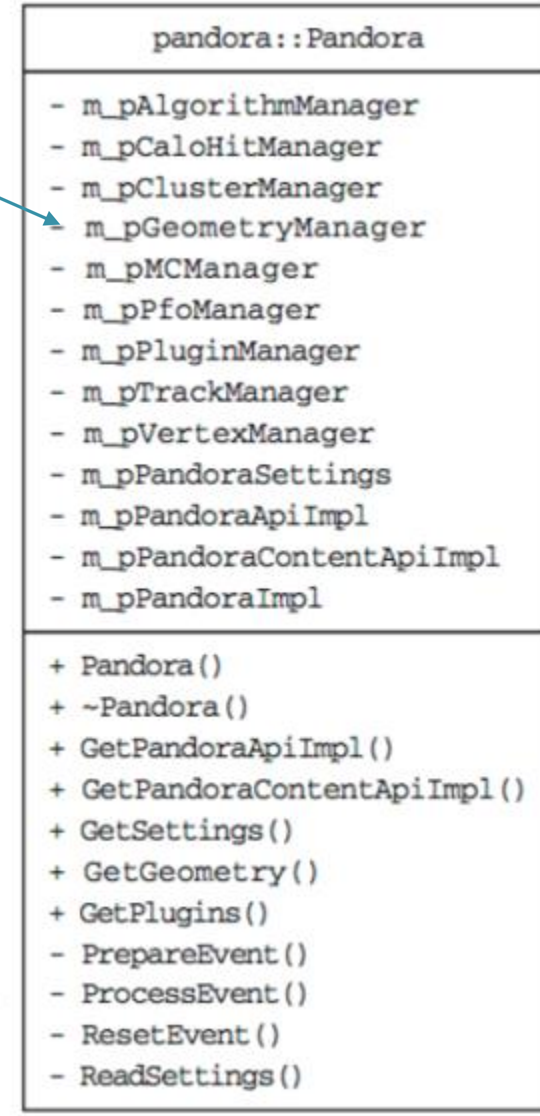


Enjoy yourself !

Back up

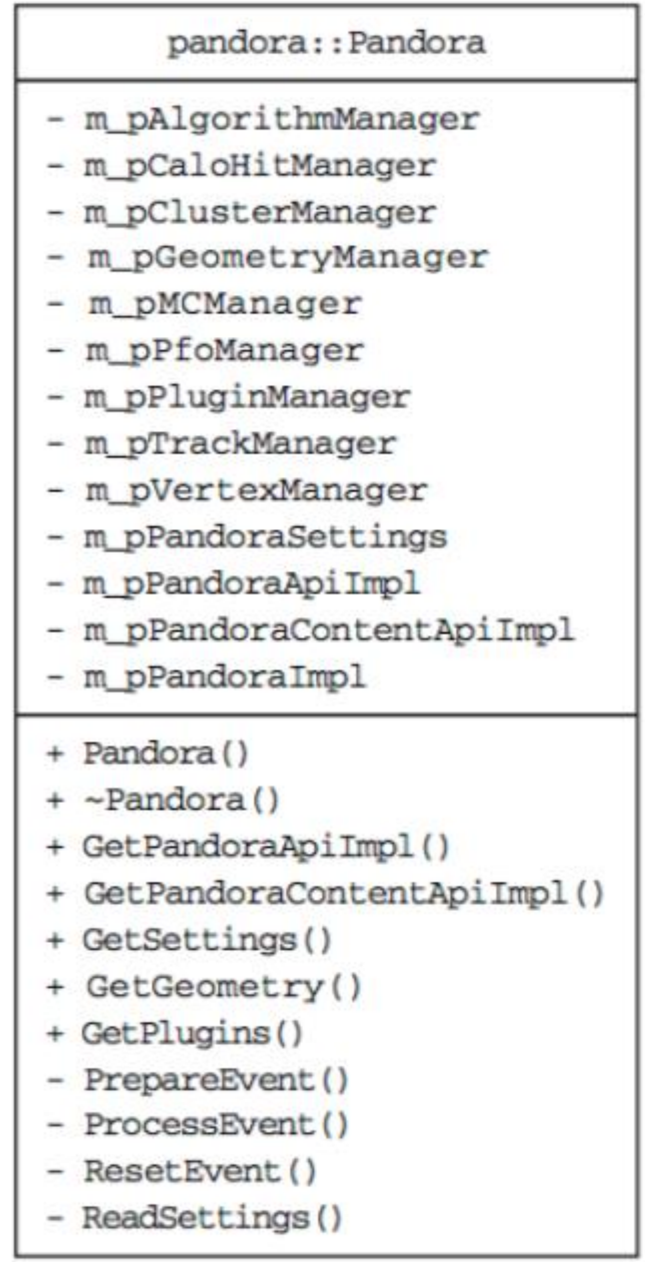
Geometry information in Pandora

- The geometry information is saved in pandora's geometry manager in client application once.
- It includes the sub-detector type (e.g. ECAL_BARREL, ECAL_ENDCAP), R, Z, layer information and so on.
- The algorithms will use PandoraContentApi to get the geometry manager of pandora and get the needed geometry information.



Managers

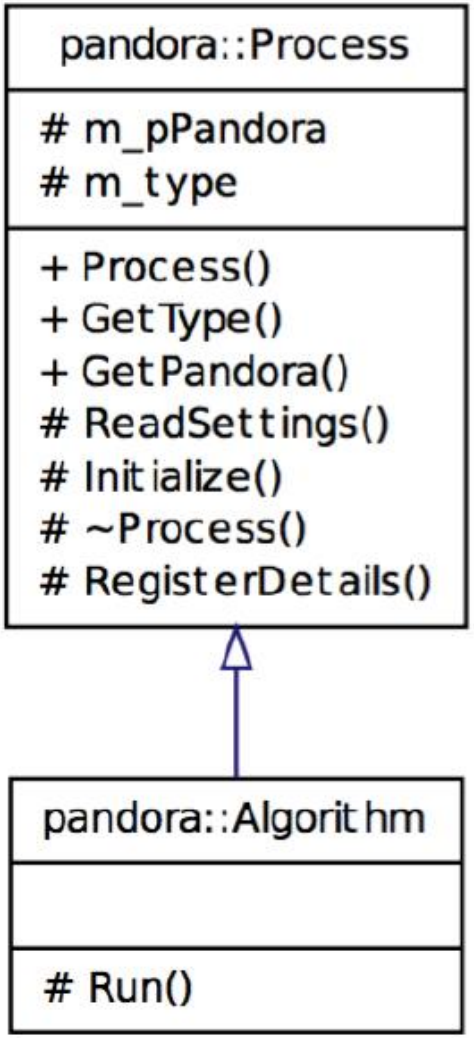
- **At very heart of Pandora design are the Managers, which own all instances of objects in Pandora EDM.**
  - The Managers are designed to provide a complete set of low-level object manipulation functions.
  - Algs request high-level services (e.g. merge two Clusters), which are then satisfied when the hidden implementation calls the low-level Manager functions in the correct order.
  - Approach helps ensure that implementation is extensible, easy to maintain and rather human-readable.
  - Key part of design is that algorithms can *only* access or
  - modify managed objects via the APIs, so Managers are able to perform memory-management.



A Pandora instance is simply a container of Manager instances and API implementation instances

Algorithms

- **Algs contain step-by-step instructions, using Pandora APIs to request object creation/modification services.**
  - Algs inherit from the Pandora Process abstract base class. Inherited functionality controls handshaking between Pandora instance and algorithm instance.
  - Process provides ability to receive a ReadSettings callback with an XML handle (tiny xml) from which configurable parameters can be extracted. Also an Initialize callback.
  - The Algorithm purely abstract base class provides the interface for the Run callback, which is called each event and is the entry point for all event processing.



- **Algorithm Factories registered (under a specific name), by the client app are extremely simple:**
  - Must allocate instance of derived algorithm type and return pointer to Algorithm base class.