Introduction to PFA and Pandora

FANGYI GUO

Particle Flow Algorithm

Motivation: give di-jet mass resolution similar to Gauge boson width(ILC original aim)

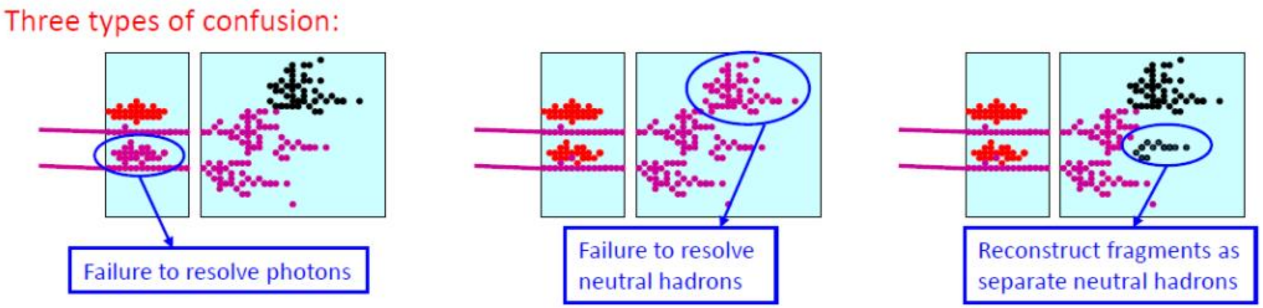
$$\sigma_m/m = 2.7\% \approx \Gamma_W/m_W \approx \Gamma_Z/m_Z$$

$$\frac{\sigma_E}{E} = \frac{\alpha}{\sqrt{\mathrm{E(GeV)}}} \oplus \beta$$

→ Stochastic term must be $\lesssim 30\%/\sqrt{\mathrm{E(GeV)}}$

→ $\sigma_E/E < 3.8\%$

New concept of jet reconstruction: use track+Ecal+Hcal to measure jet energy, to avoid confusion and increase jet energy resolution.



Particle Flow Algorithm

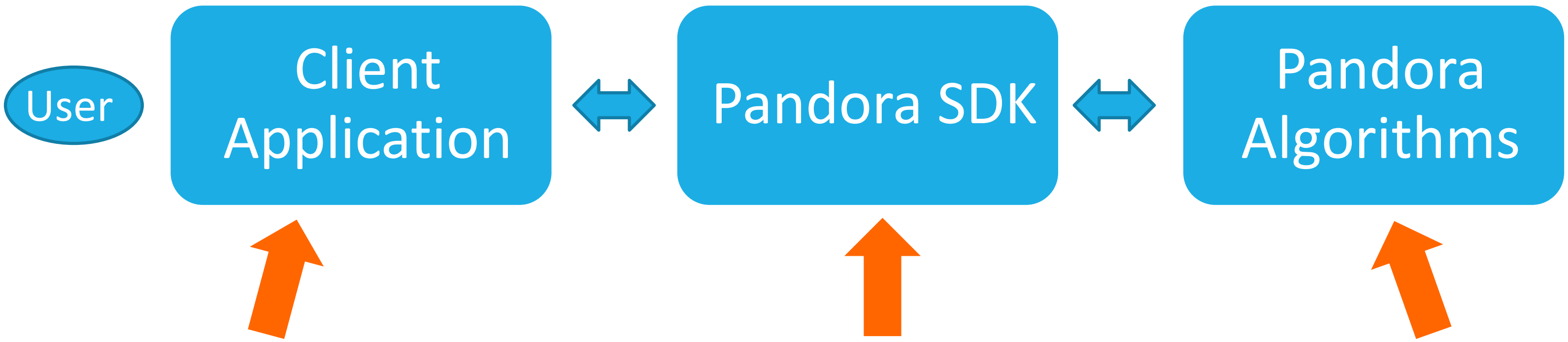
Component	Detector	Energy Fract.	Energy Res.	Jet Energy Res.
Charged Particles ($X^{\pm}$)	Tracker	$\sim 0.6\,E_j$	$10^{-4}\,E_{X^{\pm}}^2$	$< 3.6 \times 10^{-5}\,E_j^2$
Photons (γ)	ECAL	$\sim 0.3\,E_j$	$0.15\,\sqrt{E_{\gamma}}$	$0.08\,\sqrt{E_j}$
Neutral Hadrons (h^0)	HCAL	$\sim 0.1\,E_j$	$0.55\,\sqrt{E_{h^0}}$	$0.17\,\sqrt{E_j}$

Table 1: Contributions from the different particle components to the jet-energy resolution (all energies in GeV). The table lists the approximate fractions of charged particles, photons and neutral hadrons in a jet of energy, E_j, and the assumed single particle energy resolution.

Requirement for PFA:

- Hardware: high granularity calorimeter
- Software: pattern recognition algorithm. The confusion rather than calorimetric performance is the limiting factor in Particle Flow calorimetry.

PandoraPFA



MarlinPandora, XML setting files
- Provide input for Pandora
- Create required Pandora instances(e.g. Manager)
- Register the required algorithms.
- ……

PandoraManager
- Application Programming Interfaces(APIs)
- Event Data Model(EDM)
- Managers

Core algorithm for pattern recognition.

Event Data Model

Input object: building-blocks for pattern recognition, created by the client app.

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.

```
<processor name="MyMarlinPandora" type="PandoraPFANewProcessor">
  <parameter name="PandoraSettingsXmlFile" type="String">PandoraSettingsDefault.xml </parameter>
  <!-- Collection names -->
  <parameter name="TrackCollections"         type="StringVec">MarlinTrkTracks</parameter>
  <parameter name="ECalCaloHitCollections"   type="StringVec">ECALBarrel ECALEndcap ECALOther</parameter>
  <parameter name="HCalCaloHitCollections"   type="StringVec">HCALBarrel HCALEndcap HCALOther</parameter>
  <parameter name="LCalCaloHitCollections"   type="StringVec">LCAL</parameter>
  <parameter name="LHCalCaloHitCollections"  type="StringVec">LHCAL</parameter>
  <parameter name="MuonCaloHitCollections"   type="StringVec">MUON</parameter>
  <parameter name="MCParticleCollections"    type="StringVec">MCParticle</parameter>
```

Event Data Model

Algorithm object: higher-level structures created in order to solve pattern-recognition problems.

- **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.

```
<parameter name="KinkVertexCollections"   type="StringVec">KinkVertices</parameter>
<parameter name="ProngVertexCollections"  type="StringVec">ProngVertices</parameter>
<parameter name="SplitVertexCollections"  type="StringVec">SplitVertices</parameter>
<parameter name="V0VertexCollections"     type="StringVec">V0Vertices</parameter>
<parameter name="ClusterCollectionName"   type="String">   PandoraClusters</parameter>
<parameter name="PFOCollectionName"       type="String">   PandoraPFOs</parameter>
```

Client Application

- **A client application is responsible for controlling Pandora pattern recognition: it creates the Pandora instance(s) and uses the Pandora APIs to request services.**
- It registers algorithm factories, giving Pandora instances the ability to instantiate specific algorithms, and it provides the algorithm configuration via an XML file.
- Each event, it asks Pandora to create the input 'building blocks' (e.g. Hits and, optionally, MCParticles) to describe an event and it receives the final output Particles.

- To create an input building blocks must provide a list of simple parameters: energy, position, etc.
- Algorithms access information stored in building blocks but do not need to know how information was obtained.
- **Client application isolates Pandora algorithms from host framework.**

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
```

Client App MarlinPandora

```
CaloHitCreator.cc                  GeometryCreator.cc     PandoraPFANewProcessor.cc    SimpleBFieldCalculator.cc
ExternalClusteringAlgorithm.cc     MCParticleCreator.cc   PfoCreator.cc                TrackCreator.cc
```

```
void PandoraPFANewProcessor::processEvent(LCEvent *pLCEvent)
{
    static int eventCounter = 0;
    m_pLcioEvent = pLCEvent;

    try
    {
        streamlog_out(DEBUG5) << "PandoraPFANewProcessor, Run " << m_nRun << ", Event " << ++m_nEvent << std::endl;

        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateMCParticles(pLCEvent));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pTrackCreator->CreateTrackAssociations(pLCEvent));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pTrackCreator->CreateTracks(pLCEvent));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateTrackToMCParticleRelationship
s(pLCEvent));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pCaloHitCreator->CreateCaloHits(pLCEvent));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pMCParticleCreator->CreateCaloHitToMCParticleRelationsh
ips(pLCEvent));

        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, PandoraApi::ProcessEvent(*m_pPandora));
        PANDORA_THROW_RESULT_IF(pandora::STATUS_CODE_SUCCESS, !=, m_pPfoCreator->CreateParticleFlowObjects(pLCEvent));

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

Pandora SDK

Provide key services for pattern recognition algorithms, and algorithms can only access managed objects via Pandora APIs.

/PandoraSDK/include/

Api	Helpers	Managers	Objects	Pandora	Persistency	Utilities	Xml

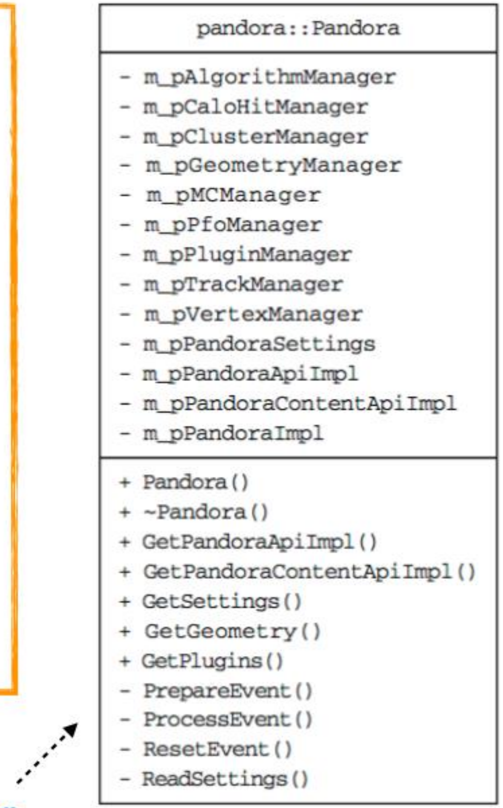
- **Algorithms can use the Pandora APIs to receive const references to the object lists from the Managers. Algorithms can access lists by name or ask for the current list.**

PandoraContentApi.h

```
/**
 *  @brief  Get the current list
 *
 *  @param  algorithm the algorithm calling this function
 *  @param  pT to receive the address of the current list
 *  @param  listName to receive the current list name
 */
template <typename T>
static pandora::StatusCode GetCurrentList(const pandora::Algorithm &algorithm, const T *&pT, std::string &listName);
```

Pandora SDK

- **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

- Managers hold address of associated Pandora instance and record details of all algs running: e.g. current list name when alg began, names of any temporary lists created.

Pandora SDK

```xml
<!-- Pandora settings xml file -->

<pandora>
    <!-- HELPER CLASS SETTINGS -->
    <IsMonitoringEnabled> true </IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo> false </ShouldDisplayAlgorithmInfo>

    <!-- Plugin helper functions -->
    <HadronicEnergyCorrectionFunctions> CleanClusters ScaleHotHadrons </HadronicEnergyCorrectionFunctions>
    <EmShowerFastFunction> FineGranularityEmShowerId </EmShowerFastFunction>
    <PhotonFastFunction> FineGranularityPhotonId </PhotonFastFunction>
    <ElectronFastFunction> FineGranularityElectronId </ElectronFastFunction>
    <MuonFastFunction> FineGranularityMuonId </MuonFastFunction>

    <!-- CaloHit helper settings -->
    <CaloHitHelper>
        <ShouldCalculateDensityWeight>false</ShouldCalculateDensityWeight>
        <ShouldCalculateSurroundingEnergy>false</ShouldCalculateSurroundingEnergy>
    </CaloHitHelper>

    <!-- PANDORA ALGORITHM SETTINGS -->

    <!-- Standalone photon clustering -->
    <algorithm type = "PhotonReconstruction">
        <algorithm type = "ConeClustering" description = "PhotonClusterFormation">
            <ClusterSeedStrategy>0</ClusterSeedStrategy>
            <ShouldUseTrackSeed>false</ShouldUseTrackSeed>
            <ShouldUseOnlyECalHits>true</ShouldUseOnlyECalHits>
            <ConeApproachMaxSeparation>250.</ConeApproachMaxSeparation>
        </algorithm>
        <ClusterListName>PhotonClusters</ClusterListName>
        <ShouldMakePdfHistograms>false</ShouldMakePdfHistograms>
        <NEnergyBins>9</NEnergyBins>
        <HistogramFile>PandoraLikelihoodData9EBin.xml</HistogramFile>
    </algorithm>
```

Algorithm

Eight main steps:

- Track Selection/ Topology:
  - track topologies such as kinks and decays of neutral particles in the detector volume are identified
- Calorimeter hit selection and ordering:
  - isolated hits, defined by proximity to others, are removed
  - remaining hits are ordered into pseudo-layers
- Clustering
  - Cone-based forward projective method. Seeding clusters using projections of rec. tracks onto the front face of ECAL
  - Photon clustering:
    - Consider ECAL hits to identify energy deposits from photons
    - Clustering on remaining hits
- Topological Cluster Merging
  - Apply only to clusters which have not been indentified as photon
- Statistical reclustering
  - For jet with E>50 GeV, attempts are made to recluser hits by applying clustering with different parameters until cluster splits, until $E_{clus} \sim E_{track}$
- Photon recovery and identification
  - More sophisticated photon id applied, improving photon tagging.
  - Recover cases where a primary photon is merged with had. shower
- Fragment removal: identify fragments of charged particle hadronic shower
- Formation of Particle Flow Objects (PFOs) – reconstructed particles

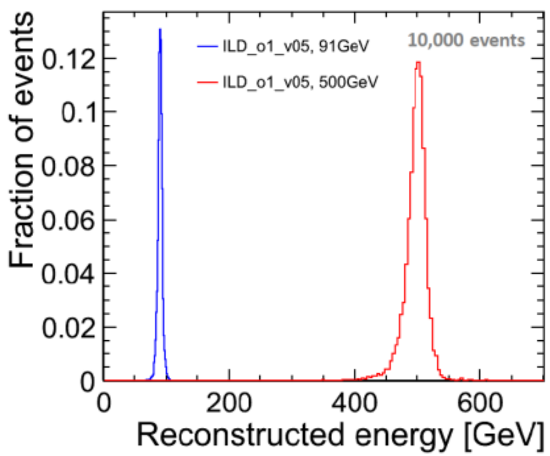
Pandora Performance (ILD)

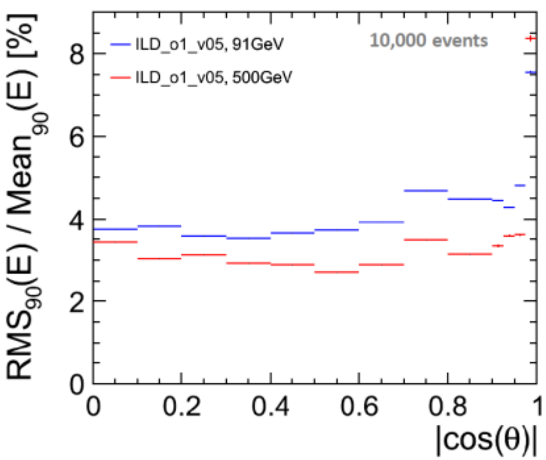
Jet energy resolution (for ILD)

- Study of jet energy resolution for ILD_o2_v05 using Z→qqbar events with Z decaying at rest (Z→uds)
- at CM energies: 91, 500 GeV

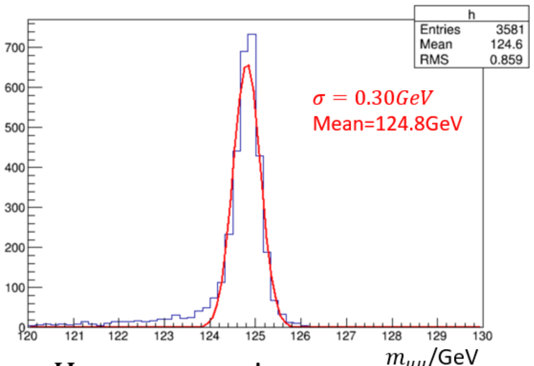
E_{jj} (= 2 * E_j)	91GeV	500GeV
ILD_o1_v05 v01-15-02-pre05	σ_E/E: 3.65 ± 0.05 mean: 90.49 GeV	σ_E/E: 2.97 ± 0.04 mean: 500.57 GeV

J. Marshall, ILD meeting 26/9/2012
(calibration constants were optimised for JER)

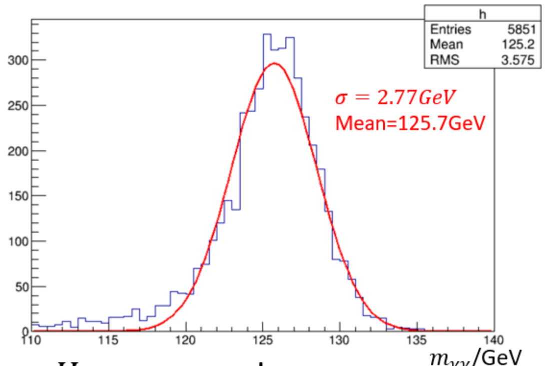




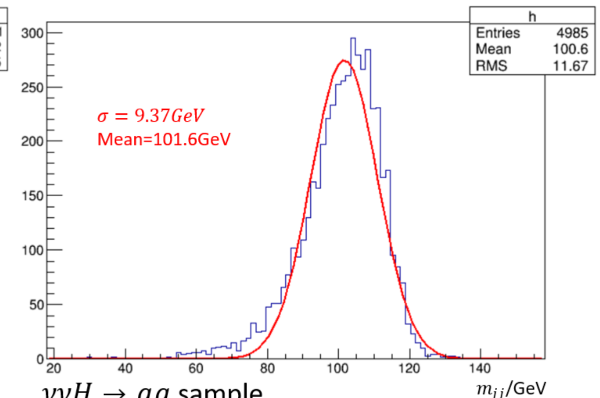
Pandora Performance (CEPC)



$\nu\nu H \rightarrow \mu\mu$ sample,
$\mu\mu$ invMass, fit with Gaus



$\nu\nu H \rightarrow \gamma\gamma$ sample,
$\gamma\gamma$ invMass, fit with Gaus

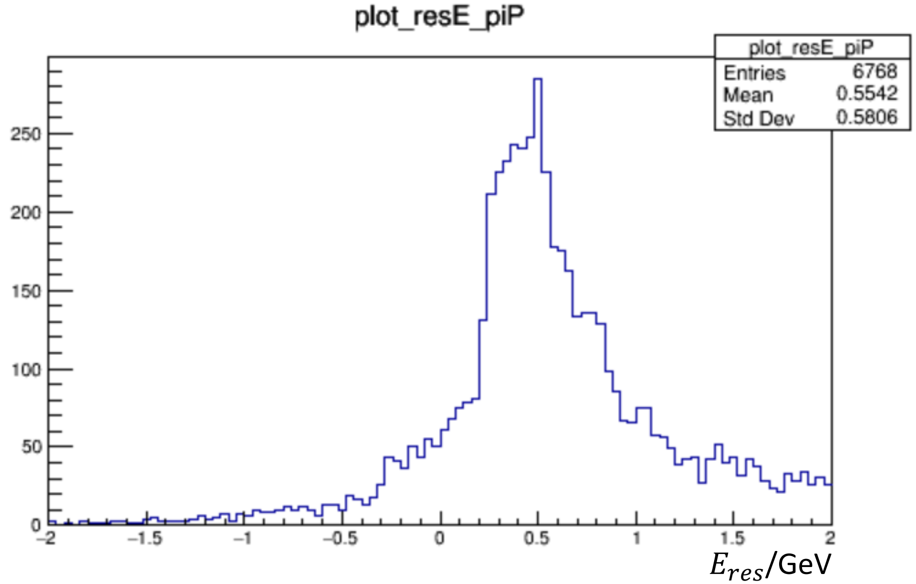


$\nu\nu H \rightarrow gg$ sample,
di-jet invMass, fit with Gaus

Could run at Marlin framework
No problem in charged lepton and photon reconstruction.
Some energy shift in jet energy reconstruction.

Pandora Performance (CEPC)

Charged Hadron energy deposition in calorimeter ($\nu\nu H \to gg$ sample)



$E_{res} = E_{reco} - E_{cluster}$. For Charged hadron $E_{reco} \approx E_{trk}$, so this means some mismatch in track-cluster association.

Summary

Pandora: an individual package for PFA.

- 3 part: client app, SDK, core algorithm.
- For new framework: client app needs new development, might be a big project.
- Performance: need to do some study in parameters. Undergoing.