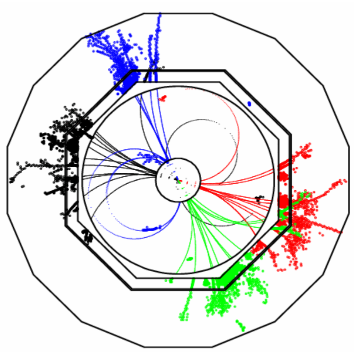


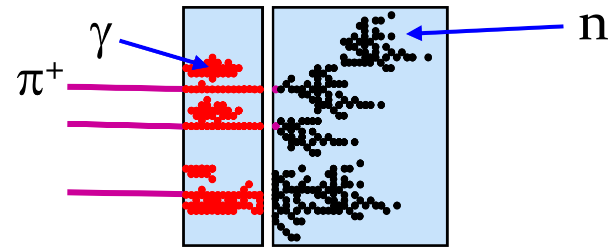
# Redesign of Pandora PFA

John Marshall,
University of Cambridge

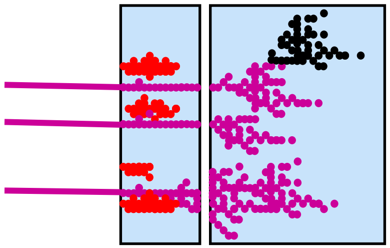
ILD Workshop, January 27 2010

# Pandora PFA

- In a typical jet:
  - 60% of jet energy is in the form of charged hadrons
  - 30% is in photons (mainly from $\pi^0 \rightarrow \gamma\gamma$)
  - 10% is in neutral hadrons (mainly $n$ and $K_L$ )
- Particle flow calorimetry aims to improve jet energy resolution by:
  - Measuring charged particles in detector tracker (essentially perfectly)
  - Measuring photon energies in the ECAL $\sigma_E/E < 20\% / \sqrt{E(GeV)}$,
  - Only measuring neutral hadron energies in the HCAL, largely avoiding the intrinsically poor HCAL resolution.




$$E_{JET} = E_{ECAL} + E_{HCAL}$$



$$E_{JET} = E_{TRACK} + E_{\gamma} + E_{n}$$

| $E_{JET}$ | $\sigma_E/E$ ($rms_{90}$) |
|---|---|
| 45 GeV | 3.7 % |
| 100 GeV | 2.9 % |
| 180 GeV | 3.0 % |
| 250 GeV | 3.1 % |

- The Pandora Particle Flow Algorithm:
  - Initially developed for the ILD detector concept.
  - The most mature PFA, giving the best performance.
  - Its algorithms are now well tested and understood.
  - Fully documented, NIMA 611 (2009) 25-40
  - Meets ILC jet energy goal of ~3.5 % at all relevant jet energies.

# Pandora Redesign

- Whilst Pandora works well, current code has reached a point where it is extremely difficult to extend. It is not flexible enough to try out new ideas and improvements...

- ILD Letter of Intent version of Pandora has been frozen and a new version is being written from scratch.

- This is much more than just a re-implementation; Pandora is now a framework for running decoupled particle flow algorithms:
  - Increased flexibility, designed to make it easy to try out new ideas
  - Independent of any specific software framework and any specific detector details
  - Properly designed code, taking findings from previous PFAs into account, makes it easier to maintain
  - Easier for other people to get involved; simple for users to create and run their own algorithms
  - Pandora framework helps separate physics in particle flow algorithms from C++ memory management

- The new Pandora is a separate library, with no dependencies. A user application, in any framework, accesses the library via a simple C++ API (application programming interface).

Will now give a brief overview of new Pandora framework and summarise current status...

# New Pandora Structure

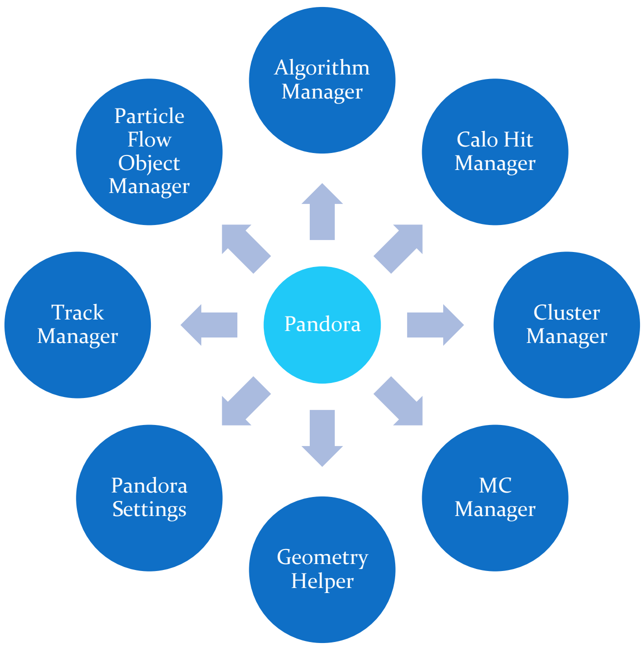
## User Application:

- Create Calo Hits
- Create Tracks
- Create MC Particles
- Specify Geometry
- Register user algorithm

- Get Particle Flow Objects

Pandora API

## Pandora Framework, can treat as "black box":




Pandora Content API

## Pandora Algorithms:

- Clustering Algorithm
- Topological Associations Algorithm
- Iterative Reclustering Algorithm
- Photon ID Algorithm
- Fragment ID Algorithm
- PFO construction Algorithm

# Pandora API

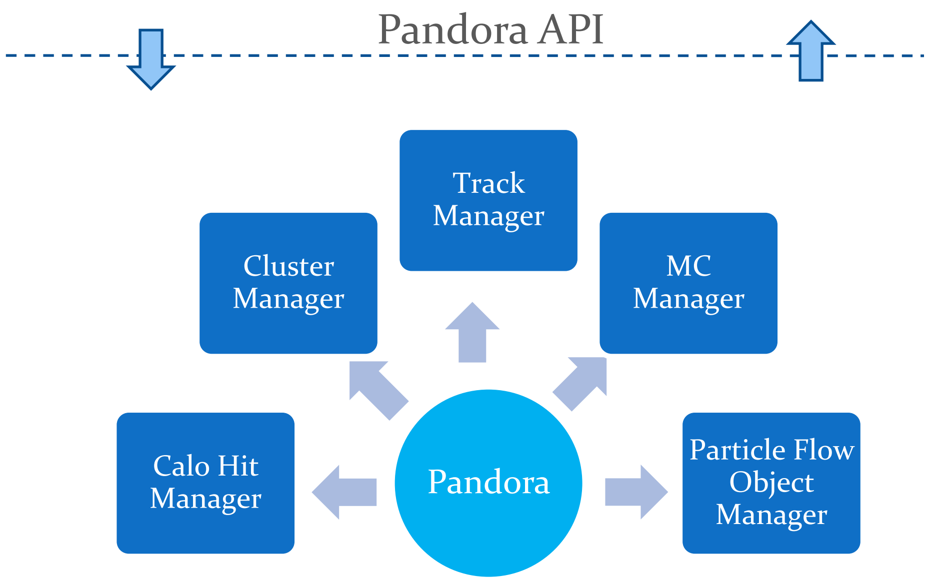
- To run Pandora, a user needs to write a small application in their chosen software framework.
- This application uses the Pandora API to supply Pandora with details of the detector geometry and of the calo hits and tracks in each event.
- Pandora then builds its own simple objects.

- Construction of these objects is simple; the user makes a Parameters class, fills the member variables and then calls the API Create function.
- Example member variables for a track:
  d0, z0, track state at start, track state at ECal, etc.
- All member variables must be specified, or an exception will be thrown when Create is called.

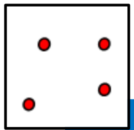
- The user can provide this information in any order, then call the API ProcessEvent function.
- Finally, user calls the API GetParticleFlowObjects function.

User Application, e.g. ILDPandora

```
PandoraApi::Track::Parameters parameters;
parameters.m_d0 = ...;
 ...
PandoraApi::Track::Create(pandora, parameters);
```

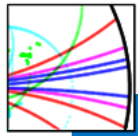


# Pandora Objects



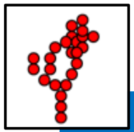
## Calo Hit

- Position + normal vectors
- Calorimeter cell size
- Absorber material in front of cell
- Time of first energy deposition
- Calibrated energy (mip equivalent, EM, Had)
- Layer + pseudolayer
- Hit type + detector region
- Density weight
- Surrounding energy
- IsDigital, IsIsolated + IsPossibleMip flags
- Associated MC particle
- Associated user object



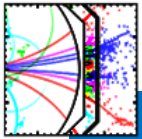
## Track

- 2D impact parameters
- Momentum at d.c.a
- Particle mass
- Charge sign
- Start track state
- End track state
- ECal track state
- ReachesECal flag
- List of track state projections to calorimeter surfaces
- Associated cluster
- Associated MC particle
- Associated user object



## Cluster

- List of constituent calo hits, ordered by pseudolayer
- Mip fraction
- EM energy measure
- Had energy measure
- Initial direction
- Current direction
- Result of linear fit to all hits in cluster
- Energy-weighted centroid
- ShowerMax layer
- List of associated tracks



## Particle Flow Object

- PDG Code
- Charge sign
- Mass
- Energy
- Momentum
- List of tracks
- List of clusters

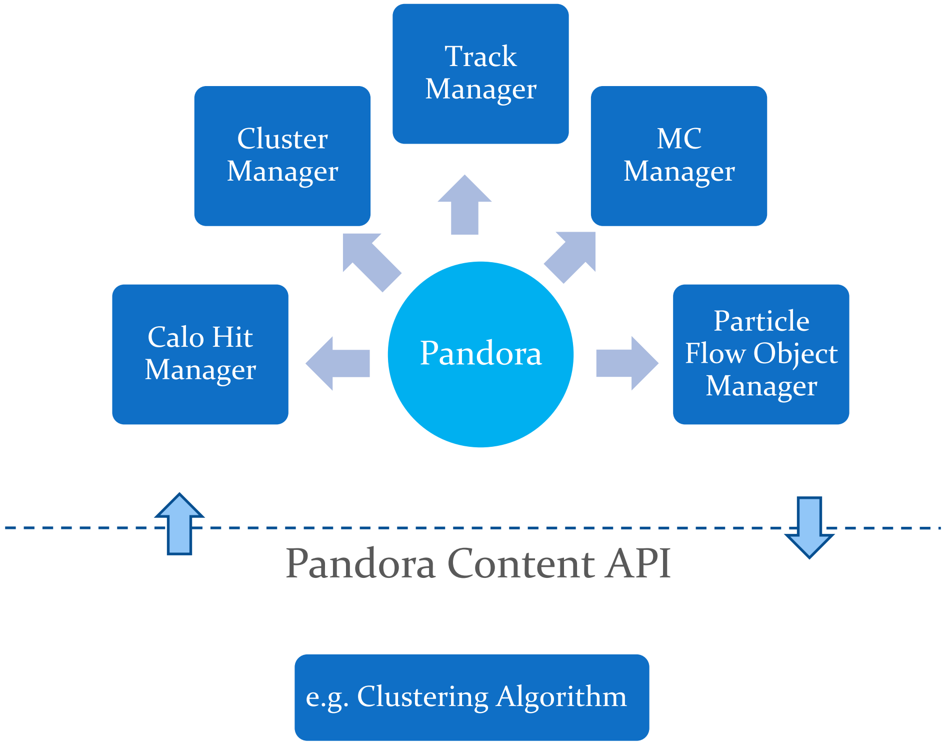
Mixture of properties specified by user and value-added properties, but all simple and well defined physics quantities for use in particle flow algorithms.

# Pandora Managers

- Pandora Managers are designed to store named lists of their respective objects.
- These objects can be accessed by the Pandora Algorithms, which perform the reconstruction.
- The algorithms interact with the Managers in a controlled way, via PandoraContentAPI, and the Managers perform the memory management.

- At any instant each Manager has a "current" list, which can be accessed by an algorithm.
- Parent algorithms can manipulate the current list in order to control scope and behaviour of daughter algorithms.

- The Managers store information about currently running algorithms so they can keep track of lists.
- Algorithms can use the PandoraContentAPI to modify lists and/or save new lists.




Algorithms can use the API without worrying about how the managers work – separation of physics and C++ memory management!

# Pandora Algorithms

- In the new Pandora framework, the algorithms contain almost exclusively physics-driven code, alongside the following typical usages of the PandoraContentAPI:
    - Create new clusters and particle flow objects
    - Modify clusters, by adding hits, merging or deleting
    - Access the current lists of Pandora objects
    - Save new lists of clusters, calo hits or tracks
    - Run a daughter algorithm, etc...

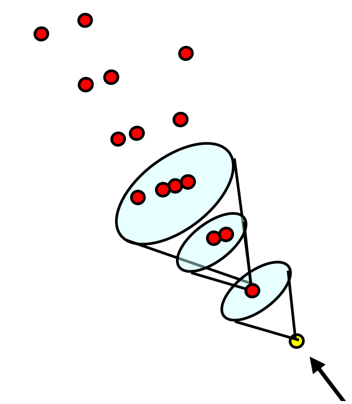
- Static helper functions are provided to perform tasks that are useful to multiple algorithms, such as functions to evaluate the overlap between two clusters or to perform a linear fit to (layers of) a cluster.

- The Pandora algorithms are configured via xml and can be swapped in/out without recompiling. The algorithms required to reproduce old Pandora performance are:
    - Clustering
    - Topological associations
    - Photon Id
    - Fragment Id
    - Iterative reclustering
    - Particle flow object formation

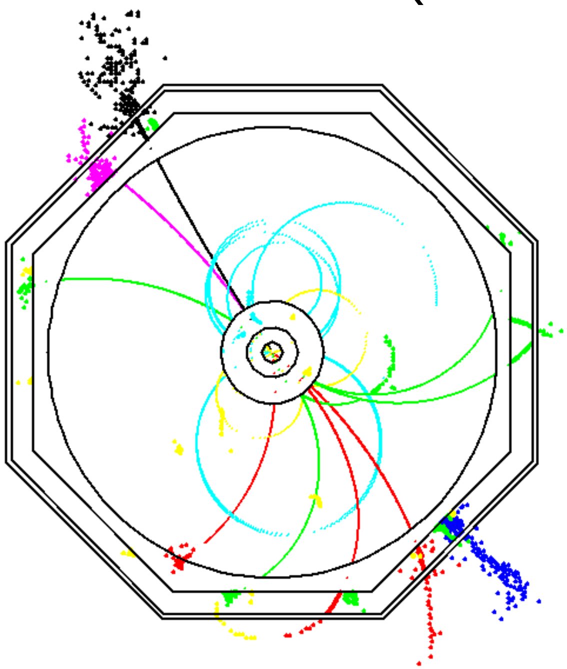
# Algorithm Status

## Clustering

- The main Pandora clustering algorithm is a cone-based forward projective method.
- Working from innermost to outermost pseudolayer, the algorithm either adds hits to existing clusters or uses them to seed new clusters.



- This algorithm has now been fully implemented in the new framework and tested extensively; the new code exactly reproduces the old Pandora clusters.



- This re-implementation was an opportunity to 'clean up' an algorithm that had changed many times during development; now more efficient.
- Code is clean and readable and have now fully separated the framework/objects from the actual algorithm.
- Configuration options have been tweaked to identify independent and physically motivated parameters.

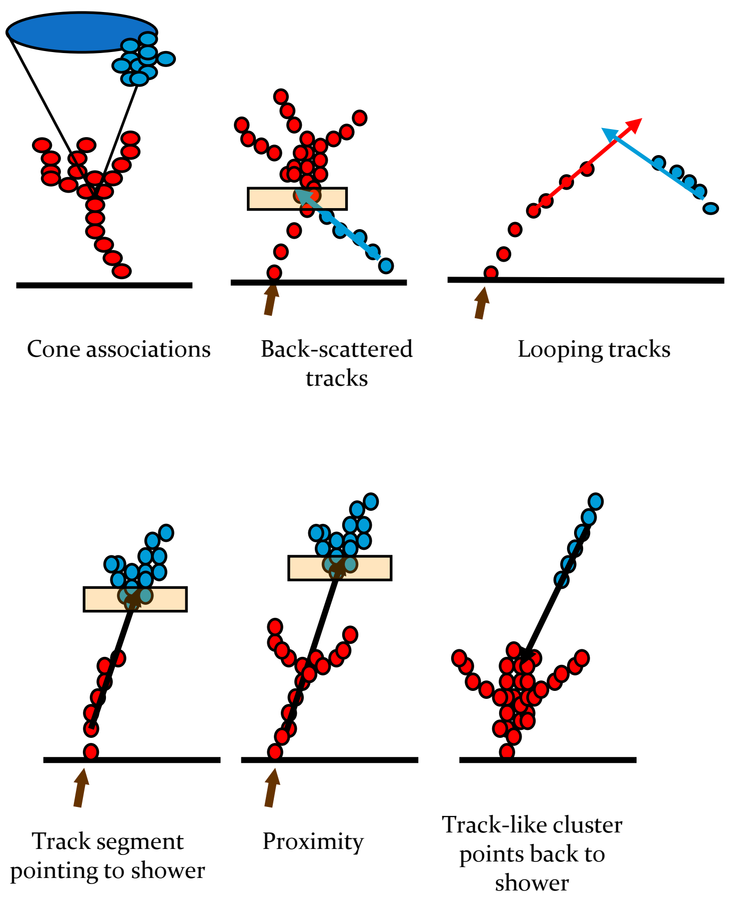
# Algorithm Status

## Topological Associations

- The approach in Pandora is to err on side of splitting up true clusters, then merge the clusters following a number of topological rules.
- Each of these rules have now been implemented via algorithms in the new Pandora framework.
- The algorithms essentially compare pairs of clusters, applying a series of cuts to identify whether the clusters should be merged.

- Many of the quantities, upon which cuts are placed, are useful properties characterising cluster interactions. As such, they are calculated by static helper functions.

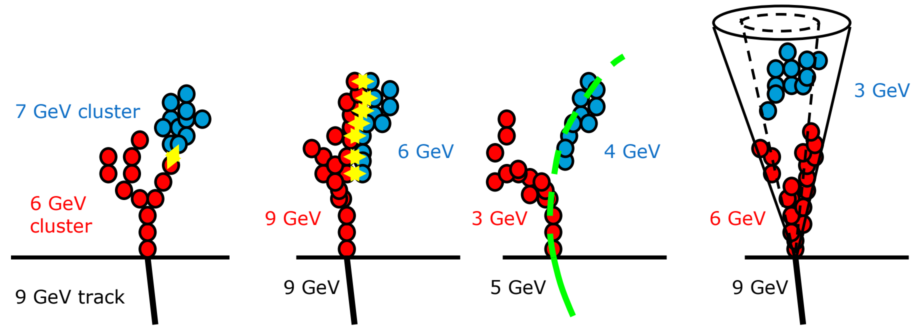
- The topological algorithms essentially reproduce the cluster associations made by old Pandora. However, have identified potential improvements during rewrite.
- Need to investigate and implement these improvements.



# Algorithm Status

## Fragment ID

- Neutral clusters that are fragments of charged particle hadronic showers are identified through examination of cluster contact and proximity.
- Fragment Id helper functions, which quantify the cluster contact, have been fully implemented.
- Fragment Id algorithm itself is almost complete, just needs another few days of development.

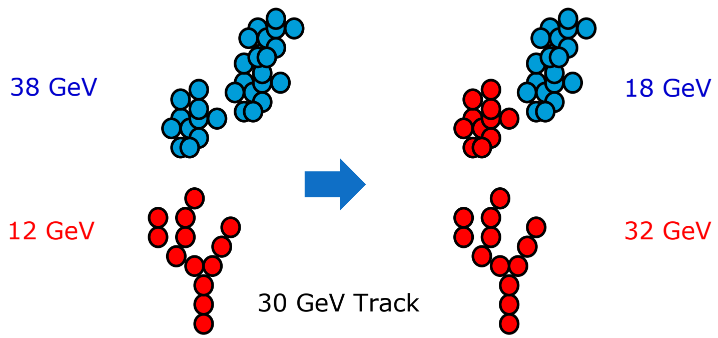



Distance of closest approach

Layers in close contact

Distance to track extrap.

Fraction of energy in cone



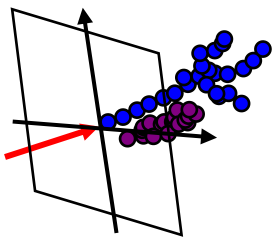

## Iterative reclustering

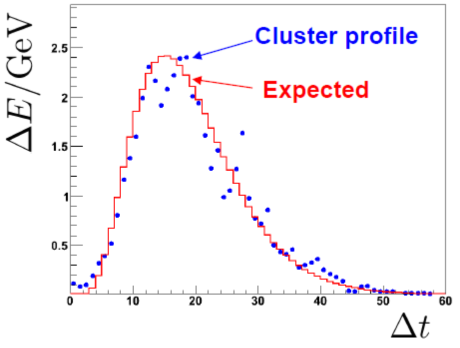
- Clusters that have been incorrectly merged together are identified via comparison of cluster energy and associated track momentum.
- Attempts are made to redistribute the hits by using different clustering parameters or algorithms (the new framework was specifically designed to make this easy).
- Have now implemented most of the algorithms that “steer” the reclustering. There are opportunities here to tidy these steering algorithms to clearly define role of each.

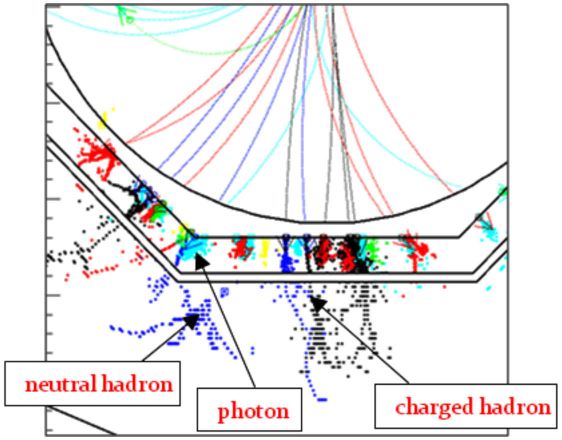
# Algorithm Status

## Photon ID

- Peter Speckmayer has been working on a photon id algorithm.
- This takes the output from the clustering algorithm, applies a shower-profile based selection and saves the photon clusters as a separate named cluster list.
- The removal of these clusters allows for improved identification of hadronic showers when the clustering algorithm is called again.






## PFO formation

- Final step is to identify the particle flow objects from the tracks and clusters and to write them out for analysis.
- A simple PFO formation algorithm is now complete.
- However, old Pandora had a more sophisticated treatment, with improved identification of track-cluster associations, and this needs to be implemented.

# Summary

- The new Pandora framework is complete and has been rigorously tested. It is stable and has been unchanged for several months now.
- The focus is now on re-implementing the original Pandora algorithms:
  - Clustering algorithm, topological association algorithms completed,
  - Significant progress made with fragment id, reclustering and photon id algorithms.
- Should also mention progress with the applications that use the Pandora library:
  - ILD Pandora application completed (our default test application),
  - Norman Graf has recently started a SlicPandora application,
  - Both applications are Marlin Processors that use the PandoraAPI to access the Pandora library.
- Aim to have a full re-implementation of original Pandora by LCWS10 in Beijing. Then have many ideas for moving forwards and improving Pandora within the new framework:
  - New clustering algorithms,
  - Long list of possible topological association improvements, etc…
  - Very easy for other people to get involved and write new algorithms or contribute ideas.