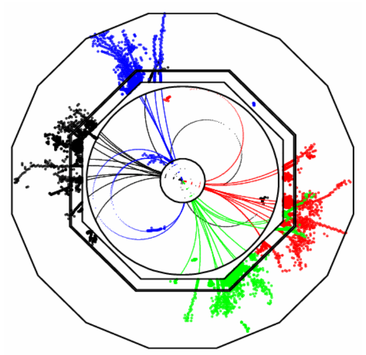


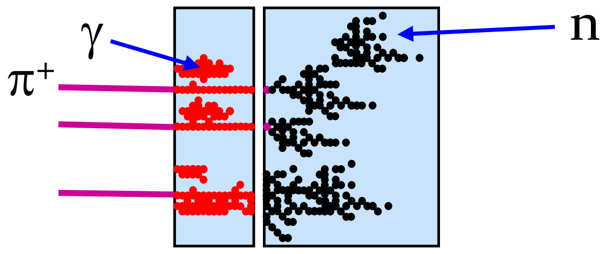
Redesign of Pandora PFA

John Marshall,
University of Cambridge

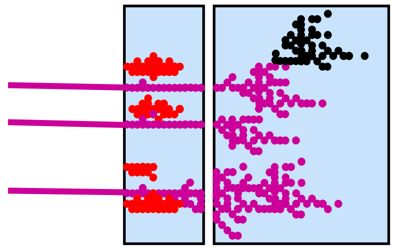
CLIC'09 Workshop, October 15 2009

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 $\sigma_E/E < 20\% / \sqrt{E(\mathrm{GeV})}$,
  - Only measuring neutral hadron energies in the HCAL, largely avoiding the intrinsically poor HCAL resolution.



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



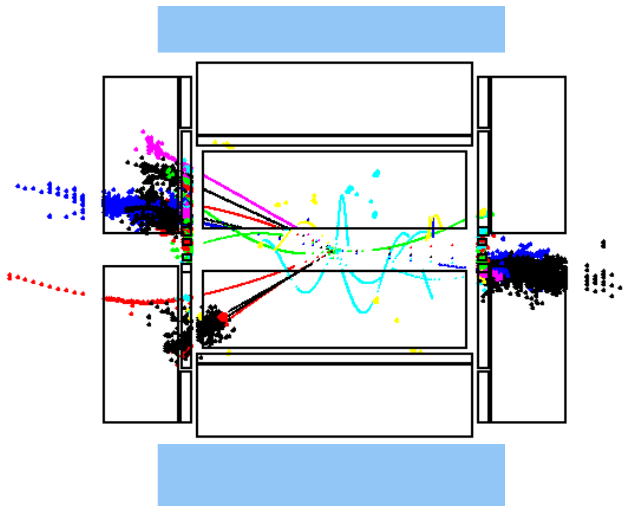
$$E_{JET} = E_{TRACK} + E_{\gamma} + E_{n}$$

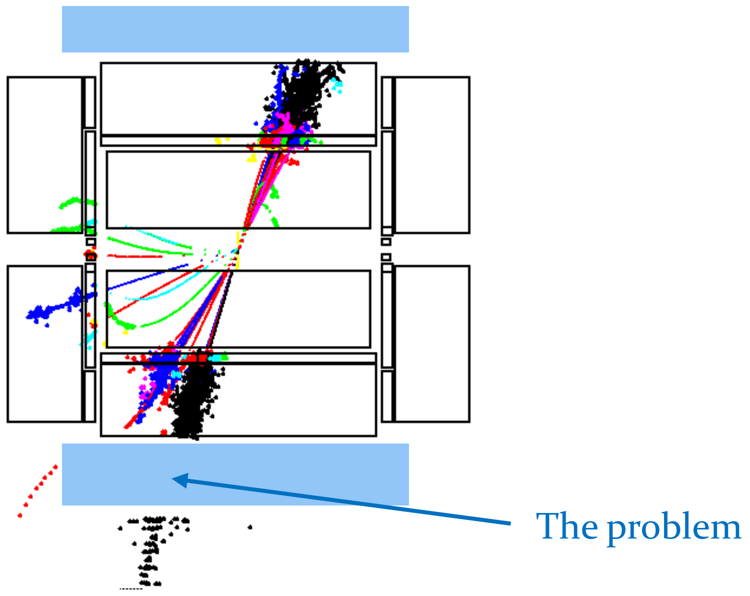
E_{JET}	σ_E/E (rms$_{90}$)
45 GeV	3.7 %
100 GeV	2.9 %
180 GeV	3.0 %
250 GeV	3.1 %

- 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, paper accepted by NIMA.
- ILD / PandoraPFA meets the ILC jet energy goal of ~3.5 % at all relevant jet energies. However, there are problems at CLIC-like energies...

Leakage

- For high energy jets, non-containment of showers is significant. This is a major issue at CLIC energies.
- Pandora PFA uses MUON chamber information to estimate leakage and energy deposited in coil:
  - Simple standalone clustering in MUON chambers.
  - Fairly simple matching to CALO clusters, with energy/momentum veto.
  - Simple energy estimator (digital), plus some estimate for loss in the coil.



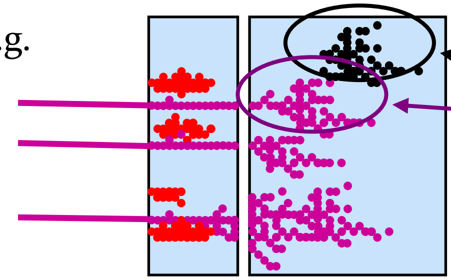


- Results for this simple treatment are encouraging, but effectiveness is limited by thick solenoid.
- Room for improvement, but difficult to address problem in current code.
- Need to make Pandora more flexible, with ability to easily swap different algorithms in/out and try new ideas.

Confusion

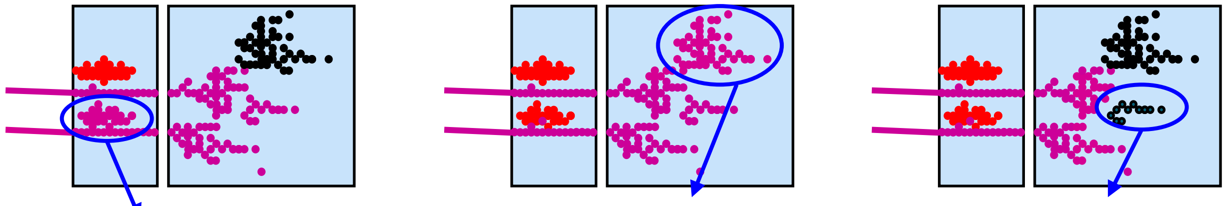
- At high jet energies, PFA performance degrades due to increasing overlap between hadronic showers from different particles:

e.g.



If these hits are clustered together with these, lose energy deposit from this neutral hadron (now part of track particle) and ruin jet energy measurement.

Three types of Confusion:



Failure to resolve photon

Failure to resolve neutral hadron

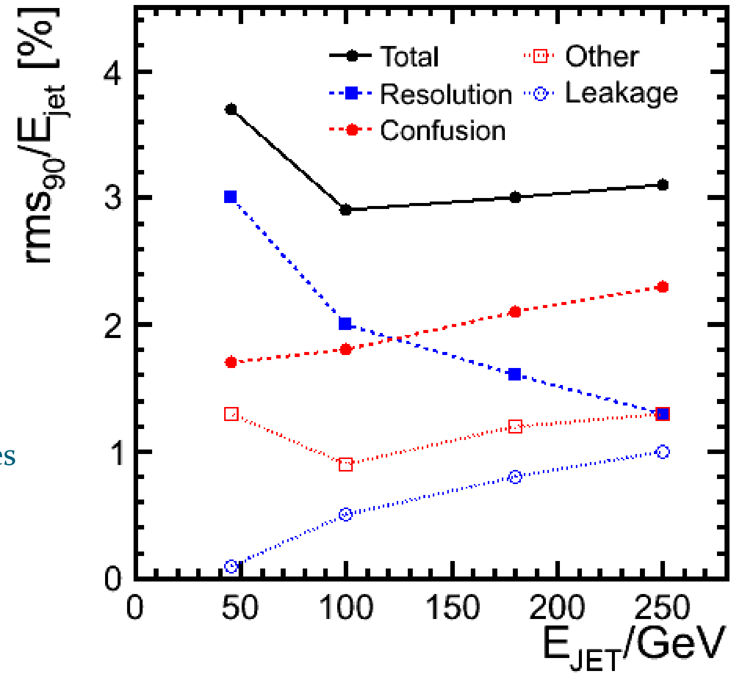
Reconstruct fragment as separate neutral hadron

- Pandora PFA addresses this problem with statistical reclustering:
  - Clusters likely to have been created by merging showers from more than one particle are identified on the basis of compatibility between the cluster energy and the associated track momentum.
  - Attempts are made to redistribute the hits by reapplying the clustering algorithm with different parameters.
- With more flexible code, and ability to run entirely different clustering algorithms this could become more powerful still.

Resolution

- It is informative to look at the contributions to the jet energy resolution as a function of the jet energy:
  - The impact of leakage can be investigated by varying the depth of the HCAL.
  - The effects of confusion can be estimated by using MC information to "cheat" various aspects of particle flow.

- Tests using ILD samples indicate that:
  - For low energy jets: Calorimetric energy resolution dominates
  - For high energy jets: Confusion term dominates
  - For very high energy jets: Leakage proves important

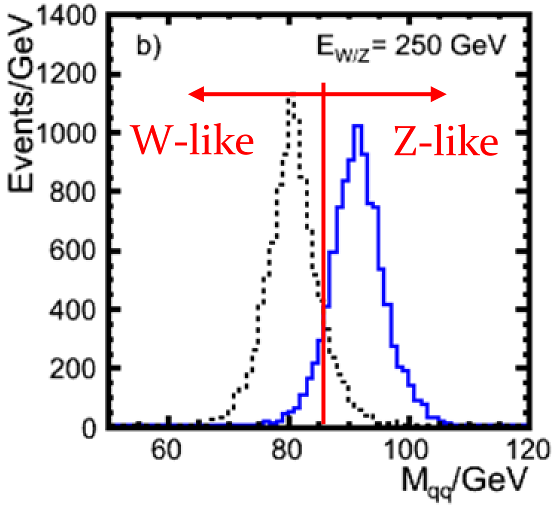


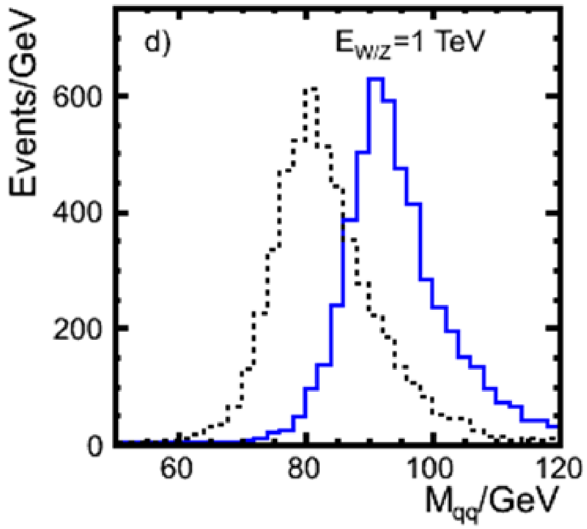
250 GeV jets: Total resolution	**3.1 %**
Confusion	2.3 %
i) Photons	1.3 %
ii) Neutral hadrons	1.8 %
iii) Charged hadrons	0.2 %

- At highest energy considered in this study (250GeV), neutral hadron confusion is the largest contribution to the jet energy resolution.

Current Performance

- The particle flow reconstruction of boosted gauge bosons has been investigated using ILD MC samples of $ZZ \rightarrow d\bar{d}\nu\bar{\nu}$ and $W^+W^- \rightarrow u\bar{d}\mu^-\nu_\mu$ events at different $\sqrt{s}$ values:





$E_{W/Z}$	$rms_{90}(m)$	σ_m/m	W/Z sep	Efficiency of W/Z cut
125 GeV	2.8 GeV	2.9 %	2.7 σ	91 %
250 GeV	3.0 GeV	3.5 %	2.5 σ	89 %
500 GeV	3.9 GeV	5.1 %	2.1 σ	84 %
1000 GeV	6.4 GeV	7.0 %	1.5 σ	78 %

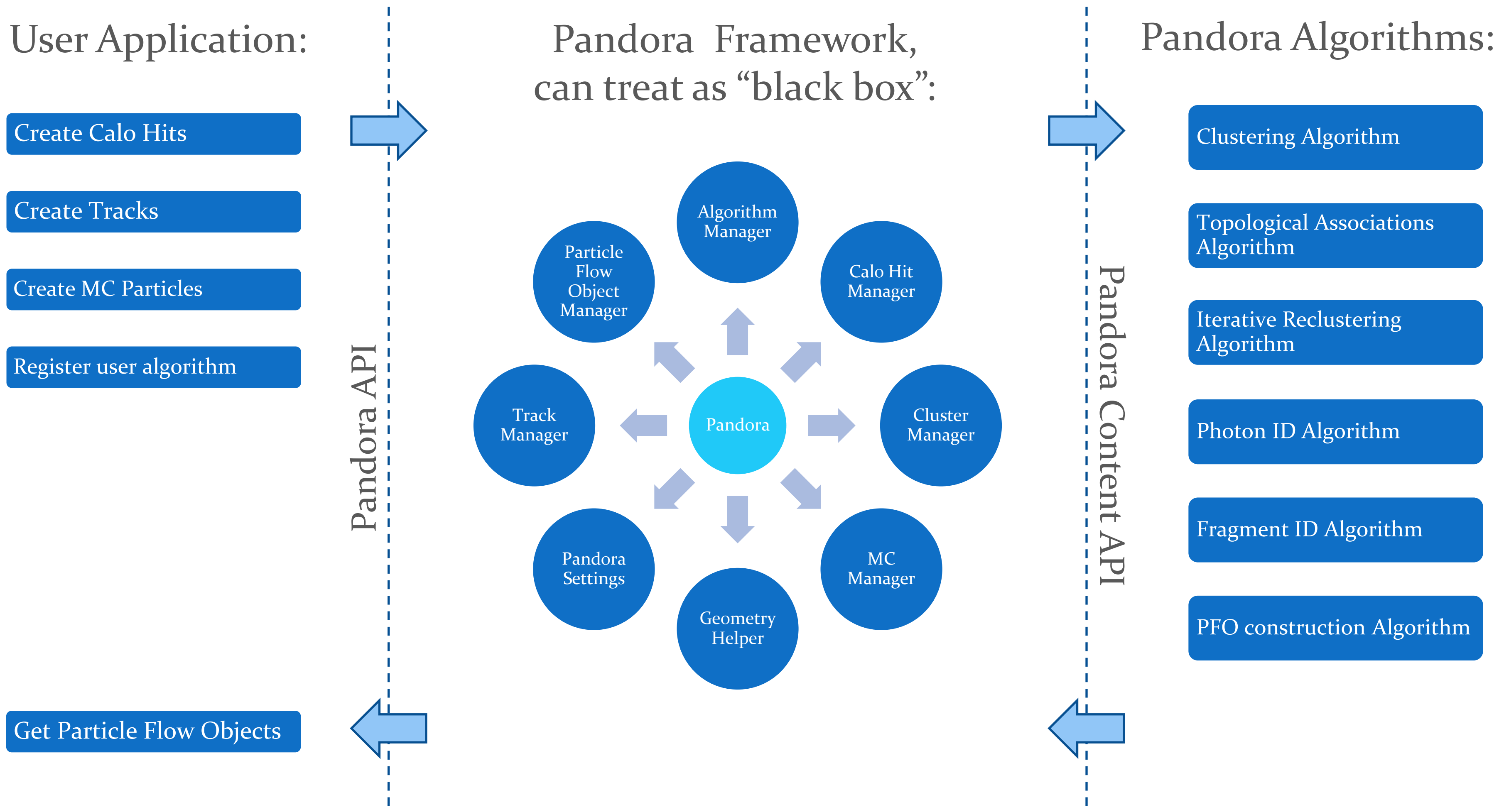
- For CLIC, relevant gauge boson energies are likely to be in the range 0.5 – 1.0 TeV.
- At low end of this range, mass resolution allows 2.1σ separation between the W and Z decays.
- For 1 TeV W/Z decays, achieved mass resolution allows for 1.5σ separation.

Pandora Redesign

- Would like better performance for 1 TeV W/Z separation and have some good ideas for achieving this (have not yet made any serious attempts to optimise Pandora performance for such high energy jets).
- However, whilst Pandora works well, the current code has reached a point where it is difficult to extend. It is simply not flexible enough to try out new ideas and meet challenges of high jet energies...

- ILD Letter of Intent version of Pandora has been frozen, no further development.
- A new version is being written from scratch (Cambridge/CERN):
    - Properly designed code, taking findings from previous PFAs into account.
    - Increased flexibility, designed to make it easy to try out new ideas:
        - Pandora is now a framework for running decoupled particle flow algorithms.
        - Can change algorithms that are used without recompiling.
        - Instead of simply changing clustering parameters in reclustering, can now slot in entirely different algorithms to find the best solution.
    - Improved memory footprint and speed.
    - Easier to maintain, significant effort to make very readable code.
    - Easier for other people to get involved – users can easily create and run their own algorithms. Pandora helps to separate physics in particle flow algorithms from the C++ memory management.
    - Now independent of Marlin framework – application in any framework uses a simple C++ API (application programming interface) to access the Pandora library.
    - Independent of specific detector details.
    - If necessary, this more flexible framework will allow a move from Particle Flow to Energy Flow calorimetry in "difficult" detector regions.

New Pandora Structure

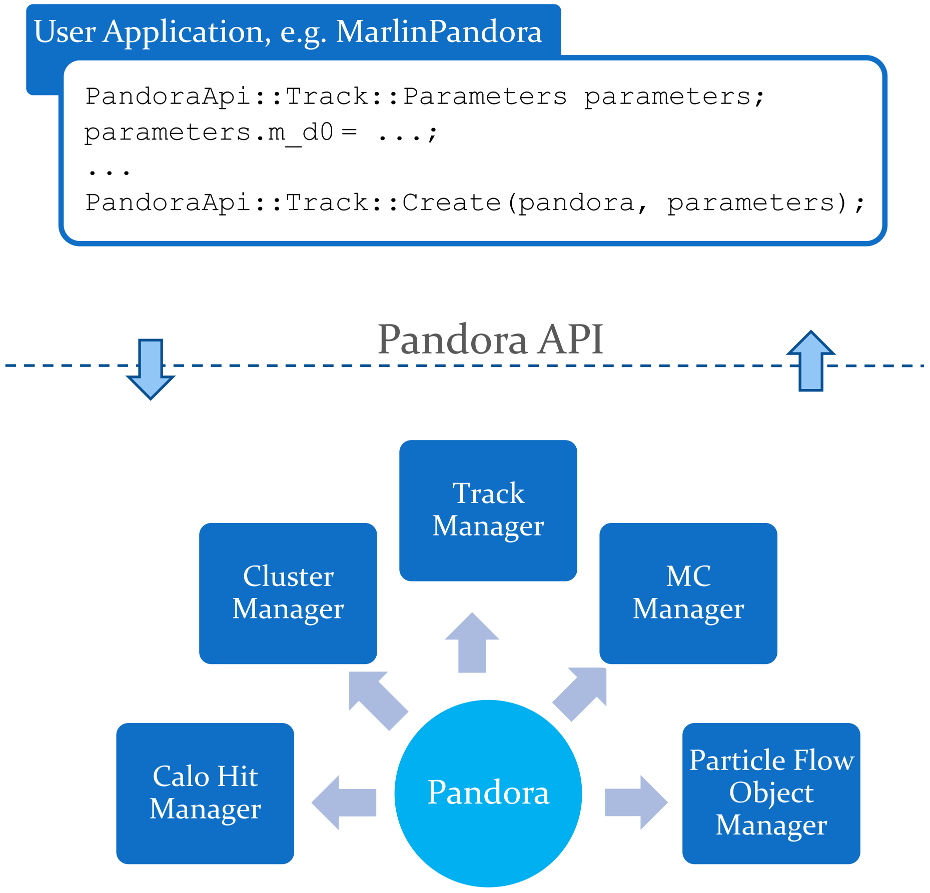


Pandora API

- The new version of Pandora is a separate library, which has no dependencies on any other software framework.
- In order for an application to use the library, a simple C++ API has been provided.
- The user application supplies Pandora with details of its calo hits, tracks and (if required) MC particles and MC relationships.
- Pandora then builds its own simple objects.

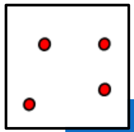
- Construction of these objects is simple: the user makes a Parameters class, fills the member variables and then calls the API Create function.
- All the member variables must be specified, or an exception will be thrown when Create is called.

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



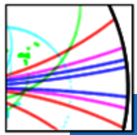
```
PandoraApi::Track::Parameters parameters;
parameters.m_d0 = ...;
...
PandoraApi::Track::Create(pandora, parameters);
```

Pandora Objects



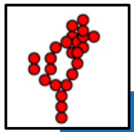
Calo Hit

- 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



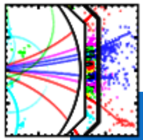
Track

- 2D impact parameters
- Momentum at d.c.a
- 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
- Energy-weighted centroid
- Radial direction cosine
- RMS w.r.t. linear fit
- ShowerMax layer
- List of associated tracks



Particle Flow Object

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

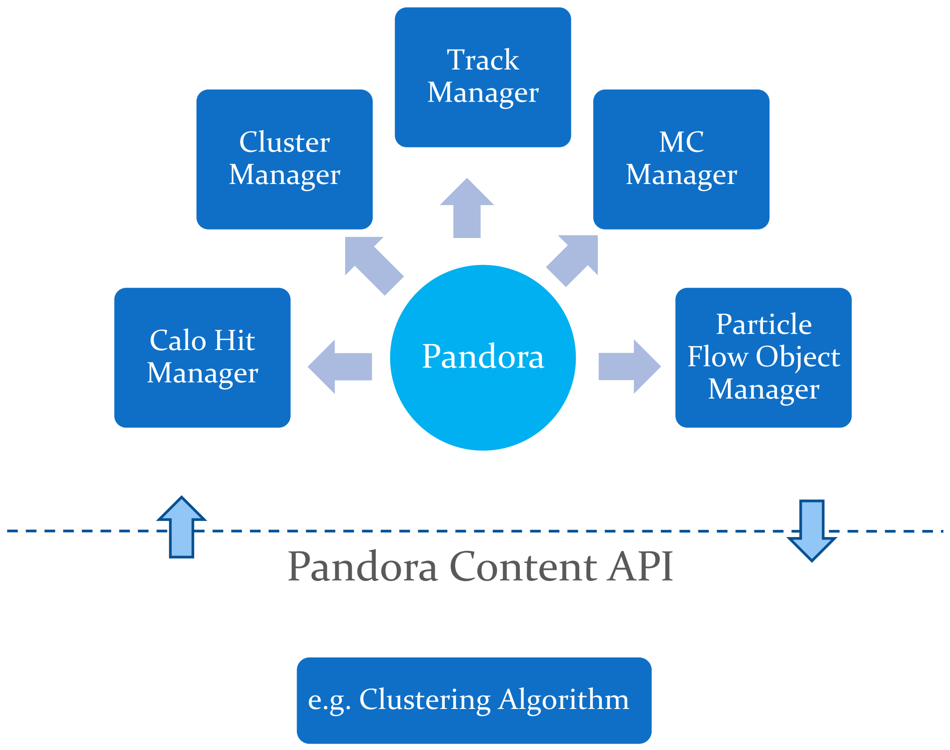
Mixture of properties specified by user and value-added properties, but all simple and well defined physics quantities for use in particle flow algorithms.

Pandora Managers

- Pandora Managers are designed to store named lists of their respective objects (like a very lightweight version of an LCIO collection).
- Pandora Algorithms interact with the Managers in a controlled manner, via the API, and Managers will perform memory management.
- For example, in order to create or destroy an object, you need to ask the Manager via the relevant API .

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

- The Managers store information about the algorithms currently running so that they can keep track of lists.
- Algorithms can modify lists, and save new lists.



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

Pandora Algorithms

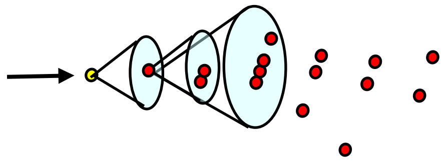
- Whilst the Pandora Managers look after memory management, the Pandora Algorithms are free to do physics.
- The algorithms use a simple API to access the Pandora objects in a controlled, but flexible manner:
    - Create new cluster, Create a particle flow object
    - Add a calo hit to a cluster, merge two clusters, delete a cluster
    - Get current cluster, calo hit or track lists
    - Change the current cluster, calo hit or track lists
    - Run a daughter algorithm
    - Save new lists of clusters, calo hits or tracks
    - …

- The algorithms are designed to be "nested" (parent algorithms run daughter algorithms to manipulate objects):
    - This helps to make the reclustering very simple and flexible (see later).
    - Keeps the algorithms simple and independent; each will have a well-defined aim. E.g. Topological clustering algorithm could choose to run a daughter algorithm for each of the possible types of association.

- Algorithms are simple to configure, using an xml file, and can be easily swapped in/out without recompiling:
    - Easy to investigate effects of running different algorithms, or running algorithms in a different order.
    - Simple to run entirely different sets of algorithms in parallel and identify the best.

Typical Algorithms

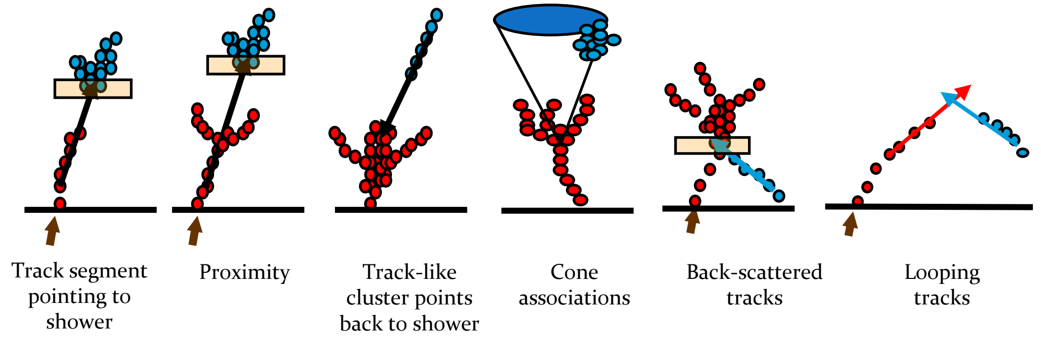
i). Clustering



The main Pandora clustering algorithm is a cone-based forward projective method working from innermost to outermost pseudolayer. In this way hits are either added to existing clusters or they are used to seed new clusters.

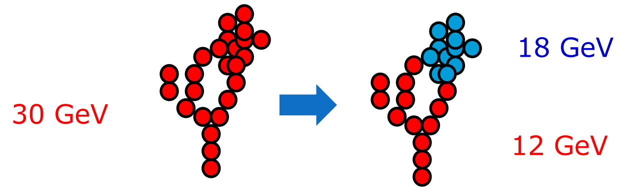
ii). Topological Associations

The initial Pandora clustering algorithm errs on the side of splitting up true clusters, rather than risking merging independent energy deposits. Hence, the next process is to merge clusters, following a number of topological rules.



Track segment pointing to shower
Proximity
Track-like cluster points back to shower
Cone associations
Back-scattered tracks
Looping tracks

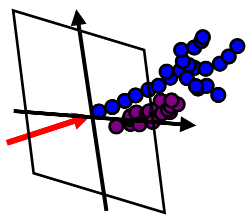
iii). Iterative reclustering



Clusters likely to have been created by merging showers from more than one particle 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.

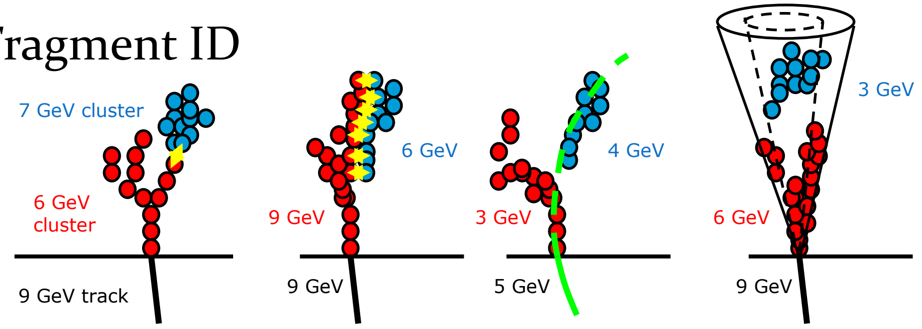
Typical Algorithms

iv). Photon ID



A shower-profile based photon identification algorithm is applied to the clusters, improving photon tagging and recovering cases where a primary photon is merged with a hadronic shower from a charged particle.

v). Fragment ID



Distance of closest approach

Layers in close contact

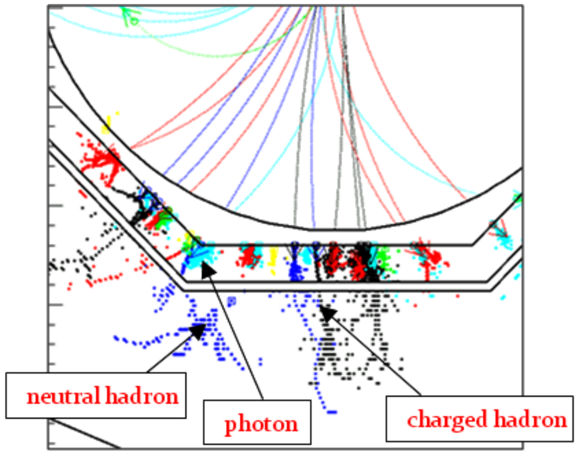
Distance to track extrap.

Fraction of energy in cone

At this late stage there are still a significant number of “neutral clusters” (not identified as photons) which are fragments of charged particle hadronic showers. Attempt to identify these clusters and merge them with the appropriate parent charged cluster.

vi). PFO formation

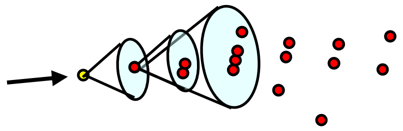
Final stage is to create PFOs from the clusters. Tracks are matched to clusters on basis of the distance of closest approach of the track projection into the first layers of the calorimeter.



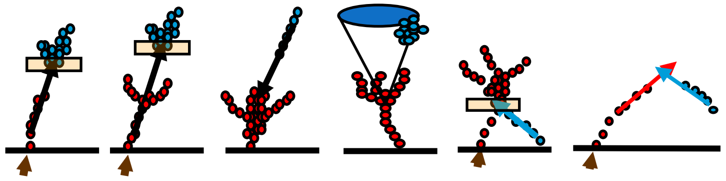
Example Algorithm – Clustering

- To form clusters in the new Pandora framework, you need:
  1. A parent algorithm to control operations,
  2. A cluster formation algorithm,
  3. A topological association algorithm (which may itself run multiple daughter algorithms).

- Parent algorithm asks to run a clustering algorithm; cluster manager then creates a new temporary cluster list, associated with the parent algorithm, sets this as "current", and allows new clusters to be formed.



- Daughter clustering algorithm gets current calo hit and track lists, uses its logic to populate temporary cluster list and returns control to parent algorithm.



- Parent algorithm then calls topological association algorithm, which cannot form new clusters, but can modify existing clusters, even merging multiple clusters.

- Parent algorithm can then save (a subset of) the temporary clusters as a new named cluster list.
- If desired, parent algorithm can set this new list to be the current list for future algorithms, otherwise current cluster list will return to that when parent algorithm was initialised. Any remaining temporary cluster s will be tidied automatically.

Example Algorithm – Reclustering

- Algorithm framework and idea of parent algorithms controlling daughter algorithms is (to some extent) designed to make reclustering simple, flexible and elegant.
- Parent reclustering algorithm needs only to perform following operations:

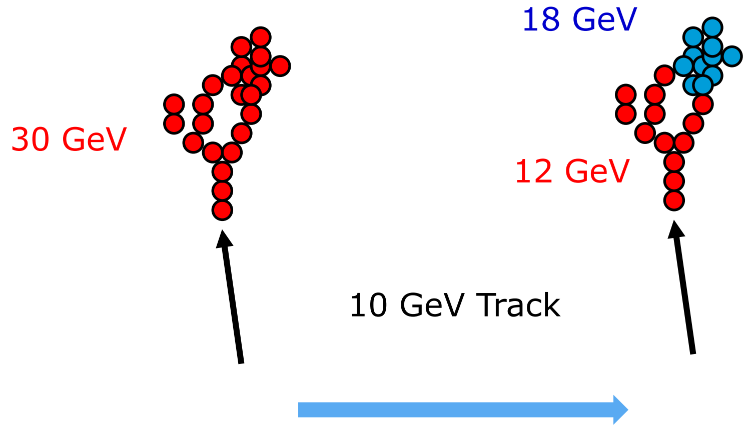
i. Identify inconsistent pairing of track and cluster(s) and ask to recluster these.
   - Relevant clusters will be moved to a new temporary cluster list, associated with the parent algorithm. Current calo hit/track lists changed.

ii. Ask to run a clustering algorithm.
   - This will create another uniquely named temporary cluster list, which will be filled by the daughter clustering algorithm.

iii. Calculate a figure of merit for the consistency of the track and new cluster(s).

iv. Repeat stages ii. and iii. as required.
   - Can run copies of the same algorithm, with different clustering parameters, or use entirely different approaches to the problem.

v. Choose most appropriate cluster(s).
   - Cluster lists will be reorganised accordingly. Temporary lists will be tidied.



Change clustering parameters and/or clustering algorithm until cluster splits and get sensible track-cluster match

Configuring Pandora Algorithms

- Pandora algorithms are configured via a simple xml file, a very natural way to configure nested algorithms:
  - Ideal for quickly experimenting with running new algorithms, or exploring new methods to address problems such as leakage. Easy to mix “real” and “cheating” algorithms.
  - The complicated process of reclustering is reduced to the following simple configuration:

```
<!-- Parent reclustering algorithm runs multiple clustering algorithms -->
<algorithm type = "Reclustering”>
    <!-- List of daughter clustering algorithms -->
    <clusteringAlgorithmList>
        <algorithm type = "ClusteringType1”> ClusteringType1 parameters... </algorithm>
        <algorithm type = "ClusteringType2”> ClusteringType2 parameters... </algorithm>
        <algorithm type = "ClusteringTypeN”> ClusteringTypeN parameters... </algorithm>
    </clusteringAlgorithmList >

    <!-- Other parent reclustering algorithm properties ... -->
</algorithm>
```

- Pandora will provide a comprehensive library of built-in algorithms to experiment with. However, we want people to get involved quickly and easily, so users can create and register their own algorithms:
  - Quick and easy to create an algorithm and we provide a template; just need to inherit from the Pandora Algorithm base class. User then implements logic and calls functions available via API.
  - There is a simple API for registering the new type of algorithm. Can then simply add the settings to the xml file (there is an XmlHelper to read custom settings in your algorithm).
  - There is no need to recompile Pandora.

Current Status

- The redesign of Pandora is well underway. The framework, including the APIs and managers, have been fully implemented and fully tested. It works!
  - Have written a MarlinPandora package for using the Pandora library in Marlin and also an associated monitoring package.
  - The first algorithms have been implemented and tested (although some of these have been designed to exercise the framework rather than being properly physics driven!)
  - Aim to have full re-implementation of existing Pandora algorithms by 1/1/2010.
  - Constant benchmarking against existing code during this process.

- The final result should be a flexible framework for running particle flow algorithms.
  - It should be much simpler to experiment with new ideas and to address the complications of particle flow calorimetry at high jet energies.
  - The new Pandora library is independent of any experimental software framework (no dependencies), but should be **easy to use** by all.
  - Have (hopefully) helped to separate physics from the code!
  - Should be very easy for people to get involved.