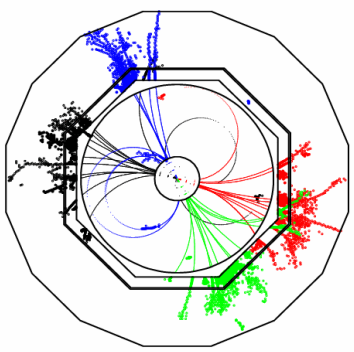


Redesign of Pandora PFA

John Marshall,
University of Cambridge

ILD Workshop, DESY, July 6 2010

Pandora Redesign

- Whilst the Pandora algorithms work very well, the old code reached a point where it was extremely difficult to extend. It was not flexible enough to try out new ideas and improvements…
- ILD Letter of Intent version of Pandora was frozen and a new version written from scratch.
- This is much more than just a re-implementation, it is a major redesign; 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; users can register 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).
- In this talk, will revisit some important points presented at LCWS10, before describing progress leading up to recent release of PandoraPFANew.

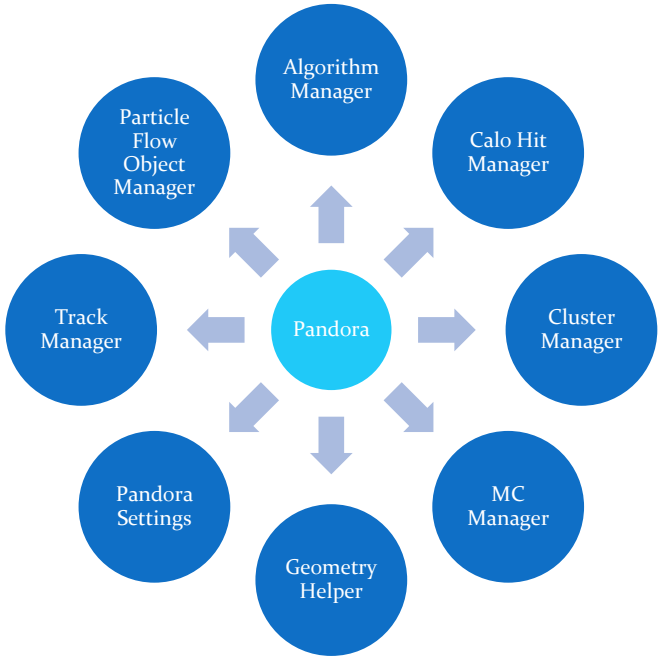
Pandora Structure

User Application:

- Specify Geometry
- Create Calo Hits
- Create Tracks
- Create MC Particles
- Register User Content

Pandora API

Pandora Framework, treat as "black box":



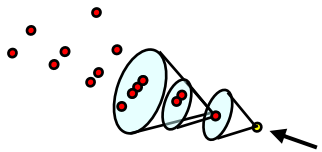
Pandora Content API

Pandora Algorithms:

- Clustering Algorithm
- Topological Association Algorithms
- Statistical Reclustering Algorithm
- Photon Recovery Algorithm
- Fragment Removal Algorithms
- Track-cluster Association Algorithms
- PFO Construction Algorithm

Get Particle Flow Objects

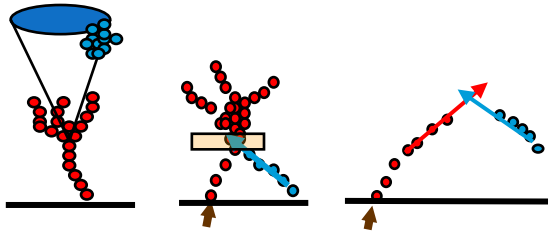
Pandora Algorithms

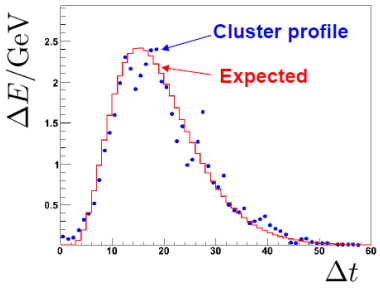


Cone-based forward projective method

Clustering Algorithm

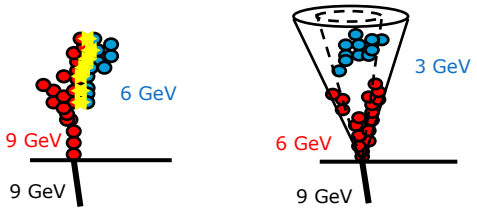
Topological Association Algorithms

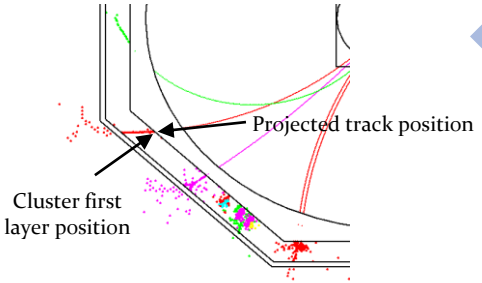




Photon ID Algorithms

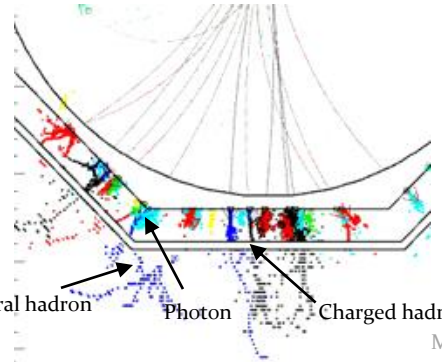
Fragment Removal Algorithms



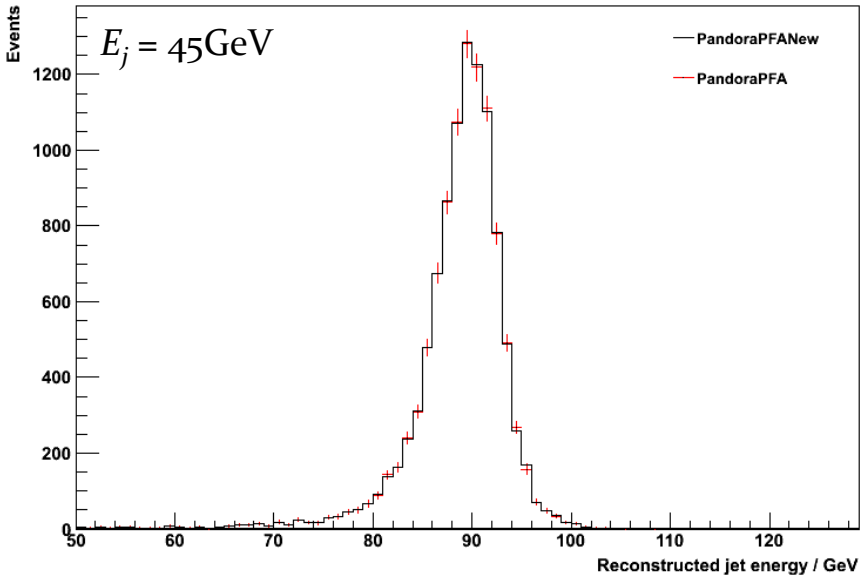


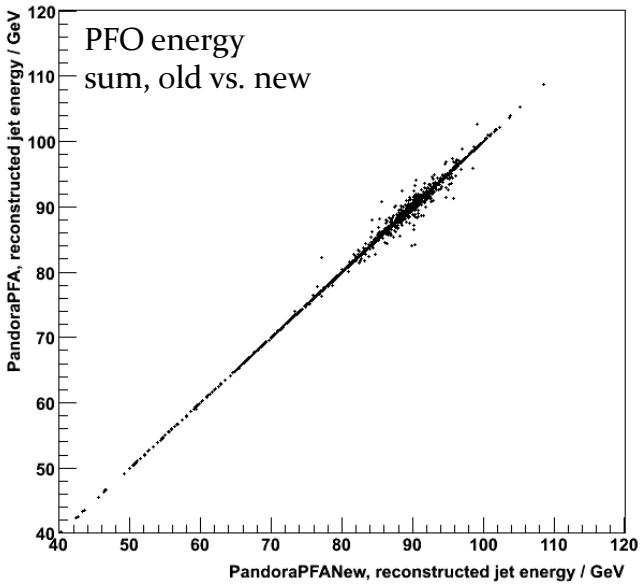
Track-cluster Association Algorithms

PFO Construction Algorithm



Status at LCWS10

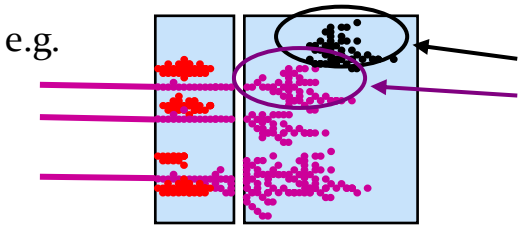




- First tests were performed for the ILD detector concept, using MC samples of approximately 10,000 $Z \rightarrow uds$ generated with the Z decaying at rest with E_z = 91.2GeV.
- At this energy, all newly implemented code and framework is exercised, without (a strong) need for statistical reclustering and/or leakage corrections.
- For fair comparison, reclustering, leakage corrections and all use of track-relationship information was turned off in old Pandora (at the time, was still work in progress for new Pandora).
- **Excellent agreement observed, by construction.**

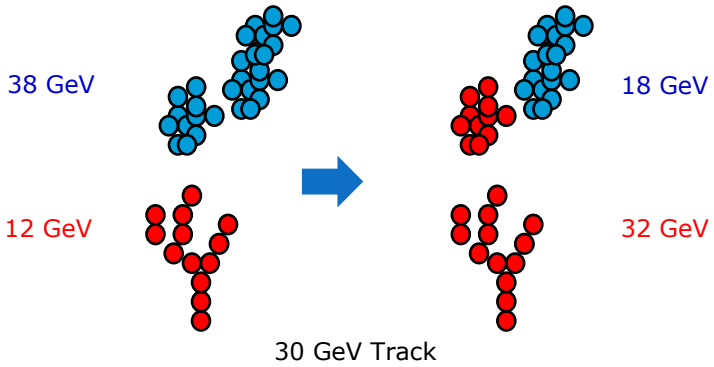
Reclustering

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



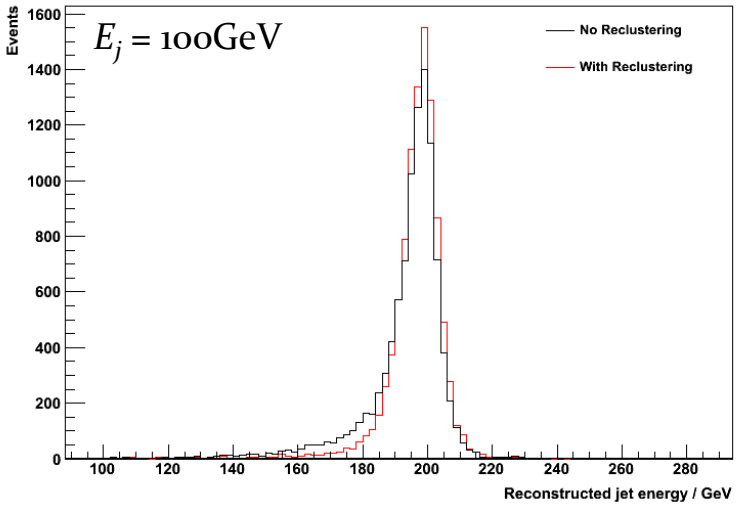
If these hits are clustered together with these, lose energy deposit from neutral hadron (now part of track particle) and ruin jet energy measurement.

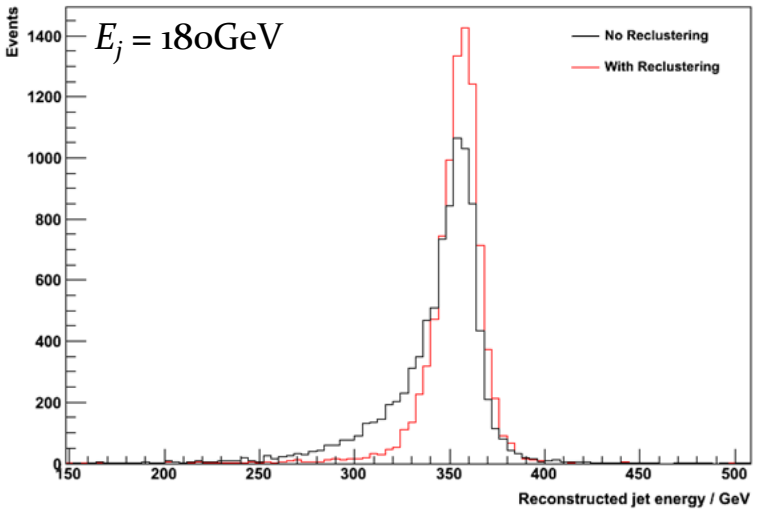
- Pandora addresses this problem with **statistical reclustering:**
  - Clusters that have been incorrectly merged together are identified via consistency of cluster energy and associated track momentum.
  - Attempts are made to redistribute the hits by using different clustering parameters or entirely different clustering algorithms.

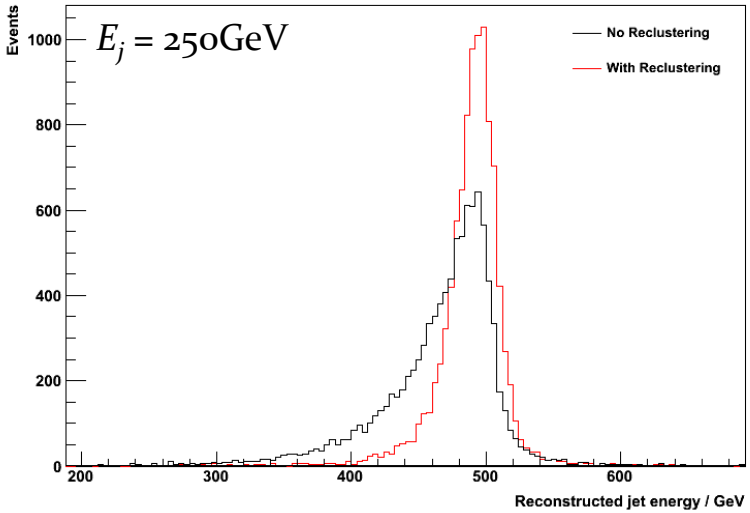


- Algorithms that "steer" the reclustering have now been fully implemented and validated.
- Have deviated from old Pandora here, tidying the algorithms and more clearly defining the role of each.
- Introduces some small differences between new and old Pandora output. However, extensive validation shows that results remain statistically identical to old Pandora.

Reclustering



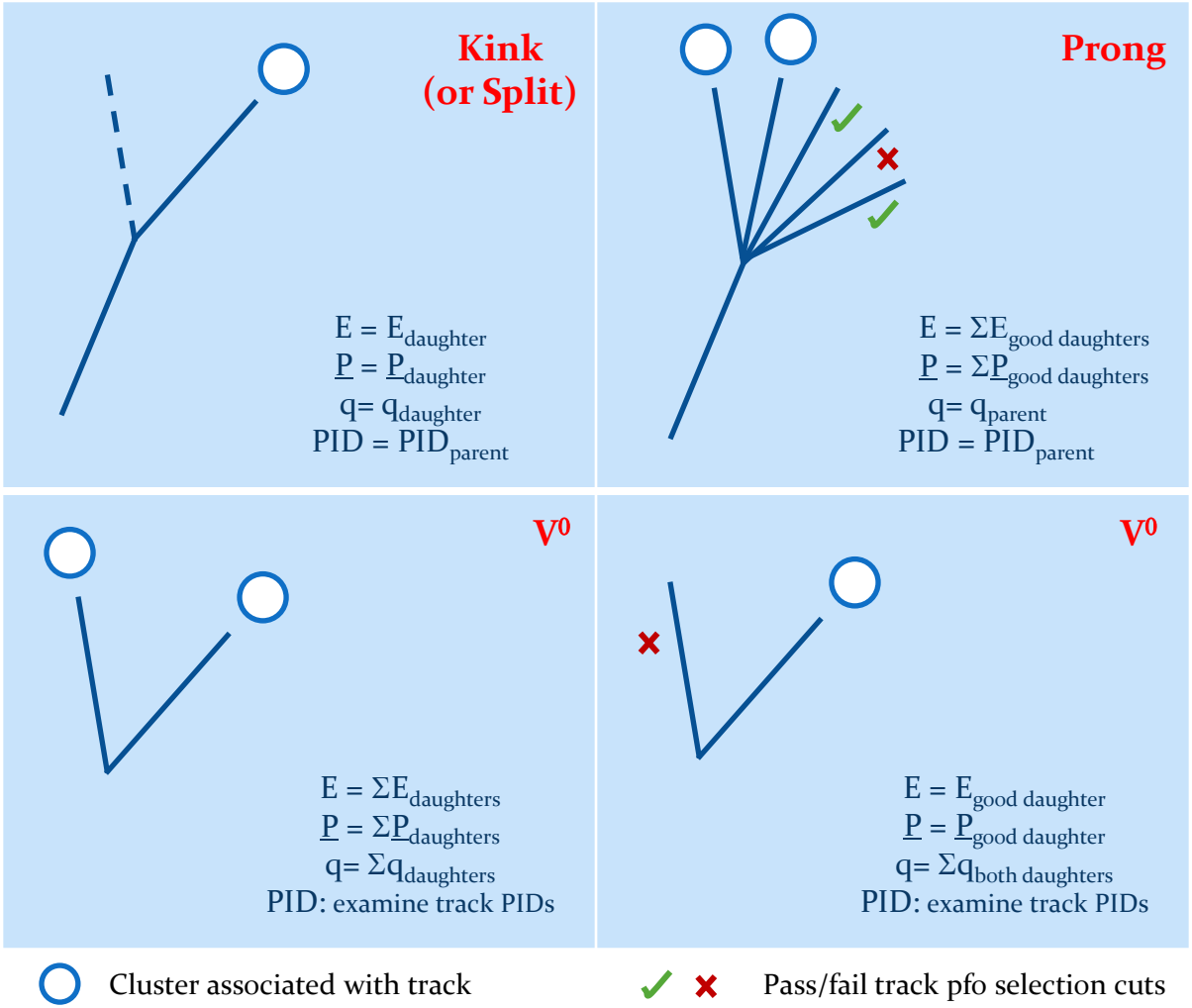




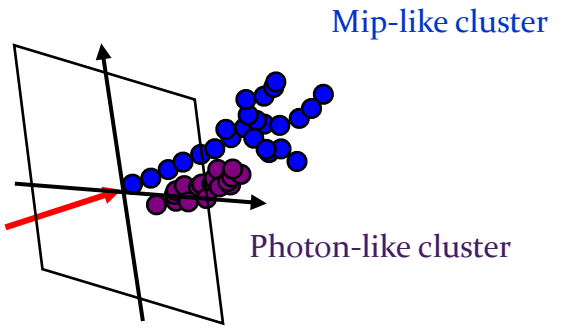
As expected, reclustering algorithms are very important at higher energies.

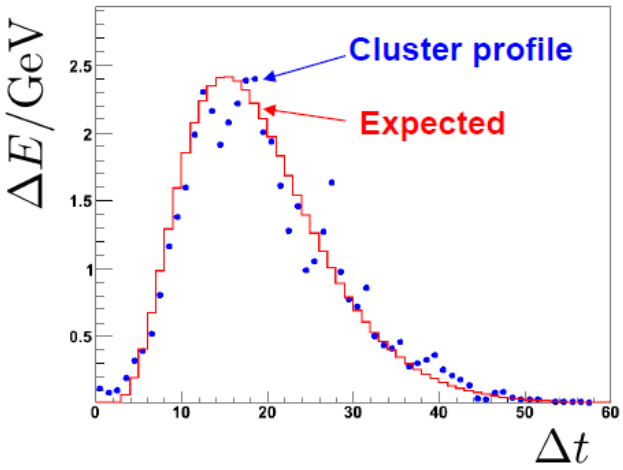
Track Relationships

- New Pandora code allows track parent, daughter and sibling relationships to be specified.
- Uses this information when determining which tracks can be associated to clusters and when building charged PFOs.
- For ILD, extract track relationships from v^0 and kink finders in MarlinReco.
- Charged PFOs are built using illustrated rules. Require at least one track with a cluster association, or a track which is deemed to be low p_T



Photon Recovery

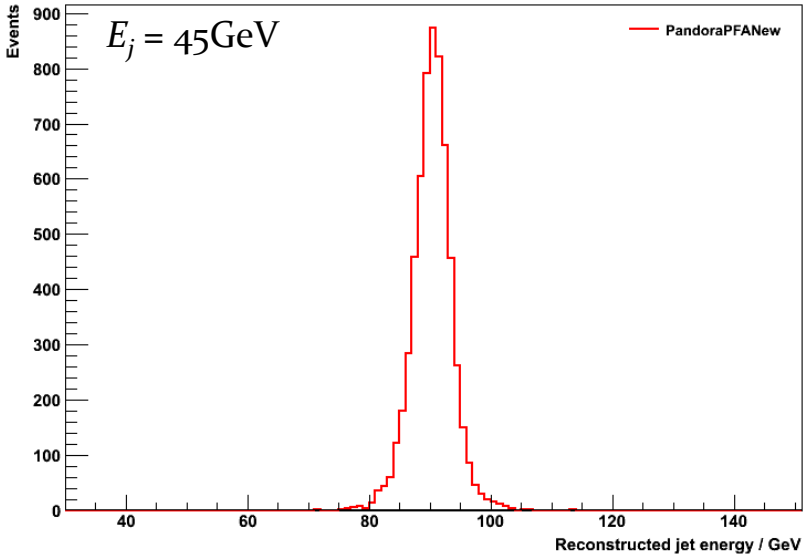




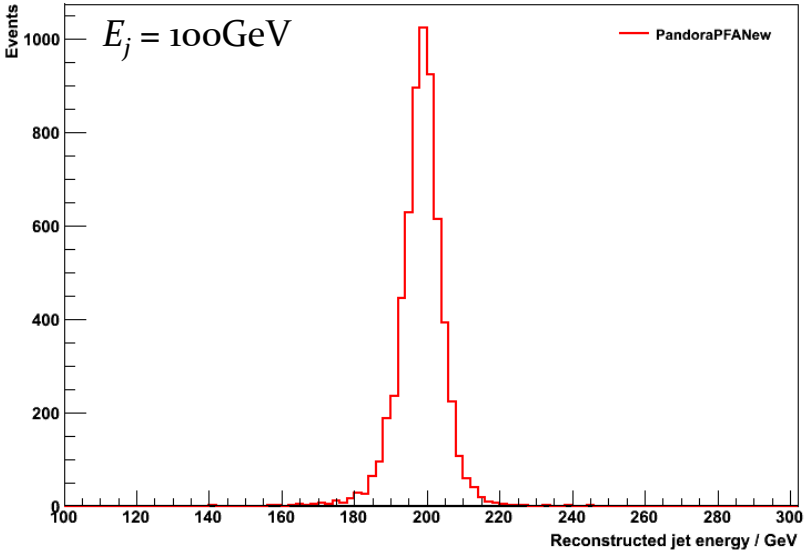
- Have also implemented a cluster fragmentation mechanism. This can be considered as a "local reclustering":
  - A cluster can be broken up into different combinations of smaller fragments.
  - The original cluster and all fragment combinations can still be accessed and their properties queried.
  - When the best fragment combination has been identified, call EndFragmentation API.
  - ClusterLists will be tidied accordingly, any temporary clusters created will be deleted.
- This functionality has been used in a newly implemented PhotonRecovery algorithm, which splits clusters up into photon-like fragments and mip-like fragments.
- Further photon id helper functions, based on shower-profiles, have also been implemented.

Reconstruction Performance

- Performance has been studied for the ILD detector concept, using MC samples of approximately 10,000 $Z \rightarrow uds$ generated with the Z decaying at rest, with E_z = 91.2, 200, 360 and 500GeV.
- The performance is quoted in terms of rms_{90}, defined as the rms in the smallest range of reconstructed energy containing 90% of the events. A cut on the polar angle is applied to avoid the barrel/endcap overlap region: $|\cos\theta| < 0.7$
- For each set of events, the total energy was reconstructed and the jet energy resolution obtained by dividing the total energy resolution by √2.



$rms_{90}(E_j) / E_j = 3.63 \pm 0.05$

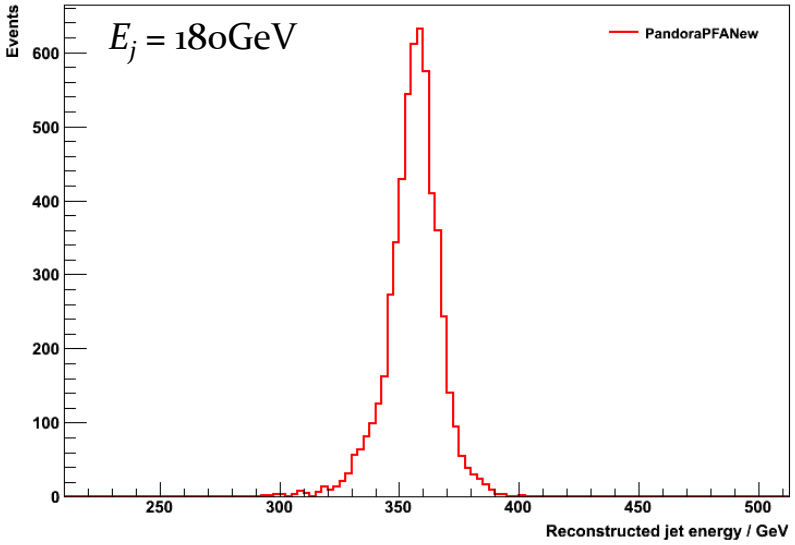


$rms_{90}(E_j) / E_j = 2.94 \pm 0.04$

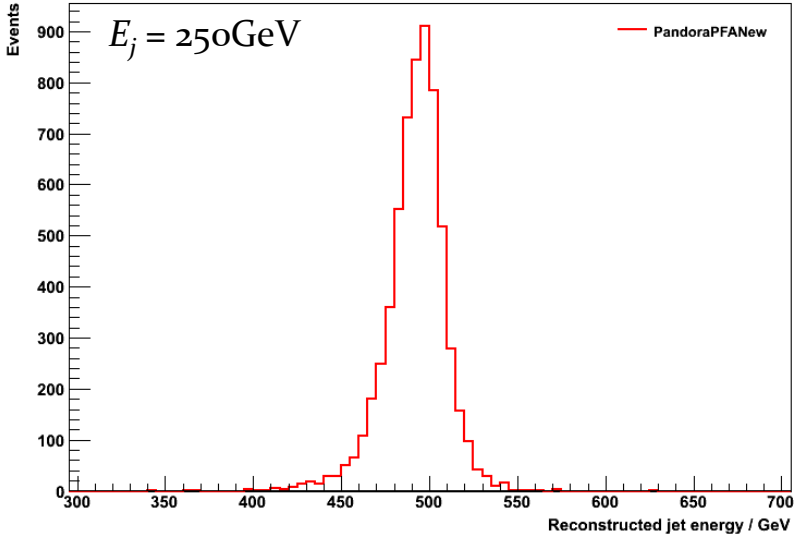
Reconstruction Performance

E_j	45GeV	100GeV	180GeV	250GeV
PandoraPFANew, $rms_{90}(E_j) / E_j$	3.63 ± 0.05	2.94 ± 0.04	3.09 ± 0.04	3.32 ± 0.04
Original Pandora, $rms_{90}(E_j) / E_j$	3.66 ± 0.05	2.99 ± 0.04	3.13 ± 0.04	3.31 ± 0.05

Note: these are results obtained without PhotonClustering algorithms.



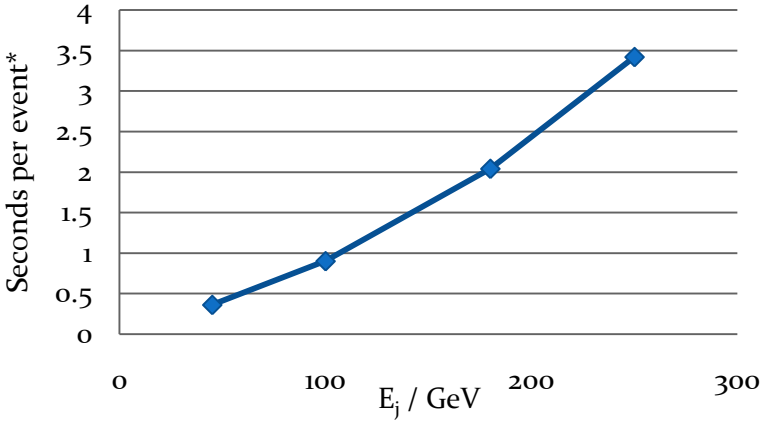
$rms_{90}(E_j) / E_j = 3.09 \pm 0.04$



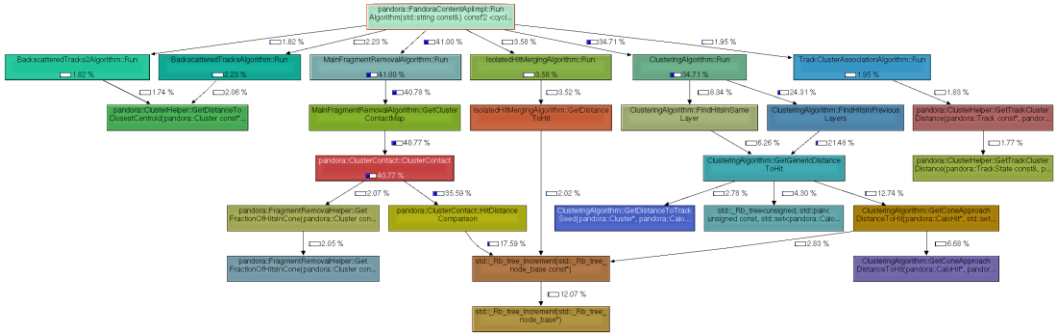
$rms_{90}(E_j) / E_j = 3.32 \pm 0.04$

CPU/Memory Performance

- Performance of PandoraPFANew has been examined using the Valgrind callgrind tool.
- Huge performance gains were obtained by optimising several FragmentRemovalHelper functions, which were called for many different permutations of CaloHits and Clusters.
- With recent improvements, PandoraPFANew is 2-3 times faster than old code. It is also encouraging to see that framework makes negligible contribution to reconstruction time – time is spent in physics algorithms, performing nested loops.



*For KinkFinder, V^0Finder, MarlinPandora and LCIO output processors on Intel E8500 @ 3.16GHz



- Valgrind memcheck tool gives PandoraPFANew a clean bill of health. It appears that the memory footprint is rather small.
- Running KinkFinder, V^0Finder, MarlinPandora and LCIO output processor uses less than 250MB on typical SLC5 machine.

Pandora Plugins

- PandoraPFANew has been designed to make it easy for people to get involved and try out new ideas. Users can therefore register and use their own content, including:
  - Particle identification helper functions,
  - Hadronic and electromagnetic energy correction helper functions,
  - Particle flow algorithms, allowing for a completely different reconstruction.

Particle Identification Functions

1. Implement particle id function:

```
static bool MyClass::MyParticleIdFunction(const pandora::Cluster *const pCluster);
```

2. Register function with Pandora under a specific name:

```
PandoraApi::RegisterParticleIdFunction(pandora, "MyFunctionName", &MyClass::MyParticleIdFunction);
```

3. In PandoraSettings xml file, assign named function to one of a number of particle id 'slots':

```
PhotonFast, PhotonFull, ElectronFast, ElectronFull, MuonFast, MuonFull, ...
```

- Similar procedure for energy correction functions (different function prototype). User creates and registers separate hadronic and electromagnetic correction functions, then specifies ordered list of correction function names via xml.
- Existing Pandora algorithms and functions have been written to ensure they have no dependencies and that they are (subject to changing steering parameters) largely detector independent, this need not apply to custom content.

Pandora Plugins

- Creating a custom algorithm allows access to the full range of functions in the PandoraContentAPI. This allows implementation of almost any conceivable particle flow algorithm.
- Can write new algorithms to complement existing Pandora reconstruction, or can write simple 'wrapper' algorithms to bring results of other packages right to heart of Pandora reconstruction.
- For example, can very simply (~50 lines of code) use output from e.g. GARLIC to replace Pandora photon-clustering stage:

Particle Flow Algorithms

1. Copy and rename Pandora template algorithm class, then register new algorithm with Pandora:
   ```
   PandoraApi::RegisterAlgorithmFactory(pandora, "MyAlgorithmName", new MyAlgorithm::Factory);
   ```
2. Add algorithm to the PandoraSettings.xml file; it will then be called automatically for each event.
3. Algorithm reads existing photon cluster collection, then uses unique identifiers in Pandora CaloHits to simply recreate the photon clusters within the Pandora framework.
4. Algorithm uses simple API call to save new photon clusters in a named cluster list, and to remove them from subsequent reconstruction.
5. ClusterPreparation algorithm configured to add photon clusters back into reconstruction when desired; they are probably best added at Pfo construction stage.
6. Should observe an immediate improvement in jet energy resolution.

Summary

- The first release of PandoraPFANew (and its associated MarlinPandora processor) is now available. This is a complete replacement for the old version of Pandora and all users are encouraged to upgrade.
- The PandoraPFANew framework offers a great deal of flexibility for investigating new ideas and, in particular, allows for custom energy corrections, particle id and implementation of custom algorithms.
- The existing particle identification and energy correction functions are quite simplistic and are only placeholders. We are going to be looking for new ideas, so please get in touch if you'd like to be involved…
- Immediate plans now concern improving the reconstruction at high energies. Also have many other interesting ideas for moving forwards and improving Pandora within the new framework:
  - New clustering algorithms,
  - Proper treatment of muons (Kalman filter),
  - Improved treatment of leakage.
- Finally, should advertise fact that Norman Graf and Jeremy McCormick are working on a SlicPandora application:
  - With MarlinPandora (ILD) and SlicPandora applications, every algorithm in the PandoraPFANew library is automatically and instantly available for both ILD and SiD concepts.