Calorimetry Clusterizer on DSTs

TRISTAN PROTZMAN, LEHIGH UNIVERSITY

Demo Code

https://github.com/tlprotzman/calo_clustering_demo

What's Available?

- V3 Clustering
  - Default for BECAL
- MA Clustering
  - Default for FEMC, LFHCAL, EHCAL
- Other algorithm can be implemented if desired
  - Trivially: C3, C5, 3x3, 5x5, V1
  - Code already exists, just needs to be ported

Running the Clusterizer

- Needs `libEICCaloReco.so`
- Includes:
  - `#include <eiccaloreco/RawClusterBuilderHelper.h>` // Base class for V3 and MA
  - `#include <eiccaloreco/RawClusterBuilderkV3.h>`
  - `#include <eiccaloreco/RawClusterBuilderkMA.h>`
- Registering a calorimeter for clustering:
  - `RawClusterBuilderHelper *becal_clusterbuilder = new RawClusterBuilderkV3("BecalRawClusterBuilderkV3");` // Creates V3 clusterizer
  - `becal_clusterbuilder->Detector("BECAL");` // Sets the target detector
  - `becal_clusterbuilder->set_seed_e(0.5);` // Sets the seed energy for clustering
  - `becal_clusterbuilder->set_agg_e(0.1);` // Sets the aggregation energy
  - `se->registerSubsystem(becal_clusterbuilder);` // Register the clusterizer with Fun4All

```
RawClusterBuilderHelper *becal_clusterbuilder = new RawClusterBuilderkV3("BecalRawClusterBuilderkV3");
becal_clusterbuilder->Detector("BECAL");
becal_clusterbuilder->set_seed_e(0.5);
becal_clusterbuilder->set_agg_e(0.1);
se->registerSubsystem(becal_clusterbuilder);
```

Running the Clusterizer

- Or just turn it on in the default macro!
- Adds CLUSTER_BECAL to the node tree

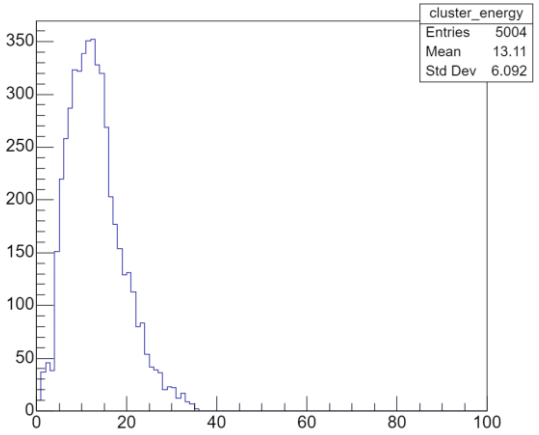
```
307   Enable::BECAL = true;
308   Enable::BECAL_CELL = Enable::BECAL && true;
309   Enable::BECAL_TOWER = Enable::BECAL_CELL && true;
310   Enable::BECAL_CLUSTER = Enable::BECAL_TOWER && true;
311   Enable::BECAL_EVAL = Enable::BECAL_CLUSTER && true;
```

Using Clusters

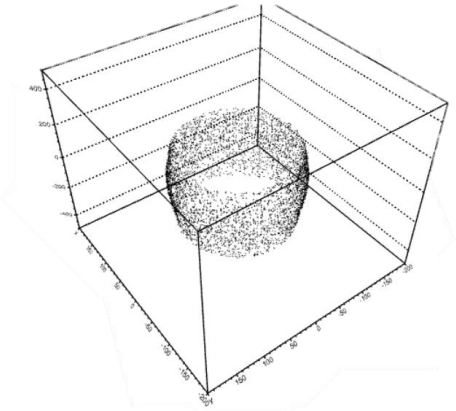
- The cluster interface is similar to the tower interface
  - Clusters are stored within a RawClusterContainer
  - We can get an iterator and loop over all the RawClusters,
  - RawClusters contain energy, position, and map of towers

```
// Grab the calorimeter's cluster node
std::string cluster_node_name = "CLUSTER_" + detector;
RawClusterContainer *clusters = findNode::getClass<RawClusterContainer>(topNode, cluster_node_name);
if (clusters == nullptr) {
  std::cerr << PHWHERE << "Could not find cluster node " << cluster_node_name << std::endl;
  return Fun4AllReturnCodes::ABORTEVENT;
}

// Add the cluster energies to the histogram
RawClusterContainer::ConstRange cluster_begin_end = clusters->getClusters();
for (RawClusterContainer::ConstIterator itr = cluster_begin_end.first; itr != cluster_begin_end.second; ++itr) {
  RawCluster *cluster = itr->second;
  cluster_energy->Fill(cluster->get_energy());
  cluster_xyz->Fill(cluster->get_x(), cluster->get_y(), cluster->get_z());
}
```



Cluster Energy


Cluster Position

Using Clusters

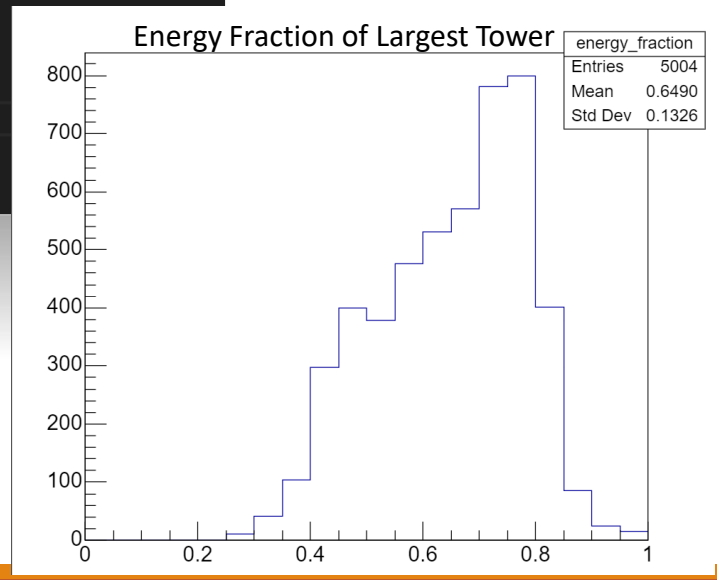
- The tower map lets us work directly with the towers in the cluster!
  - Find the tower node and tower geometry node as well
    - `RawTowerContainer *towers = findNode::getClass<RawTowerContainer>(topNode, "TOWER_CALIB_" + detector);`
    - `RawTowerGeomContainer *tower_geom = findNode::getClass<RawTowerGeomContainer>(topNode, "TOWERGEOM_" + detector);`
- The tower map maps towers in the cluster to their energy contribution
  - `twr_itr = clusters->get_towermap().begin()`
  - Get the towers **energy** with `twr_itr->second`
  - Get the **tower** with `RawTower *towers->getTower(twr_itr->first)`
  - Get tower **geometry** with `RawTowerGeom *tower_geom->get_tower_geometry(twr_itr->first)`
- From here, we could calculate observables such as cluster shape

Using Clusters

```
// Let's play with the towermap...
RawCluster::TowerMap tower_map = cluster->get_towermap();
RawCluster::TowerIterator twr_itr;
float max_e = -1;
for (twr_itr = tower_map.begin(); twr_itr != tower_map.end(); ++twr_itr) {
  max_e = twr_itr->second > max_e ? twr_itr->second : max_e;  // If we just want the energy, use the second element

  RawTower *cluster_tower = towers->getTower(twr_itr->first); // Or we can get the entire tower for calculating things like shape
  RawTowerGeom *cluster_tower_geom = tower_geom->get_tower_geometry(twr_itr->first);
  float weighted_x = cluster_tower->get_energy() * cluster_tower_geom->get_center_x(); // for example

  weighted_x++; // to avoid unused-variable warning at compilation, meaningless
}
energy_fraction->Fill(max_e / cluster->get_energy());
```



Extra

Cluster Properties

virtual **RawClusterDefs::keytype**	**get_id** () const cluster ID
virtual float	**get_energy** () const total energy
virtual size_t	**getNTowers** () const Tower operations.
virtual **TowerConstRange**	**get_towers** () const
virtual const **TowerMap** &	**get_towermap** () const return tower map for c++11 range-based for-loop
virtual **CLHEP::Hep3Vector**	**get_position** () const
virtual float	**get_phi** () const access to intrinsic cylindrical coordinate system
virtual float	**get_r** () const
virtual float	**get_z** () const
virtual float	**get_x** () const access Cartesian coordinate system
virtual float	**get_y** () const
virtual float	**get_ecore** () const
virtual float	**get_chi2** () const reduced chi2 for EM shower
virtual float	**get_prob** () const cluster template probability for EM shower
virtual float	**get_et_iso** () const isolation ET default
virtual float	**get_et_iso** (const **int**, bool, bool) const isolation ET the radius and hueristic can be specified