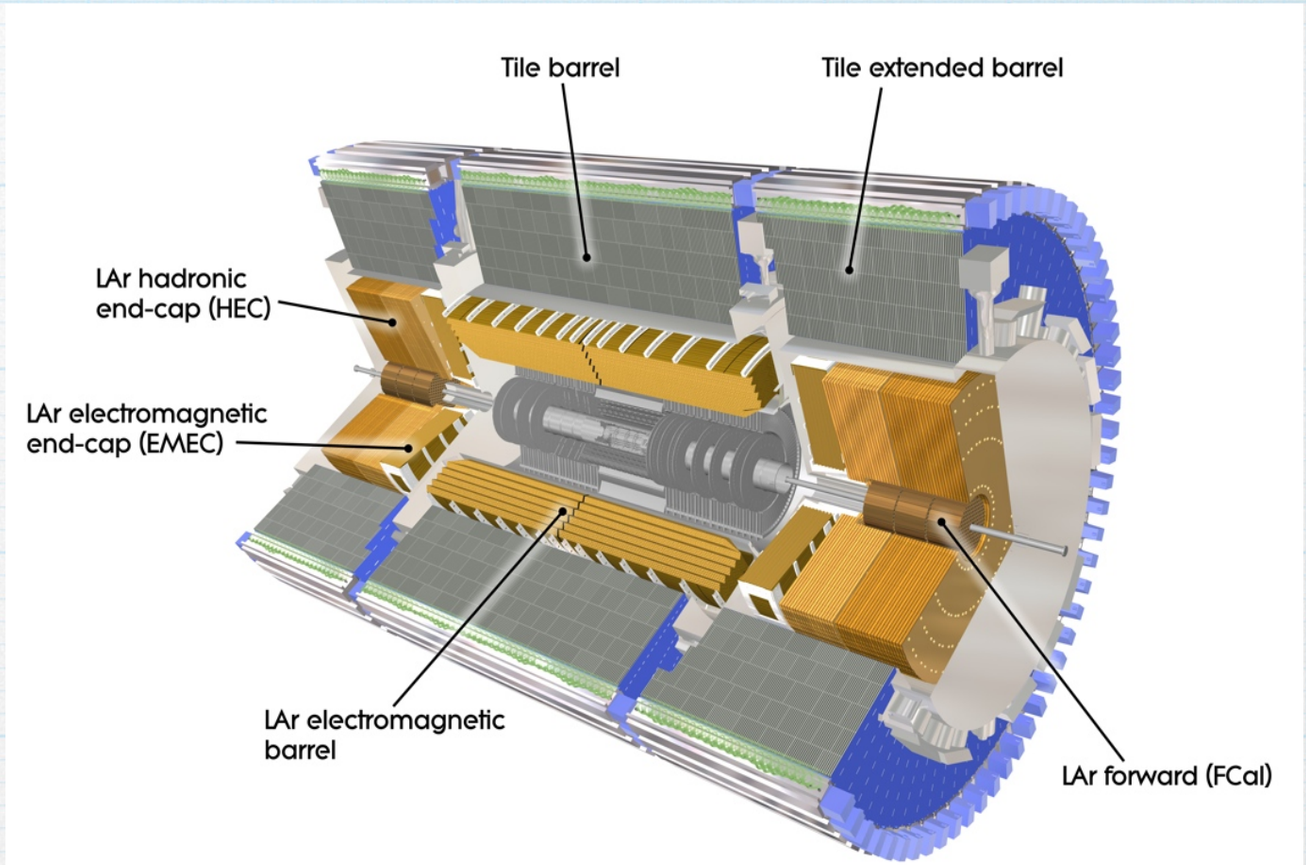
Calorimeter EDM

by Denis Oliveira Damazio
(BNL)

Calorimeters

* Different calorimeter technologies.
* lead/Liquid Argon (EM - up to |η|<3.2), copper/ Liquid Argon (HEC - 1.5<|η|<3.1) and copper-tungsten/Liquid Argon (FCAL - 3.0<|η|<4.9).
* Iron/Scintillating tiles (Hadronic - |η|<1.75).
* Highly segmented, specially in central EM region (>187k).
* Minimum Bias Trigger Scintillators (MBTS) using Tile hardware.
* Gains to increase dynamic range : LAr (high, medium, low), Tile (high, low).
* Used also for Trigger purposes (L1 and HLT).
* Tile has two channels used for a single cell (two sides of the scintillator plates).
* Upgrades trying to re-use a lot of the already developed code, specially EDMs.



How these calorimeter work ?
Check : https://www.youtube.com/watch?v=WAHTMG5nMK4
https://www.youtube.com/watch?v=mwpY7GamN_w
https://www.youtube.com/watch?v=si5QYCeQOOU

Main classes

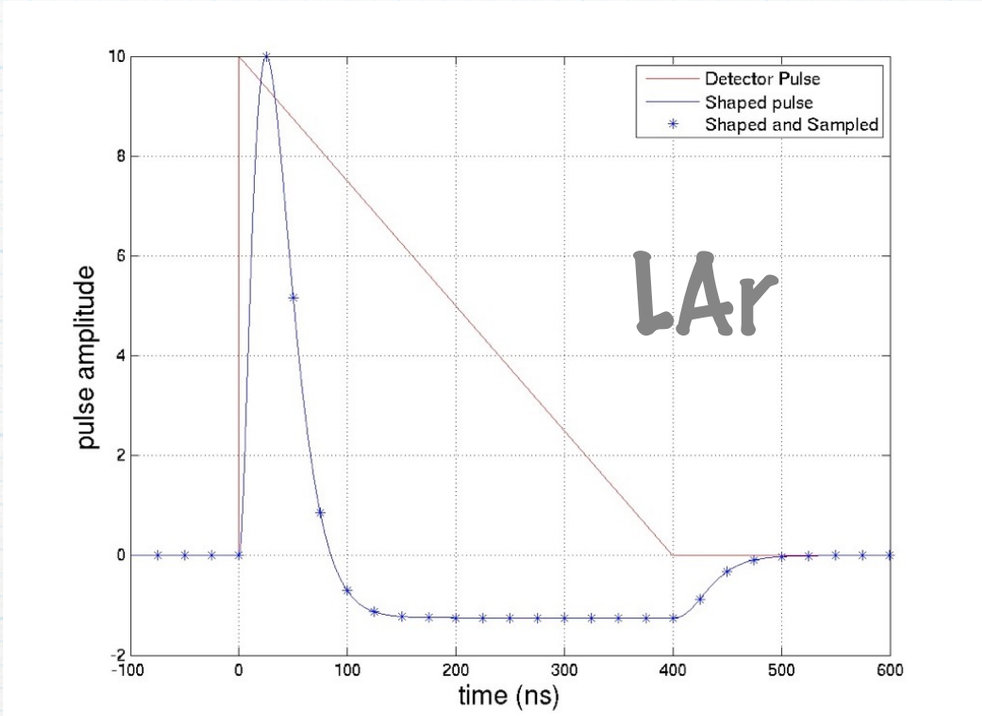
* hardware-related classes : Raw Channels / Digits.

* Physics related classes : CaloCell, LArCell, TileCell, geometry.

* CaloCluster. Signal State change (look at different calibrations).

Digits

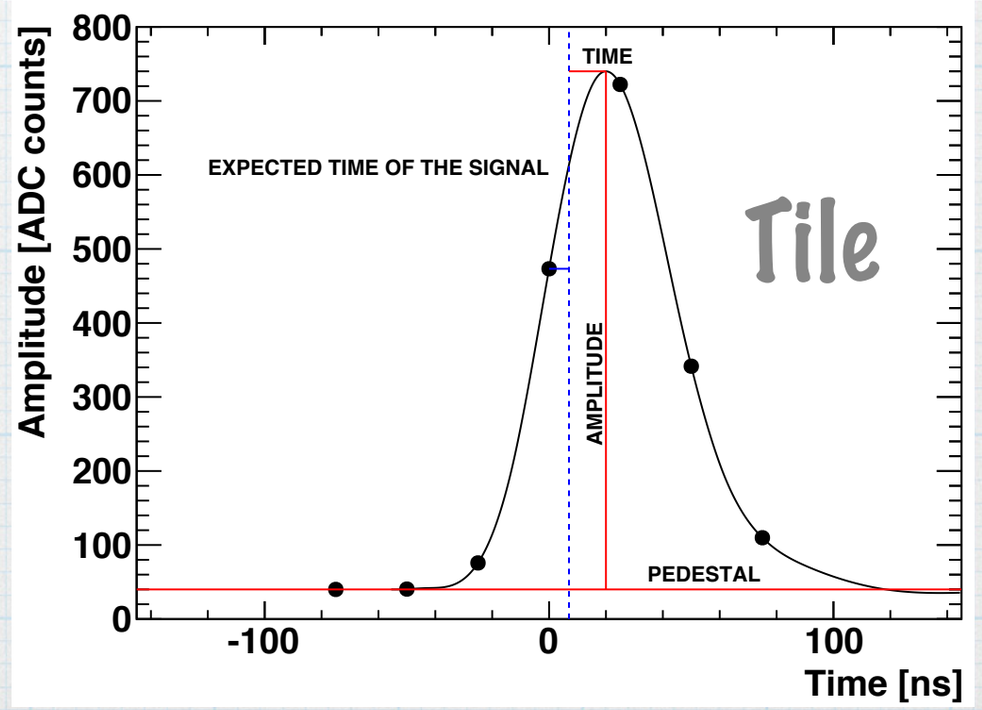
* LAr and Tile Digits represent the sampled signal from the analogue calorimeter pulse.
* usually only available in special MC productions or specific data samples, but also available in RAW for every cell above a given energy threshold.
* Associates to hardware identifier and Gain and samples.




```
class LArDigit  {

 private:

  /** @brief Online Identifier */
  HWIdentifier m_hardwareID;

  /** @brief gain */
  CaloGain::CaloGain m_gain;

  /** @brief vector of ADC samples */
  std::vector < short > m_samples;
```

TileDigits :

* hardware ID (with Gain information associated)
* samples



Raw Channels

* Representation of the hardware channel associated to a cell (two/cell for Tile) : Direct dump of Digital Signal Processing (DSP) units. No fancy calibration.
* hardware ID connected to front-end channel (readout). Energy directly related to pulse amplitude, time related to shift (LHC jitter, time of flight), quality of the pulse fitting by optimal filter algorithm.
* Usually available in RDOs (special production for trigger purposes).

LArRawChannel :
* hardware ID (with Gain information associated)
* energy
* time (shift from reference)
* quality/provenance (related to how well signal was fitted)
* gain

TileRawChannel :
* hardware ID (with Gain information associated)
* amplitude
* time (shift from reference)
* quality (related to how well signal was fitted)
* pedestal

CaloCell, LAr and Tile

* Associates energy, time, quality to physics relevant position (η,ϕ, layer, detector type coordinates).
* all coordinates come via geometry description (see next slide).
* Many possible improvements on top of RawChannels (average energy for missing cells, pedestal train-bunch structure compensation, High-Voltage).
* Full container available ("AllCalo") in ESDs level : no xAOD direct (ROOT-like) access.
* Derived classes LArCell and TileCell used for very specific functions (e.g: access to independent Tile readouts). In Tile, "Gain" comes independently from each photomultipliers.

CaloCell :
* Offline ID.
* energy
* time
* quality/provenance
* gain
* CaloDetDescrElement (geometry).

Geometry

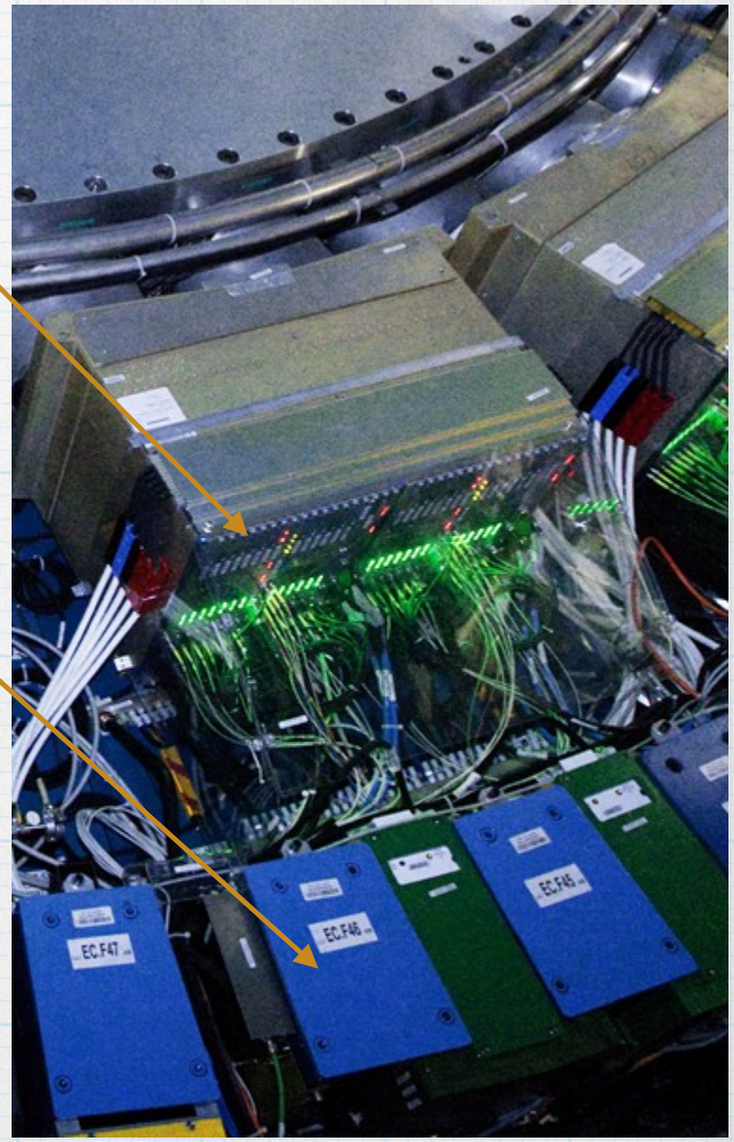
* Detector geometry has a raw version defined by xml dictionaries.
* cells are later re-aligned from DB in the "Start of Run" event.
* Precise position (η,ϕ, raw η, raw ϕ, x, y, z, r, raw x, raw y, raw z, raw r) and size (Δη, Δϕ, volume, Δr, Δx, Δy, Δz). Some methods (e.g: sampling layer) comes from yet another class (CaloDetDescriptor).
* Because loading these conditions depends on run number->IOV access, cells access cannot happen at ROOT level.

Containers

* LArRawChannels are organised by Front-End boards (128 channels) in LArRawChannelCollections. A Final LArRawChannelContainer contains all collections.

* TileRawChannels are organised by Tile Drawer (46 channels - a phi-wedge) into TileRawChannelCollections. The TileRawChannelContainer contains all collections as well.

* CaloCellContainer has all the cells (usually in ESD) and has fast cell access methods, specially if the container is complete (organisation by hash ID).

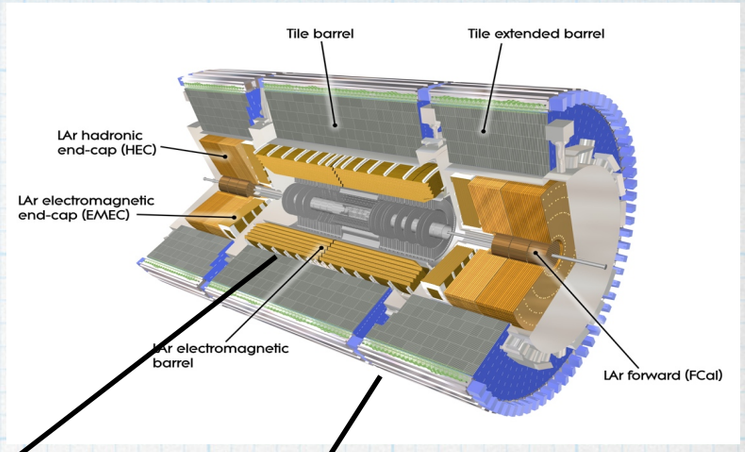


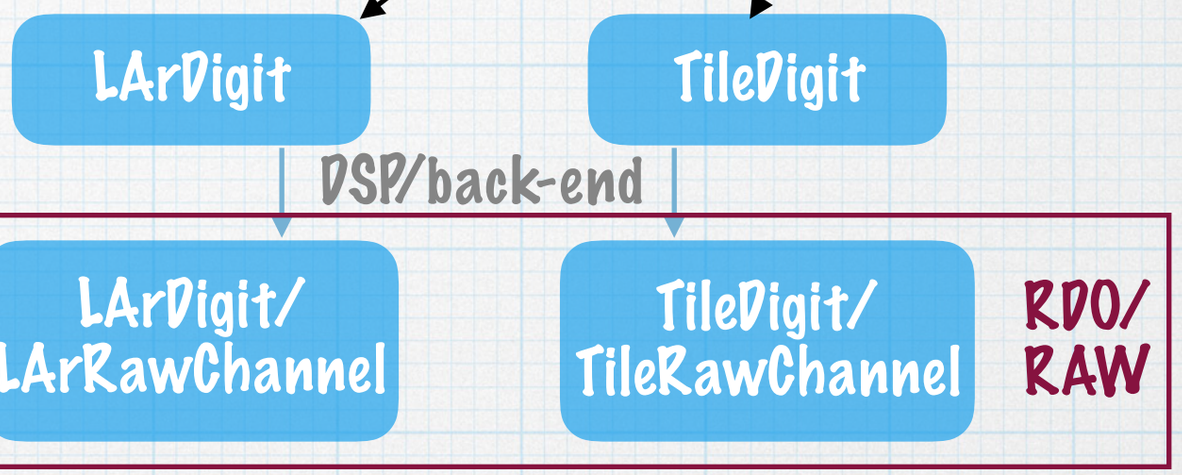
CaloCluster

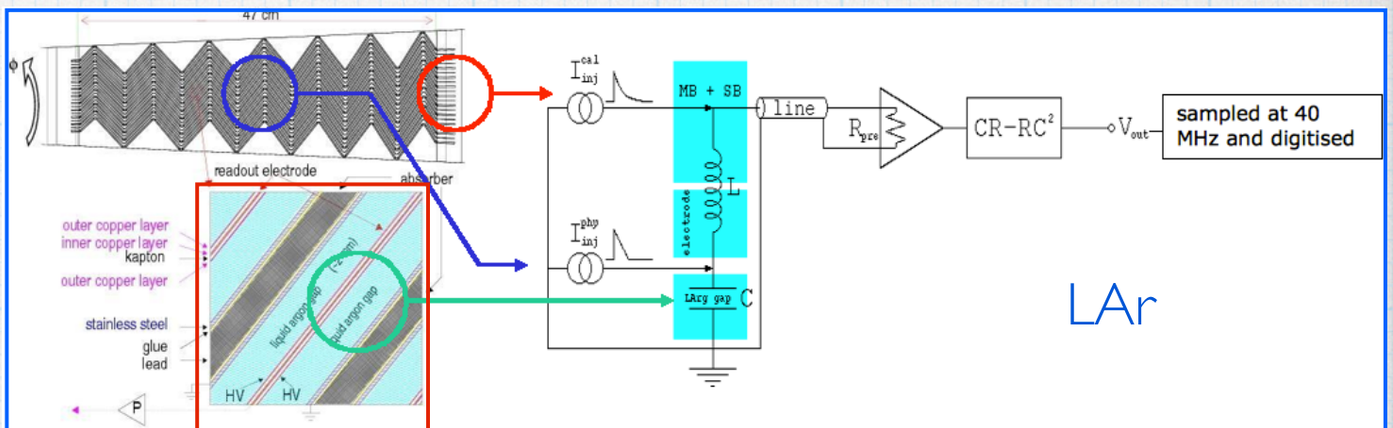
* CaloCluster available in xAOD. SignalState (calibrated, uncalibrated, etc) is available.
* CaloCluster has global properties (η,ϕ, energy, mass, time) as well as quantities by layer (eSample, etaSample, phiSample). Also FourMon_t.
* Multiple "size"s are defined to help differentiate between cluster types (like Sliding-window - window size, Topo Cluster - cluster growing thresholds).
* possibility to loop over cells composing the cluster (no ROOT-access!).
* Large number of Moments (variables associated to the cluster : cluster centroid, fraction of energy in Core, calibration, etc).

Calorimeter Reconstruction

* The digits are used to recover the information about amplitude.
* Detailed information about the detectors (obtained with special calibration data taking) is stored in DBs organised by hardware channel ID.
* Many special classes defined for DB access.





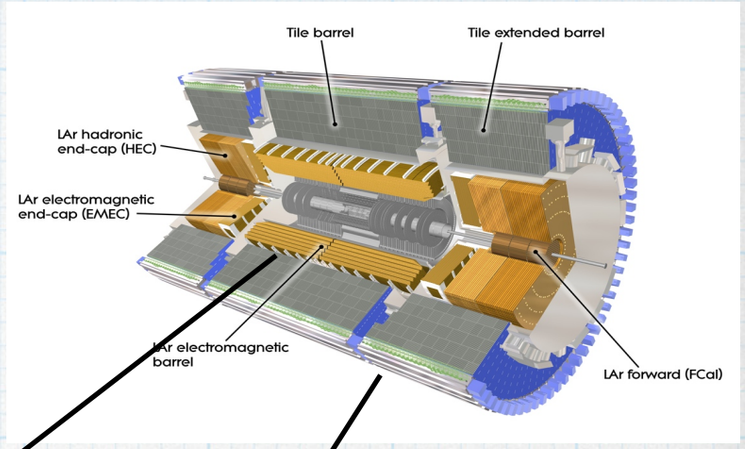


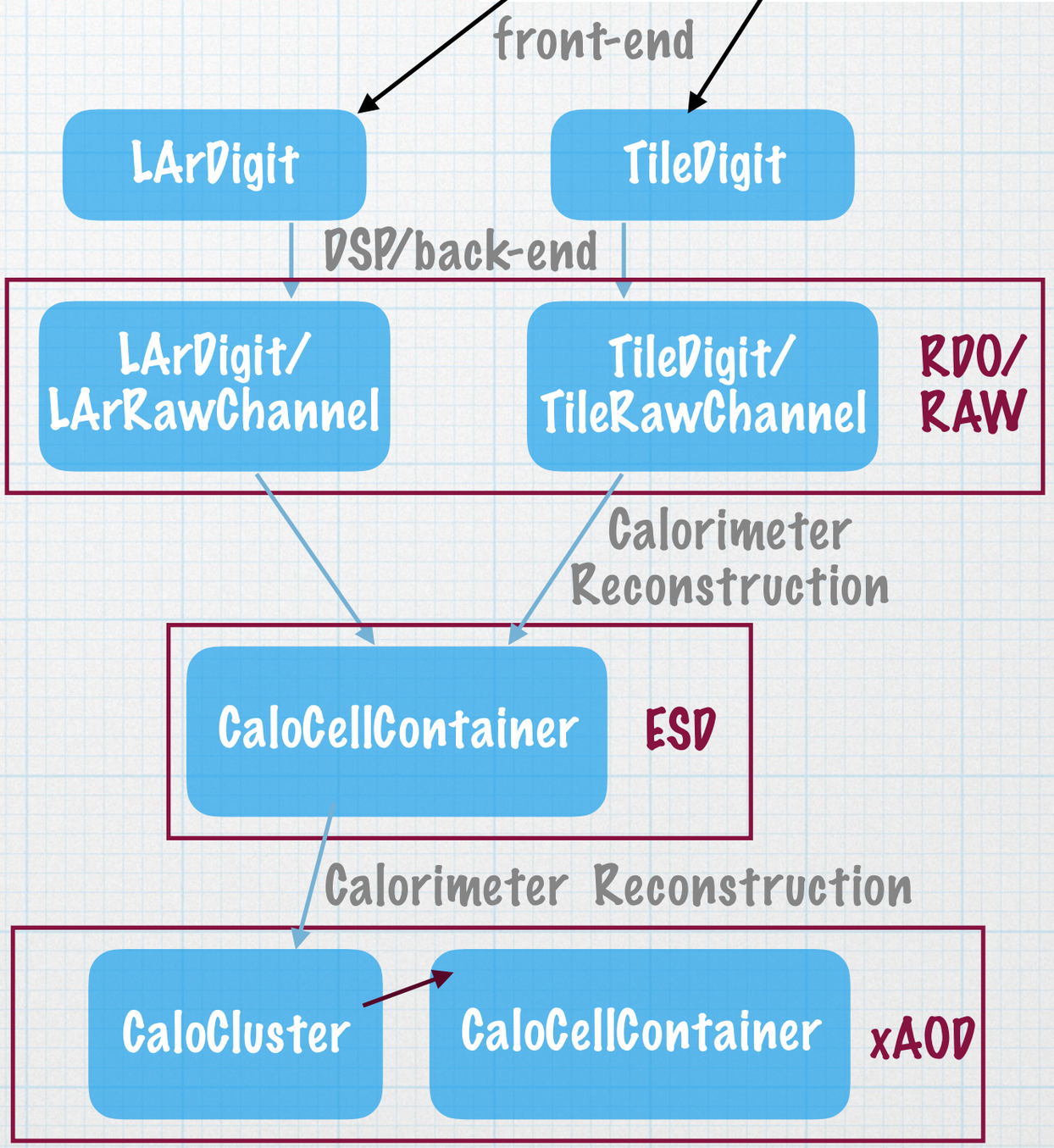
$$E_{\text{cell}} = F_{\mu A \to MeV} \cdot F_{DAC \to \mu A} \cdot \frac{1}{\frac{M_{phys}}{M_{cali}}} \sum_{i=1}^{M_{ramps}} R_i \left[\sum_{j=1}^{N_{samples}} a_j \left(s_j - p \right) \right]^i$$

Sampling Fraction : relation between the amount of energy in the LAr Gap and the lead absorber.

Calorimeter Reconstruction

* The acquisition chain starting from the front-end digitisation of the analog pulses.
* DSPs produce RawChannels (and for cells above an energy threshold, Digits are attached).
* Reconstruction produces CaloCells from RawChannels (or from Digits refitted at this stage). Many cell corrections are applied.
* Clusters are built from cells and recorded into xAOD. Group of cells associated to each CaloCluster can be stored (but only accessed within athena!).



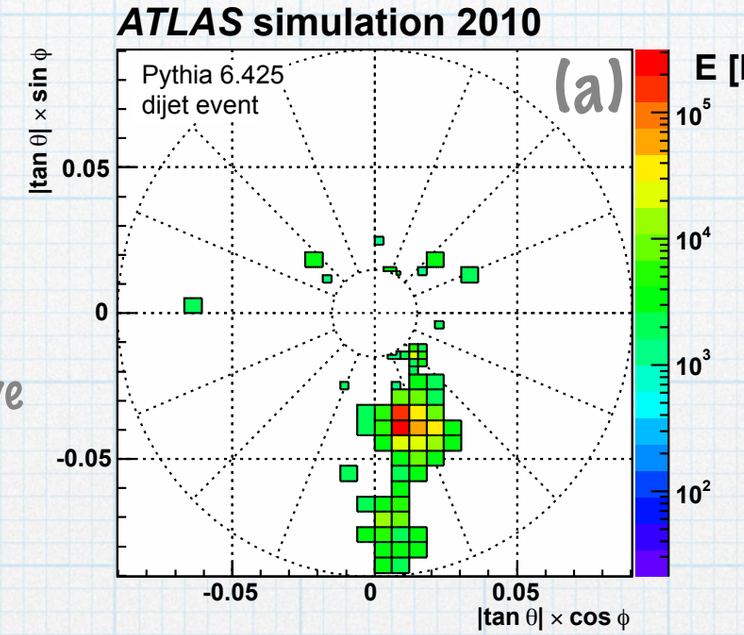


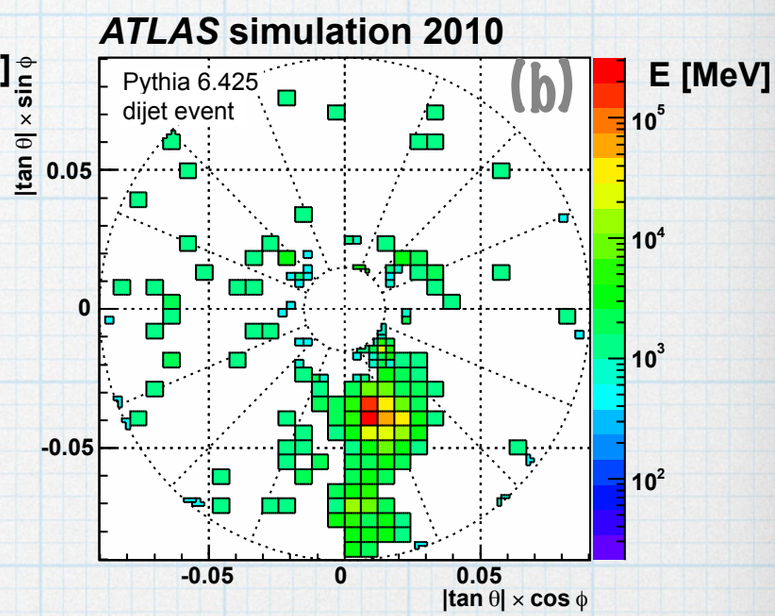
Clustering

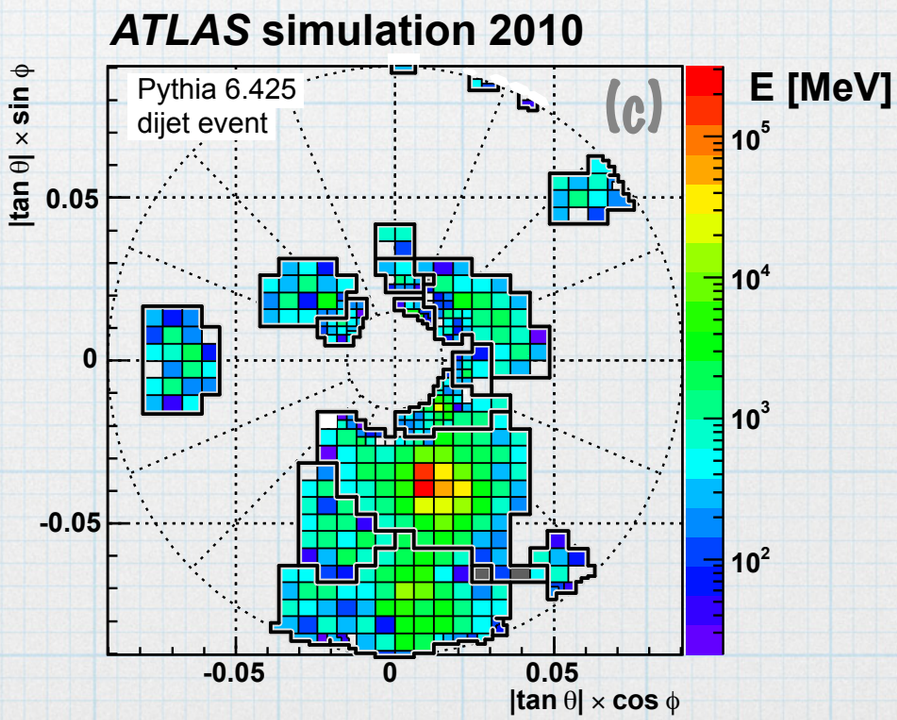
* two main approaches : Sliding window and topological clustering. Special calibration routines to each case.
* Sliding window clusters are found when a pre-defined size window of cells (say 3x3) in the full detector (or only EM) has more than a threshold of energy and is a local maximum.
* The window slides over the full detector.
* Single layer (or groups of layers) can be used to seed (pre-cluster window finding). Later other layers can be added. Cluster position is energy weighted position of the cells in its window.

topological clustering

* clusters do not have a fixed form.
* initial (a) seeds are found when cell energy > 4*sigma of noise (including pile-up noise).
* cluster grows (b) as touching cells have energy > 2*sigma.
* Final layer (c) added with cluster energy > 0*sigma.
* Cluster positions derived from cells (usually energy weighted sums of cells properties- UpdateKine).
* Special algorithms separate contributions (weights) of cells to two different clusters.
* Adjustable thresholds. Resilience to pile-up.







Associations

* Using online helpers for cells "hardware features" :

```
const HWIdentifier chid = (*cont_it)->channelID();
HWIdentifier FebID = m_onlineHelper->feb_Id(chid); // LArOnlineID
```

* Using offline helpers for cells "physics coordinates" or hash organisation. :

```
Identifier theCellID = aCell->ID();
IdentifierHash theCellHashID = m_calo_id->calo_cell_hash(theCellID);  //CaloCell_ID
```

* Cabling (connect online world - rawchannel/digit - to offline - cells) :

```
const Identifier id = m_larCablingSvc->cnvToIdentifier(chid); //LArCablingService
HWIdentifier onlId=m_larcablingSvc->createSignalChannelID(id);
```

Change State

* It is possible to change the state of a clusters and later come back to the original state :

```
{ // start scope
        CaloClusterChangeSignalStateList list;
        for(xAOD::CaloClusterContainer::const_iterator itr = m_pCaloClusterContainer->begin();
            itr!=m_pCaloClusterContainer->end(); ++itr)
                list.add( (*itr), xAOD::CaloCluster::UNCALIBRATED );
        std::cout << "CLUSTER ISSUE check the first now it should be uncalibrated : "
        << (*m_pCaloClusterContainer->begin())->et() << " " << (*(m_pCaloClusterContainer->begin()))->et()
        << std::endl;
} // destroy scope // cluster should come back to its original state of mind...
```

Calorimeter EDM

* Calorimeter digits, rawchannels, cells, cluster presented in the order they appear in reconstruction.

* More detailed information (like cells geometry) stored in DB. No ROOT access.

* Clusters can be formed by different algorithms (sliding window or topological clustering) and accessed in xAOD level.

Event displays