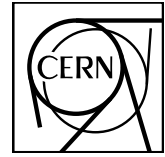


1



ALICE-INT-2012-xxx
April 20, 2013

2

3

EMCal Offline Documentation

4

EMCal collaboration

5

Email:alice-emcal-offline@cern.ch

6

Abstract

7

This document describes the EMCAL, it's offline geometry, the software to run a simulation or reconstruct data, the strategy to control the quality of the data, the trigger code and the analysis format.

8

9

10 **Contents**

11 **1 Introduction** **5**

12 1.1 Mechanical description of the EMCAL - Federico 5

13 1.2 Functional description of the EMCAL - Terry 9

14 **2 EMCAL geometry software - Marco +++** **11**

15 2.1 Classes description 11

16 2.2 Accessing the geometry 11

17 2.3 Geometry configuration options 12

18 2.4 Mapping 12

19 2.5 Tower index transformation methods 12

20 2.5.1 Absolute tower ID to Row/Column index 12

21 2.6 Tower index to local / global reference system position 15

22 2.6.1 Local coordinates 15

23 2.6.2 Global coordinates 17

24 2.7 Geometry Alignment 18

25 **3 EMCal OCDB/OADB - Marcel** **20**

26 3.1 Accessing a different OCDB 20

27 3.2 Energy calibration 21

28 3.3 Bad channels 22

29 3.4 Reconstruction parameters 22

30 3.5 Simulation parameters 25

31 3.6 Alignment 25

32 **4 Simulation code** **26**

33 4.1 Step Manger and Hits Creation 26

34 4.1.1 The EMCal Step Manager 26

35 4.1.2 Step Manager and Monte Carlo Setting 29

36 4.2 Digitization: SDigits and Digits - Evi 36

37	4.3	Raw data - David	36
38	4.4	How to make a simulation	36
39	5	Reconstruction code	38
40	5.1	Offline data base access	38
41	5.1.1	Energy calibration	38
42	5.1.2	Bad channels - Marie, Alexis	38
43	5.1.3	Alignment - Marco	38
44	5.2	Raw data fitting: from ADC sample to digits - David	38
45	5.3	Clusterization: From digits to clusters - Adam	39
46	5.3.1	Clusterization in the EMCal	40
47	5.3.2	Clusterizer V1	41
48	5.3.3	Clusterizer V2	42
49	5.3.4	Clusterizer V1 with unfolding	42
50	5.3.5	Clusterizer NxN	44
51	5.3.6	Cluster in the EMCal	45
52	5.4	Cluster-Track matching - Rongrong, Shingo, Michael	47
53	5.5	How to execute the reconstruction	48
54	6	Calibration and detector behavior	50
55	6.1	Calibration	50
56	6.1.1	Energy calibration: MIP calibration before installation - Julien	50
57	6.1.2	Energy calibration: π^0 - Catherine	50
58	6.1.3	Energy calibration: Run by run temperature gain variations - Evi, David	54
59	6.1.4	Time calibration - Marie	54
60	6.2	Alignment - Marco	56
61	6.3	Bad channel finding - Alexis	57
62	7	Trigger	58
63	7.1	L0 - Jiri	58

64	7.2	L1 - Rachid	58
65	7.3	L0-L1 simulation - Rachid	58
66	8	The EMCal HLT online chain - Federico	58
67	8.1	Reconstruction components	59
68	8.1.1	RawAnalyzer	59
69	8.1.2	DigitMaker	60
70	8.1.3	Clusterizer	60
71	8.2	Trigger components	61
72	8.2.1	Cluster trigger	62
73	8.2.2	Electron trigger	62
74	8.2.3	Jet trigger	63
75	8.3	Monitoring components	64
76	9	Analysis format and code	67
77	9.1	Calorimeter information in ESDs/AODs	67
78	9.1.1	AliVEvent (AliESDEvent, AliAODEvent)	67
79	9.1.2	AliVCaloCluster (AliESDCaloCluster, AliAODCaloCluster)	68
80	9.1.3	AliVCaloCells (AliESDCaloCells, AliAODCaloCells)	70
81	9.1.4	AliVCaloTrigger (AliESDCaloTrigger, AliAODCaloTrigger) - Rachid	71
82	9.2	Macros	71
83	9.3	Code example	71
84	9.4	Advanced utilities : Reconstruction/corrections of cells, clusters during the	
85		analysis	72
86	9.4.1	AliEMCALRecoUtils	72
87	9.4.2	Tender : AliEMCALTenderSupply	72
88	9.4.3	Particle Identification with the EMCal	72
89	10	Run by run QA, how to and code	76
90	10.1	Online - Francesco, Michael	76
91	10.1.1	Creation and checking of online QA histograms (AMORE)	76

92	10.1.2 How to's for EMCAL AMORE experts	77
93	10.1.3 Some more informations	79
94	10.2 Offline - Marie	79
95	10.2.1 Creation of offline QA histograms	80
96	10.2.2 Run by Run Quality Assurance	81
97	10.2.3 Period Quality Assurance and stability of EMCAL	82
98	10.3 Event display	84
99	10.4 Logbook tips	84

100 **1 Introduction**

101 This document is addressed to those who want to work with the EMCal software. It explains
102 the different steps to have the data taken ready to be analyzed. It is divided in 2 blocks: a first
103 one with the description of the procedures needed to cook the data and a second one with the
104 reconstruction and simulation offline code.

105 For a fast introduction on the code and how it works you can have a look to the EMCal for
106 beginners guide [1]. Some other interesting references are the AliRoot primer [3], the offline
107 AliRoot page [2], and the installation page from Dario Berzano [5].

108 **1.1 Mechanical description of the EMCAL - Federico**

109 The chosen technology is a layered Pb-scintillator sampling calorimeter with a longitudinal pitch
110 of 1.44 mm Pb and 1.76 mm scintillator with longitudinal Wavelength Shifting Fiber (WLS) light
111 collection. The full detector spans $\eta = -0.7$ to $\eta = 0.7$ with an azimuthal acceptance of $\Delta\phi = 107^\circ$
112 and is segmented into 12,288 towers, each of which is approximately projective in η and ϕ to
113 the interaction vertex. The towers are grouped into super modules of two types: full size which
114 span $\Delta\phi = 20^\circ$ and 1/3 size which span $\Delta\phi = 6.67^\circ$. There are 10 full size and 2, 1/3-size super
115 modules in the full detector acceptance (Fig. 1).

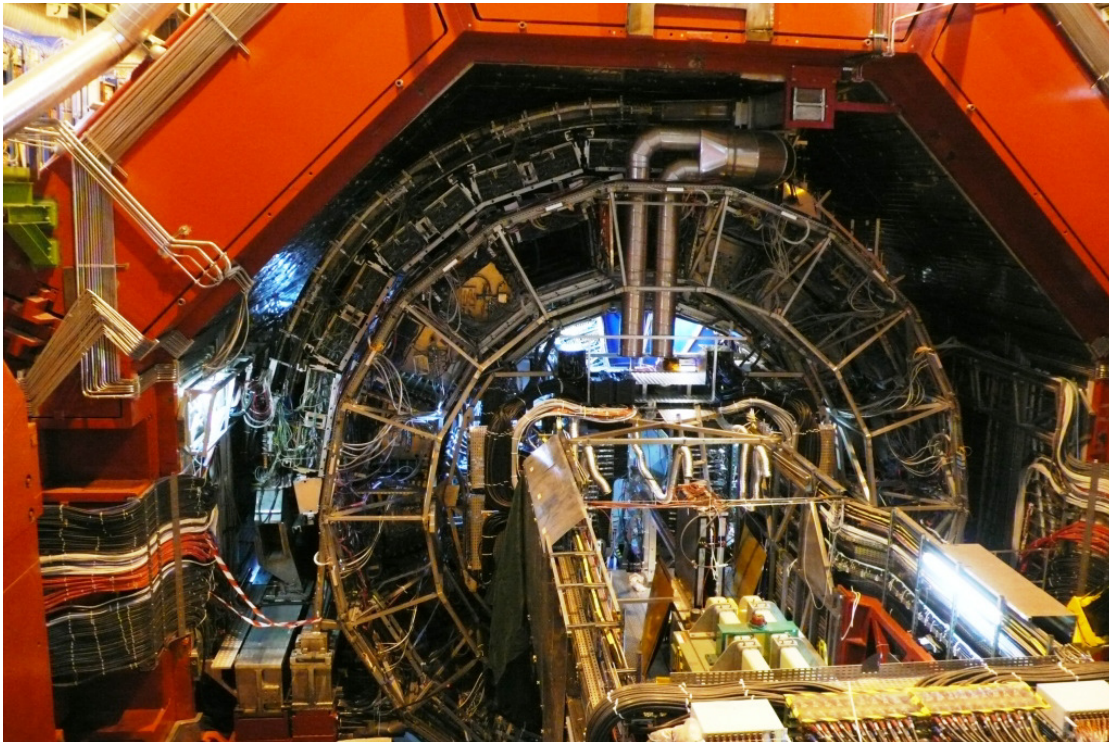


Fig. 1: Azimuthal view from the A-side (opposite to the di-muon arm) of the full EMCal as installed into the ALICE detector. The two 1/3-size super-modules are visible at 9 o'clock position.

116 The super module is the basic structural units of the calorimeter. These are the units handled as
117 the detector is moved below ground and rigged during installation.

118 Fig. 2 shows a full size super module with 12×24 modules configured as 24 strip modules of
119 12 modules each. The supporting mechanical structure of the super module hides the stacking
120 into a nearly projective geometry which can be inferred by the different tilt of the strip modules
121 going from the left to the right part of the picture. The electronics integration pathways are also
122 visible. Each full size super module is assembled from $12 \times 24 = 288$ modules arranged in 24
123 strip modules of 12 modules each.



Fig. 2: View of one EMCal super-module during the installation into the ALICE detector. The cradle holds the 24 strip modules into a mechanically rigid unit. Each strip module holds 12 unit modules. On the right side the two electronics crates are visible.

124 Each module has a rectangular cross section in the ϕ direction and a trapezoidal cross section
125 in the η direction with a full taper of 1.5° . The resultant assembly of stacked strip modules is
126 approximately projective with an average angle of incidence of less than 2° in η and less than 5°
127 in ϕ . An assembled strip module is shown in Fig. 3.

128 The smallest building block of the calorimeter is the individual module illustrated in Fig. 4 Each
129 individual module contains $2 \times 2 = 4$ towers built up from 77 alternating layers of 1.44 mm Pb
130 and 1.76 mm polystyrene, injection molded scintillator. White, acid free, bond paper serves as
131 a diffuse reflector on the scintillator surfaces while the scintillator edges are treated with TiO₂



Fig. 3: View of a fully assembled strip module. The photo shows the APD+CSP package and copper shielding mounted light guide fixture. On the right part of the photo the LED UV optical fiber distribution system is visible. Each strip module, is cabled via 3 T-Cards visible in the center of the assembly.

132 loaded reflector to provide tower to tower optical isolation and improve the transverse optical
 133 uniformity within a single tower. The Pb-scintillator stack in a module is secured in place by the
 134 static friction between individual layers under the overall load of 350 kg. The module is closed
 135 by a skin of 150 μm thick stainless steel screwed by flanges on all four transverse surfaces to
 136 corresponding front and rear aluminum plates. This thin stainless skin is the only inert material
 137 between the active tower volumes. The internal pressure in the module is stabilized against
 138 thermal effects, mechanical relaxation and long term flow of the Pb and/or polystyrene by a
 139 customized array of 5 non-linear spring sets (Bellville washers) per module. In this way, each
 140 module is a self supporting unit with a stable mechanical lifetime of more than 20 years when
 141 held from its back surface in any orientation as when mounted in a strip module.

142 All modules in the calorimeter are mechanically and dimensionally identical. The front face
 143 dimensions of the towers are $6 \times 6 \text{ cm}^2$ resulting in individual tower acceptance of $\Delta\eta \times \Delta\phi =$
 144 0.014×0.014 at $\eta=0$. The EMCAL design incorporates a moderate detector average active vol-
 145 ume density of 5.68 g/cm^3 which results from a 1:1.22 Pb to scintillator ratio by volume. This
 146 results in a compact detector consistent with the EMCAL integration volume at the chosen detec-
 147 tor thickness of 20.1 radiation lengths.



Fig. 4: The first 1.5° tapered module of the EMCal generation II prototype produced in EU shown. The module's internal compression is maintained by a set of 5 Bellville washers (non linear springs) acting between the top and bottom containment Al plates to prevent the delamination of the internal Pb-scintillator sandwich.

148 As described above, the super module is the basic building block of the calorimeter. Starting
149 with 288 individual modules which are rather compact and heavy, the main engineering task is
150 to create a super module structure which is rigid, with small deflections in any orientation yet
151 does not require extensive, heavy external stiffening components that would reduce the volume
152 available for the active detector. The solution adopted for the ALICE EMCal is to develop
153 a super module crate which functions not as a box for the individual modules but rather an
154 integrated structure in which the individual elements contribute to the overall stiffness. The
155 super module crate is effectively a large I-beam in which the flanges are the long sides of the
156 crate and the 24 rows of strip modules together. This configuration gives to the super module
157 good stiffness for both the 9 o'clock and 10 o'clock locations. For the 12 o'clock location, the
158 I-beam structure of the super module is augmented by a 1 mm thick stainless steel forward sheet
159 (traction loaded), which controls the bending moment tending to open the crate main sides, and
160 helps to limit deflection of strip modules. Ridges are provided on the interior surfaces of the crate
161 to allow precision alignment of the strip modules at the correct angle. The stiffness given by this
162 I-beam concept allows the use of non-magnetic light alloys for main parts of the super module
163 crate. Parts of the super module crate will be made mainly from laminated 2024 aluminum alloy

164 plates. The two main sides (flanges of the I-beam) of the crate will be assembled from 2 plates,
165 25 mm and 25 mm thick, bolted together and arranged so as to approximately follow the taper
166 of the 20 degree sector boundary. Each of the 24 rows of a super module contain 12 modules
167 as described above. Each of the modules is attached to a transverse beam by 3.4 mm diameter
168 stainless steel screws. The 12 modules and the transverse beam form a strip module. The strip
169 module is 1440 mm long, 120 mm wide, 410 mm thick. The total weight of the strip module is
170 approximately 300 kg and like module, it is a self supporting unit. The transverse beam, which is
171 the structural part of the strip module, is made from cast aluminum alloy with individual cavities
172 along its length where the fibers emerging from towers are allowed to converge. The casting
173 process is well suited to forming these cavities and the overall structure, saving considerable
174 raw material and machining time.

175 In addition to functioning as a convenient structural unit which offers no interference with the
176 active volume of the detector and forming the web of the I-beam structure of the super module,
177 the transverse beam of the strip module provides protection for the fibers, a structural mount
178 for the light guide, APD and charge sensitive preamplifier and a light tight enclosure for these
179 elements.

180 **1.2 Functional description of the EMCAL - Terry**

181 **** need some additional info on PE APDs *****

182 Particles traversing the calorimeter, in particular photons and electrons, will deposit energy in
183 different towers. The EMCAL reconstruction measures such energy per tower, forms clusters of
184 cells produced by a given particle, and if possible matches them with particles detected by the
185 tracking detectors in front of EMCAL (charged particles).

186 Scintillation photons produced in each tower are captured by an array of 36 Kuraray, Y-11,
187 double clad, WLS fibers that run longitudinally through the Pb/scintillator stack. Each fiber
188 terminates in an aluminized mirror at the front face end of the module and is integrated into
189 a polished, circular group of 36 at the photo sensor end at the back of the module. The fiber
190 bundles are pre-fabricated and inserted into the towers after the module mechanical assembly
191 is completed. The 36 individual fibers are packed into a circular array 6.8 mm in diameter and
192 held in place inside a custom injection molded grommet by Bicon BC-600 optical cement. An
193 optical quality finish is applied to the assembled bundle using a diamond polishing machine. At
194 the other end of the bundle, individual fibers are similarly polished and mirrored with a sputtered
195 coat of aluminum thick enough to ensure the protection of the inner mirror. The response of the
196 Al-coated fiber is considerably flatter with an overall increase in efficiency in the range of about
197 25% in the vicinity of shower maximum (i.e. the location of the highest energy deposition for
198 an electromagnetic shower). This number accounts for material immediately in front of the
199 detector; which ranges between 0.4 and 0.8 radiation lengths, and assumes 5.5 - 6.0 radiation
200 lengths for shower maximum for 10 GeV photons. At this depth in the detector, the mirrored
201 fiber response is very uniform does not contribute to the non-linearity of the detector as a whole.

202 Other factors which can significantly impact the electromagnetic performance of the calorime-
203 ter, include scintillator edge treatment and the density of the wavelength shifting fiber readout

204 pattern and the material chosen for the interlayer diffuse reflector. For scintillator edge treatment
205 and fiber density, advantage was taken from the extensive studies made by the LHCb collabora-
206 tion for their ECAL. In particular, a diffuse reflector edge treatment was adopted, such as that
207 obtained with Bicron Titanium Dioxide loaded white paint (BC622A) with a total fiber density
208 of about one fiber per cm^2 . In the case of the interlayer diffuse reflector, a white, acid free, bond
209 paper was used in place of the Teflon based commercial TYVEK. While TYVEK produces
210 slightly better surface reflectivity, its coefficient of friction is too low to permit its use in this
211 design where the module's mechanical stability depends somewhat on the interlayer friction.

212 The 6.8 mm diameter fiber bundle from a given tower connects to the APD through a short light
213 guide/diffuser with a square cross section of 7 mm \times 7 mm that tapers slowly down to 4.5 mm
214 \times 4.5 mm as it mates (glued) to the 5 mm \times 5 mm active area of the photo sensor. The 4
215 pre-fabricated fiber bundles are inserted into the towers of a single module.

216 The selected photo sensor is the Hamamatsu S8664-55 Avalanche Photo Diode *****

217 This photodiode has a peak spectral response at a wavelength of 585 nm compared to an emission
218 peak of 476 nm for the Y-11 fibers. However, both the spectral response and the quantum
219 efficiency of the APD are quite broad with the latter dropping from the maximum by only 5%
220 at the WLS fiber emission peak. At this wavelength, the manufacturer's specification gives a
221 quantum efficiency of 80%.

222 **2 EMCAL geometry software - Marco +++**

223 This page is intended for a description of the EMCAL geometry and the methods to access it.
224 *This is a very preliminary version that needs work.*

225 **2.1 Classes description**

226 The EMCAL geometry is implemented in several classes : (right now very brief description, it
227 should be completed)

- 228 – AliEMCALGeoUtils: Steering geometry class. No dependencies on STEER or EMCAL
229 non geometry classes. Can be called during the analysis without loading all aliroot classes.
- 230 – AliEMCALGeometry: Derives from AliEMCALGeoUtils, contains dependencies on other
231 EMCAL classes (AliEMCALRecPoint).
- 232 – AliEMCALEMCGeometry: Does the geometry initialization. Does all the definitions of
233 the geometry (towers composition, size, Super Modules number ...)
- 234 – AliEMCALGeoParams: Class container of some of the geometry parameters so that it can
235 be accessed everywhere in the EMCAL code, to avoid "magic numbers". Its use has to be
236 propagated to all the code.
- 237 – AliEMCALShishKebabTrd1Module: Here the modules are defined and the position of
238 the modules in the local super module reference system is calculated

239 **2.2 Accessing the geometry**

240 One can get the geometry pointer in the following ways:

- 241 – If galice.root is available:

```
242 1 AliRunLoader *rl = AliRunLoader::Open("galice.root",AliConfig::  
    GetDefaultEventFolderName(),"read");  
2 rl->LoadgAlice();//Needed to get geometry  
3 AliEMCALLoader *emcalLoader = dynamic\_cast<AliEMCALLoader*>(rl->  
    GetDetectorLoader("EMCAL"));  
4 AliRun * alirun = rl->GetAliRun();  
5 AliEMCAL * emcal = (AliEMCAL*)alirun->GetDetector("EMCAL"); AliEMCALGeometry  
    * geom = emcal->GetGeometry();  
6 else, if galice.root is not available:  
7 AliEMCALGeometry * geom = AliEMCALGeometry::GetInstance("EMCAL\_COMPLETE") ;
```

243

244 In this case you might need the file geometry.root if you want to access to certain methods
245 that require local to global position transformations. This file can be generated doing a simple
246 simulation, it just contains the transformation matrix to go from global to local.

247 The way to load this file is:

```
248 TGeoManager::Import ("geometry.root");
```

249 The transformation matrices are also stored in the ESDs so if you do not load this file, you can
250 have to load these matrices from the ESDs.

251 If you want to see different parameters used in the geometry printed (cells centers, distance to
252 IP, etc), one just has to execute the method PrintGeometry().

253 **2.3 Geometry configuration options**

254 Right now the following geometry options are implemented:

- 255 – EMCAL_COMPLETE: 12 Super Modules (2 half Super Modules)
- 256 – EMCAL_FIRSTYEAR: 4 Super Modules (year 2010)
- 257 – EMCAL_FIRSTYEARV1: 4 Super Modules, corrected geometry (year 2010)
- 258 – EMCAL_COMPLETEV1: 10 Super Modules, corrected geometry (year 2011)
- 259 – EMCAL_COMPLETE12SMV1: 12 Super Modules (10+2/3), corrected geometry (year
260 2012)

261 Other options exists but need to be removed as they **should not be used**:

- 262 – EMCAL_PDC06: Old geometry, for reading old data (which do not exist anymore).
- 263 – EMCAL_WSU: Prototype geometry.

264 By default, the geometry is loaded with the EMCAL_COMPLETE12SMV1 configuration.

265 **2.4 Mapping**

266 The tower row/column mapping online and offline follows the alice numbering convention. Fig-
267 ures 5 to 7 display the position of the super modules from different points of view and the
268 position of the tower index in them.

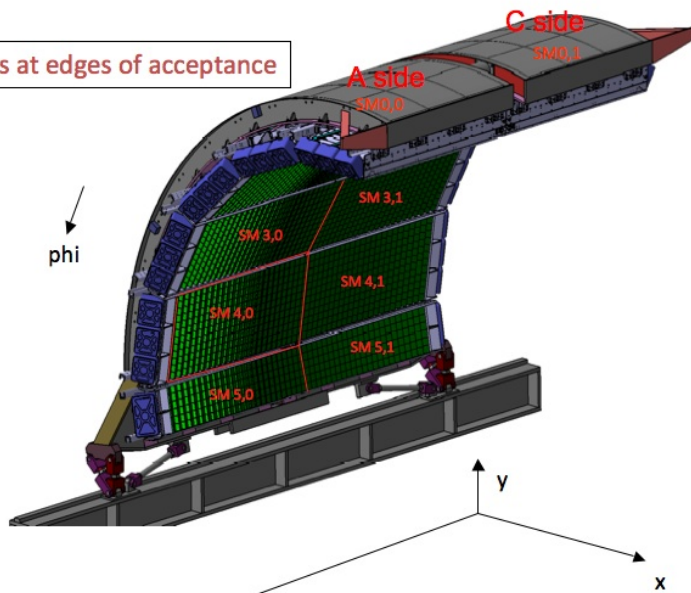
269 **2.5 Tower index transformation methods**

270 **2.5.1 Absolute tower ID to Row/Column index**

271 Each EMCAL supermodule is composed of 24x48 towers (phi,eta), grouped in 4x4 modules.
272 Each tower (even each module) has a unique number assigned, called in the code "absolute
273 ID" number (absId). This number can be transformed into a row (phi direction) or column (eta
274 direction) index. The procedure to go from the absId to the (row, col) formulation or viceversa
275 is as follow:

2 x(5+1/3) SM's

Note: FEE crates at edges of acceptance



z Coordinate system according ALICE-INT-2003-038

Fig. 5: Position of the super modules

276 – From absId to col-row:

```

1  Int_t nSupMod, nModule, nIphi, nIeta, iphi, ieta;
2  //Check if this absId exists
3  if(!CheckAbsCellId(absId)) return kFALSE;
4  // Get from the absId the super module number, the module number and the eta-phi index
277  (0 or 1) in the module
5  GetCellIndex(absId, nSupMod, nModule, nIphi, nIeta);
6  // Get from the the super module number, the module number and the eta-phi index (0
   or 1) in the module the tower row (iphi) and column (ieta)
7  GetCellPhiEtaIndexInSModule(nSupMod, nModule, nIphi, nIeta, iphi, ieta);

```

278

279 – From col-row to absId, following the same notation as above:

```

1
280 2  absid = GetAbsCellIdFromCellIndexes(nSupMode, iphi, ieta);

```

EMCAL, seen from back/magnet side – looking towards IP through EMCAL from the top of the CalFrame. 4 installed SuperModules; sector 0 is the top/highest sector. Standard view. Row as Y-axis, and Column as X-axis (LED amplitude plots).

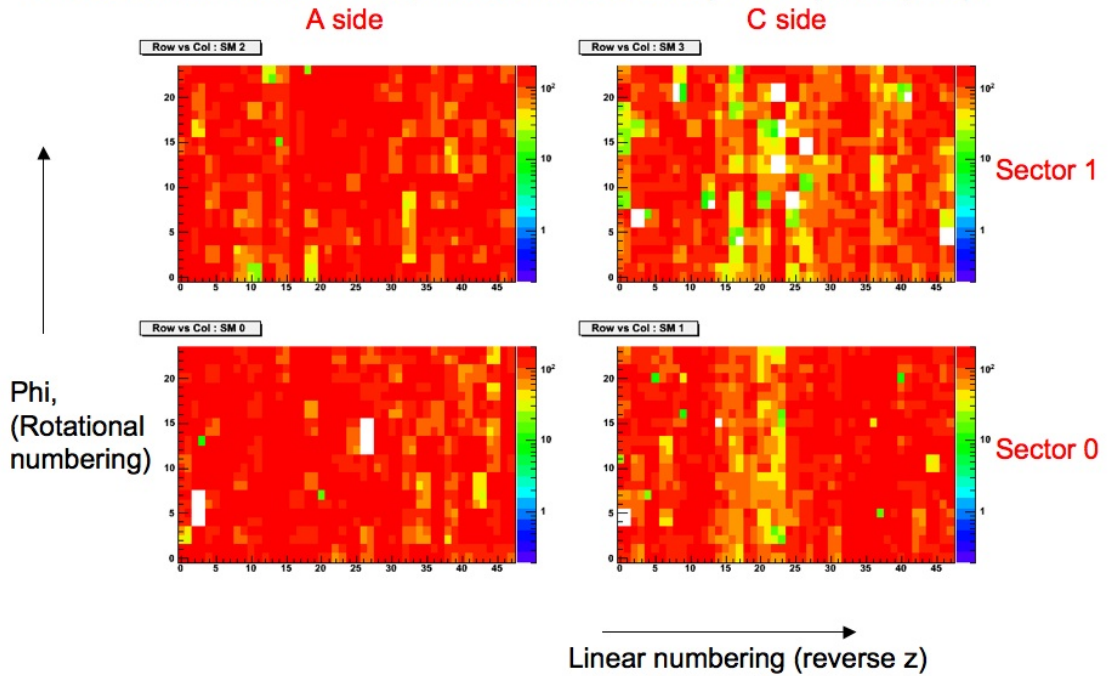


Fig. 6: EMCAL seen from the magnet side with 4 SMs.

281

282 OR

```
283 1
284 2 absid = GetAbsCellId(nSupMod, nModule, nIphi, nIeta);
```

284

285 – Other interesting method is

```
286 1
287 2 Int\_t GetSuperModuleNumber(Int\_t absId)
```

287

EMCAL, seen from back/magnet side – looking towards IP through EMCAL from the bottom (alternative view) of the CalFrame.

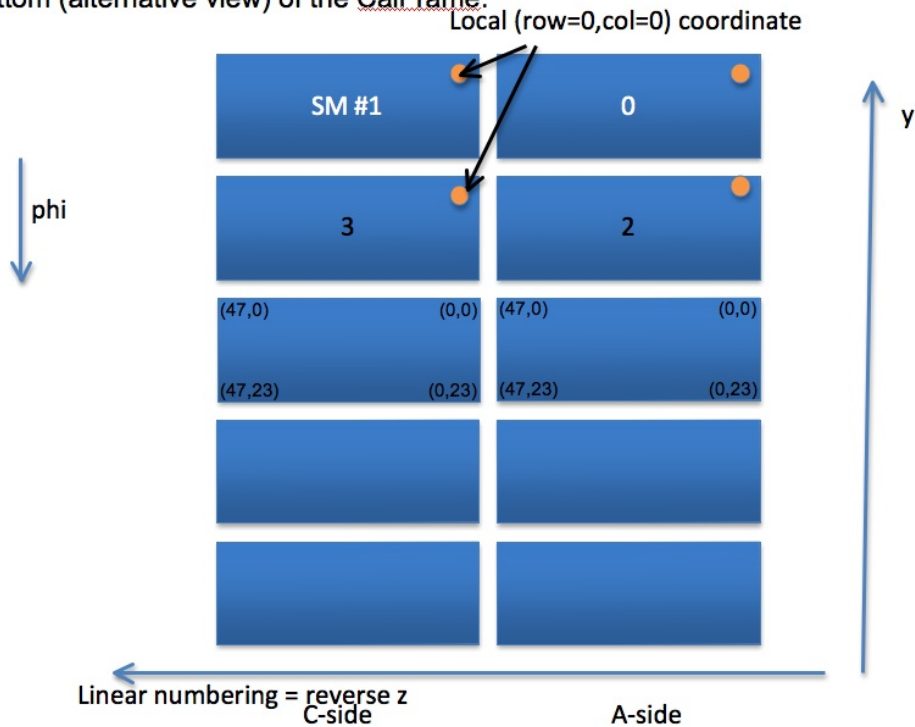


Fig. 7: EMCAL geometrical numbering.

288 2.6 Tower index to local / global reference system position

289 2.6.1 Local coordinates

290 To correlate the tower index and its position in local coordinates, the following methods are
291 available:

```

1  Bool_t AliEMCALGeoUtils::RelPosCellInSModule(Int_t absId, Double_t &xr,
2  Double_t &y, Double_t &zr) const;
3  Bool_t AliEMCALGeoUtils::RelPosCellInSModule(Int_t absId, Double_t loc[3])
292  const;
4
5  Bool_t AliEMCALGeoUtils::RelPosCellInSModule(Int_t absId, TVector3 &vloc)
const;
```

293

294 To which the input is the absId and the output are the coordinates of the center of towers in the

295 local coordinates of the Super Module. This method gets the column and row index of the cell
296 from the absId, independently of the Super Module (like above), and gets the center of the cell
297 from 3 arrays (x,y,z) filled with such quantities. Such central positions are calculated during the
298 initialization of the geometry, where the arrays are filled, in the method :

```
1  
299 2  AliEMCALGeoUtils::CreateListOfTrd1Modules()
```

300

301 «««Someone else should explain how it works»»»

302 In case we calculate the cluster position, things are a bit different.

303 ««« This explanation should go to the clusterization section»»»

304 This is done in

```
3051 void AliEMCALRecPoint::EvalLocalPosition()
```

306

307 First we calculate the cell position with the method

```
1  AliEMCALGeometry::RelPosCellInSModule(Int_t absId, Int_t maxAbsId, Double_t  
308    tmax, Double_t &xr, Double_t &yr, Double_t &zr)
```

309

310 The calculation of the cell position done here is different in the "x-z" but the same in "y".

311 ««« «««Someone else should explain how it works»»»»»

312 In this particular case the position calculation per tower depends on the position of the maxi-
313 mum cell, and the sum of the energy of the cells of the cluster. The maximum depth (tmax) is
314 calculated with the method

```

1   Double\ _t AliEMCALRecPoint::TmaxInCm(const Double\ _t e){
2
3       //e: energy sum of cells
4
5   static Double\ _t ca = 4.82; // shower max parameter – first guess; ca=TMath::Log(1000./8.07)
6
7       static Double\ _t x0 = 1.23; // radiation lenght (cm)
3158
9       static Double\ _t tmax = 0.; // position of electromagnetic shower max in cm
10
11      tmax = TMath::Log(e) + ca+0.5;
12
13      tmax *= x0; // convert to cm
14
15  }

```

316

317 After the cells position of the cluster is accessed, the position of the cluster is calculated averaging
318 the cell positions with a logarithmic weight:

```

1   w(cell i) = TMath::Max( 0., logWeight + TMath::Log( energy[cell i] / summed\
319   _cluster\_cell\_energy ));

```

320

321 where the logWeight was chosen to be 4.5 (this value was taken from PHOS, never optimized as
322 far as I know)

323 So in the end the position, is

```

3241 f = Sum(f(i) * w(i))/Sum(w(i))

```

325

326 where f=x,y,z.

327 2.6.2 Global coordinates

328 To transform from local to global we have the methods

```

1
2 void GetGlobal(const Double\<_t *loc, Double\<_t *glob, int ind) const;
3
4 void GetGlobal(const TVector3 \&vloc, TVector3 \&vglob, int ind) const;
329
5
6 void GetGlobal(Int\<_t absId, Double\<_t glob[3]) const;
7
8 void GetGlobal(Int\<_t absId, TVector3 \&vglob) const;

```

330

331 These methods take the local coordinates and transform them into global coordinates using the
332 transformation matrix of the Super Module.

```

1
2
3333 TGeoHMatrix* m = GetMatrixForSuperModule(nSupMod);
4
5 if(m) m->LocalToMaster(loc, glob);

```

334

335 GetGlobal is called in the following useful methods in the geometry class:

336 – Return the eta and phi angular position of the cell from the AbsId

```

1 void EtaPhiFromIndex(Int\<_t absId, Double\<_t \&eta, Double\<_t \&phi) const
337 ;
2 void EtaPhiFromIndex(Int\<_t absId, Float\<_t \&eta, Float\<_t \&phi) const;

```

338

339 – Print information of the cells. For "pri>0" returns more information. "tit" has not much
340 use, this value is printed.

```

341 1 void PrintCellIndexes(Int\<_t absId, int pri, const char *tit)

```

342

343 2.7 Geometry Alignment

344 AliRoot contains a frame for the correction of the misplacement of geometry objects with respect
345 to the ideal positions which are kept in the STEER/ directory of the following classes:

```
1 AliAlignObj  
2 AliAlignObjMatrix  
346 AliAlignObjParams  
3  
4 AliAlignmentTracks
```

347

348 The class AliEMCALSurvey creates the corrections to the alignable objects. The class AliEM-
349 CALSurvey was established to take the survey parameters from OCDB, calculate the shift in
350 position of the center of the end faces of the supermodules from the nominal position, and con-
351 vert this to a transformation matrix for each supermodule which is applied to correct the global
352 position of the supermodules. All calculations of global positions would then use these corrected
353 supermodule positions to determine their locations within the ALICE global coordinate system.

354 **3 EMCAL OCDB/OADB - Marcel**

355 OCDB is the Offline data base. It contains the different parameters used for simulation or re-
356 construction of the detectors or even the LHC machine parameters that might change for the
357 different run conditions.

358 OADB is the Offline Analysis data base.

359 The EMCAL OCDB (and other detectors OCDB) is divided in 3 directories that can be found in

360 `$ALICE_ROOT/OCDB/EMCAL`

361 – Calib: Very different type of information, from hardware mapping to calibration paramete-
362 ters.

363 – Align: Survey misplacements in geometry.

364 – Config: Detector configuration, temperatures

365 Inside these directories you will find other subdirectories with more specific types of parameters.
366 Each of the directories contains a file named in this way:

367 `Run(FirstRun)_(LastRun)_v(version)_s(version).root`

368 being the default and what you will find in the trunk

369 `Run0_999999999_v0_s0.root`

370 What is actually used for the real data reconstruction can be found in alien here:

371 `/alice/data/20XX/OCDB/EMCAL`

372 There are different repositories for different years (20XX). For the simulation productions, there
373 is another repository on the grid:

374 `/alice/simulation/2008/v4-15-Release/XXX/EMCAL`

375 which is divided into 3 other repositories: Ideal, Full and Residual. Each one is meant to repro-
376 duce the detector with different precision. For EMCAL, right now these 3 repositories contain
377 the same parameters.

378 The following section explain the elements stored and how to read and fill OCDB parameters.

379 **3.1 Accessing a different OCDB**

380 In the simulation/reconstruction macro a default OCDB needs to be specified if different from

381 `$ALICE_ROOT/OCDB`.

382 When running on the grid, one needs to set for example in a reconstruction of simulated data:

```
383 reco.SetDefaultStorage("alien://Folder=/alice/simulation/2008/v4-15-Release/  
384 Residual/");
```

385 If one or several OCDB files have been modified, the following line has to be added in the
386 simulation or reconstruction macro:

```
387 reco.SetSpecificStorage("EMCAL/Calib/Pedestals", "local://your/modified/local/OCDB  
388 ");
```

389 The file with the calibration coefficients needs to be stored in the directory :

```
390 /your/modified/local/OCDB/EMCAL/Calib/Data
```

391 If more of the OCDB files are modified, add the following line :

```
392 reco.SetSpecificStorage("EMCAL/Calib/", "local:/your/modified/local/OCDB");
```

393 with all the directories inside

```
394 \begin{lstlisting}  
395 /your/modified/local/OCDB/EMCAL/Calib/
```

396 3.2 Energy calibration

397 Calibration Coefficients tower by towers are stored in the following directory :

```
398 EMCAL/Calib/Data
```

399 What is stored is an object of the class AliEMCALCalibData which is a container of gains and
400 pedestals per tower. These coefficients are used in:

- 401 – Simulation: during the digitization, in AliEMCALDigitizer::Digitizer(), when calling
402 AliEMCALDigitizer::DigitizeEnergy(), to transform the deposited energy into ADC counts.
- 403 – Reconstruction: in AliEMCALClusterizerV1::Calibrate() called in AliEMCALCluster-
404 izer::MakeClusters(), when forming the cluster, to get the final cluster energy.

405 The macro

```
406 $ALICE_ROOT/EMCAL/macros/CalibrationDB/AliEMCALSetCDB.C
```

407 is an example on how to set the calibration coefficients per channel, or how to read them from
408 the OCDB file. This macro can set all channels with the same selected value or with random
409 values given a uniform or gaussian smearing of a selected input value. A simple example that
410 shows how to print the parameters is PrintEMCALCalibData.C

411 All channels in the simulation have the same value for the gains (0.0153 GeV/ADC counts) and
412 pedestal (set to 0 since the calorimeter works with Zero Suppressed data).

413 3.3 Bad channels

414 Storage for the bad channels map found in hardware are here :

415 `EMCAL/Calib/Pedestals`

416 The object stored is from the class `AliCaloCalibPedestal` used for monitoring the towers calibra-
417 tion and functionality. This class has the data member `TObjArray *fDeadMap` which consists
418 of an array of 12 TH2I (as many as Super Modules), and each TH2I has the dimension of 24x48
419 (number of towers in $\phi \times \eta$ direction), each bin corresponds to a tower. The content of each
420 entry in the histogram is an integer which represents the possible status:

```
421 enum kDeadMapEntry{kAlive = 0, kDead, kHot, kWarning, kResurrected,  
422     kRecentlyDeceased, kNumDeadMapStates};
```

423 Right now only the status `kAlive`, `kDead`, `kHot` and soon `kWarning` (soon, not yet) are set but,
424 the code is basically skipping all the channels that are `kDead` and `kHot`. The bad channel map is
425 used in the reconstruction code in 3 places:

- 426 – `AliEMCALRawUtils::Raw2Digits()` : Before the raw data time sample is fitted, the status
427 of the tower is checked, and if bad (`kHot` or `kDead`), the fit is not done. This avoids trying
428 to fit ill shaped samples. This step is optional though, right now default is to skip the bad
429 channels here. With the `RecParam OCDB` we can select to use it or not.
- 430 – `AliEMCALClusterizerV1::Calibrate()`: once the cluster is formed, to get the cluster energy
431 from its cells.
- 432 – `AliEMCALRecPoint::EvalDistanceToBadChannels()`: Evaluate the distance of a cluster
433 to the closest bad channel. During the analysis we may want to skip clusters close to a
434 bad channel. This time a bad channel is whatever is not `kAlive`.

435 The macro

436 `$ALICE_ROOT/EMCAL/macros/PedestalDB/AliEMCALPedestalCDB.C`

437 is an example on how to set the bad channel map and how to read it from a file. When executed,
438 it displays a menu that allows to set randomly as bad a given % of the towers. It also allows to
439 set the map from an input txt file, with the format like
440 `$ALICE_ROOT/EMCAL/macros/PedestalDB/map.txt` (this map file is the one used in the last
441 mapping in the raw OCDB). It can also read the OCDB file and display the 12 TH2I histograms
442 on screen.

443 3.4 Reconstruction parameters

444 The storage of the parameters used in reconstruction is done in

445 `EMCAL/Calib/RecoParam`

446 What is stored is an object of the class AliEMCALRecParam which is a container for all the
447 parameters used. There are different kind of parameters, we can distinguish them depending on
448 which step of the reconstruction are used as explained below.

449 **Raw data fitting and mapping**

- 450 – Double_t fHighLowGainFactor; // gain factor to convert between high and low gain
- 451 – Int_t fOrderParameter; // order parameter for raw signal fit
- 452 – Double_t fTau; // decay constant for raw signal fit
- 453 – Int_t fNoiseThreshold; // threshold to consider signal or noise
- 454 – Int_t fNPedSamples; // number of time samples to use in pedestal calculation
- 455 – Bool_t fRemoveBadChannels; // select if bad channels are removed before fitting
- 456 – Int_t fFittingAlgorithm; // select the fitting algorithm
- 457 – static TObjArray* fgkMaps; // ALTRO mappings for RCU0..RCUX

458 **Clusterization**

- 459 – Float_t fClusteringThreshold ; // Minimum energy to seed a EC digit in a cluster
- 460 – Float_t fW0 ; // Logarithmic weight for the cluster center of gravity calculation
- 461 – Float_t fMinECut; // Minimum energy for a digit to be a member of a cluster
- 462 – Bool_t fUnfold; // Flag to perform cluster unfolding
- 463 – Float_t fLocMaxCut; // Minimum energy difference to consider local maxima in a cluster
- 464 – Float_t fTimeCut ; // Maximum difference time of digits in EMC cluster
- 465 – Float_t fTimeMin ; // Minimum time of digits
- 466 – Float_t fTimeMax ; // Maximum time of digits

467 **Track Matching**

- 468 – Double_t fTrkCutX; // X-difference cut for track matching
- 469 – Double_t fTrkCutY; // Y-difference cut for track matching
- 470 – Double_t fTrkCutZ; // Z-difference cut for track matching
- 471 – Double_t fTrkCutR; // cut on allowed track-cluster distance

```

472  – Double_t fTrkCutAlphaMin; // cut on 'alpha' parameter for track matching (min)
473  – Double_t fTrkCutAlphaMax; // cut on 'alpha' parameter for track matching (min)
474  – Double_t fTrkCutAngle; // cut on relative angle between different track points for track
475  matching
476  – Double_t fTrkCutNITS; // Number of ITS hits for track matching
477  – Double_t fTrkCutNTPC; // Number of TPC hits for track matching

```

478 PID

```

479  – Double_t fGamma[6][6]; // Parameter to Compute PID for photons
480  – Double_t fGamma1to10[6][6]; // Parameter to Compute PID not used
481  – Double_t fHadron[6][6]; // Parameter to Compute PID for hadrons
482  – Double_t fHadron1to10[6][6]; // Parameter to Compute PID for hadrons between 1 and
483  10 GeV
484  – Double_t fHadronEnergyProb[6]; // Parameter to Compute PID for energy ponderation
485  for hadrons
486  – Double_t fPiZeroEnergyProb[6]; // Parameter to Compute PID for energy ponderation for
487  Pi0
488  – Double_t fGammaEnergyProb[6]; // Parameter to Compute PID for energy ponderation
489  for gamma
490  – Double_t fPiZero[6][6]; // Parameter to Compute PID for pi0

```

491 The macro

```
492 $ALICE_ROOT/EMCAL/macros/RecParamDB/AliEMCALSetRecParamCDB.C
```

493 is an example on how to set the parameters. There are different event types that we might record,
494 and each event type may require different reconstruction parameters. The event types that are
495 now defined in STEER/AliRecoParam.h are:

```
496 enum EventSpecie_t {kDefault = 1, kLowMult = 2, kHighMult = 4, kCosmic = 8,
497                    kCalib = 16};

```

498 The default event species that we have is kLowMult (low multiplicity). For AliRoot versions
499 smaller than release 4.17 it was set to be kHighMult (high multiplicity). Today, the code is as
500 follow :

```
501 kDefault=kLowMult=kCosmic=kCalib.
```

502 kHighMult differs only from the other two in 2 clusterization parameters, for low multiplicity
503 they are fMinECut=10 MeV and fClusteringThreshold=100 MeV and for high multiplicity they
504 are fMinECut=0.45 GeV and fClusteringThreshold=0.5 GeV.

505 A simple example that shows how to print the parameters for the different event species is
506 PrintEMCALRecParam.C

507 **3.5 Simulation parameters**

508 The parameters used in the simulation are stored in EMCAL/Calib/SimParam. What is stored
509 is an object of the class AliEMCALSimParam which is a container of all the parameters used.
510 There are different kind of parameters depending on the step of the simulation :

511 **SDigitization**

- 512 – Float_t fA ; // Pedestal parameter
- 513 – Float_t fB ; // Slope Digitization parameters
- 514 – Float_t fECPrimThreshold ; // To store primary if Shower Energy loss > threshold

515 **Digitization**

- 516 – Int_t fDigitThreshold ; // Threshold for storing digits in EMC = 3 ADC counts
- 517 – Int_t fMeanPhotonElectron ; // number of photon electrons per GeV deposited energy =
518 4400 MeV/photon
- 519 – Float_t fPinNoise ; // Electronics noise in EMC = 12 MeV
- 520 – Double_t fTimeResolution ; // Time resolution of FEE electronics = 600 ns
- 521 – Int_t fNADCEC ; // number of channels in EC section ADC =

522 The macro \$ALICE_ROOT/EMCAL/macros/SimParamDB/AliEMCALSetSimParamCDB.C, is
523 an example on how to set the parameters. A simple example that shows how to print the param-
524 eters is PrintEMCALSimParam.C

525 **3.6 Alignment**

526 4 Simulation code

527 The class AliSimulation manages this part. An example is here : “\$ALICE_ROOT/EMCAL/
528 macros/TestEMCALSimulation.C”. The simulation consists of different steps: geometry and
529 event definition, particle generation, transport of the particle in the material (GEANT) and fi-
530 nally digitization. Note that the final output from the digitization process is different from the
531 processing of real experimental Raw Data. The process of converting the digitized data to Raw
532 Data is discussed in Sec. 4.2. Sec. 4.4 gives the recipe to do all the steps of the simulation.

533 4.1 Step Manger and Hits Creation

534 The majority of time and effort associated with a detector Monte Carlo is involved in the trans-
535 port of the particles, one at a time typically, though the detector geometry. This is handled by a
536 routine typically called the “step manager”. This routine, in general, does a lot of stuff with con-
537 siderable help from other sub-packages. It must determine what size step to make based on the
538 distance to the next volume, the curvature of the track, the probability of some non-continuum
539 process occurring (an interaction), and deal with particles no longer being transported (dropping
540 below cuts); computing the effects of all continuum process (energy loss, fluctuations, and mul-
541 tiple scattering); and outputting, when relevant, any information to the “user”. The majority of
542 these tasks are common to all detectors and are therefor done for us with the help of the geo-
543 metrical modeler and/or simulation framework. To deal with the outputting of information an
544 EMCal specific **StepManager** routine located in the **AliEMCALv1**¹ module, or equivalent is
545 used. Information outputted by this routine are called “hits”, in the AliRoot terminology, and
546 are written to a file called `EMCAL.Hits.root`. Often in production simulations this file will be
547 deleted afters the digits are produced.

548 The EMCal StepManager is called from the Alice implementation of the ROOT virtual Monte
549 Carlo step manager, specifically the routine **AliMC::Stepping**. It inquires, from the transport
550 engine and its geometrical modeler, what material the presently transporting particle is in and
551 deals with a couple of remaining particle transport issues and then calls the detector specific
552 step manager routine. This decoding is done quickly through an array of material ID numbers
553 indexed to their corresponding detectors. Consequently, each sub-detector must have its own
554 unique material definitions obeying the ALICE material numbering conventions. In this way,
555 the addition or absence of a sub-detector is dynamically handled via the initialization of the
556 material/detector array and the `TObject` array of sub-detectors (all derived from the **AliModule**
557 class). This initialization is done by the `Config.C` script.

558 4.1.1 The EMCal Step Manager

559 The first difficulty faced in the EMCal step manager is the extremely large number of tracks
560 produced and all of their individual steps. Recording each step location, momentum, energy
561 loss, and the like, for all of those shower particles would overwhelm most IO systems and create

¹There are more than one version of **AliEMCALv1** depending on differences in geometry and some physics. All are derived from the EMCal class **AliEMCALv0** which is derived from **AliEMCAL**, which is derived from **AliDetector** which is derived from **AliModule**.

562 too much data to try to deal with further on in the simulation. Yet we need the particle transport
 563 engine to generate and transport the majority of these shower particles, otherwise, the signals in
 564 the neighboring towers would be grossly incorrect and any part of a shower which goes beyond
 565 the EMCal would also not be dealt with properly. In much thinner detectors, like the ITS, the
 566 particles parameters at each step is recorded directly into the hits.

567 For each particle entering the EMCal, what we want to record is the energy lost by it and all of
 568 its shower daughters and in which tower this energy loss occurred (and only for the “sensitive”
 569 materials/volumes in the towers). In fact we really only want to associate this tower-wise energy
 570 loss to the “primary” particle. To do this, for as long as the “primary” track hasn’t changed and
 571 the present track is still in the same tower, the signals are added together (by adding their hits
 572 together. This is done in `AliEMCALv1::AddHit`). The determination of the “primary” parent
 573 particle isn’t so difficult, but one need to deal with a number of special cases, and search back
 574 through the parentage tree in some cases.

575 This leads to the 2nd major task of the EMCal step manger routine. It must determine which
 576 tower the transported particle is in. This is very dependent on the details of the geometry and
 577 how it has been coded. We know which volume the particle is in, but since there are many copies
 578 (of the directory like geometry structure. figure 8) of the tower volume, we also need to find the
 579 necessary copy index numbers associated with the specific volume. This is easy to get from the
 580 geometric modeler (ROOT’s TGeo package in our case), but it can be non-trivial to convert these
 581 numbers into the tower, module, super-module index wanted by the following simulation and
 582 reconstruction routines. The present geometry, where a single tower sized scintillator volume
 583 has the lead radiators embedded into it, simplifies this tower determination because there is only
 584 one sensitive tower sized volume and not a lot of individual sheets of scintillator to decode.

585 For the EMCal we do something a bit special (but not untypical for a calorimeter using organic
 586 scintillators). We correct for the diminished light output due to the ionization produced by the
 587 particles proceeding it. This is done by rescaling the energy deposited using Birk’s law, equation
 588 1, as copied from GEANT3’s G3BRIRK routine [1]. This can be switched on or off from the
 589 EMCal creation section of `Config.C` via the `fBirkC0` variable in `AliEMCAL` class². There has
 590 been some debate about the proper way to deal with this in the collaboration, mostly dealing
 591 with the limitations of any Monte Carlo which transports particles one at a time, but it has been
 592 agreed that including such a correction is better than none at all.

$$\begin{aligned}
 \text{Light yield} &= \frac{\Delta E_{\text{deposited}}}{1 + C_1 \delta + C_2 \delta^2} & (1) \\
 \delta &= \frac{1}{\rho} \frac{dE}{dx} \left[\frac{\text{MeV cm}^2}{\text{g}} \right] \\
 C_1 &= \begin{cases} 0.013 \left[\frac{\text{g}}{\text{MeV cm}^2} \right] & Z = 1 \\ 0.00743 \left[\frac{\text{g}}{\text{MeV cm}^2} \right] & Z > 1 \end{cases}
 \end{aligned}$$

²A function needs to be added to this class to allow for setting this value and the Birk’s law constants `fBirkC1` and `fBirkC2`

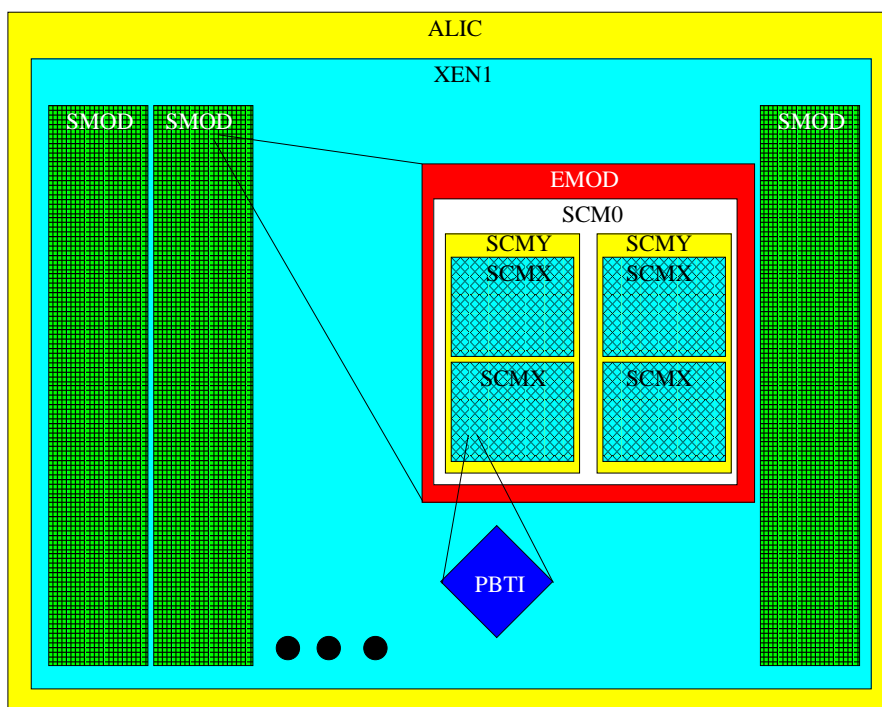


Fig. 8: Here is shown a typical hierarchical geometry structure. This is similar to a directory structure except at each level one or more copies, including translation and rotation operators, of the daughters can be specified.

$$C_2 = 9.6 \times 10^{-6} \left[\frac{g^2}{MeV^2 cm^4} \right]$$

593 The remaining tasks of the EMCal step manager is mostly book-keeping. We only want to
 594 go to all of this effort if there is energy being deposited in the sensitive scintillator volume,
 595 and not the lead radiators or other structural materials. All of the relevant information for the
 596 EMCal hit needs to be gathered. Lastly, the **AliEMCALHit** class needs to be created within
 597 the `TClonesArray` of EMCal hits. This leads to some convoluted looking code involving the
 598 `TClonesArray fHits`, the new operator and the `AliEMCALHit` copy constructor (see **AliEM-**
 599 **CAL::AddHit**).

600 The structure of these `EMCALHit` class (data structure) is simple. It starts with the `AliHit` infor-
 601 mation which consists of the `tTrack` number of the track which entered the EMCal and its `x,y,z`
 602 global position (in cm). The EMCal specific derivation includes the absolute tower ID where
 603 the hit signal is from, the energy deposited by the showering particles originating from this track
 604 in that tower, and the relative time (with respect to the initial event) when this energy was de-
 605 posited, the particle ID of the particle entering the EMCal, the entrance energy of the particle
 606 entering the EMCal, and the energy and momentum of the primary particle entering the EMCal.

607 Just a note, although not included in the code, the addition of the signals from the APD, primarily
608 due to neutrons interacting with the APD, needs to be added. CMS has found that including this
609 effect measurably improves the response of their simulations. This will require an addition to
610 the EMCal step manager, but hopefully not the `EMCALHit` structure.

611 **4.1.2 Step Manager and Monte Carlo Setting**

612 In the EMCal geometry description there are also settings done, on a medium by medium basis,
613 which are used in the non-EMCal specific step manager code. In `AliEMCAL` where ever a medium
614 is defined (either by a call to `AliMedium` or equivalently to a call to `TGeoMedium`) a list of
615 parameters must be given which effects the size of a step. These parameters are given in Table
616 1.

Table 1

Type	Variable	Description
Int_t	isvol	Sensitive volume flag. 0 Not a Sensitive volume. 1 Sensitive volume.
Int_t	ifield	Magnetic field flag. 0 No magnetic field. -1 User decision in <code>guswim</code> . Not supported in <code>AliRoot</code> . 1 Tracking performed with Runge Kutta. 2 Tracking performed with helix. 3 constant magnetic field along z.
Float_t	fieldm	Maximum magnetic field [kG].
Float_t	tmaxfd	Maximum deflection angle due to magnetic field [degrees].
Float_t	stemax	Maximum step allowed [cm].
Float_t	deemax	Maximum fractional energy loss in one step. $dee = \frac{\Delta E}{E_k}$
Float_t	epsil	Tracking precision [cm]. This effects transition to new volumes.
Float_t	stmin	Minimum step due to continuous processes [cm]. This must be set to 0 so that <code>GEANT3</code> will computing it correctly. Not doing so will adversely effect the simulation.

Table 1: Parameters and flags defined in the EMCal geometry via a call to `AliMedium` or `TGeoMedium`. Because we use a version of `GEANT3` which has its geometrical modeler replaced by `TGeo` geometry, they are the same. This is also true for both `GEANT4`[2] and `Fluka`[3] particle transport Monte Carlos. See figure 4.1.2 for a geometrical description of some of these parameters. This information comes from the `GEANT3` documentation CONS200-1 [1]. .

617 This isn't the whole story in regards to the step manager. There are a number of things which
 618 we need to set/control that don't appear in any EMCal code, but are dealt with in the transport
 619 engines part of the step manager. Such settings and controls are very dependent on the specific
 620 transport Monte Carlo being used. All of these settings and controls have ALICE wide default
 621 values, but most of them we will need to change to get optimal performance and accuracy from
 622 our EMCal simulation. These switches and settings are settable for specific materials. If a
 623 material does not have a set of switches or settings set, the ALICE wide defaults are used.

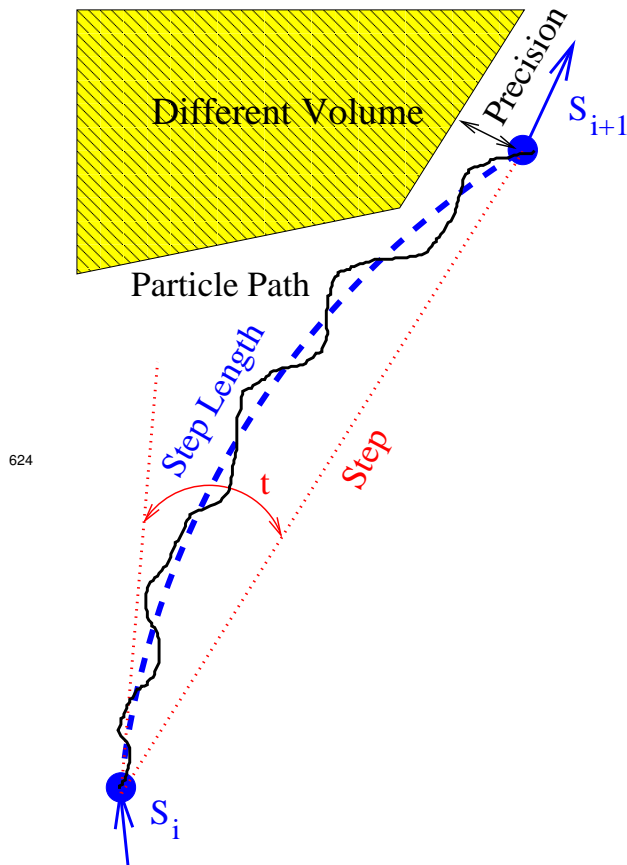


Figure 4.1.2 An exaggerated Monte Carlo step showing some of the considerations associated with transporting a Monte Carlo particle through a step. One step between s_i to s_{i+1} is shown in the dotted (red) line. The dashed (blue) line shows the step length taken due to a magnetic field. The solid (black) line show what the particle path might really be. A new/different volume is shown as the hashed (yellow) area. A new momentum and energy are computed at the end of each step taking into account the energy loss and multiple scattering. Also indicated is the deviation in the step due to an applied magnetic field t , and the precision with which the step has missed the other volume.

625 **GEANT3 Switches and Settings** GEANT3 was the first particle transport Monte Carlo inte-
 626 grated into AliRoot and ROOT's virtual Monte Carlo and so has some of the oldest and simplest
 627 interfaces. For simulation, AliRoot sets many default settings and switches. This is done in
 628 `Config.C` which you can find in `$ALICE_ROOT/macros`. There you will see a number of line of
 629 the form `gMC->SetProcess(char *name, int value)`. The switch names and there ALICE
 630 default values are shown in table 2. There are limits both to the computer's capabilities in deal-
 631 ing with the number of particles to transport and with the physics models used by GEANT3.
 632 Consequently there are "cuts" used to stop the transport of particles which are below some en-
 633 ergy or are taking too long. The ALICE wide default values are also set in `Config.C` using the
 634 function `gMC->SetCut(char *name, double value)`. All of these values, and those included

635 in the `galice.cuts` file are listed in table 3.

636 The `galice.cuts` file has a fixed format as indicated in `AliMC::ReadTransPar` function. In
 637 this file, lines starting with an “*” are ignored. The remaining lines are required to contain, in
 638 order separated by one or more spaces, `Detector_Name`, `Detector’s_media_number`, and then
 639 the numbered cuts and flags listed in tables 2 and 3.

Table 2

Switch	ALICE Default values	Description
13 ANNI	1	Positron annihilation. The e^+ is stopped. 0 No positron annihilation. 1 Positron annihilation with generation of γ . 2 Positron annihilation without generation of γ .
14 BREM	1	bremsstrahlung. The interaction particle (e^- , e^+ , μ^- , μ^+) is stopped. 0 No bremsstrahlung. 1 bremsstrahlung with generation of γ . 2 bremsstrahlung without generation of γ .
15 COMP	1	Compton scattering. 0 No Compton scattering. 1 Compton scattering with generation of e^- . 2 Compton scattering without generation of e^- .
16 DCAY	1	Decay in flight. The decaying particles stops. 0 No decay in flight 1 Decay in flight with generation of secondaries 2 Decay in flight without generation of secondaries
17 DRAY	0	δ -ray production. 0 No δ -ray production. 1 δ -ray production with generation of e^- . 2 δ -ray production without generation of e^- .
18 HADR	1	Hadronic interactions. The particle is stopped in case of inelastic interactions, while it is not stopped in case of elastic interactions. 0 No hadronic interactions. 1 Hadronic interactions with generation of secondaries. 2 Hadronic interactions without generation of secondaries. > 2 can be used in the user code <code>GUPHAD</code> and <code>GUHADR</code> to choose a hadronic package. These values have no effect on the hadronic packages themselves. Not supported in AliRoot.

Table 2 continued on next page.

Table 2 continued

Switch	ALICE Default value	Description
19 LOSS	2	<p>Continuous energy loss.</p> <p>0 No continuous energy loss, DRAY is forced to 0.</p> <p>1 Continuous energy loss with generation of δ-rays which have an energy above DCUTE and restricted Landau-fluctuations[4] for δ-rays which have an energy below DCUTE (no δ-ray produced).</p> <p>2 Continuous energy loss without generation of δ-rays and full Landau-Vavilov-Gauss[?] fluctuations. In this case DRAY is forced to 0 to avoid double counting of fluctuations.</p> <p>3 Same as 1, kept for backwards compatibility.</p> <p>4 Energy loss without fluctuations. The value obtained from the tables is used directly.</p>
20 MULS	1	<p>Multiple scattering.</p> <p>0 No multiple scattering.</p> <p>1 Multiple scattering according to Moliere[?] theory.</p> <p>2 Same as 1. Kept for backwards compatibility.</p> <p>3 Pure Gaussian scattering according to the Rossi formula[7].</p>
21 PAIR	1	<p>Pair production. The interacting γ is stopped.</p> <p>0 No pair production.</p> <p>1 Pair production with generation of e^+/e^-.</p> <p>2 Pair production without generation of e^+/e^-.</p>
22 PHOT	1	<p>Photoelectric effect. The interacting photon is stopped.</p> <p>0 No photo-electric effect.</p> <p>1 Photo-electric effect with generation of e^-.</p> <p>2 Photo-electric effect without generation of e^-.</p>
23 RAYL	1	<p>Rayleigh effect[?]. The interacting γ is not stopped.</p> <p>0 No Raylieght effect.</p> <p>1 Rayleigh effect.</p>
24 STRA	0	<p>Turns on the collision sampling method to simulate energy loss in thin materials, particularly gasses.</p> <p>0 Collision sampling is off.</p> <p>1 Collision sampling is on.</p>
PFIS	0	<p>Nuclear fission induced by a photon The photon stops.</p> <p>0 No photo-fission.</p> <p>1 Photo-fission with generation of secondaries.</p> <p>2 Photo-fission without generation of secondaries.</p>
MUNU	1	<p>Muon-nucleus interactions. The muon is not stopped.</p> <p>0 No muon-nucleus interactions.</p> <p>1 Muon-nucleus interactions with generation of secondaries.</p> <p>2 Muon-nucleus interactions without generation of secondaries.</p>

Table 2 continued on next page.

Table 2 continued

Switch	ALICE Default value	Description
CKOV	1	Light absorption. This process is the absorption of light photons in dielectric materials. It is turned on by default when the generation of Čerenkov[9] light is requested (in GEANT manual it is LABS). 0 No absorption of photons. 1 Absorption of photons with possible detection.
SYNC	0	Synchrotron radiation in magnetic fields. 0 Synchrotron radiation is not simulated. 1 Synchrotron photon are generated, at the end of the tracking step. 2 Photons are not generated, the energy is deposited locally. 3 Synchrotron photons are generated, distributed along the curved path of their particle.

Table 2: GEANT3 physics process flags. These flags can be set on a material by material basis. The ALICE Default values are set in the `Config.C` file uses the `gMC->SetProcess` function. The setting of these specific flags for any specific material is done in `$ALICE_ROOT/data/galice.cuts` file. The number on the left of the switch name is the column in the `galice.cuts` file that this switch is expected to be found. This information comes from the GEANT3 documentation PHYS001-3 [1].

Table 3

Parameter	ALICE value	Default	Description
3 CUTGAM	$1. \times 10^{-3}$ GeV		Threshold for gamma transport.
4 CUTELE	$1. \times 10^{-3}$ GeV		Threshold for electron and positron transport.
5 CUTNEU	$1. \times 10^{-3}$ GeV		Threshold for neutral hadron transport.
6 CUTHAD	$1. \times 10^{-3}$ GeV		Threshold for charged hadron and ion transport.
7 CUTMUO	$1. \times 10^{-3}$ GeV		Threshold for muon transport.
8 BCUTE	$1. \times 10^{-3}$ GeV		Threshold for photons produced by electron bremsstrahlung.
9 BCUTM	$1. \times 10^{-3}$ GeV		Threshold for photons produced by muon bremsstrahlung.
10 DCUTE	$1. \times 10^{-3}$ GeV		Threshold for electrons produced by electron δ -rays.
11 DCUTM	$1. \times 10^{-3}$ GeV		Threshold for electrons produced by muon or hadron δ -rays.
12 PPCUTM	$1. \times 10^{-3}$ GeV		Threshold for e^\pm direct pair production by muons.
TOFMAX	$1. \times 10^{10}$ sec		Threshold on time of flight counted from primary interactions time.

Table 3 continued on next page.

Table 3 continued

Parameter	ALICE value	Default	Description
-----------	-------------	---------	-------------

Table 3: GEANT3 physics process limits. These “cuts” can be set on a material by material basis. The ALICE Default values are set in the `Config.C` file uses the `gMC->SetCuts` function. The setting of these specific flags for any specific material is done in `$ALICE_ROOT/data/galice.cuts` file. The number on the left of the cut name is the column in the `galice.cuts` file that this cut is expected to be found. This information comes from the GEANT3 documentation ZZZZ010-2 [1]

640 **GEANT4 Switches and Settings**

641 **Fluka Switches and Settings**

642 **References**

- 643 [1] CERN Program Library. *GEANT Detector description and Simulation Tool*, CERN Pro-
644 gram Library Long writeup W5013, October 1994
- 645 [2] GEANT4 Working Group, “User Documentation”, Accessed Feb. 4 2013.
646 <http://geant4.cern.ch/support/suerdocumentatoin.shtml>
- 647 [3] FLUKA Team, “FLUKA”, Access Feb. 4 2013,
648 http://www.fluka.org/fluka.php?id=man_onl
- 649 [4] *GEANT Detector description and Simulation Tool*, PHYS332. and L. Landau. *On the En-
650 ergy Loss of Fast Particles by Ionisation*, J. Phys. 8:201, 1944. and K. S. Kölbig and B.
651 Schorr. *Asymptotic expansion for the Landau density and distribution functions*, Comp.
652 Phys. Comm., 32:121,1984
- 653 [5] *GEANT Detector description and Simulation Tool*, PHYS332. and P. V. Valilov. *Ionisation
654 losses of high energy heavy particles*, Soviet Physics JETP, 5:749, 1957
- 655 [6] *GEANT Detector description and Simulation Tool*, PHYS320. and G. Z. Moliere *Theorie
656 de Streuung schneller geladener Teilchen I: Einzelstreuung am aberschmitteten Coulomb-
657 Feld*, Z. Naturforsch., 2a:133, 1947 and G. Z. Moliere, *Teeorie der Steuung schneller
658 geladerner Teichen II: Merfach- und Vielfachstreuung* Z. Naturforsh., 3a:87, 1948 and
659 W. T. Scott. Rev. Mod. Phys. 35:231 1963
- 660 [7] *GEANT Detector description and Simulation Tool*, PHYS320 and R. Rossi, Prentice-Hall,
661 Englewood Clifgs, 1962 R. Rossi, and K. Greisen, Rev. Mod. Physics, 13-240, 1942
- 662 [8] *GEANT Detector description and Simulation Tool*, PHYS250. and W. R. Nelson, H. Hi-
663 ayama, and D. W. O. Rogers. Technical Report 265, SLAC, 1985

664 [9] *GEANT Detector description and Simulation Tool*, PHYS260, J. D. Jackson *Classical*
665 *Electrodynamics*, J. Wiley et Sons, Inc. New York, 1975

666 4.2 Digitization: SDigits and Digits - Evi

667 We want to generate events which look like the real data collected by the experiment. In the end,
668 we want to have an amplitude in ADC counts and a time (when particle traverse a cell) per each
669 cell (tower) of the calorimeter. In the code for calorimeters, it is done in the following steps:

- 670 1. **SDigit** objects are created, they consist of the sum of deposited energy by all Hits in a cell
671 (a particle can create Hits in different cells but only one in a single cell), so there is only
672 one SDigit per fired cell.
- 673 2. **Digit** objects are created, they are like the SDigits but the energy in the cell is transformed
674 into the ADC amplitude units, the electronic noise is added and Digits whose energy does
675 not pass an energy threshold (3 ADC counts) are eliminated. SDigits and Digits are stored
676 in the files **EMCAL.SDigits.root** and **EMCAL.Digits.root**, respectively.

677 4.3 Raw data - David

678 The experiment does not record Digits directly, but instead a series of so-called time samples
679 with 10-bit ADC counts per channel. Each time bin is 100 ns wide, corresponding to a 10
680 MHz readout. These samples are referred to as **Raw Data**. The samples follow a certain signal
681 shape, more complicated than a Gaussian distribution, which is fitted offline. The simulated
682 signal (Gamma-2) shape is described in the AliEMCALRawResponse class, in the RawResponseFunction
683 method. With real data, which is zero-suppressed, i.e. has the pedestal subtracted
684 online, the Digits amplitude is just the maximum of the distribution obtained with the fit to the
685 sample. The Digit time (defined by the time the particle hits the active volume of the detector) is
686 the time value at the maximum signal fit. There are methods to go from Digits to Raw and vice
687 versa in the AliEMCALRawUtils class: Raw2Digits and Digits2Raw, respectively. For the re-
688 construction step Digits are needed. The generation of Raw Data is optional during simulations
689 and the generated data can be reconstructed directly from Digits, but Raw data is the initial step
690 when reconstructing real data.

691 4.4 How to make a simulation

692 TestEMCALSimulation.C is a very simple macro where we specify all the simulation parameters
693 and process the simulation. Below is a similar but a bit more elaborated macro:

```

1 void TestEMCALSimulation() {
2
3 TString detector="EMCAL TPC"; // Define in this variable the detectors that you want to be
   included in the simulation for the digitization . They can be less detectors than the
   detectors defined in the Config.C file , imagine that you want all the detectors in front
   of EMCAL present to consider the conversion of particles but you are not really
   interested in the output from these detectors .
4 // Option detector="ALL" makes all detectors .
5
6 AliSimulation sim ; //Create simulation object
7
8 // Generation and simulation
9
10 sim.SetRunGeneration(kTRUE) ; //Default value is kTRUE, make generation
11 // For some reason we may want to redo the Digitization , without redoing the generation , in
   this case it must set to kFALSE
12
13 // Making SDigits
14 sim.SetMakeSDigits(detector) ; //We want to make SDigits
15 // set no detectors if SDigits are already made
16
17 // Making Digits
18 sim.SetMakeDigits(detector) ; //We want to make Digits
19 // set no detectors if SDigits are already made
20
21 //Merging
22 //sim.MergeWith("bgrd/galice.root") ; // If we want to merge a signal and a background, the
   merging is done at the SDigit level . The background must be located in the repertory
   defined in the method.
23
24 //Write Raw Data, make Raw data from digits
25 //sim.SetWriteRawData(detector) ;
26 //sim.SetConfigFile (" somewhere/ConfigXXX.C"); //Default is Config.C
27
28 //Make the simulation
29 sim.Run(3) ; //Run the simulation and make 3 events

```

695

696 **5 Reconstruction code**

697 The energy deposited by the particles in the towers produces scintillating light that is propagated
698 with optic fibers through the different layers to APD placed at the base of the cells. The APDs
699 amplify the signal and generate an electronic pulse shape that is stored in the raw data format.
700 From this pulse shape, we extract the signal amplitude and the arrival time. The pulse shape is
701 fitted during the reconstruction via a parametrized function and TMinuit, and those 2 values are
702 extracted.

703 A particle produces signals in different towers (electromagnetic shower expands more than its
704 Molière radius which is a cell size). The next step is the formation of clusters of cells that belong
705 to the same particle, although depending on the energy, granularity, clusterization algorithm or
706 event type, those clusters might have contributions from different particles. The default algo-
707 rithm in pp collisions is a simple aggregation of neighboring cells until there is no more cells
708 above a certain energy threshold (named *clusterizer V1*). In case of Pb-Pb collisions environ-
709 ment, where particle showers merge quite often, we apply another algorithm that aggregates
710 cells to the clusters until reaching a cell with more energy than the precedent (named *cluster-*
711 *izer V2*). Depending on the analysis type, one might want to use one or the other clusterization
712 type. For this reason, a re-clusterization is also possible at the analysis level. A last clusterizer
713 is implemented, which makes 3x3 clusters. It has been used in jet analysis for instance in order
714 to avoid biasing jet reconstruction where one is interested in the energy flow over a large area
715 without explicit reconstruction of photon showers and where the driving consideration is that
716 the wide clusterizer does not interfere with the jet finder. For π^0 , η , and direct γ analyses, V2 is
717 most likely preferable).

718 Once the cluster is defined, we calculate cluster parameters, shower shape parameters, that will
719 help at the analysis level to identify each cluster as one particle type. Also, we compare the
720 cluster position information with the propagation of tracks measured in the central barrel to the
721 EMCAL surface, to identify the clusters generated by charged particles.

722 The final analysis objects, ESDs and AODs, contain all the cluster and cell basic informations
723 allowing to redo the clusterization if needed at the analysis level.

724 **5.1 Offline data base access**

725 How to create explained OCDB/OADB section.

726 **5.1.1 Energy calibration**

727 **5.1.2 Bad channels - Marie, Alexis**

728 **5.1.3 Alignment - Marco**

729 **5.2 Raw data fitting: from ADC sample to digits - David**

730 As also discussed in Sec. 4.3, the recorded Raw data consists of instead a series of so-called
731 time samples with 10-bit ADC counts per channel. Each time bin is 100 ns wide, corresponding
732 to a 10 MHz readout. The expected signal (Gamma-2) shape is described e.g. in the AliEM-

733 CALRawResponse class, in the RawResponseFunction method. The reconstruction from Raw
734 data to Digits is done in the AliEMCALRawUtils class, Raw2Digits method. The Raw ADC
735 time samples data is kept in AliCaloBunchInfo objects, which are given as input to an AliCalo-
736 RawAnalyzer object, which returns the signal amplitude and time information (in the form of
737 an AliCaloFitResults object). There are several different AliCaloRawAnalyzer versions, which
738 can be selected via AliEMCALRawUtils::SetFittingAlgorithm(). They are:

- 739 – kStandard: AliCaloRawAnalyzerKStandard, which is a (slower but simple) Gamma-2 fit
740 implementation.
- 741 – kFastFit: AliCaloRawAnalyzerFastFit, which is a faster Gamma-2 fit implementation
742 from Aleksei Pavlinov.
- 743 – kNeuralNet: AliCaloRawAnalyzerNN, which is a neural network implementation from
744 Paola La Rocca and Franco Riggi.
- 745 – kPeakFinder: AliCaloRawAnalyzerPeakFinder, which is a fast (parameterized vector op-
746 erations) implementation from Per Thomas Hille.
- 747 – kCrude: AliCaloRawAnalyzerCrude, which is the simplest possible algorithm: just take
748 the maximum ADC value as the signal amplitude.
- 749 – kFakeAltro: AliCaloRawAnalyzerFakeALTRO, which is an algorithm intended for the
750 Trigger/TRU raw data analysis, i.e. not for the regular FEE or cell/tower data.

751 **5.3 Clusterization: From digits to clusters - Adam**

752 The set of information related to one cell - (between each other) the position of cell and the
753 energy deposited in it is called a digit. The digit is represented by the AliEMCALDigit class. A
754 group of digits which are related somehow between each other is called a cluster. The cluster is
755 represented by the AliEMCALRecPoint class.

756 Transformation from digits to clusters is done during clusterization phase. There are many ideas
757 how to form cluster. Each applied idea is called clusterizer. Currently, there are four types of
758 clusterizers in the EMCal:

- 759 – Clusterizer V1,
- 760 – Clusterizer V2.
- 761 – Clusterizer V1 with unfolding,
- 762 – Clusterizer NxN,

763 Technically it is organized in the following way. AliEMCALClusterizer is a base clusterizer
764 class. The V1 and NxN clusterizer classes (AliEMCALClusterizerV1 and AliEMCALClusterizerNxN,

765 respectively) inherit from the base class. The clusterizer class V2 (`AliEMCALClusterizerV2`)
 766 inherits from `AliEMCALClusterizerV1`. The third case of clusterization (clusterizer V1 with
 767 unfolding) is realized via settable option in the `AliEMCALClusterizerV1` class. The dedicated
 768 class `AliEMCALUnfolding` was written for the purpose of unfolding. The description of each
 769 clusterizer type separately and common clusterization structure together with a description of
 770 the cluster is explained below.

771 5.3.1 Clusterization in the EMCal

772 A clusterizer is called in the `AliEMCALReconstructor` class. The set of clusterizer parameters
 773 is initialised. Usually parameters are taken from the OCDB. The full set of parameters which
 774 are used during clusterization is given in Tab. 4. The other fields of the `AliEMCALClusterizer`
 class are given in Tab. 5. Also input and output are connected. Finally, a clusterization phase is

Name	Type	Explanation
<code>fTimeMin</code>	<code>Float_t</code>	minimum time of physical signal in a cell/digit
<code>fTimeMax</code>	<code>Float_t</code>	maximum time of physical signal in a cell/digit
<code>fTimeCut</code>	<code>Float_t</code>	maximum time difference between the digits inside EMC cluster
<code>fToUnfold</code>	<code>Bool_t</code>	says if unfolding should be performed
<code>fECAClusteringThreshold</code>	<code>Float_t</code>	minimum energy to seed a EC digit in a cluster
<code>fECALocMaxCut</code>	<code>Float_t</code>	minimum energy difference to distinguish local maxima in a cluster
<code>fECAW0</code>	<code>Float_t</code>	logarithmic weight for the cluster center of gravity calculation
<code>fMinECut</code>	<code>Float_t</code>	minimum energy for a digit to be a member of a cluster
<code>fSSPars[8]</code>	<code>Double_t</code>	shower shape parameters
<code>fPar5[3]</code>	<code>Double_t</code>	shower shape parameter 5
<code>fPar6[3]</code>	<code>Double_t</code>	shower shape parameter 6

Table 4: Parameter name, type and explanation.

775 done in a `Digits2Clusters` method. Main steps run as follow:
 776

- 777 1. Get calibration parameters (method `GetCalibrationParameters`),
- 778 2. Get pedestal parameters (method `GetCaloCalibPedestal`),
- 779 3. Make clusters (method `MakeClusters`),
- 780 4. Make unfolding or not (method `MakeUnfolding`),
- 781 5. Evaluate cluster properties (`AliEMCALRecPoint` class methods),
- 782 6. Store clusters.

783 In the first two steps calibration (ADC counts to energy conversion) and pedestal parameters
 784 are read from a database. The third step, where clusters are formed, is different for each clus-
 785 terizer. However, some beginning parts of this step are common for each algorithm. The input

Name	Type	Explanation
fADCchannelECA	Float_t	width of one ADC channel for EC section (GeV)
fADCpedestalECA	Float_t	pedestal of ADC for EC section (GeV)
fTimeECA	Float_t	calibration parameter for channels time
fIsInputCalibrated	Bool_t	to enable reclusterization from ESD cells
fJustClusters	Bool_t	false for standard reco
fDigitsArr	TClonesArray*	pointer to array with EMCAL digits
fTreeR	TTree*	pointer to tree with output clusters
fRecPoints	TObjArray*	pointer to array with EMCAL clusters
fGeom	AliEMCALGeometry*	pointer to geometry for utilities
fCalibData	AliEMCALCalibData*	pointer to calibration database if available
fCaloPed	AliCaloCalibPedestal*	pointer to tower status map if available
fDefaultInit	Bool_t	says if the task was created by default ctor
fNumberOfECAClusters	Int_t	number of clusters found in EC section
fClusterUnfolding	AliEMCALUnfolding*	pointer to unfolding object

Table 5: Other fields in the `AliEMCALClusterizer` class.

786 is the same. It is an array of fired digits with electronic signal registered in each of them.
787 Also calibration and cleaning the array of digits, to work with reduced sample of digits, is the
788 same for each algorithm. This common part is done in the common method `Calibrate` of the
789 `AliEMCALClusterizer` class. Here, we require the proper timing (via selection of `fTimeMax`
790 and `fTimeMin` of a given digit), status (check of dead channel map) and calibrate energy and
791 time of each digit. If any digit fails to pass one of requirement it is rejected from a “working
792 array” of digits (pool of digits). In `make clusters` step the `AreNeighbours` method is used. The
793 method could differ for each clusterization algorithm. The explanation how clusters are formed
794 is given in next four subsections for each clusterization algorithm, respectively. The next step
795 (unfolding) is an option in each clusterizer, but currently is used only for the V1 clusterizer.
796 The last two steps (evaluation of a cluster properties and its recording) are the same for each
797 algorithm.

798 5.3.2 Clusterizer V1

799 Having obtained “working array” of digits in the V1 algorithm we additionally reject digits with
800 energy smaller than `fMinECut`, which is set to be 10 MeV in the database by default.

801 After selecting digits we form clusters. We loop over all digits to find the first seed digit with
802 energy greater than `fECAClusteringThreshold` (default value is 100 MeV). When the seed digit
803 is found it is associated to a new cluster and removed from the “working array” of digits. We
804 loop over all remaining digits to look for neighbours of the seed digit. The neighbour digits
805 are called digits which have at least common side (i.e.: row index difference or column index
806 difference must be equal 1, but not both of them at the same time are equal 1, so one cell can
807 have four neighbours at maximum). The additional requirement on neighbour digits is applied.
808 The absolute value of time difference for two digits must be less or equal `fTimeCut`.

809 The neighbour digits are associated to the cluster (also removed from the “working array” of
810 digits) and we keep on looking for neighbours of each digit associated to the cluster. When a
811 digit is associated to one cluster it cannot be associated to other one. When there are no more
812 neighbours of digits in one cluster one can say that this cluster is formed. Once the cluster is
813 formed and there are still remaining digits in the “working array” the procedure starts to check
814 seeds and all the story repeats until no seed digit is found. The consequence of such algorithm
815 is that one cluster can contain all digits in the super-module. The other thing is that there can
816 be digits which are not associated to any cluster. The special case when cluster is formed from
817 digits in two super-modules at the same SM- ϕ angle is also supported.

818 **5.3.3 Clusterizer V2**

819 The algorithm starts with the pool of digits. Then the most energetic digit with the energy
820 over `fECAClusteringThreshold` is taken. It is the seed of a cluster. We scan over digits
821 already associated to the cluster and check for neighbours. It is the iterative procedure. Here,
822 the absolute value of the time difference of seed and neighbours digits should be less or equal
823 `fTimeCut`. The definition of neighbours is the same like in V1 clusterizer, however, energy
824 of neighboring digit should be smaller in order to become a neighbor. `fDoEnGradCut` flag is
825 responsible for application of the last condition. If the process of one cluster formation ends we
826 start from the point where new and the most energetic digit is found in the pool of digits and
827 repeat other steps until no digit remains in the pool.

828 **5.3.4 Clusterizer V1 with unfolding**

829 The main goal of unfolding is to divide multi-maxima clusters into single-maxima clusters and
830 split energy of unfolded clusters. The unfolding is an option which is switched off by default.
831 It can be switched on in every clusterizer. However, as the output of V2 and NxN algorithms
832 clusters are already small and contains only one maximum. The V1 clusterizer provides multi-
833 maxima clusters, so unfolding can be reasonably used only in V1 method of clusterization.

834 Unfolding uses already reconstructed clusters from V1 as an input and modify (split and share
835 energy in digits among several clusters) them if necessary. The unfolding scheme can be divided
836 into several steps:

- 837 1. Maxima finding.
- 838 2. Fit.
- 839 3. Reclustering.

840 **Maxima finding.** As the first step number of local maxima is defined in one cluster. A cell
841 is a local maximum when its energy deposit is greater than the energy deposit in each of its
842 neighboring cells (by neighboring cell we understand here two cells touched by side or corner,
843 so one cell can have 8 neighbours at maximum) by at least some constant value. This constant
844 value is called the minimum energy difference between two local maxima (`fECALocMaxCut`)
845 and as a default is equal to 30 MeV. In the particular case where two neighboring cells have a

846 similar energy deposit (the difference between energy of two cells is below a certain value) the
 847 cluster is treated as a flat one with no maximum.

848 The outcome of the maximum finding procedure is divided in two cases. In the case no pro-
 849 nounced maximum or only one maximum is found unfolding is not applied and cluster is not
 850 touched. If there are at least two maxima the procedure of unfolding starts running.

851 **Fit.** The next step is the fitting procedure which allows to disentangle overlapping clusters
 852 based on the knowledge of what should be the typical shower shape of a γ particle. The energy
 853 distribution in a single photon cluster (shower shape) is described by following function:

$$f(r) = P_0 \cdot \exp(-(2.332 \cdot r)^{P_1}) \cdot \left(\frac{1}{P_3 + P_4 \cdot (2.332 \cdot r)^{P_1}} + \frac{P_5}{1 + (2.332 \cdot r)^{P_2} \cdot P_6} \right), \quad (2)$$

854 where $P_{0,1,2,3,4,5,6}$ are parameters and r is a distance between a cell center and the center of
 855 gravity of a cluster. This function is constant for given ϕ region and it is our reference to start to
 856 unfold clusters. One single photon cascade can be described by the shower shape function with
 857 fixed parameters. However to locate it in the detector we need 3 parameters: position of center
 858 of gravity of cluster in ϕ and η coordinates and a cluster's energy. In case of a single photon
 859 cluster we could fit just mentioned 3 parameters. If there are more maxima we start with more
 860 parameters. The correlation is very simple. One maximum found corresponds to 3 parameters
 861 which we want to fit. The initial value of parameters for one maximum are following: position
 862 of the local maximum cell in ϕ and η coordinates and its energy. TMinuit package is called to
 863 minimize the χ^2 between the shower shape function of a single γ and the shower shape spectrum
 864 of the cluster being unfolded. The outcome of the fit is the set of parameters which describe
 865 center of gravity and energy of each unfolded cluster.

866 **Reclustering.** The last step is to build in terms of cell energy attribution the two (or more)
 867 clusters obtained splitting the original big blob cluster. There is an obvious constraint for the
 868 energy in each cell. The sum of the energies associated to the different clusters returns the
 869 measured energy associate at the cell. For each cell, and for each split cluster, the fit result from
 870 the previous step provides an expected value which can be used as weight to distribute the cell
 871 energy among the different unfolded clusters. The total (measured) signal present in the cell is
 872 shared among the split clusters with the proportion given by the fit function values. The split
 873 (unfolded) clusters are built based on these new cell entries.

874 The energy of each cell in the unfolded cluster should be above a certain energy threshold E_{th}
 875 ($fThreshold$). By default energy threshold is set to be $E_{th} = 10$ MeV. If a cell after unfolding in
 876 a given cluster has an energy below threshold, this cell is rejected from this cluster and its energy
 877 is shared among other clusters, proportionally to the energy of this cell in other clusters. If after
 878 unfolding only one cluster contains a cell with energy above threshold cells below threshold
 879 are rejected from other clusters and the full energy is associated to the cell with energy above
 880 threshold. If after unfolding each energy of cell is below threshold then the whole energy is
 881 associated to the most energetic cell.

882 The number of new (unfolded) clusters will be the same as the number of local maxima found

883 during the first step above. When unfolding succeeds the original big blob cluster is replaced by
 884 several unfolded clusters. Unfolding method is precisely described in [?].

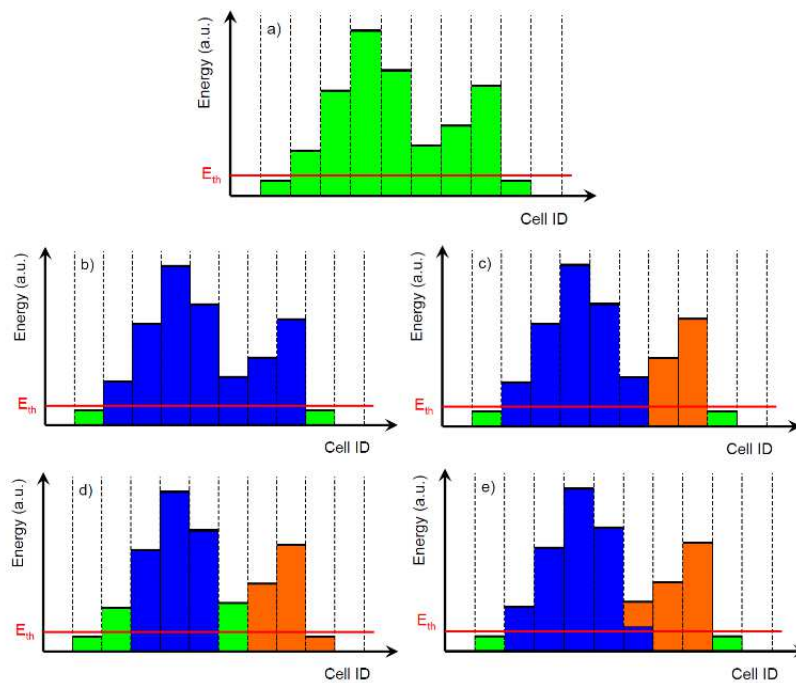


Fig. 9: Comparison of different algorithms of clusterization. Boxes represent energy in cells. E_{th} is clusterization threshold $fMinECut$. a) Energy in cells before clusterization marked by green color. b) Result of V1 clusterizer. There is one big cluster made of cells in blue color. Green cells are below threshold and not associated to the cluster. c) Result of V2 algorithm. There are two clusters made of blue and orange cells. Green cells are below threshold and not associated to any cluster. d) Result of NxN clusterizer. There are two clusters made of blue and orange cells. Green cells are not associated to any cluster. e) Result of V1 algorithm with unfolding. There are two clusters made of blue and orange cells. One cell is associated to two clusters and its energy is shared. Green cells are below threshold and not associated to any cluster.

884

885 5.3.5 Clusterizer NxN

886 The highest energy digit which exceeds energy threshold $fMinECut$ is looked for in the pool of
 887 digits. This digit is a seed for a new precluster. To form a precluster we loop over remaining
 888 digits and check whether they are neighbours of the seed digit. The energy of a neighbour
 889 should be smaller than the energy of the seed. Here neighbours are defined in the other way than

890 in the V1 clusterizer: row index difference or column index difference must be less or equal
891 1. In such requirement one cell can have eight neighbours at maximum. Here neighbours must
892 fulfill also timing condition. The absolute value of time difference for two digits must be less or
893 equal `fTimeCut`. The precluster starts to be a cluster only if a precluster energy is larger than
894 clustering threshold given by `fECAClusteringThreshold`. If the requirement is satisfied a new
895 cluster is formed from the precluster and digits which belong to precluster are removed from the
896 pool of digits. Otherwise only the seed digit is removed from the pool of digits. The procedure
897 is repeated but with a new seed if available. Here maximum size of a cluster is 3×3 cells.
898 However, digits associated to the cluster do not fulfill the energy threshold condition (energy
899 of digit greater than `fMinECut`). The special case when cluster is formed from digits in two
900 super-modules at the same SM- ϕ angle is also supported.

901 Different methods of clusterization are compared in Fig. 9.

902 **5.3.6 Cluster in the EMCal**

903 A cluster is represented by the `AliEMCALRecPoint` class. This class contains an information
904 about cluster itself (energy, multiplicity, local or global position, etc.), features of the cluster
905 (shower ellipse axes, dispersion, etc.) and digits belonging to the cluster (index, energy, etc.).
906 The list of important fields is shown in Tab. 6.

Name	Type	Explanation
fAmp	Float_t	summed amplitude of digits
fIndexInList	Int_t	the index of this RecPoint in the list stored in TreeR (to be set by analysis)
fGlobPos	TVector3	global position
fLocPos	TVector3	local position in the sub-detector coordinate
fMulDigit	Int_t	total multiplicity of digits
fMulTrack	Int_t	total multiplicity of tracks
fDigitsList	Int_t*	[fMulDigit] list of digit's indexes from which the point was reconstructed
fTracksList	Int_t*	[fMulTrack] list of tracks to which the point was assigned
fClusterType	Int_t	type of cluster stored: v1
fCoreEnergy	Float_t	energy in a shower core
fLambda[2]	Float_t	shower ellipse axes
fDispersion	Float_t	shower dispersion
fEnergyList	Float_t*	[fMulDigit] energy of digits
fAbsIdList	Int_t*	[fMulDigit] absId of digits
fTime	Float_t	Time of the digit with maximal energy deposition
fNExMax	Short_t	number of (Ex-)maxima before unfolding
fCoreRadius	Float_t	The radius in which the core energy is evaluated
fDETracksList	Float_t*	[fMulTrack] list of tracks to which the point was assigned
fMulParent	Int_t	Multiplicity of the parents
fMaxParent	Int_t	Maximum number of parents allowed
fParentsList	Int_t*	[fMulParent] list of the parents of the digits
fDEParentsList	Float_t*	[fMulParent] list of the parents of the digits
fSuperModuleNumber	Int_t	number identifying super-module containing recpoint, reference is cell with maximum energy
fDigitIndMax	Int_t	Index of digit with max energy in array fAbsIdList
fDistToBadTower	Float_t	Distance to nearest bad tower
fSharedCluster	Bool_t	States if cluster is shared by 2 super-modules in same phi rack (eg. 0,1)

Table 6: Basic fields in the AliEMCALRecPoint class.

907 5.4 Cluster-Track matching - Rongrong, Shingo, Michael

908 Even though EMCal is intended to measure the energy of particles that interact with EMCal via
909 electromagnetic showering, e.g. photons and electrons, charged hadrons can also deposit energy
910 in EMCal, most commonly via minimum ionization, but also via nuclear interactions generating
911 hadronic showers. In the analysis where the distinction between hadronic and electromagnetic
912 showers is necessary, cluster-track matching is often used to meet this requirement.

913 The main method used to extrapolate tracks in the ALICE software framework is:

```
914 1 static Bool_t PropagateTrackToBxByBz (AliExternalTrackParam *track, Double_t x,  
Double_t m, Double_t maxStep, Bool_t rotateTo=kTRUE, Double_t maxSnp=0.8, Int_t  
sign=0, Bool_t addTimeStep=kFALSE);
```

915 which takes the following arguments: “*track*” stores all the information of the starting point for
916 the extrapolation; “*x*” is the coordinate of the destination plane in the local coordinate system;
917 “*m*” is the mass assumption for the track; “*maxStep*” is the step size used in the extrapolation.
918 This method extrapolates the track trajectory to a destination plane in a magnetic field, taking
919 into account the energy loss of the tracks when going through detector materials. However, the
920 energy loss model is tuned for charged hadrons, so it does not work very well for electrons or
921 positrons whose primary energy loss process is bremsstrahlung.

922 For EMCal, the track-cluster matching is done by default in the reconstruction chain and the
923 code is implemented in:

```
924 AliEMCALTracker::PropagateBack (AliESDEvent* esd)
```

925

926 The logic of the matching procedure is the following:

- 927 – Check whether TPC is available in DAQ/reco. See `AliEMCALTracker::LoadClusters()`. In
928 case there is no TPC, ITS specific extrapolation will be used.
- 929 – Find all the EMCal clusters in the event. See `AliEMCALTracker::LoadClusters()`.
- 930 – Find all the good tracks in the event. See `AliEMCALTracker::LoadTracks()`. Several cuts
931 are applied to select good tracks
 - 932 – Minimum p_T cut, which can be set during the reconstruction.
 - 933 – Cut on number of TPC clusters, which can be set during the reconstruction. This
934 specific cut is avoided in case there is only ITS available in reconstruction.
 - 935 – $|\eta| < 0.8$ and $20^\circ < \phi < 120^\circ$. These fiducial cuts are hard coded since tracks out of
936 this range should never make it to EMCal.

- 937 – For each good track, find the nearest cluster as matched if their residuals fall within the
938 cuts. See `AliEMCALTracker::FindMatchedCluster()`, which follows the following steps:

- 939 – Get the starting point: if the *friendTrack* is available, use the last point on the TPC.
940 Otherwise, use the point at the inner wall of the TPC.

- 941 – If only ITS tracks are available in reconstruction, the propagation will use the track
942 information from the vertex.

- 943 – Extrapolate tracks to the EMCal surface at 430 cm, and apply fiducial cuts on the ex-
944 trapolated points: $|\eta| < 0.75$ and $70^\circ < \varphi < 190^\circ$. The step size in the extrapolation
945 can be set in the reconstruction, and the default value is 20 cm.

- 946 – Extrapolate tracks further, with 5 cm step size, to the positions of all the EMCal
947 clusters which are in the vicinity of the extrapolated points from last step. Then the
948 distance between extrapolated tracks to the clusters are calculated, and the nearest
949 cluster is assigned as matched if the residuals fall within cuts. By fitting the distri-
950 butions of the residuals using Gaussian functions, we can choose to cut on $N\sigma$ of the
951 residuals. To further improve the matching performance, p_T and charge dependent
952 cuts can be used.

953 **5.5 How to execute the reconstruction**

954 Executing the reconstruction is very similar to the simulation case, see the macro `TestEMCAL-`
955 `Reconstruction.C` (a bit more detailed than the one in `$ALICE_ROOT/EMCAL/macros`) :

```

1 void TestEMCALReconstruction() {
2
3 TString detector="EMCAL TPC"; //Same function as in Simulation.C
4 // TString detector="EMCAL ITS"; if user wants ITS tracking to be used.
5
6 AliReconstruction rec; //Create reconstruction object
7
8 //Making Tracking
9 rec.SetRunTracking(detector) ;
10
11 // Particle Reconstruction . Make Rec Points
12 rec.SetRunReconstruction(detector);
9563
14 //read RAW data. Give directory where raw data is stored
15 // rec.SetInput ("RawDataDirectory/raw.root");
16
17 //Make vertex finder
18 rec.SetRunVertexFinder(kFALSE) ; //false only if the tracking detectors are not included.
19
20 // Fill ESD file with RecPoints information .
21 rec.SetFillESD(detector) ;
22
23 //Run Reconstruction
24 rec.Run() ;
25 }

```

957

958 **6 Calibration and detector behavior**

959 **6.1 Calibration**

960 This section describes how different correction factors are obtained: the energy calibration (MIP,
961 π^0 , run by run), the time calibration and the bad channel mask.

962 All these correction factors or masks are stored in the OCDB but also the OADB. Since these
963 calibration parameters do not arrive before the full ALICE data reconstructions of the first pe-
964 riods are completed, the parameters are stored not only in the OCDB but also in the OADB so
965 that the clusters can be corrected at the analysis level. For the moment we do not store the time
966 calibration and run by run correction factors in OCDB just in OADB.

967 **6.1.1 Energy calibration: MIP calibration before installation - Julien**

968 First, the calibration is done on cosmic measurements before installing the SuperModules at P2,
969 but the accuracy obtained using MIPs is not good enough.

970 **6.1.2 Energy calibration: π^0 - Catherine**

971 The energy calibration relies during data taking on the measurement of the π^0 mass position
972 per cell. Each tower has a calibration coefficient. In what follows, a calibration parameter is
973 equal to the result of the fitted mass over the PDG mass value, where the fitted mass denotes
974 the mass given by a gaussian fit on the π^0 invariant mass peak distribution in a given tower
975 (plus a combinatorial background, fitted by a 2nd degree polynomial). About 100-200 M events
976 EMCAL (L0) triggered (trigger threshold at 1.5-2 GeV) allow to calibrate a majority of the
977 towers. The towers located on rows 0 and 23 of each super modul (SM) and those behind the
978 support frame (about 5 columns per SM) have much fewer statistics and would need a minimum
979 of 150 Mevts (probably more). It is to be noted that the run-to-run temperature variations change
980 the towers' response in a non-uniform way, i.e. the width of the π^0 peak increases, and the mean
981 π^0 mass is shifted differently for the various towers. Also the π^0 mass shifts to lower values for
982 the towers with material in front, due to photoconversion close to the EMCAL surface.

983 A few iterations on the data, obtaining in each iteration improved calibration coefficients, are
984 needed to achieve a good accuracy (1-2%). Since the online calibration has a strong effect on
985 the trigger efficiency, the voltage gains of the APDs are varied after each running period, to get a
986 uniform trigger performance. Still, some towers are difficult to calibrate because they are behind
987 of a lot of material (TRD support structures). For those MIPs or J/Ψ measurements could help.

988 **π^0 Calibration Procedure**

989 Since π^0 s decay into 2 gammas, their invariant mass is calculated from the energy of 2 clusters
990 (and angle between the clusters). The position of the invariant mass peak of a tower therefore
991 doesn't depend only on its response and calibration coefficient, but also on an average of the
992 responses and calibration coefficients of all the other towers of the SM, weighted by how often
993 they appear in combination with a cluster in the considered tower. The 2nd effect, of weaker
994 magnitude maybe, originates from the fact that a cluster most often covers more than the con-
995 sidered tower. To simplify the calibration process, the calibration coefficient is calculated as if

996 the whole energy of the cluster was contained in the tower of the cluster which has the largest
997 signal. So the position of the invariant mass peak of a tower also depends on an average of the
998 responses and calib coeffs of its neighbouring towers. For these reasons, the calibration of the
999 calorimeter with the π^0 is an iterative procedure :

- 1000 – Set all calib coeffs to 0 in OCDB.
- 1001 – Reconstruct the π^0 's with these OCDB coeffs.
- 1002 – Run the analysis code on this data to produce the analysis histograms and a 1st version of
1003 the calib coeffs.
- 1004 – Look at the fits on the towers invariant mass histograms and discard the value (or set it by
1005 hand) of the calib coeff of the towers for which the fit can't be trusted.
- 1006 – Create a 1st set of OCDB coeffs.
- 1007 – Reconstruct the π^0 's with these OCDB coeffs.
- 1008 – Run the analysis code on this data to produce the analysis histograms and a 2nd version
1009 of the calib coeffs.
- 1010 – Look at the fits on the towers invariant mass histograms and discard the value (or set it by
1011 hand) of the calib coeff of the towers for which the fit can't be trusted.
- 1012 – Create a 2nd set of OCDB coeffs.
- 1013 – Etc..., until the invariant mass is satisfactory in all the towers.

1014 When the statistics is enough, 4 iterations should be enough to finalize the calibration (in prac-
1015 tice, more are needed, due to outliers or studies that are needed).

1016 There are 3 sets of codes :

- 1017 – Reco code : reads the data, reconstructs the π^0 inv mass distrib in each tower after it applies
1018 some cuts on the clusters and π^0 parameters. The output is a root file with invariant mass
1019 histograms (per tower, and summed-up per SM, per pT-bin).
- 1020 – Analysis code : reads the file produced by the reco code and analyses the histograms to
1021 produce the calib coeffs. This code is the one I present in what follows.
- 1022 – A code which reads the calib coeffs and writes them into a format that is loadable to
1023 OCDB.

1024 The code is located in EMCAL/calibPi0/ :

- 1025 – macros/ : contains the various macros.

- 1026 – input/ : contains the root files produced by the analysis code for the various iterations
1027 ("passes"). It has subdirectories "pass0/", "pass1/", etc... with, in each dir, the root file.
- 1028 – output/ : contains the various files produced by the analysis code for the various passes.
1029 It has subdirectories "pass0/", "pass1/", etc... with, in each dir, the various output files
1030 related to the pass.³

1031 The cuts which must be put in the reconstruction are :

- 1032 – Bad towers masked.
- 1033 – Both clusters in the same SM (to avoid misalignment effects).
- 1034 – Cut the 1-tower clusters out.
- 1035 – 20 ns timing cut.
- 1036 – Non-linearity correction (for the cluster energy)– from beam test AFAIK.
- 1037 – No asymetry cut.
- 1038 – $E_{cluster} > 0.8$ GeV, or 0.7 GeV if there is little statistics. Tests showed that to remove the
1039 residual non-linearity (the π_0 invariant mass rises with p_T), tightening the cut on $E_{cluster}$
1040 was more efficient than requiring symmetric decays (both gamma's of similar energy) (e.g.
1041 $asym < 0.5$ with $E_{gamma} > 0.5$ GeV).

1042 It has the possibility to mask some areas. This is useful to disentangle the zones which have
1043 more material in front of them from those which don't. In the invariant mass distributions, the
1044 π^0 candidates kept are only those for which both clusters belong to the non-masked zones. In
1045 2011, we considered masking the zones behind the support frame (in all the SMs or only in the
1046 SMs with TRD modules in front of them, i.e. SM 6-9 that year), plus additionnal problematic
1047 zones, to avoid taking clusters in these zones for the calculation of the average invariant mass in
1048 the towers with less material. (NB : not used for final calibration results, but for studies).

1049 The analysis code has 3 input files :

- 1050 – the root file f05 with inv mass histograms produced by the reconstruction code,
- 1051 – a file txtFileIn (output_calibPi0_parameters.txt) that contains the values of various param-
1052 eters of the fit for each tower, at the previous pass,
- 1053 – a file txtFilePrevCalib (output_calibPi0_coefs_clean.txt) that contains the value of the
1054 calibration coefficient for each tower, at the previous pass (and after the hand-made cor-
1055 rections).

³Note that it wouldn't necessarily help to set-up a code that automatically reads and writes the pass number to avoid the hardcoded directories in the code, because it happens to do several times the same pass with various parameters (e.g. cuts in the reconstruction, or more statistics, or various masked zones, or hand-customization of a few calib coeffs, etc...).

1056 The 2 last files are therefore useless for the "pass0". To run the code for "pass0" (1st iteration),
1057 put the name of a valid file (e.g. one of last year) and just ignore the plots (red colour, in the last
1058 section – see below).

1059 There are 4 output files, that are written in the current directory (calibPi0/) : be careful not to
1060 overwrite an existing file ! After the code has been run, simply move those files to the relevant
1061 passXX directory := output/passXX/=.

- 1062 – a postscript file psfile (output_calibPi0.ps) with the plots described below,
- 1063 – a root file rootFileOut (output_calibPi0.root) that contains the same plots in root format,
- 1064 – a file txtFileOut (output_calibPi0_parameters.txt) that contains the values of various pa-
1065 rameters of the fit for each tower, for the current pass,
- 1066 – a file outputFile (output_calibPi0_coefs.txt) that contains the value of the calib coeff for
1067 each tower, for the current pass. Once the code has been run and the output files copied to
1068 the relevant output directory, I copy output_calibPi0_coefs.txt to output_calibPi0_coefs_clean.txt,
1069 and modify the latter by hand to put the desired calib coefs where we estimate that they
1070 can't be trusted.

1071 9 parameters are defined to qualify the invariant mass distribution in each tower : the distribution
1072 is fitted by a gaussian + pol2 for the combinatorial background. The parameters are :

- 1073 – amplitude of gaussian fit,
- 1074 – mean of the gaussian fit,
- 1075 – sigma of the gaussian fit,
- 1076 – c, b and a parameters of the combinatorial background fit $ax^2 + bx + c$, I (histo integral),
- 1077 – I-S, S (integral of the gaussian fit). Minimal and maximal cut values are hardcoded (and
1078 to be changed at each iteration) for each parameter.

1079 When the value of all the parameters lie between both extremes, the tower (i.e. the fit values,
1080 hence the mean, hence the calculated calib coeff) is "trusted". If one or more parameter has a
1081 value beyond the max cut value or below the min cut value, the tower is "untrusted". Because
1082 these cut values can't be guessed in advance, the analysis code must be run twice per pass.
1083 The 1st time, so as to get the distributions of all 9 parameters, and decide on the basis of those
1084 distributions what are the suitable cut values to separate the towers to be trusted and those not
1085 to be trusted. The values are plugged in the code, and the code is then run a 2nd time, for
1086 real this time. The macro (currently called DrawJulienFullEMCAL6.C) runs with 1 parameter
1087 in argument (set to 10 by default) : choice, which sets the number of SMs that one desires to
1088 include in the analysis. The values are either 4 (for the older SMs), or 6 (for only the newer
1089 SMs), or 10 (for the whole EMCAL). Here is the code. The macro is run this way :

1090 `aliroot -b -q 'macros/DrawJulienFulleMCAL6.C++(10)'`

1091 There are various places where things must be customized before running the code ; they can be
1092 spotted by searching for this line : `//CUSTOMIZE customize ;.`

- 1093 – testChoice : this variable is a flag that allows to shorten the execution time for tests. 0 =
1094 not a test ; 1 = runs with only the 2 first columns of each SM ; 2 = runs with only the 2
1095 first columns of the first SM,
- 1096 – the root input file f05,
- 1097 – the text input file txtFilePrevCalib (in principle not the name, only the path),
- 1098 – the text input file txtFileIn (in principle not the name, only the path),
- 1099 – if necessary : the min and max range values for the parameter histograms : tabMin and
1100 tabMax,
- 1101 – the min and max cut values for the parameters cutMin and cutMax,
- 1102 – if necessary : the number of bins in pT (for the 1st section, see below) nbPtBins and their
1103 range tabPtBins.
- 1104 – Text output on the standard output ("printf's") :

1105 Finally, the first iteration needs the recalibration factors. This file is made running macros/Recal-
1106 ibrationFactors_TextToHistoJulien_mult_2012.C on the output_calibPi0_coefs.txt file. Once
1107 the RecalibrationFactors.root file is created it needs to be linked properly to re-run the recon-
1108 struction.

1109 **6.1.3 Energy calibration: Run by run temperature gain variations - Evi, David**

1110 The SuperModules calibration depends on the temperature dependence of the different tow-
1111 ers gains. We observe that from one period to other, where the T changes, the π^0 peak po-
1112 sitions also changes. There are 2 ways to correct for this effect : either measure the mean
1113 T per run, and get the gain curves per tower a calculate the corresponding correction; or use
1114 the calibration LED events to quantify the variation from one reference run. Each of those 2
1115 procedures have problems, poor or lack of knowledge of the gain curves of some towers or
1116 bad performance of the LED system in certain regions. These temperature or time-dependent
1117 corrections are still under study: for further, and up-to-date, information, please see the wiki:
1118 <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EMCalTimeDependentCalibrations>

1119 **6.1.4 Time calibration - Marie**

1120 The time of the amplitude measured by a given cell is a good candidate to reject noisy towers,
1121 identify pile up events when coming from different Bunch Crossing, or even identify heavy
1122 hadrons at low energy. The average time is around 580 ns. The aim of the time calibration

1123 is to do a relative calibration between cells to align all cells to a mean value of 0 ns, with as
1124 small spread as possible (negative values are unavoidable for the moment). The time calibration
1125 coefficient for each cell is the result of the average time of the cell when belonging to a cluster
1126 with enough energy (>1GeV). The calibration coefficient have to be subtracted to the cell time.

1127 **Time Calibration Procedure**

1128 Since the some variations of mean time have been observed depending on the bunch cross num-
1129 bers (BC) % 4 the computation of the time coefficients is done for each bunch cross numbers
1130 BC% 4 scheme.

1131 The time calibration coefficient computing is done in 2 iterations.

1132 – 1st iteration:

1133 Get Bunch Cross Number for the event. Loop on all cluster of the event.
1134 Loop on all cells in the cluster. If cell amplitude is > 0.9 GeV and 500ns < cell time <
1135 700ns then compute average per cell per BC%4 and fill in 1D histogram with calibration
1136 coefficients: hAveragesBCx where x stands for the result of BC%4.

1137 – 2nd iteration:

1138 Get Bunch Cross Number for the event.
1139 Loop on all cluster of the event.
1140 Loop on all cells in the cluster.
1141 Get Calibration coefficient for BC%4 from hAveragesBCx. If cell amplitude is > 0.9 GeV
1142 and -20ns < (cell time-cell Calibration Coefficient) < 20ns.
1143 Compute average time per cell per BC%4 and fill in 1D histogram with those new calibra-
1144 tion coefficients: hAllAveragesBCx.

1145 **Acces time calibration coefficients**

1146 The time calibration coefficient are stored in OADB in TH1D histograms named hAllTimeAvBCx
1147 where x stands for the BC%4 value. They have the usual structure: runrange/pass/

1148 To use them in your analysis you may do the following:

```

1  AliOADBContainer *contTRF=new AliOADBContainer("");
2  contTRF->InitFromFile(Form("%s/EMCALTimeCalib.root", foADBFilePathEMCAL.Data()), "
    AliEMCALTimeCalib");
3  TObjArray trecal=(TObjArray) contTRF->GetObject(runnumber);
4  if(trecal)
5  {TObjArray trecalpass=(TObjArray) trecal->FindObject(pass);
6  if(trecalpass)
11497 {printf("AliCalorimeterUtils::SetOADBParameters() - Time Recalibrate EMCAL \n");
8  for (Int_t ibc = 0; ibc < 4; ++ibc)
9  {
10 TH1F *h = GetEMCALChannelTimeRecalibrationFactors(ibc);
11 if (h)
12 delete h;
13 h = (TH1F*) trecalpass->FindObject(Form("hAllTimeAvBC%d", ibc));
14 }}

```

1150

1151 As already mentioned the time calibration of the cells is not done at the reconstruction level but
1152 offline during analysis. The methods to recalibrate cells is implemented in the class \$ALICE_ROOT/
1153 EMCAL/AliEMCALRecoUtils. The method RecalibrateCellTime(absId, bc, time): is called
1154 by the method SwitchOnTimeRecalibration, modifies the provided time with the calibration
1155 parameters. The inputs are the bunch crossing number that can be recovered from the event with
1156 InputEvent()->GetBunchCrossNumber(), and the absolute ID of the cell. The way to pass the
1157 calibration parameters to AliEMCALRecoUtils is the following.

```

1  AliCalorimeterUtils *cu = new AliCalorimeterUtils ;
2  TGeoManager::Import("geometry.root") ; //need file "geometry.root" in local dir !!!!
3  AliEMCALRecoUtils * reco = cu->GetEMCALRecoUtils();
1158 4  and then
5  for(Int_t i =0; i< 4; i++) reco->SetEMCALChannelTimeRecalibrationFactors( i, (TH1F
    *) file->Get(Form("hAllTimeAvBC%d", i)))

```

1159 , where TFile *file is in the file containing the time calibration coefficients.

1160 Some more details on time recalibration can be found in twikis: https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EMCalCode:HowTo#Energy_and_Time_calibration and in
1161 <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EMCALTimeCalibration>
1162

1163 6.2 Alignment - Marco

1164 CERN provides survey measurements of the position of different EMCAL Supermodules points
1165 at the beginning of the running period (and on request?). As soon this information is available,
1166 the ideal EMCAL positions used in the reconstruction by default, are corrected with special
1167 position matrices calculated from the measurements. Finally, once the data is reconstructed,
1168 the accuracy of the alignment is cross checked with track matching and π^0 mass measurements,

1169 since those values change depending on variations on the positions of the SuperModules.

1170 **6.3 Bad channel finding - Alexis**

1171 The analysis is done on the output of offline Quality assurance see section 10.2 histograms TH2F
1172 EMCAL_hAmpId containing the distribution of amplitudes (energy) of cell versus cell Absolute
1173 ID number (AbsId). The idea is to check distributions over the cells of different observables
1174 extracted from this histograms. Then each cell is tested regarding to the distribution over all the
1175 cells for each observable. The different tests are the following:

- 1176 1. average energy (average computed for $E_{min} < E < E_{max}$)
- 1177 2. average number of hit per event (average computed for $E_{min} < E < E_{max}$)
- 1178 3. Shape criteria : A fit of the cell energy (amplitude) distribution is performed with the
1179 function: $A * e^{-B*x} / x^2$ between E_{min} and E_{max} . Then the χ^2/ndf , A and B which are
1180 parameters from the fit of each cell amplitude.

1181 Each criteria is tested at least once (they can be tested also for different energy interval). At
1182 the end of each test the marked cells are excluded (if above $n\sigma$ from mean value, usually
1183 $n\sigma$ is taken equal 4 or 5) before computing the next distribution.

1184 The typical $n\sigma$ used is 4 or 5. The min energy considered is 0.1 GeV/0.3 GeV. And the
1185 maximum energy depends on the data (minbias or triggered data).

1186 There are different levels of classification for cells which will enter the deadMap:

- 1187 – kAlive = 0: cell is OK
- 1188 – kDead = 1: cell is dead
- 1189 – kHot = 2: cell is bad
- 1190 – kWarning=3: cell may have problems (warm or miscalibrated)

1191 The distinction of the bad/warm status is done by visual check of the energy distribution of the
1192 cells detected by the different tests described above.

1193 In the reconstruction pass the only the cells marked as kHot and kDead are not reconstructed.

1194 The macro to compute the BadCellAnalysis can be found on
1195 `$ALICE_ROOT/PWGGA/CaloTrackCorrelations/macros/QA/BadChannelAnalysis.C`. This macro
1196 allows first to merge all the individual run histograms in order to get the energy distribution over
1197 a large statistics. Usually this is done at the end of a data taking period. Then it computes the
1198 different tests and produces the text file with the dead and bad cells detected. Finally the macro
1199 produce a pdf file containing the individual energy distribution for all the detected cells.

1200 All the results and OCDB created are updated on the following twiki:
1201 <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EMCalQABadChannels>.

1202 **7 Trigger**

1203 **7.1 L0 - Jiri**

1204 Documented in [10]. Add Summary or more info here.

1205 **7.2 L1 - Rachid**

1206 **7.3 L0-L1 simulation - Rachid**

1207 **8 The EMCal HLT online chain - Federico**

1208 The EMCal L0 or L1 hardware trigger decisions provide the input for a dedicated on-line event
1209 processing chain running on the HLT cluster, where further refinement based on criteria using the
1210 full event reconstruction information is performed. In fact, the detector optical link transports
1211 the raw data to the Read-Out Receiver Card (RORC) in the local data collector of the data
1212 acquisition system, which sends a complete copy of the readout to a set of specialized nodes in
1213 the HLT cluster (FEP or Front End Processors). Each FEP node is equipped with RORC cards in
1214 analogy to the collector nodes used by the data acquisition. The FEP nodes are physically linked
1215 to the detector hardware and reflect the geometrical partitioning of each ALICE sub-system.
1216 The 10 full-size super-modules are read out using 2 Read-Out Control Units (RCUs) for a total
1217 of 20 optical links running into the HLT FEPs. The reduced-size super-modules were installed
1218 prior to the 2012 LHC run and are not discussed in the present report. In addition to the 20
1219 links from the super-module readout, the HLT receives also a copy of the L0/L1 trigger data
1220 stream via an additional optical link from the EMCal jet trigger unit (STU) data collector. The
1221 different stages of data processing are then performed by the software analysis chain executed on
1222 the HLT cluster: a set of general purpose nodes (Computing Nodes or CNs) perform the higher
1223 level operations on the data streams which have been already pre-processed on the FEPs at the
1224 lower level. The EMCal software components form a specialized sub-chain executed at run time
1225 together with all other ALICE sub-systems participating in the HLT event reconstruction.

1226 The functional units of the EMCal HLT online chain are presented in Figure 10 where the online
1227 reconstruction, monitoring, and trigger components are shown together with their relevant data
1228 paths. The lower-level EMCal online component (*RawAnalyzer*) is fed by the detector front
1229 end electronics and performs signal amplitude and timing information extraction. Intermediate
1230 components (*DigitMaker*) use this information to build the digitized data structures needed for
1231 the clusterizer components to operate on the cell signals. Alternatively, the digitized signals can
1232 be generated via monte carlo simulations (*DigitHandler*).

1233 At the top of the EMCal reconstruction chain, the digits are summed by the *Clusterizer* compo-
1234 nent to produce the cluster data structures. The calorimeter clusters are then used to generate
1235 the different kinds of EMCal HLT trigger information: a single shower trigger (γ) with no track
1236 matching, an electron trigger using the matching with a corresponding TPC track, and a jet
1237 trigger also using the TPC tracks information and the V0 multiplicity dependent threshold.

1238 The trigger logic generated by the EMCal chain is evaluated (together with the outputs of the
1239 HLT trigger components coming from other ALICE detectors) within the HLT Global Trigger

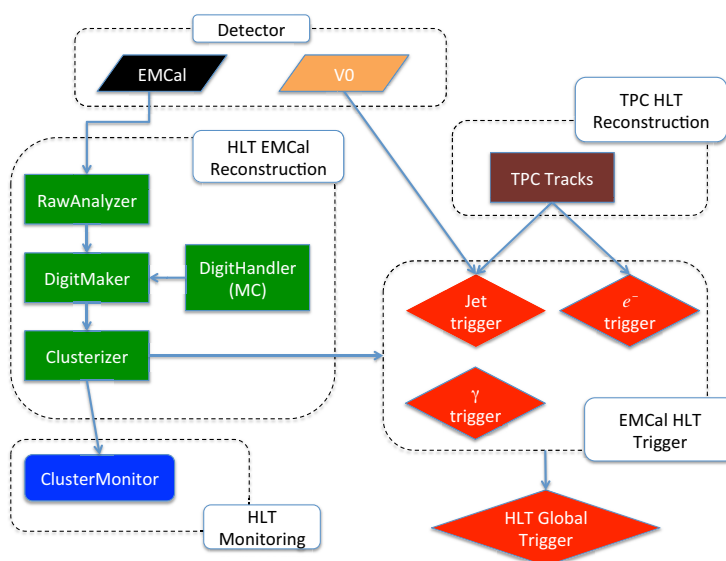


Fig. 10: Functional diagram of the EMCal online reconstruction components (signal processing, data structure makers, and clusterizers) shown in green. The EMCal chain is fed by the detector raw data. Trigger components are shown in red. EMCal-specific triggers operate on the calorimeter clusters and perform TPC track-matching when needed (electron and jet triggers). Monitoring components are shown in blue and live in a separate monitoring chain. The EMCal triggers are evaluated within the Global Trigger which is aware of the full HLT trigger logic of the other ALICE detectors.

1240 which produces the final high level decision based on the reconstructed event. The ALICE data
 1241 acquisition system will then discard, accept or tag the event according to the HLT decision.

1242 For performance and stability reasons, the full on-line HLT chain contains only analysis and
 1243 trigger components. On the other hand, monitoring components typically make heavy use of
 1244 histogramming packages and ESD objects, hence they are kept in a separate chain. The isolation
 1245 of the monitoring from the reconstruction chain gives additional robustness since a crash in a
 1246 monitoring component will not affect the reconstruction chain and the data taking.

1247 8.1 Reconstruction components

1248 As shown in Figure 10 the EMCal HLT analysis chain provides all the necessary components
 1249 to allow the formation of a trigger decision based on full event reconstruction. The following
 1250 subsections are devoted to a detailed discussion of each processing stage, starting from the most
 1251 basic, i.e. signal extraction, to the highest stage: the HLT trigger decision.

1252 8.1.1 RawAnalyzer

1253 The *RawAnalyzer* component extracts energy and timing information for each calorimeter cell.
 1254 Extraction methods implemented in the offline code (AliRoot) typically use least squares fitting
 1255 algorithms, and cannot be used in online processing for performance reasons. Conversely, the
 1256 HLT signal extraction is done without need of fitting using two possible extraction methods. The

1257 first method, referred to as *kCrude*, simply produces an amplitude using the difference between
1258 the maximum and the minimum values of the digitized time samples and associates the time
1259 bin of the maximum as the signal arrival time. The *kCrude* method was used during the 2011
1260 data taking: it has the advantage of being extremely fast and fully robust since no complex
1261 algorithms are used. On the other hand, it produces a less accurate result than the processing
1262 of the full signal shape. An alternative method (*kPeakFinder*) evaluates the amplitude and peak
1263 position as a weighted sum of the digitized samples. This approach is not as fast as *kCrude* but
1264 is a few hundred times faster than least squares fitting.

1265 **8.1.2 DigitMaker**

1266 The *DigitMaker* component essentially transforms the raw cell signal amplitudes produced by
1267 the *RawAnalyzer* into digit structures by processing the cell coordinates and by the application
1268 of dead channel maps and the appropriate gain factors (low and high-gain).

1269 **8.1.3 Clusterizer**

1270 The *Clusterizer* component merges individual signals (digits) of adjacent cells into structures
1271 called clusters. At transverse momenta $p_T > 1$ GeV/c most of the clusters are associated to elec-
1272 tromagnetic showers in EMCal from π^0 and η mesons decays. Other sources of electromagnetic
1273 showers are direct photons and electrons from semi-leptonic decays of c and b hadrons. Since
1274 the typical cluster size in the EMCal can vary according to the detector occupancy due to shower
1275 overlap effects, which are much different for pp and heavy-ion collisions, clustering algorithms
1276 with and without a cutoff on the shower size are available (both in offline and in the HLT) to
1277 optimize the cluster reconstruction for the different cases. Events originating from pp collisions
1278 tends to generate smaller, spherical and well-separated clusters in the EMCal, at least up to 10
1279 GeV/c. At higher transverse momenta, overlapping of the showers requires a shape analysis to
1280 extract the single shower energy. Above 30 GeV/c the reconstruction can be performed only
1281 with more sophisticated algorithms such as isolation cuts to identify direct photons.

1282 The identification of an isolated single electromagnetic cluster in the EMCal can be performed
1283 using different strategies: summing up all the neighboring cells around a seed-cell over threshold
1284 until no more cells are found or adding up cells around the seed until the number of clustered
1285 cells reaches the predefined cutoff value.

1286 The first approach is more suitable for an accurate reconstruction. A further improvement to this
1287 clustering algorithm would be the ability to unfold overlapping clusters as generated from the
1288 photonic decay of high-energy neutral mesons, however this procedure usually requires comput-
1289 ing intensive fitting algorithms.

1290 Such performance penalty must be avoided in the online reconstruction so the cutoff technique
1291 is preferred. In the EMCal HLT reconstruction a cutoff of 9 cells is used (according to the
1292 geometrical granularity of the single cell size), so the clusterization is performed into a square
1293 of 3×3 cells. The cutoff and non-cutoff algorithms are referred to as $N \times N$ and $V1$, respectively.

1294 In pp collisions the response of the two methods is very similar since the majority of clusters are
1295 well separated, while in $PbPb$ collisions, especially in central events, the high particle multiplic-

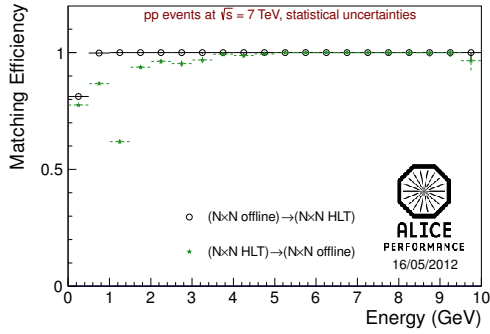


Fig. 11: Reconstruction efficiency for the $N \times N$ algorithm (cutoff) in offline and HLT. The notation $(A) \rightarrow (B)$ indicates the fraction of clusters found using method A that are also found using method B (data from run 154787, period LHC11c).

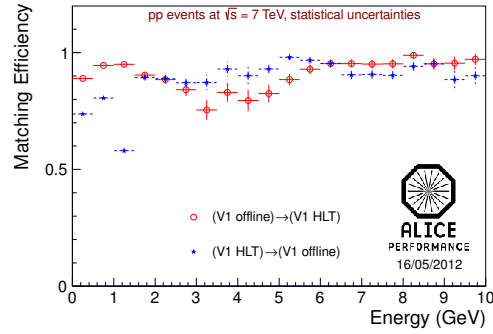


Fig. 12: Reconstruction efficiency for the V1 algorithms (no cutoff) in offline and HLT. The notation $(A) \rightarrow (B)$ indicates the fraction of clusters found using method A that are also found using method B (data from run 154787, period LHC11c).

1296 ity requires the use of the cutoff (or unfolding in offline) to disentangle the cluster signals from
 1297 the the underlying event to avoid the generation of artificially large clusters.

1298 The quality of the EMCal online clusterizer algorithms implemented in the HLT chain were
 1299 checked against offline, as shown in Figures 11 and 12 where it can be seen that the perfor-
 1300 mance is in a reasonable agreement in all cases. The low point at 1.25 GeV is due to bad towers,
 1301 which are assigned an energy of 1 GeV. Bad clusters are removed in later stages of the analysis,
 1302 but that is not yet reflected in Figures 11 and 12. This effect leads to an excess of clusters that
 1303 are found by the HLT clusterizer, but not by the offline clusterizer.

1304

1305 Since the EMCal HLT reconstruction is mainly targeted for triggering, a small penalty in the
 1306 accuracy of the energy reconstruction of the clusters is accepted as a trade off in favor of faster
 1307 performance, and for this reason the cutoff clustering method was used, especially in $PbPb$
 1308 collisions.

1309 8.2 Trigger components

1310 The online HLT chain is capable of producing trigger decisions based on full event reconstruc-
 1311 tion. In terms of EMCal event rejection the following relevant trigger observables have been
 1312 implemented:

- 1313 – neutral cluster trigger
- 1314 – electron and jet trigger

1315 **8.2.1 Cluster trigger**

1316 The single shower triggering mode is primarily targeted to trigger on photons and neutral mesons.
 1317 In all collision systems, the high level trigger post-filtering can improve the hardware L0 and
 1318 L1 trigger response by using the current bad channels map information and calibration factors
 1319 (which could be recomputed directly in the HLT).

1320 **8.2.2 Electron trigger**

1321 For this trigger the cluster information reconstructed online by the EMCal HLT analysis chain is
 1322 combined with the central barrel tracking information to produce complex event selection as a
 1323 single electron trigger (matching of one extrapolated track with an EMCal cluster. Performance
 1324 and accuracy studies of the track matching component developed for this purpose have been
 1325 done using simulated and real data taken during the 2011 LHC running period. Results are
 1326 shown in Figures 13 and 14 where the cluster - track residuals in azimuth and pseudo-rapidity
 1327 units are to be compared with a calorimeter cell size of 0.014×0.014 .

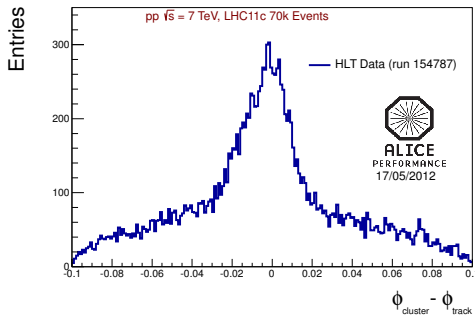


Fig. 13: Distribution of the residuals in azimuth ($\Delta\phi$) for the EMCal cluster and central barrel tracks obtained using the HLT online chain for run 154787 (LHC11c), 70 k events reconstructed.

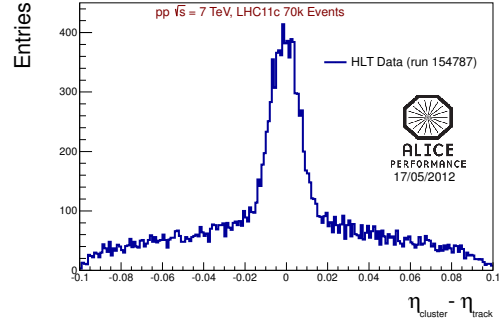


Fig. 14: Distribution of the residuals in pseudo-rapidity ($\Delta\eta$) for the EMCal cluster and central barrel tracks obtained using the HLT online chain for run 154787 (LHC11c), 70 k events reconstructed.

1328 In addition to the extrapolation of the track from the central barrel to the EMCal interaction plane
 1329 and the matching with a compatible nearby cluster, the electron trigger component must finally
 1330 perform particle identification to issue a trigger decision. The selection of electron candidates is
 1331 done using the E/pc information where the energy is measured from the EMCal cluster and the
 1332 momentum from the central barrel track. The trigger component is initialized with default values
 1333 for the cut of $0.8 < E/pc < 1.3$. The default cuts are stored in the HLT conditions database and
 1334 can be overridden via command line arguments at configuration time (usually at start of run).

1335 The performance of the electron trigger was studied using pp minimum bias data at 7 TeV with
 1336 embedded J/Ψ events. Figure 15 shows the good agreement of the E/pc distributions obtained
 1337 with the track extrapolation - cluster matching performed using the online algorithms compared
 1338 to the ESD-based tracking (red).

1339 To determine the possible improvement of the event selection for electrons with energies above

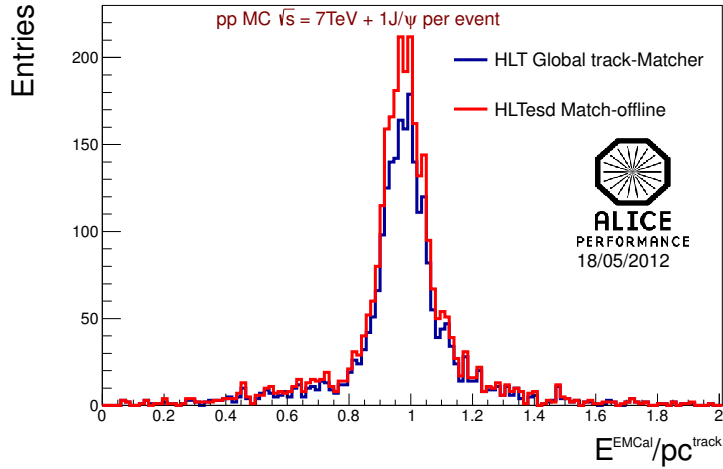


Fig. 15: E/p_c distributions obtained with the track extrapolation - cluster matching via the online algorithms compared to the ESD-based tracking (red).

1340 1 GeV, AliRoot simulations of the HLT chain using LHC11b10a pp minimum bias data at 2.76
 1341 GeV and the EMCal full geometry (10 super-modules) have been used. These studies have
 1342 shown that at least a factor 5 to 10 in event selection can be gained compared to the single
 1343 shower trigger, as shown in Figure 16.

1344 8.2.3 Jet trigger

1345 The EMCal online jet trigger component was developed to provide an unbiased jet sample by
 1346 refining the hardware L1 trigger decisions. In fact, the HLT post-processing can produce a
 1347 sharper turn on curve using the track matching capabilities of the online reconstruction chain.
 1348 In addition, a more accurate definition of the jet area than the one provided by the hardware
 1349 L1 jet patch, can be obtained choosing a jet cone based on the jet direction calculated online.
 1350 The combination of the hadronic and electromagnetic energy provides a measurement of the
 1351 total energy of the jet by matching the tracks identified as part of the jet with the corresponding
 1352 EMCal neutral energy.

1353 The use of the HLT jet trigger also allows a better characterization of the trigger response as
 1354 a function of the centrality dependent threshold by re-processing the information from the V0
 1355 detector directly in HLT.

1356 Performance considerations, due to the high particle multiplicity in $PbPb$ collisions, impose that
 1357 the track extrapolation is done only geometrically without taking into account multiple scattering
 1358 effects introduced by the material budget in front of the EMCal. The pure geometrical extrapo-
 1359 lation accounts for a speedup factor of 20 in the execution of the track matcher component with
 1360 respect to the full-fledged track extrapolation used in pp collisions.

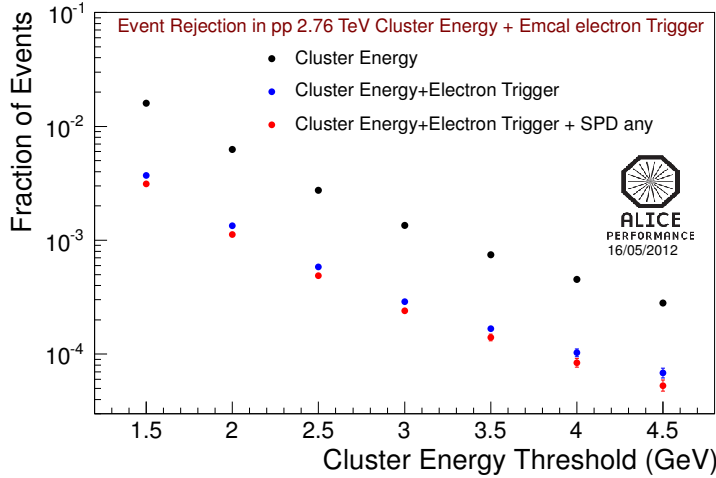


Fig. 16: Improvement in the event selection for $E_{e^-} > 1$ GeV from AliRoot simulation (anchor to LHC11b10a) with minimum bias pp at $\sqrt{s} = 2.76$ TeV (EMCal full geometry). The red points are obtained with the requirement of one hit in one of the silicon pixel (SPD) layers to reject a higher fraction of photon conversions.

1361 The identification of the jet tracks is performed using the anti- k_T jet finder provided by the
 1362 FastJet package.

1363 The EMCal jet trigger was only partially tested during the 2011 data taking period and will be
 1364 fully commissioned for the LHC pPb run period in 2012.

1365 **8.3 Monitoring components**

1366 The role of the EMCal HLT reconstruction in pp collisions is targeted mainly on the monitoring
 1367 functions since the expected event sizes are small enough for the complete collision event to be
 1368 fully transferred to permanent storage.

1369 In this respect, two monitoring components have been developed and deployed in the online
 1370 chain. The first component currently monitors reconstructed quantities, such as the cluster en-
 1371 ergy spectra and timing, the cluster position in the η and ϕ coordinates, and the number of cells
 1372 per cluster as a function of the cluster reconstructed energy as shown in Figure 17.

1373 The second component re-evaluates the EMCal hardware trigger decisions by recalculating the
 1374 cluster energy spectrum for all the clusters with the L0 trigger bit set as shown in Figure 18.
 1375 The L0 turn on curve can then be calculated online as the ratio between the triggered and the
 1376 reconstructed cluster spectra and monitored for the specific run.

1377 No recalculation of hardware L1 trigger primitives was possible during the 2011 data taking
 1378 since the optical link from the EMCal L1 trigger unit could only be installed during the 2011-2012
 1379 winter shutdown of the LHC hence the software development for the L1 trigger monitoring is

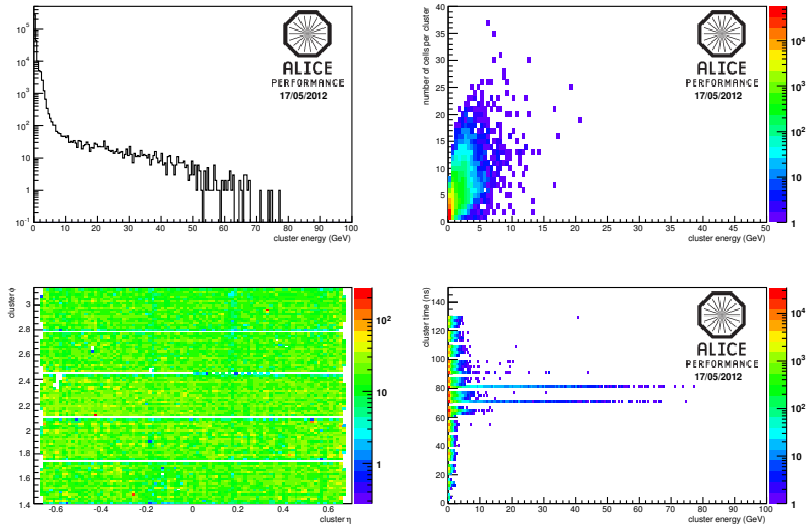


Fig. 17: Output from the EMCal HLT monitoring component. Top left: cluster energy spectra as a function of the reconstructed cluster energy; bottom left: cluster position in η and ϕ coordinates; bottom right: cluster time distribution; top right: number of cells per cluster vs cluster energy. LHC11b period, $\sqrt{s} = 7$ TeV pp data, 10 kEvent analyzed.

1380 still underway.

1381 Documented in [11]. Add Summary or more info here.

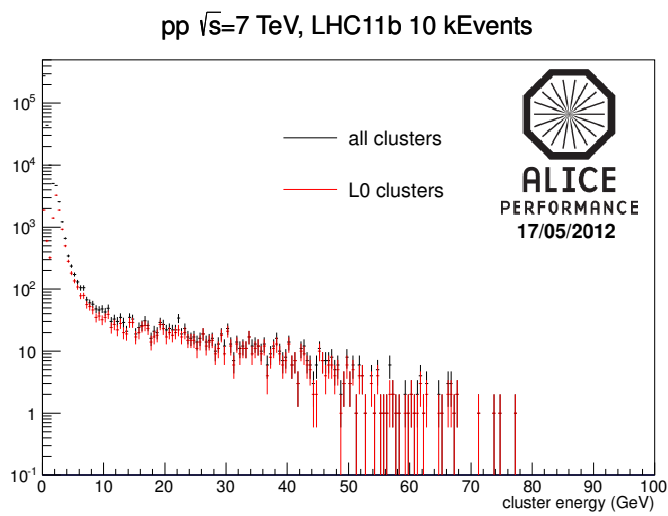


Fig. 18: Energy spectrum for all clusters reconstructed by the EMCal (black points) superposed with the triggered cluster spectrum (i.e. clusters reconstructed which also carry the L0 hardware trigger bit set, red points).

1382 9 Analysis format and code

1383 All the reconstructed particles of all the detectors are kept in a file called **AliESDs.root**. The
1384 detectors must store there the most relevant information which will be used in the analysis.
1385 Together with the **AliESDs.root** file, another file is created with some reference tags of the
1386 simulated events, containing for example the number of events per run. This file is named
1387 **Run0.Event0_1.ESD.tag.root** (1 means that only 1 event was simulated).

1388 In order to do the analysis with the data contained in the ESDs, the only file needed is **AliESDs.root**
1389 in the local directories or a grid collection. No other files are needed in the working directory
1390 (such as **galice.root** nor **EMCAL.*.root**) unless one needs to access the primary particles gener-
1391 ated during the simulation. In that case, the files **galice.root** and **Kinematics.root** are needed
1392 locally. Also, if one want to access to some information of the detector geometry, the **geome-**
1393 **try.root** file is needed.

1394 There are other data analysis containers created from the ESD, the AOD (Analysis Object Data)
1395 with smaller quantity of data for most of the subsystems but for the calorimeters, where we copy
1396 all the information⁴.

1397 9.1 Calorimeter information in ESDs/AODs

1398 The basic calorimeter information needed for analysis is stored in the ESDs or AODs in the
1399 form of **CaloClusters** and **CaloCells** (cell = **EMCal Tower** or **PHOS crystal**). Also there is some
1400 information stored in the AOD/ESD event classes, it will be detailed more in the lines below.
1401 Both AOD and ESD classes derive from virtual classes so that with a similar analysis code and
1402 access methods, we can read both kind of data formats.

1403 9.1.1 *AliVEvent (AliESDEvent, AliAODEvent)*

1404 Those are manager classes for the event information retrieval. Regarding the calorimeters they
1405 have the following access information (getters) methods⁵:

- 1406 – `AliVCaloCluster *GetCaloCluster(Int_t i)` : Returns a **CaloCluster** listed in position "i"
1407 in the array of **CaloClusters**. It can be either **PHOS** or **EMCal** (**PHOS** list of clusters is
1408 before the **EMCal** list).
- 1409 – `TClonesArray *GetCaloClusters()` : Returns the array with **CaloClusters** **PHOS+EMCAL**,
1410 Only defined for AODs
- 1411 – `Int_t GetEMCALClusters(TRefArray *clusters); Int_t GetPHOSClusters(TRefArray *clus-`
1412 `ters)` : Returns an array with only **EMCal** clusters or only with **PHOS** clusters.
- 1413 – `Int_t GetNumberOfCaloClusters()`: Returns the total number of clusters **PHOS+EMCAL**.
- 1414 – `AliVCaloCells *GetEMCALCells(); AliESDCaloCells *GetPHOSCells()` : Returns the
1415 pointer with the **CaloCells** object for **EMCal** or **PHOS**.

⁴until half 2012 everything but the time of the cells was stored

⁵There are the equivalent setters just have a look to the header file of the class

- 1416 – AliVCaloTrigger *GetCaloTrigger(TString calo) : Access to trigger patch information,
1417 for calo="PHOS" or calo="EMCAL"
- 1418 – const TGeoHMatrix* GetPHOSMatrix(Int_t i); const TGeoHMatrix* GetEMCALMa-
1419 trix(Int_t i): Get the matrices for the transformation of global to local. The transformation
1420 matrices are not stored in the AODs.

1421 **9.1.2 AliVCaloCluster (AliESDCaloCluster, AliAODCaloCluster)**

1422 They contain the information of the calorimeter clusters. Note that PHOS and EMCAL Calo-
1423 Clusters are kept in the same TClonesArray (see above). The information stored in each Calo-
1424 Cluster is :

- 1425 – General
 - 1426 – Int_t GetID(): It returns a unique identifier number for a CaloCluster.
 - 1427 – Char_t GetClusterType(): It returns kPHOSNeutral (kPHOSCharged exists but not
1428 used) or kEMCALClusterv1. Another way to get the origin of the cluster:
 - 1429 – Bool_t IsEMCAL(); Bool_t IsPHOS().
 - 1430 – void GetPosition(Float_t *pos) : It returns a x,y,z array with the global positions of
1431 the clusters in centimeters.
 - 1432 – Double_t E() : It returns the energy of the cluster in GeV units.
 - 1433 – void GetMomentum(TLorentzVector& p, Double_t * vertexPosition) : It fills a TLorentzVec-
1434 tor pointing to the measured vertex of the collision. It also modifies the cluster global
1435 positions to have a vector pointing to the vertex, this has to be corrected. Assumes
1436 that cluster is neutral. To be used only for analysis with clusters not matched with
1437 tracks.
- 1438 – Shower Shape
 - 1439 – Double_t GetDispersion(): Dispersion of the shower.
 - 1440 – Double_t Chi2(): Not filled.
 - 1441 – Double_t GetM20() Double_t GetM02() : Ellipse axis.
 - 1442 – UChar_t GetNExMax() : Number of maxima in cluster. Not filled.
 - 1443 – Double_t *GetPID(): PID weights array, 10 entries corresponding to the ones de-
1444 fined in AliPID.h
 - 1445 – enum EParticleType kElectron = 0, kMuon = 1, kPion = 2, kKaon = 3, kProton = 4,
1446 kPhoton = 5, kPi0 = 6, kNeutron = 7, kKaon0 = 8, kEleCon = 9, kUnknown = 10; :
1447 PID tag numbers, corresponding to the PID array
 - 1448 – Double_t GetDistanceToBadChannel() : Distance of the cluster to closest channel
1449 declared as kDead, kWarm or kHot.
 - 1450 – Double_t GetTOF() : Measured Time of Flight of the cluster.

- 1451 – Track-Cluster matching

- 1452 – TArrayI * GetTracksMatched(): List of indexes to the likely matched tracks. Tracks
1453 ordered in matching likeliness. If there is no match at all, by default it contains one
1454 entry with value -1. Only in ESDs.
- 1455 – Int_t GetTrackMatchedIndex(Int_t i): Index of track in position "i" in the list of
1456 indices stored in GetTracksMatched(). Only in ESDs
- 1457 – Int_t GetNTracksMatched() : Total number of likely matched tracks. Size of Get-
1458 TracksMatched() array.
- 1459 – Double_t GetEmcCpvDistance() : PHOS method, not used anymore. Use instead
1460 those below.
- 1461 – Double_t GetTrackDx(void), Double_t GetTrackDz(void): Distance in x and z to
1462 closest track.
- 1463 – TObject * GetTrackMatched(Int_t i): References to the list of most likely matched
1464 tracks are stored in a TRefArray. This method retrieves the one in position "i". Tracks
1465 are listed in order of likeliness. The TObject is a AliAODTrack. Only for AODs

- 1466 – MonteCarlo labels:

- 1467 – TArrayI * GetLabels(): List of indexes to the MonteCarlo particles that contribute to
1468 the cluster. Labels ordered in energy contribution.
- 1469 – Int_t GetLabel(): Index of MonteCarlo particle that deposited more energy in the
1470 cluster. First entry of GetLabels() array.
- 1471 – Int_t GetLabelAt(UInt_t i): Index of MonteCarlo particle in position i of the array
1472 of MonteCarlo indices.
- 1473 – Int_t GetNLabels() : Total number of MonteCarlo particles that deposited energy.
1474 Size of GetLabels() array.

- 1475 – Cluster cells

- 1476 – Int_t GetNCells() : It returns the number of cells that contribute to the cluster.
- 1477 – UShort_t *GetCellsAbsId(): It returns the array with absolute id number of the cells
1478 contributing to the cluster. Size of the array is given by GetNCells().
- 1479 – Double32_t *GetCellsAmplitudeFraction(): For cluster unfolding, it returns an array
1480 with the fraction the energy that a cell contributes to the cluster.
- 1481 – Int_t GetCellAbsId(Int_t i) : It returns the absolute Id number of a cell in the array
1482 between 0 and GetNCells()-1.
- 1483 – Double_t GetCellAmplitudeFraction(Int_t i) : It returns the amplitude fraction of a
1484 cell in the array between 0 and GetNCells()-1.

1485 **9.1.3 AliVCaloCells (AliESDCaloCells, AliAODCaloCells)**

1486 They contain an array with the amplitude or time of all the cells that fired in the calorimeter
1487 during the event. Notice that per event there will be a CaloCell object with EMCAL cells and
1488 another one with PHOS cells.

- 1489 – Short_t GetNumberOfCells(): Returns number of cells with some energy.
- 1490 – Bool_t IsEMCAL(); Bool_t IsPHOS(); Char_t GetType(): Methods to check the origin of
1491 the AliESDCaloCell object, kEMCALCell or kPHOSCell.
- 1492 – Short_t GetCellNumber(Short_t pos): Given the position in the array of cells (from 0 to
1493 GetNumberOfCells()-1), it returns the absolute cell number (from 0 to NModules*NRows*NColumns
1494 - 1).
- 1495 – Double_t GetCellAmplitude(Short_t cellNumber): Given absolute cell number of a cell
1496 (from 0 to NModules*NRows*NColumns - 1), it returns the measured amplitude of the
1497 cell in GeV units.
- 1498 – Double_t GetCellTime(Short_t cellNumber): Given absolute cell number of a cell (from
1499 0 to NModules*NRows*NColumns - 1), it returns the measured time of the cell in second
1500 units.
- 1501 – Double_t GetAmplitude(Short_t pos): Given the position in the array of cells (from 0 to
1502 GetNumberOfCells()-1), it returns the amplitude of the cell in GeV units.
- 1503 – Double_t GetTime(Short_t pos): Given the position in the array of cells (from 0 to GetNumberOfCells()-
1504 1), it returns the time of the cell in second units.
- 1505 – Double_t GetCellMCLable(Short_t cellNumber): Given absolute cell number of a cell
1506 (from 0 to NModules*NRows*NColumns - 1), it returns the index of the most likely MC
1507 label.
- 1508 – Double_t GetCellEFraction(Short_t cellNumber): Given absolute cell number of a cell
1509 (from 0 to NModules*NRows*NColumns - 1), it returns the fraction of embedded energy
1510 from MC to real data (only for embedding)
- 1511 – Double_t GetMCLLabel(Short_t pos): Given the position in the array of cells (from 0 to
1512 GetNumberOfCells()-1), it returns the index of the most likely MC label.
- 1513 – Double_t GetEFraction(Short_t pos): Given the position in the array of cells (from 0 to
1514 GetNumberOfCells()-1), it returns the fraction of embedded energy from MC to real data
1515 (only for embedding)
- 1516 – Bool_t GetCell(Short_t pos, Short_t &cellNumber, Double_t &litude, Double_t &time,
1517 Short_t &mclabel, Double_t &frac); : For a given position of the list of cells, it fills the
1518 amplitude, time, mc lable and fraction of energy.

1519 **9.1.4 AliVCaloTrigger (AliESDCaloTrigger, AliAODCaloTrigger) - Rachid**

1520 **9.2 Macros**

1521 You can find example macros to run on ESDs or AODs in

1522 `$ALICE_ROOT/EMCAL/macros/TestESD.C` or `TestAOD.C`

1523 All the ESDs information is filled via the AliEMCALReconstructor/AliPHOSReconstructor
1524 class, in the method FillESD(). The AODs are created via the analysis class

1525 `$ALICE_ROOT/ANALYSIS/AliAnalysisTaskESDfilter.cxx, .h`

1526 and as already mentioned, for the calorimeters it basically just copies all the information from
1527 ESD format to AOD format.

1528 Below is a description of what information is stored and how to retrieve it. The location of the
1529 corresponding classes is

1530 `$ALICE_ROOT/STEER`

1531 **9.3 Code example**

1532 The analysis is done using the data stored in the ESD. The macro

1533 `$ALICE_ROOT/EMCAL/macros/TestESD.C`

1534 is an example of how to read the data for the calorimeters PHOS and EMCal (just replace where
1535 it says EMCAL by PHOS in the macro to obtain PHOS data). For these detectors we have
1536 to use the ESD class AliESDCaloCluster or AliESDCaloCells to retrieve all the calorimeters
1537 information. For the tracking detectors, the class is called AliESDtrack, but the way to use it is
1538 very similar (see “`$ALICE_ROOT/STEER/AliESDtrack.*`”

1539 and “`$ALICE_ROOT/STEER/AliESDCaloCluster*`” for more details). In AliESDCaloCluster
1540 we keep the following cluster information: energy, position, number of Digits that belong to the
1541 cluster, list of the cluster Digits indeces, shower dispersion, shower lateral axis and a few more
1542 parameters. In AliESDCaloCells we keep the following tower information: amplitude (GeV),
1543 time (seconds), absolute cell number.

1544 The structure of the ESD testing macro (TestESD.C) is the following:

- 1545 – Lines 0-29: This macro is prepared to be compiled so it has “includes” to all the Root and
1546 AliRoot classes used.
- 1547 – Lines 30-36: This macro prints some information on screen, the kind of information is
1548 set here. We print by default clusters information and optionally, the cells information,
1549 the matches information, the cells in the clusters information or the MonteCarlo original
1550 particle kinematics.
- 1551 – Lines 40-64: Here are the methods used to load AliESDs.root , geometry or kinematics
1552 files. Also loop on ESD event is here.

- 1553 – Lines 65-66 Gets the measured vertex of the collision.
- 1554 – Lines 69-78 Loops on all the CaloCell entries and prints the cell amplitude, absolute
1555 number and time.
- 1556 – Lines 84- end: We access the EMCAL AliESDCaloCluster array and loop on it. We get
1557 the different information from the CaloCluster.
- 1558 – Lines 111-130: Track Matching prints. Access to the matched track stored in AliESD-
1559 track.
- 1560 – Lines 133-159: Cells in cluster prints
- 1561 – Lines 161 - end: Access the stack with the MC information and prints the parameters of
1562 the particle that generated the cluster.

1563 **9.4 Advanced utilities : Reconstruction/corrections of cells, clusters during the analysis**

1564 **9.4.1 AliEMCALRecoUtils**

1565 **9.4.2 Tender : AliEMCALTenderSupply**

1566 **9.4.3 Particle Identification with the EMCal**

1567 In the EMCal we have two different ways to obtain the PID of a given particle:

- 1568 1. Shower shape of the cluster: Distinguish electrons/photons and π^0 from other particles.
1569 This is done without any track information in the class AliEMCALPID.
- 1570 2. Ratio between energy of the cluster and the momentum of a matched track: Distinguish
1571 electrons from other particles. This is done in the combined PID framework by the class
1572 AliEMCalPIDResponse.

1573 **AliEMCALPID (AliEMCALPIDutils)** The idea for particle identification with EMCal clus-
1574 ters is that the shower shapes produced in the EMCal are different for different particle species.
1575 The long axis of the cluster (λ_0) is used for this purpose and parametrized for electrons/photons
1576 (should have the same electromagnetic shower), for π^0 (two photons merging in one cluster give
1577 a different shape than one photon), and hadrons (MIP signal).

1578 The main method in this class is RunPID(), which calls AliEMCALPIDutils::ComputePID(Double_t
1579 energy, Double_t lambda0) for each cluster with cluster energy energy and long axis of the
1580 cluster lambda0 in the event. Inside this method first the energy dependent parametrizations
1581 for the three cases (photons, π^0 , and hadrons) are retrieved. The parametrization is done here
1582 with a combination of a Gaussian and a Landau (at the moment there are two parameter sets
1583 available: low and high flux, which can be set by AliEMCALPID::SetLowFluxParam() and
1584 AliEMCALPID::SetHighFluxParam()). Then a conditional probability is assigned to the clus-
1585 ter for each of these three species depending on lambda0. Finally a probability for a cluster

1586 being of a certain particle species is calculated with the Bayesian approach that can be retrieved
 1587 by `AliVCluster::GetPID()`.

```

1 // compute PID weights
2 if( (fProbGamma + fProbPiZero + fProbHadron)>0.){
3     fPIDWeight[0] = fProbGamma / (fProbGamma + fProbPiZero + fProbHadron); // gamma
4     fPIDWeight[1] = fProbPiZero / (fProbGamma+fProbPiZero+fProbHadron); // pi0
5     fPIDWeight[2] = fProbHadron / (fProbGamma+fProbPiZero+fProbHadron); // hadron
6 }
7 ...
8 // default particles
9 fPIDFinal[AliPID::kElectron] = fPIDWeight[0]/2; // electron
10 fPIDFinal[AliPID::kMuon] = fPIDWeight[2]/8;
11 fPIDFinal[AliPID::kPion] = fPIDWeight[2]/8;
12 fPIDFinal[AliPID::kKaon] = fPIDWeight[2]/8;
13 fPIDFinal[AliPID::kProton] = fPIDWeight[2]/8;
1588 // light nuclei
14
15 fPIDFinal[AliPID::kDeuteron] = 0;
16 fPIDFinal[AliPID::kTriton] = 0;
17 fPIDFinal[AliPID::kHe3] = 0;
18 fPIDFinal[AliPID::kAlpha] = 0;
19 // neutral particles
20 fPIDFinal[AliPID::kPhoton] = fPIDWeight[0]/2; // photon
21 fPIDFinal[AliPID::kPi0] = fPIDWeight[1] ; // pi0
22 fPIDFinal[AliPID::kNeutron] = fPIDWeight[2]/8;
23 fPIDFinal[AliPID::kKaon0] = fPIDWeight[2]/8;
24 fPIDFinal[AliPID::kEleCon] = fPIDWeight[2]/8;
25 //
26 fPIDFinal[AliPID::kUnknown] = fPIDWeight[2]/8;

```

1589

1590 **AliEMCalPIDResponse** The idea for particle identification with EMCal clusters AND the
 1591 track information is that electrons are losing their total energy in an electromagnetic shower
 1592 inside the EMCal whereas all other charged particles only part of it. The main observable is
 1593 E/p with the energy of the EMCal cluster (E) and the momentum of a matched track (p). This
 1594 ratio is $E/p \sim 1$ for electrons and $E/p < 1$ for other charged particles.

1595 The decision about a particle species is done within the PID framework provided by ALICE. The
 1596 main classes are: STEER/STEERBase/AliPIDCombined and STEER/STEERBase/AliPIDResponse.
 1597 There are two methods of usage:

- 1598 1. $n\sigma$ method: For each detector the multiples of σ values are given for the deviation from the
 1599 expected mean value at a given p_T (assuming a Gaussian distribution). This can be done
 1600 via: `AliPIDResponse::GetNumberOfSigmas(EDetector detCode, const AliVParticle`
 1601 `*track, AliPID::EParticleType type)`
- 1602 2. Bayesian approach: In `AliPIDCombined::ComputeProbabilities(const AliVTrack`

1603 *track, const AliPIDResponse *response, Double_t* bayesProbabilities) for
 1604 each detector (included in the analysis via
 1605 AliPIDCombined::SetDetectorMask(AliPIDResponse::EDetector)) the conditional
 1606 probability for the respective detector observable is calculated for each particle species
 1607 (selected via
 1608 AliPIDCombined::SetSelectedSpecies(AliPID::EParticleType)). Then the prob-
 1609 ability for a track being of a certain particle type is calculated with the Bayesian approach.
 1610 The initial particle abundances (priors) can be activated via
 1611 AliPIDCombined::SetEnablePriors(kTRUE) and either own priors can be loaded
 1612 (AliPIDCombined::SetPriorDistribution(AliPID::EParticleType type, TH1F *prior))
 1613 or default ones can be chosen (AliPIDCombined::SetDefaultTPCPriors()).

1614 For the case of the EMCAL the $n\sigma$ as well as the conditional probability are calculated in
 1615 AliEMCALPIDResponse::GetNumberOfSigmas(Float_t pt, Float_t eop, AliPID::EParticleType
 1616 n, Int_t charge) and
 1617 AliEMCALPIDResponse::ComputeEMCALProbability(Int_t nSpecies, Float_t pt, Float_t
 1618 eop, Int_t charge, Double_t *pEMCAL), respectively. These methods are called from AliPIDCombined
 1619 and AliPIDResponse internally, so usually the user does not use them.

1620 To calculate $n\sigma$ and the conditional probability a parametrization of E/p for the different par-
 1621 ticle species at different momenta is needed. This was retrieved from data in a clean PID sam-
 1622 ple with the help of $V0$ decays for electrons, pions and protons ($\gamma \rightarrow e^+e^-$, $K^0 \rightarrow \pi^+\pi^-$, and
 1623 $\Lambda \rightarrow p + \pi^- / \bar{p} + \pi^+$) for different periods. Electrons are parametrized with a Gaussian distribu-
 1624 tion (mean value and σ), all other particles are parametrized with a Gaussian for $0.5 < E/p < 1.5$
 1625 and the probability to have a value in this E/p interval (this is small, since the maximum of the
 1626 distribution lies around 0.1). Here we distinguish between protons, antiprotons (higher proba-
 1627 bility for higher E/p values due to annihilation) and other particles (pions are used for these). At
 1628 the moment this parametrization is not done for all periods so far, as default LHC11d is taken.
 1629 There might be especially some centrality dependence on the E/p parametrization (because of
 1630 the different multiplicities of track-cluster matches).

1631 In addition to that the purity of the electron identification can be enhanced by using shower
 1632 shape cuts in addition. At the moment this can be done by getting them together with $n\sigma$:
 1633 AliEMCALPIDResponse::NumberOfSigmasEMCAL(const AliVParticle *track,
 1634 AliPID::EParticleType type, Double_t &eop, Double_t showershape[4]) In future,
 1635 a full treatment inside the PID framework is planned (by combining with AliEMCALPID).

1636 Some more information can be found on following TWiki pages:

- 1637 – <https://twiki.cern.ch/twiki/bin/view/ALICE/AlicePIDTaskForce>
- 1638 – <https://twiki.cern.ch/twiki/bin/view/ALICE/PIDInAnalysis>
- 1639 – <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EMCALPIDResponse>

1640 Here follows an example how to include the EMCAL PID in an analysis task:

```

1 // in analysis header:
2
3 AliPIDCombined fPIDCombined;
4 AliPIDResponse fPIDResponse;
5
6
7 // in Constructor:
8
9 fPIDCombined(0),
10 fPIDResponse(0),
11
12
13 // in UserCreateOutputObjects
14
15 // Set up combined PID
16 AliAnalysisManager *man=AliAnalysisManager::GetAnalysisManager();
17 AliInputEventHandler* inputHandler = (AliInputEventHandler*) (man->
    GetInputEventHandler());
18 fPIDResponse = (AliPIDResponse*)inputHandler->GetPIDResponse();
19
1641 fPIDCombined = new AliPIDCombined;
20
21 fPIDCombined->SetSelectedSpecies(AliPID::kSPECIES);
22 fPIDCombined->SetDetectorMask(AliPIDResponse::kDetEMCAL);
23 fPIDCombined->SetEnablePriors(kFALSE);
24
25
26 // in UserExec:
27
28 Double_t pEMCAL[AliPID::kSPECIES];
29 Double_t pBAYES[AliPID::kSPECIES];
30 Double_t eop;
31 Double_t ss[4]; //shower shape parameters (number of cells , M02, M20, Dispersion)
32
33 // NSigma value for electrons
34 nSigma = fPIDResponse->NumberOfSigmas(kEMCAL,track,AliPID::kElectron);
35 // or with getting also the E/p and shower shape values
36 nSigma = fPIDResponse->NumberOfSigmasEMCAL(track,AliPID::kElectron,eop,ss);
37
38 // Bayes probability
39 fPIDCombined->ComputeProbabilities(track, fPIDResponse, pBAYES);

```

1642

1643 **10 Run by run QA, how to and code**

1644 **10.1 Online - Francesco, Michael**

1645 **10.1.1 Creation and checking of online QA histograms (AMORE)**

1646 The QA histograms for the EMCal are created and filled by the class:

1647 `$ALICE_ROOT/EMCAL/AliEMCALQADataMakerRec`

1648 It is run both during the offline reconstruction and by the online data quality monitoring frame-
1649 work. Its main methods are:

1650

1651 `InitRaws()`: All the QA histograms are created here. Their titles should be self-explaining.
1652 Each of them has 2 important flags:

1653 `expert` - if true, the histogram is shipped only to the AMORE expert agent (run in the EMCal
1654 station), otherwise it is also shipped to the AMORE shifter agent (at the moment we have 4
1655 histograms monitored by the DQM shifter).

1656 `image` - if true, the plots is saved to the loogbok (there should be 9 plots in the logbook).

1657 Each of those histograms is replicated 4 times by the amore framework and filled according to
1658 the event species (Calib, Cosmic, LowMultiplicity, HighMultiplicity).

1659

1660 `MakeRaws(AliRawReader* rawReader)`: Here we loop over the input raw data stream and we
1661 fill the histograms.

1662

1663 `MakeRawsSTU(AliRawReader* rawReader)`: STU raw data decoding, provided EMCal trig-
1664 ger experts.

1665

1666 `GetCalibRefFromOCDB()`: Get the reference histograms from the OCDB. The ratio between
1667 histograms from current run and the reference run is calculated in the `MakeRaws` method.

1668

1669 The QA histograms are then analyzed by the class:

1670 `$ALICE_ROOT/EMCAL/AliEMCALQAChecker`

1671 It checks the data contained in the 4 non-expert histograms. They are at the moment:

- 1672 1. Ratio distribution of towers' amplitude in the current run w.r.t. the reference run.
- 1673 2. Number of hits of L1 Gamma patch.
- 1674 3. Number of hits of L1 Jet patch.
- 1675 4. Number of links between TRU and STU.

1676 The main method is: `CheckRaws()`: For the first histogram, we check how many channels have
1677 the ratio in the region 0.8 – 1.2. If there are more than the threshold (default = 90%), everything

1678 is ok, otherwise a red box with a “call expert” message is displayed. For L1 Jet/Gamma patch,
1679 we check if the rate of a single patch is higher than the average rate of all other patches times a
1680 certain threshold value (default = 0.5):

$$Rate_{patch}/Rate_{Total} > Threshold/(1 + Threshold) \quad (3)$$

1681 If so red box with a “hot spot - call expert” message is displayed. Finally, if there is a number
1682 (default = 1) of missing STU-TRU links, another red box with a warning message is shown.
1683 If one wants to change the thresholds, this can be done by updating them in the database (type
1684 amoreConfigFileBrowser d emc at any machine at P2 and edit the file QAThresholds.configfile
1685 accordingly).

1686 **10.1.2 How to's for EMCAL AMORE experts**

1687 **How to run AMORE at point 2 (P2):** Some useful scripts for running AMORE at P2 are
1688 located in the ~/amore directory in the aldaqacr51 machine. StartAmore.sh is used to run
1689 the agent and the GUI (in the expert mode). It simply launches the configMonitor.sh and
1690 the checkLogs.sh (used to check the logs of the agents). configMonitor.sh setup the agent
1691 and the GUI using some kdialog commands and runs them. Basically the agent is run by the
1692 following line in the script:

```
1693 sshdqm $agentName nohup $scriptsPATH/startAmoreAgent.sh $onlineMode &
```

1694 The agent is actually run on the EMCAL DQM node via the sshdqm command, so this line
1695 basically is used to connect to our dqm node and run the agent there.

1696 The GUI is launched in our machine by the command:

```
1697 amore -c EMC -m EMCUIQA(TRU) -g configFile.txt
```

1698 The configFile.txt file contains just one line, i.e. runType i, where i runs from 1 to 4
1699 (1=LowMultiplicity, 2=HighMultiplicity, 3=Cosmic, 4=Calib) and it is used to select the event
1700 specie you want to display in the GUI (this file is automatically created each time you run
1701 configMonitor.sh).

1702 Some other useful scripts (which are launched by configMonitor.sh using the sshdqm) are
1703 located in our DQM node. One can login to it by using:

```
1704 ssdqm EMCQA
```

1705 They are located in the scripts directory.

1706 startAmoreAgent.sh basically runs the agent, it simply issues the command:

```
1707 amoreAgent -a EMCQA -s @$1: -u
```

1708 The only parameter here is \$1, which is the gdc to monitor, defined in configMonitor.sh.
1709 killAgent.sh kills any running agent. Be aware that there cannot be 2 running agents of the
1710 same kind at the same time (i.e. just one EMCQA and one EMCQAshifter at a time). It can
1711 happen that some time the agent does not start when you try to run it. Most of the time this is
1712 because the dqm shifter is already running the EMCQA agent instead of just the EMCQAshifter
1713 one (they can run it from the dqm station even if they shouldn't). To check if the dqm shifter is
1714 running it, simply do ps aux | grep EMCQA in the dmq node, and see if there is an EMCQA
1715 process belonging to the user daq. If so, kindly ask the dqm shifter to please kill it.

1716 The agent is always running using the same parameters, which are stored in a database. In order

1717 to change them, you should run the `amoreConfigFileBrowser` command in the `aldaqacr51`
1718 machine. A window will appear, there you can browse a lot of configuration files. Our files are
1719 `EMCAL_config.txt` (it contains just the libraries to be loaded by the agent, and the event species
1720 to monitor), `QAdescriptionsEMC.configfile` (it contains the descriptions of the histograms
1721 shipped to the dqm shifter), and `QAThresholds.configfile` (it contains the thresholds for the
1722 QA checker - see above). You can edit them by pressing the edit button.

1723 **How to run AMORE in the test machine:** The EMCAL AMORE test machine can be reached
1724 via `ssh emcal@pcaldbl601`. The standard setup there is identical to the setup at P2. In the
1725 home directory there are some symbolic links which need to be changed in order to change
1726 the setup and test new features. `alirootLink` usually links to `/opt/aliroot-<some-ver>`,
1727 which is the version currently at P2 (daq team updates the software in the `/opt` directory when
1728 needed). The same holds for `rootLink`, `amoreLink` and `amoreSiteLink` (you should not need
1729 to change `rootLink` and `amoreLink`). For testing some changes to the EMCAL QA classes,
1730 one has to compile an own version of `aliroot` with the new version of those classes, and then
1731 make a symbolic link to the path of this `aliroot` version. There is an `aliroot` trunk version
1732 which was used to be updated from time to time for tests in `fblanco/alisoft`. In the directory
1733 `myamoreStuff` there are 2 version of the AMORE modules: `current_deploy` and `trunk` (of
1734 course you should do `svn up` from time to time). Let's suppose you want to make some mod-
1735 ification to our AMORE GUI (the expert one) and test them. Remove the `amoreSiteLink` (it
1736 usually points to `/amoreSite`). Then do:

```
1737 ln -s myamoreStuff/amoreSite amoreSiteLink  
1738 (or any other directory you would like to use). Then:
```

```
1739 cd trunk[current_deploy]/amoreEMC
```

1740 At this point you can modify either the `src/ui/EMCUIQA` or `src/ui/EMCUIQATRU` class. This
1741 class contains just some manipulations of canvas/histograms to be shown in the expert panel and
1742 should be easy to understand. If there are some doubts, just ask. The first one also contains the
1743 hack we use in order to use our own reference file at P2 in used to calculate the ratio to refer-
1744 ence (next section will explain how to create a new reference file). `make install` will install
1745 the modified libraries into `amoreSiteLink`. You can use some of the scripts in the `fblanco`
1746 directory to run the agent and the GUI. In order to run the expert agent with the expert GUI you
1747 should do:

```
1748 amoreAgent -a EMCQA -s <some-raw-data>6 -g emcal_amore.cfg
```

1749 After doing `ssh` to the test machine in another terminal, you do

```
1750 amore -d EMC -m EMCUIQA(TRU) -g configGUI.txt
```

1751 If you want to test the shifter agent, you simply do:

```
1752 amoreAgent -a EMCQAshifter -s <some-raw-data> -g emcal_amore.cfg
```

1753 and type `amoreGui` in another terminal window.

1754 You can commit changes to the trunk (and not to the `current_deploy`) with the usual `svn commit`.

1755 Once you feel that your changes are ready to be deployed at P2, send an email with a request to

⁶If you want to create a raw data file, you should first download a chunk of raw data from alien. Then do:

```
deroot file.root file.raw.
```

Keep in mind that the test machine is shared with other detectors, so avoid to store a lot of raw data there and clean up the space from time to time.

1756 date-support.

1757 **How to create a new reference file:** In order to create a new reference file, you have to
1758 use the `doReco.sh` script. The only thing you should do there is the path to the raw data
1759 in `alien` and the number of chunks to analyze (check it in the `alien` path). Remember to do
1760 `alien-token-init` and `source /tmp/gclient_env_${UID}` before running the script. After
1761 the script is executed you will have some directories (`chunck10`, `chunk11...`). Each of them con-
1762 tains an `EMCAL.QA.<RunNumber>.root` file. You can do :

```
1763 hadd EMCAL.QA.0.root chunck10/EMCAL.QA.<RunNumber>.root  
1764 chunck11/EMCAL.QA.<RunNumber>.root
```

1765 to merge them (the output files should always be called `EMCAL.QA.0.root`). At this point the
1766 output file can be already copied to the `aldaqacr51` machine in the `emcal` station, in which we
1767 can change the `QARef` file whenever we want. In order to do that, you should copy it to your
1768 `lxplus` area, then from the `~/amore/QARef` directory in the `aldaqacr51` you can do:

```
1769 scp your-afs-account@aldaqgw01:/afs/cern.ch/<path-to-file>/<file> .
```

1770 Do:

```
1771 ln -s new-file QA.Ref.root
```

1772 and add a note to the notes file in the directory. Then you have to create an `OCDB` file. In order
1773 to do that, you must do

```
1774 aliroot Save2OCDB.C
```

1775 Standard parameters of the macro are “EMCAL” (detector name), 0 (run number) and “12”
1776 (year). You may need to change only the last one. The macro will create a directory named
1777 `QARef/EMCAL/QA/Calib/`, and the file `Run0_999999999_v0_s0.root` in it. This file has to
1778 be committed to the `QARef/EMCAL/QA/Calib/` directory of `aliroot`. You can check it with the
1779 `checkCDB.C` macro (it just displays a couple of histograms).

1780 *10.1.3 Some more informations*

1781 Further details about AMORE can be found here:

1782 <https://ph-dep-aid.web.cern.ch/ph-dep-aid/amore/>

1783 The Twiki page with the general EMCAL informations for the DQM shifter is:

1784 <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/EVEEMC> The Twiki page (DQM blackboard)

1785 with temporary informations for the DQM shifter is:

1786 <https://twiki.cern.ch/twiki/bin/viewauth/ALICE/DQMBlackboard>

1787 **10.2 Offline - Marie**

1788 The aim of the offline QA is to check EMCAL data quality but also to get information of the
1789 global run quality with respect to EMCAL. The offline QA is automaticclly processed for each
1790 official data reconstruction pass, it can be calibration passes, validation passes, production passes
1791 or specific early dedicated production such as the “muoncalo” production for calorimeters and
1792 muon detectors. The user can then analyse the outputs created during official reconstruction
1793 passes which are stored on `alien`. The output are in that case stored in `/alice/data/year/period/-`
1794 `pass/QAresults.root` files. But each analyser can also add the QA task in its ianalysis in order to
1795 perform the QA of his dataset.

1796 The following sections describes how to run the QA task then how to analyse the ouputs run by
1797 run and how to look at EMCAL stability via some trending observables for whole periods/year.

1798 **10.2.1 Creation of offline QA histograms**

1799 The offline QA histograms for the EMCal are created and filled by the class:

1800 \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/AliAnaCalorimeterQA

1801 This class takes the information on AliESD/AliAOD objects and fills histograms related to EM-
1802 CAL cluster or cells but also on correlation between EMCAL and the other detectors in the run.

1803 Its main methods are:

- 1804 – MakeAnalysisFillHistograms(): main loop where the histogram filling once the
1805 clusters and cells are loaded.
- 1806 – ClusterLoopHistograms(const TObjArray clusters, AliVCaloCells cells):
1807 fill of cluster related histograms.
- 1808 – CellHistograms(AliVCaloCells cells): fill cells related histograms.
- 1809 – Correlate(): this methods allows to correlate clusters observables to other detector
1810 observables: tracks, VO signal, PHOS clusters.
- 1811 – InvariantMassHistograms(const Int_t iclus, const TLorentzVector mom, const
1812 Int_t nModule, const TObjArray caloClusters, AliVCaloCells cells): fill In-
1813 variant mass histograms in cluster loop.

1814 The way to initialize the macro is the following:

```
1815 AddTaskCalorimeterQA(Bool_t kSimulation = kFALSE, const char *suffix = default,  
1816 TString outputFile=" ", Int_t year = 2012, Bool_t kPrintSettings = kFALSE)
```

1817 where

- 1818 – kSimulation: TRUE if you want to run all the MC histograms.
- 1819 – suffix: suffix added to *CaloQA_* to the name of the TDirectory and TList where the output
1820 histograms will be stored. This is important in case you want to add the task for different
1821 **AliVEvent** offline trigger type set by SelectCollisionCandidates(kTriggerMask).
- 1822 – outputFile: this will be let to the Analysis manager to define it;
- 1823 – year: by default to 2012 will set all the SuperModules histograms (12) even if data from
1824 previous years are analysed. (if sets to 2010 or 2011 the associated nb of SM (10 for 2011
1825 to 2013 ; 4 for 2010)
- 1826 – kPrintSettings: to have lots of printings.

1827 An example of configuration macro for this class can be found in
1828 \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/AliAnaCalorimeterQA/macros/
1829 QA/AddTaskCalorimeterQA.C
1830

1831 Lots of settings for this class can be configured via the macro such as the filling of MonteCarlo
1832 info, the range of the histograms, the different histogram class filling.

1833 The ingredient to be passed to the AddTaskCalorimeterQA.C macro are the ones of
1834 \$ALICE_ROOT/PWG/CaloTrackCorrBase/AliAnalysisTaskCaloTrackCorrelation.cxx.

1835 An example of how to call it can be found in \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/
1836 AliAnaCalorimeterQA/macros/ana.C
1837

1838 *10.2.2 Run by Run Quality Assurance*

1839 The Run by Run Quality assurance is made by analyzing different histograms. A macro
1840 \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/AliAnaCalorimeterQA/macros/QA/QAplots.C
1841 is used systematically on all the runs. This macro runs on the .root file generated by the
1842 \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/AliAnaCalorimeterQA/macros/
1843 QA/AddTaskCalorimeterQA.C and looks at several histograms generated by the AliAnaCalorimeterQA
1844 task. To run this macro it is supposed that the output (.root files) of the QA task are in a directory
1845 named **period/pass** where period is the period (e.g. LHC12h) and pass the reconstruction pass
1846 you are looking at. The .root files are supposed to be renamed **runnumber.root** in case of pro-
1847 duction passes, validation passes or dedicated calo production passes and **runnumber_outer.root**
1848 or **runnumber_barrel.root** in case of cpass (calibration pass). In the calibration pass case, the
1849 Minimum Bias triggers are reconstructed within the _barrel.root extension and the specific trig-
1850 gers (muon, EMCAL, ...) are reconstructed with the _outer.root extension. The QA task is called
1851 twice at the end of reconstruction to separate both kind of triggers.

1852 To run the macro you must have in the period/pass repository a text file called "runlist.txt con-
1853 taining the list of run you want to plot, the runlist.txt being of the type:

```
1854 1 run1  
1855 2 run2  
1856 .....
```

1857 The output plots are saved automatically in repository period/pass/runnumber so that you should
1858 create those repository prior to running the macro.
1859

1860 The arguments to pass to the macro: QAplots(TString fCalorimeter = "EMCAL", TString
1861 period = "LHC11h", TString pass = "pass", TString fTrigger= "default", TString
1862 system = "PbPb") are the following:

- 1863 – TString fCalorimeter : name of the calorimeter: "EMCAL".
- 1864 – TString period: path(name) of the repository for the period.

- 1865 – TString pass: name of the repository where you saved the .root files (in the example
1866 period/pass).
- 1867 – TString fTrigger type: "default" for the minbias; "trigEMC" or "EMC7" for EMCAL
1868 triggers. This corresponds to the TString suffix of the AddTaskCalorimeterQA() when
1869 initializing the task.
- 1870 – TString system: "PbPb" or "pp": a flag for setting/adjusting scales for some drawings to
1871 the multiplicities/energies reached in the system.

1872 For typical automatic QA of official reconstruction passes the histograms we check automati-
1873 cally with running of the QApots.C macro are the following: but the user can add as much
1874 histograms verifications as he wants.

- 1875 – Occupancy map: η ϕ distributions of cells with signal (in row/col units).
- 1876 – Summed energy map: η ϕ distribution of mean energy/event (in row/col units).
- 1877 – Time vs E of custers: Time vs Energy of clusters.
- 1878 – EMCAL Cluster versus track correlation (multiplicities/energies).
- 1879 – EMCAL Clusters vs VO signal correlation.
- 1880 – EMCAL vs PHOS clusters correlations.

1881 On top of those default settings and checks, the user can add as much histograms checks as he
1882 wants.

1883 ***10.2.3 Period Quality Assurance and stability of EMCAL***

1884 The stability of EMCAL is checked also systematically at the end of each reconstruction passes
1885 (calibration pass, validation pass, production pass). This step is done via a set of macros called
1886 “trending” macros which can be found in \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/
1887 AliAnaCalorimeterQA/macros/QA/. The goal of these two macros (trendingClusters.C and
1888 trendingPi0.C) is to look at averages observables concerning clusters and π^0 respectively, as a
1889 function of runnumber. Those macros uses the output of .root file generated by the
1890 \$ALICE_ROOT/PWGGA/CaloTrackCorrelations/AliAnaCalorimeterQA/macros/
1891 QA/AddTaskCalorimeterQA.C. It requires the same repository, naming conventions as in the
1892 previous section: period/pass/runnumber.root files and a runlist.txt file containing the list of runs
1893 to check. It can be run depending on the “fTrigger” option EMCAL triggered events and on the
1894 MinBias events.

1895 **Cluster Averages: (trendingCluster.C):**

1896 This macro computes some averages over histograms and store them in a TGraph as a fuction of
1897 runnumber. The folling histograms are used:

- 1898 – EMCAL_hE: cluster energy distribution in EMCAL.
- 1899 – EMCAL_hNClusters_Mod_xx: 2D histogram with number of cluster per supermodule.
- 1900 – EMCAL_hNCellsPerCluster_Mod_xx: 2D histogram with number of cells per supermodule.
- 1901
- 1902 – EMCAL_hE_Mod_xx: 2D histogram with Cluster Energy per supermodule.

1903 and the following averages are computed and displayed as a function of runnumber:

- 1904 – Mean number of clusters per event is computed over EMCAL and also per supermodule. An example is displayed on figure 19 for LHC12i period.
- 1905
- 1906 – Average energy of clusters (with given threshold energies) per event is computed over EMCAL and also per supermodule.
- 1907
- 1908 – Number of cells of clusters (with given threshold energies)per event is computed over EMCAL and also per supermodule.
- 1909

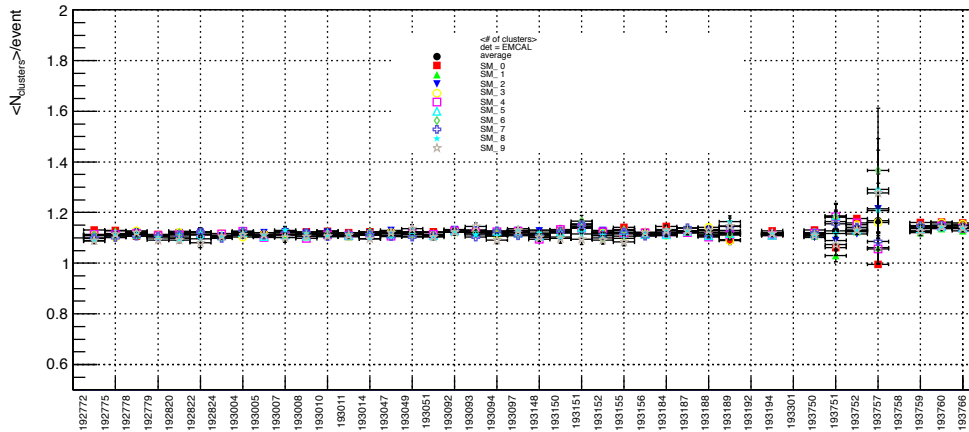


Fig. 19: Example of trending plot for period LHC12i: mean number of cluster per event as a function of runnumber for MinBias events.

1910 π^0 Averages: (trendingPi0.C):

1911 This macro computes some fits over invariant mass histograms and store the fit parameters and
 1912 π^0 computed values in a TGraph as a function of runnumber. The following histograms are used:

- 1913 – EMCAL_hIM: Invariant mass histogram filled over the whole cluster pairs.

1914 – EMCAL_hIM_Mod_xx: Invariant mass histogram filled over the cluster pairs in the same
1915 supermodule.

1916 The fit is done by a gaussian plus a 2nd order polinom for background. This is a very “raw” fitting
1917 wich may not be appropriate especially for the PbPb cases but allows to check the stabilities of
1918 averages (mass, width, number of π^0).

1919 – Mean number of π^0 per event is computed over EMCAL and also per supermodule.

1920 – Mass of the π^0 per event is computed over EMCAL and also per supermodule.

1921 – Withd of the π^0 per event is computed over EMCAL and also per supermodule.

1922 The use of these set of QA macros allows to identify problematic runs (subperiod with missing
1923 part of detectors) or more general features. Also it can be used to check global observables for
1924 different dataset used for analysis purpose.

1925 **10.3 Event display**

1926 **10.4 Logbook tips**

1927 **References**

- 1928 [1] EMCal for beginners, [https://twiki.cern.ch/twiki/pub/ALICE/EMCalOffline/](https://twiki.cern.ch/twiki/pub/ALICE/EMCalOffline/EMCalBeginners.pdf)
1929 EMCalBeginners.pdf
- 1930 [2] AliRoot: ALICE offline project, <http://aliceinfo.cern.ch/Offline/>
- 1931 [3] AliRoot Documentation, [http://aliceinfo.cern.ch/Offline/AliRoot/Manual.](http://aliceinfo.cern.ch/Offline/AliRoot/Manual.html)
1932 html
- 1933 [4] AliRoot Installation, [http://aliceinfo.cern.ch/Offline/AliRoot/Installation.](http://aliceinfo.cern.ch/Offline/AliRoot/Installation.html)
1934 html
- 1935 [5] AliRoot Installation from Dario Berzano, [http://newton.ph.unito.it/~berzano/w/](http://newton.ph.unito.it/~berzano/w/doku.php?id=alice:compile-any&redirect=1)
1936 doku.php?id=alice:compile-any&redirect=1
- 1937 [6] AliEn web page, <http://alien.cern.ch/twiki/bin/view/AliEn/Home>
- 1938 [7] AliRoot in SVN <http://alisoft.cern.ch/viewvc/?root=AliRoot>
- 1939 [8] EMCAL documentation, [http://aliceinfo.cern.ch/Offline/Detectors/](http://aliceinfo.cern.ch/Offline/Detectors/EMCALOffline.html)
1940 EMCALOffline.html
- 1941 [9] EMCal Offline twiki, <https://twiki.cern.ch/twiki/pub/ALICE/EMCalOffline>
- 1942 [10] EMCAL L0 [http://www.sciencedirect.com/science/article/pii/](http://www.sciencedirect.com/science/article/pii/S0168900212007681#)
1943 S0168900212007681#
- 1944 [11] EMCAL HLT <http://arxiv.org/abs/1209.3647>
- 1945 [12] ALICE Collaboration, ALICE EMCal Technical Design Report, CERN/LHCC 2008-014
- 1946 [13] Casalderrey-Solana and C. A. Salgado, Introductory lectures on jet quenching in heavy ion
1947 collisions, Acta Phys. Polon. B 38 (2007) 3731
- 1948 [14] ALICE Collaboration, ALICE: Physics Performance Report, Volume I .J. Phys. G, 30
1949 (2004) 1517-1763
- 1950 [15] ALICE Collaboration, ALICE: Physics Performance Report, Volume II. J. Phys. G, 32
1951 (2006) 1295-2040
- 1952 [16] P. H. Hille, Fast Signal Extraction for the ALICE Electromagnetic Calorimeter, in prepa-
1953 ration.