Table of Contents

DD4hep-based CLIC Detector Calorimeter Calibration

Introduction

This TWiki aims to document the calibration procedure for the new CLIC detector calorimeters, specifically the HCal and ECal. The philosophy behind the calibration is the one developed by S. Green (Cambridge) for the optimization studies based on the ILD detector concept. The Cambridge procedure uses simulated single particle data at various energies to set the ECal, HCal and Muon system digitization constants as well as the Pandora calibration constants. The procedure, as performed currently at CLIC, needs the following simulated single particle data (typically 1000-10000 events per point, distributed uniformly in the detector) in `lcio` format:

- 10 GeV Photons (`gamma`)
- 10 GeV muons (`mu-`)
- 50 GeV Neutral Kaons (`K0L`). More kaon energies can be used to determine corrections for non-linearity, but for the time being this is not done.

For each point, full reconstruction with `Marlin` (including digitization and PandoraPFA reconstruction) is run iteratively and in several steps so as to converge to a set of calibration constants within a required precision. In general, the steps have to be performed in a particular order since the some constants depend on the proper calibration and setting of constants determined in an earlier step. Typically the total PFO energy is collected per event and a histogram is filled from which correction factors are extracted. The necessary information is collected in ROOT files using the `PfoAnalysis` Marlin processor, run after the PFO creation, with `CollectCalibrationDetails` turned on in the processor parameters. More crucially, Steve has written several applications that can be run over these ROOT files and perform the histogram fills, fits and calibration constant extraction. The Marlin processor and the code/applications to run the extraction of every constant are contained in the `PandoraAnalysis` package (link on github) which is part of every ILCSOFT release. The package also includes some documentation on the the calibration procedure and the provided executables (a pdf copy of the documentation is statically attached).

The procedure involves several iterations and many steps therefore it is impractical to run all the steps manually. In addition, it would take too long to run the reconstruction of several thousands of events many times in series. Steve has developed a calibration script that performs the all the steps in the proper order. In addition, the script employs a computing cluster to parallelize the event reconstruction steps which significantly reduces the overall time it takes to perform a calibration. For example, it takes under 10 minutes to run the full calibration on a high performance cluster which uses 100 nodes each reconstructing 100 events. However, the calibration script and other associated files are tied to the Cambridge computing cluster configuration and could not readily be used in a different environment. Furthermore, the Cambridge scripts were strongly tied to the ILD detector geometry and, even more crucially, relied on the old reconstruction software based on `GEAR` (and used events simulated with `Mokka`). The procedure was initially modified to run at CERN with a version of the ILD detector tweaked to incorporate the CLIC detector aspects (namely deeper HCal, longer detector, different field), always using the old reconstruction framework. The CLIC detector model diverged more and more from the ILD detector model, not least of which was the use of an all-silicon tracker and associated track reconstruction software. Coupled with the introduction of the new software framework based on DD4hep and the development of the new detector simulation model, it became clear that we needed a new calibration procedure.

To maximize backwards compatibility and to minimize work, the constant extraction applications are used exactly and directly as provided in `PandoraAnalysis`, but the calibration scripts were heavily modified to accommodate the new DD4hep-based software: The geometry is initial using `DD4hep`, the `DDCaloDigi` digitizers are used and most importantly, the `DDMarlinPandora` package is used to interface with the

PandoraPFA reconstruction. Further modifications include the addition of comments and printout statements in the calibration procedure, checks at the various steps and an attempt to make the whole procedure more robust and possible to be run by anyone. With the current status of the code, anybody can check out the calibration package. Assuming a `condor` cluster is setup in the environment, the calibration can be run with minimal modification.

Description of the DD4hep-based calibration package

The calibration package is hosted under the clicdp/Users svn repository and can be checked out using:

```
svn co svn+ssh://svn.cern.ch/reps/clicdet/trunk/Users/nikiforo/DD4hepDetCalibration
```

No compilation is needed since it is merely a collection of bash and python scripts as well as some configuration templates. As mentioned above, it runs full reconstruction with `Marlin` and uses `PandoraAnalysis`, therefore a recent installation of ILCSOFT is needed. Generally one should use the same ILCSOFT installation/version as the one used to produce the simulated single particle files used as an input to the procedure.

The package contains all the scripts, configuration files and templates needed to run the calibration, assuming that the input simulation lcio files are accessible and organized appropriately. Here is a list of the most important files in the package:

- **`Calibrate.sh`** : This is THE main script file that configures and launches the calibration. Within this file and between the lines labelled **`##### CONFIGURATION #####`** and **`##### END OF MAJOR CONFIGURATION #####`** lie the major configuration variables to be configured, in the form of local shell environment variables. The main parameters to be configured are (please consult with the file for more details and more options):
  - `ilcsoftPath` : The Path to the `ILCSOFT` installation to be used.
  - `geometryModelName` : The detector model geometry name. Together with `ilcsoftPath` (which sets the version of `lcgeo`), it determines the location of the DD4hep compact xml file for the detector geometry. Override the `dd4hep_compact_xml` variable to use an xml file in a non-standard location.
  - `slcioPath`: by default it is set to `/scratch1/CalibDataForDD4hep/${geometryModelName}/` and assumes a certain structure for the directories. For example:

```
/scratch1/CalibDataForDD4hep/CLIC_o2_v04/gamma/10/
/scratch1/CalibDataForDD4hep/CLIC_o2_v04/mu-/10/
/scratch1/CalibDataForDD4hep/CLIC_o2_v04/K0L/50/
```

The script checks whether that the files exist and are accessible, preferable with sequential indices set contained in their filenames.

-
  - `slcioFormat` : By default set to ".*_([0-9]+).slcio" . Used to filter the lcio files contained in the directories (not needed in this example with a well defined directory structure) but also to determine the index of the filename (as defined by the pattern `([0-9)+)` . No need to modify this if you adhere to the file naming convention (numerical file index before the `.slcio` extension) and you use the directory structure defined above.
  - `outputPath` : Absolute path to a directory to store the output results: constant extraction log file, calibration constants summary, plots and Marlin steering xml file template with the updated calibration constants set.
  - Maximum Time Window values for ECal/HCal barrel and encap and hadronic energy truncation limit (MHHHE). Typically change with the calorimeter absorber material and readout settings.

- **`clean.sh`** : Auxiliary script to clean the directory of all results, intermediate files and root files. Useful when you want to run a new calibration.
- **`Xml_Generation/CLIC_PfoAnalysis_AAAA_SN_BBBB.xml`** : The template Marlin XML steering file to run the reconstruction. Defines and configures the processors run during the reconstruction of the calibration events. You can replace the template with your own, or modify it to include your own additional processors that you need to have to run the reconstruction for your detector. For example, you may want to change the track finding and fitting algorithms (currently this is using the track cheater) or move to a different digitizer. The calibration script copies this file and modifies it for each input file (using `sed`) by replacing the following dummy variables (identifiable by the suffix `_XXXX`):
  - `slcio_XXXX` : Replaced by the input lcio file
  - `Gear_XXXX` : Replaced by the GearOutput.xml file for the detector. Kept for backwards compatibility and the file is generated automatically using `convertToGear` .
  - `COMPACTXML_XXXX` : Replaced by the DD4hep compact xml file describing the geometry of the detector
  - `CALIBR_ECAL_XXXX`, `CALIBR_HCAL_BARREL_XXXX`, `CALIBR_HCAL_ENDCAP_XXXX`, `CALIBR_HCAL_OTHER_XXXX` : Replaced by the latest digitization constants for the current step.
  - `ECALBarrelTimeWindowMax_XXXX`, `HCALBarrelTimeWindowMax_XXXX`, `ECALEndcapTimeWindowMax_XXXX`, `HCALEndcapTimeWindowMax_XXXX`, `MHHHE_XXXX` : replaced by the chosen configuration values
  - `PSF_XXXX` : Replaced by the name of the chosen Pandora Settings xml configuration file (by default is `PandoraSettings.xml`, which is copied over from `MarlinPandora` along with the photon likelihood data file).
  - `ECalGeVToMIP_XXXX`, `HCalGeVToMIP_XXXX`, `EMT_XXXX`, `HMT_XXXX`, `ECALTOEM_XXXX`, `HCALTOEM_XXXX`, `ECALTOHAD_XXXX`, `ECALTOHAD_XXXX`, `HCALTOHAD_XXXX`, `MuonGeVToMIP_XXXX` : Replaced by the latest Pandora calibration constants for the current step.
  - `pfoAnalysis_XXXX.root` Replaced by the output root file name associated with the current input slcio file.

If you replace the steering file template, make sure you introduce at least the dummy variables listed above in the appropriate places. Please consult with the script `Xml_Generate.py` to see how and which variables are replaced.

- **`Xml_Generarion/Xml_Generate.py`** : Python script responsible to change the marlin xml steering file template and create one for every job with the appropriate settings (slcio file, constants, etc). The tokens `AAAA` and `BBBB` in the template filename are replaced by the job name and the input lcio file index, respectively, for every job xml file.
- **`Root_File_Generation/condorSupervisor_Calibration.sh`** : File responsible for generating temporary job scripts for each event, copying files to worker nodes, submitting jobs and waiting for them to finish. Only works with `condor`. This particular version was developed for use at CERN over `afs`, but should be usable on any `condor` system. If AFS or other network file system is not present, it can still run, provided that the ILCSOFT installation files are accessible on every worker node. The input, output and log files can be transferred using the condor file transfer mechanism.
- **`Root_File_Generation/batchSupervisor_Calibration.sh`** : Not working yet, but should be fixed to run over LSF batch or any other job scheduling system.
- **`Root_File_Generation/DummyCondorSupervisor.sh`** : Allows to run on a local machine (slow)
- There more auxiliary python and bash scripts in the various directories in the package that deal with extracting results from text files, modifying templates, etc.

The DD4hep-based calibration procedure as performed at CERN

As an example, we can login using the service account `clicdpsw` to a virtual machine (`clidpcalwn00`) set-up to perform the calibration. Before the calibration procedure is attempted, single particle events

(photons, muos and k-longs) need to be simulated as described above. The procedure described here can be used, though it is preferable that the events be generated on a batch farm or the grid for efficiency. To be able to parallelize the reconstruction at the calibration step, it is advised to split the events (1000, 10000 events or even more per point) over many output files (typically 100) according to the capacity of your computing cluster. A current limitation of the calibration script is that it can only launch one job per input file (no grouping files in one job or splitting files over several jobs).

We have collected the files (100x100 events for each point) under:

```
[clicdpsw@clicdpcalwn00 ~]$ ls /scratch1/CalibDataForDD4hep/CLIC_o2_v04/
gamma  K0L  mu-
```

`/scratch1` a mountpoint to an CERN cloud infrastructure volume (0.5 TB) attached to this machine. Notice that this directory (or the entire `/scratch1`) is only accessible locally to the machine (i.e. not on `AFS`). The files however will be automatically transferred to the execution nodes.

We have also checked out a version of the `DD4hepDetCalibration` calibration package under `/scratch1` . Note that also this directory is not on `AFS`. At least for this temporary calibration pool setup, due to a limitation of `condor` the directory from which you launch the calibration and submit the jobs (i.e. `/scratch1/DD4hepDetCalibration`) **can not** be on `AFS` since the condor daemons that transfer back the output files (and the `out`, `err`, and `log` files) do not authenticate on `AFS` and have not access to it. This limitation does not affect, however, the ILCSOFT installation or the input files which are accessed by the Marlin job itself, which runs under the authenticated user. The input files lcio described above could in principle actually be hosted somewhere on `AFS`, provided that the authenticated user (in this case `clicdpsw`) has access.

In principle, you can already launch the calibration for the `CLIC_o2_v04` model, as implemented in `lcgeo` for the ILCSOFT installation under `/afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19/`. You should verify that your simulated files are indeed simulated with this detector model and ILCSOFT/lcgeo version by looking at the lcio header information. For example, setup ILCSOFT and look at a photon file with `anajob`:

```
[clicdpsw@clicdpcalwn00 DD4hepDetCalibration ]$ source /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-20
[clicdpsw@clicdpcalwn00 DD4hepDetCalibration ]$ anajob /scratch1/CalibDataForDD4hep/CLIC_o2_v04/g
anajob:  will open and read from files:

     /scratch1/CalibDataForDD4hep/CLIC_o2_v04/gamma/10/DDSim_CLIC_o2_v04_gamma_10GeV0_0_250116102

  Run : 0 - CLIC_o2_v04:
 parameter CommandLine [string]: /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19/lcgeo/HEAD/bi
...
 parameter compactFile [string]: /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19/lcgeo/HEAD/CL
...
```

We can verify the ILCSOFT installation, `DDSim` version and dimulation settings, `lcgeo` version and detector model from the relevant lines.

Before starting the calibration, it is usually a good idea to clean the directory of previous output, log files, root files, xml files and other results. **Make sure you kept the information you need!**. To clean, just do:

```
[clicdpsw@clicdpcalwn00 DD4hepDetCalibration]$ ./clean.sh
```

The scripts are already configured for the appropriate file names, ILCSOFT version and detector geometry. Try to run the calibration:

```
[clicdpsw@clicdpcalwn00 DD4hepDetCalibration]$ ./Calibrate.sh
Will perform calibration with the following settings:

ILCSOFT: /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19
```

```
AnalysePerformance: /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19/PandoraAnalysis/HEAD/bin/A
dd4hep_compact_xml: /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-01-19/lcgeo/HEAD/CLIC/compact/CL
LCIO File path: /scratch1/CalibDataForDD4hep/CLIC_o2_v04/
     Photon path: /scratch1/CalibDataForDD4hep/CLIC_o2_v04//gamma/10/ [ 100 ( 0 ... 99 ) files fo
     Muon path:   /scratch1/CalibDataForDD4hep/CLIC_o2_v04//gamma/10/ [ 100 ( 0 ... 99 ) files fo
     Kaon path:   /scratch1/CalibDataForDD4hep/CLIC_o2_v04//K0L/50/  [ 100 ( 0 ... 99 ) files fou
HCALEndcapTimeWindowMax: 10 HCALBarrelTimeWindowMax: 10 ECALEndcapTimeWindowMax: 20 ECALBarrelTim
Run on batch system: Condor

Strike any key to continue...
```

If everything is properly set and accessible, you should see the proper versions of the relevant software, as well as the file counts for the input lcio files (with a printout of the first and last index of the files for crosscheck). Press any key to launch the first set of 100 jobs to reconstruct the photon events. After some initial work (including loading the geometry and creating a Gear file), the jobs will be submitted and you should see several lines of printout ending with:

```
/tmp/clicdpsw/jobs.Marlin_Runfile_10_GeV_Energy_gamma.txt.tmp is empty. All jobs submitted.
Checking whether the jobs have finished...
Not finished yet, come back later.
```

The script will now wait for the jobs to finish so it can collect the output root files and calculate the ECal digitization constants. After a few minutes, the jobs are done, the files are copied back to the submission directory and the script continues with the calculation of the constants:

```
Jobs finished.
ECAL Digitisation: Update CalibrECAL
CaloHitEnergyECal (7324 entries), rawrms: 0.647749, rmsx: 0.545863 (9.83505-11.9056), low_fit and
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
 FCN=0.26993 FROM HESSE     STATUS=OK             16 CALLS         339 TOTAL
                     EDM=7.71968e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           2.34650e+03   3.99794e+01   3.20368e-03   8.73356e-06
   2  p1           1.06986e+01   1.14501e-02   2.55075e-06  -2.52034e-02
   3  p2           2.42183e+00   9.87843e-02   4.74486e-07   9.73710e-03
Info in <TCanvas::Print>: png file /scratch1/DD4hepDetCalibration/output/Calorimeter_Hit_Energies
Info in <TCanvas::SaveSource>: C++ Macro file: /scratch1/DD4hepDetCalibration/output/Calorimeter_
```

```
Digitisation of the ECal.                          :

For Photon energy                                  : 10 : /GeV
Gaussian fit to ECal calorimeter hit energy has    :
the following parameters, fit uses 90% data with   :
min RMS:                                           :
Amplitude                                          : 2346.5 :
ECal Digi Mean                                     : 10.6986 :
Standard Deviation                                 : 0.642581 :
The total number of events considered was          : 7324 :
```

The procedure iterates some more times to converge on digitization constants that give a reconstructed energy as close as possible to 10 GeV. The procedure will continue similarly with the muon and kaon files to obtain more digitization constants. Then it will loop through the points performing more iterations to obtain the Pandora calibration constants. You can monitor the progress of each step by watching the output for the fitting of the parameters and the calculation of the updated constants as in the example for the first step above. In addition, the histograms and fits for each step are saved in `png` and macro formats.

You can open a new session to the machine from which to monitor the output plots. You can also use `condor_q` to see the list of jobs that are currently executing:

```
$ condor_q
-- Schedd: clicdpcalwn00.cern.ch : <128.142.200.224:57978>
 ID OWNER SUBMITTED RUN_TIME ST PRI SIZE CMD
1801.0 clicdpsw 1/28 16:23 0+00:00:10 R 20 0.0 krenew -t ./Marlin
1802.0 clicdpsw 1/28 16:23 0+00:00:01 I 20 0.0 krenew -t ./Marlin
1803.0 clicdpsw 1/28 16:23 0+00:00:10 R 20 0.0 krenew -t ./Marlin
...
1900.0 clicdpsw 1/28 16:23 0+00:00:02 R 20 0.0 krenew -t ./Marlin

100 jobs; 0 completed, 0 removed, 1 idle, 99 running, 0 held, 0 suspended
```

Depending on the availability of the nodes, all your jobs should quickly go into the R (i.e. running) state. If they are marked always as I (Idle) it would suggest that either all nodes are busy or unable to accept the jobs due to misconfiguration of the cluster. In case the jobs stay in the Idle, Held, or other erroneous state, you can remove (i.e. kill) them by using the command `condor_rm` . You will have to start the calibration from the beginning, however.

Further, if you are observing that your jobs are being killed or not producing the output root files (you see messages from the calibration script saying that the root files cannot be found), you can consult the condor log files under `Root_File_Generation/CondorLogs`. Inside you will see several files, one for each job executed. Usually you should check the latest ones (use `ls -lart`). The `.out` files show the standard output from the Marlin job, the `.err` contain the standard error printouts and the `.log` files collect information about the job (submission and execution nodes, resource usage, etc). There are some basic checks to ensure the jobs have finished successfully and the root files were generated and moved to the proper directories. If the move operation fails, the Calibration script should exit and the user should consult with the log files to find out what went wrong.

The last step will be the reconstruction of `K0L` and the collection of the final numbers. You should see this output:

```
Jobs finished.
Info in <TCanvas::Print>: png file /scratch1/DD4hepDetCalibration/output/PandoraPFA_Calibration_H
Info in <TCanvas::SaveSource>: C++ Macro file: /scratch1/DD4hepDetCalibration/output/PandoraPFA_C
```

```
For kaon energy                                 : 50 :
m_eCalToHadInterceptMinChi2                     : 49.5012 :
m_hCalToHadInterceptMinChi2                     : 49.5012 :
m_minChi2                                       : 4199.9 :
The total number of events considered was       : 4163 :

['Xml_Generate_Output.py', '1.00528117089', '1.00528117089', '0.958466960008', '1.05876999033', '
```

The output directory will contain text file that logs each step (`Calibration.txt`) a text file that collects all the computed constants in the form of shell variables (`calib.CLIC_o2_v04.txt`), a Marlin xml steering file template with the new constants (`CLIC_PfoAnalysis_AAAA_SN_BBBB.xml`) as well as a set of plots for each step.

Open issues and things to check

Feel free to investigate and update/remove if needed.

Running the Calibration script within a GNU `screen` session

Often it is needed to run `./Calibrate.sh` within a `screen` session if you are running remotely to avoid the ssh session dropping and having to start from scratch. However, you should make sure the screen environment is set up properly. For example, we identified the following two issues:

- **Your regular bash environment is not set up**: Solution: add the line `shell -$SHELL` to your `$HOME/.screenrc` file
- **The script crashes because of write permissions**: First, make sure your AFS credentials are up to date (if this applies to your case) by issuing `kinit` and then `aklog`. However, such an error will most likely appear because `$TMPDIR` is not set in your screen session. Add this line to your `$HOME/.screenrc` file: `export TMPDIR=/tmp/username` or whatever is appropriate for your case.

PFOAnalysis and especially need to be ported to DD4hep!

This is an important issue. Currently, the processor accesses information from gear rather than DD4hep. The information is used to calculate calibration quantities and fill histograms that are used in the calibration procedure. See `PandoraAnalysis/CalibrationHelper.cc`. **Update!** The `PandoraAnalysis` package has been modified privately and merger to production is pending.

Handling of "Other" calorimeters (i.e. HCal Ring, ECal Plug,)

We should check whether the "auxiliary" calorimeters (calorimeters other than ECal/HCal Barrel/Encap) are properly handled and calibrated. For example the ECal Plug and HCal Ring are handled as "Other". Especially the ECal Plug is handled together with the lumical which would mean that the calibration is averaging out completely different detector responses. The ECal Plug and HCal Ring (at least in the CLICdet case) have a makeup that is almost identical to the main calorimeters, therefore they should have similar calibration constants. As a first approach, DDCaloDigi should be checked whether we can set the calibration constants of these not so small auxiliary calorimeters to be the same as the ones from the main ones (**this should be already the case:** look at `MyDDCaloDigi` configuration). Then, the calibration procedure should be modified to handle the Lumical separately.

Calibration of the

The BeamCal is not included in the calibration. It shouldn't affect the calibration constants too much anyway and the detector is not modified often. It could be therefore calibrated in standalone, independently. We still have to check, however, how Pandora uses the information from BeamCal hits.

Check handling of changing ECal layer thickness

The ECal typically has two families of layers: the back layers have typically twice the absorber thickness which would require roughly digitization constants differing by a factor of two for the two layer families. This is accommodated in the `DDCaloDigi` processor configuration by accepting a vector of floats for `CalibrECAL`:

```
<parameter name="CalibrECAL" type="FloatVec">40.3366 80.6732</parameter>
```

and specifying the makeup of layers in the configuration:

```
<parameter name="ECALLayers" type="IntVec">17 100  </parameter>
```

The two lines above instruct the digitizer to use the first `CalibrECAL` constant (40.3366) for the first 17 layers and the second (80.6732) for the rest (the next switch of layer "family" would be at the unrealistically large 100-th layer). The second number is a result of a manual multiplication by a factor of 2 of the first number. Notice further down in your Marlin steering file that for the HCal digitization configuration there is only one family of layers (the switching index is set to 100). Things that can be done:

- Modify `DDCaloDigi` to get the layer info from the geometry and perhaps apply the factor of two automatically (however this removes the flexibility of specifying arbitrary constants for each layer family)

- Make sure that the calibration procedure takes into account the change of layer thickness. **My understanding is that currently this is NOT done at the moment which results in a double MIP peak in the ECal "direction corrected" digitization distributions**. However, only the first MIP peak is used (with a resulting loss of the statistics from the back layers and potentially a minor shift of the MIP peak). The constants are calculated according to the first peak.

NOTE: If you are calibrating a detector in which the ECal layer makeup changes, you need to modify your Marlin steering file template (e.g. `Xml_Generation/CLIC_PfoAnalysis_AAAA_SN_BBBB.xml`) in the line mentioned above. For example, change `17 100` to `20 100` for an ECal with changing layer thickness after the 20th layer. If you only have one type of layer, say 40 layers, you should change the line to `100`. Especially in the last case, you should perhaps modify the calibration scripts (which assume the back family of layers have twice the absorber thickness) to either stop adding the extra calibration constant, or at least fill it with the same value as the first layer family.

Muon Calibration

- Muon digitization calibration is not performed. **We need to get meaningful values for the RPC response!** This is different than the ILD case, where they have scintillators (so the response is similar to the calorimeters).
    - *Probably it doesn't matter: at least for SiD/CLIC_SID what appears to have been used is digital muon hits with a standalone muon clustering algorithm *
- Muon mip calibration is not working: histograms are not filled from the calibration helper in the PFOAnalysis processor
    - We tracked down at least a few issues (they come time and again so I leave them here for future reference):
        - Wrong Muon collection names: Was set to `MuonBarrelCollection MuonEndcapCollection` whereas it is now set to `YokeBarrelCollection YokeEndcapCollection` for the CLIC model.
        - Very high threshold to digitize Muon SimCalorimeterHits. The threshold was set to the **default (0.25)** because of the miss-spelling of the parameter in the steering file. More specifically, the **wrong case** was used for the parameter in the `DDSimpleMuonDigi` configuration: `MUONThreshold` was being used instead of the correct `MuonThreshold` .
        - The response for an RPC (used in the Yoke) is very different compared to the response expected for the scintillators in the rest of the calorimeters. Therefore, the deposited energy per hit in the muon system is out of the range of the histograms used to extract the response.
- Make sure various muon systems (plugs etc) are used.

Problems with new photon reconstruction algorithm

With the changes in the new photon reconstruction code, I see the following error during the Marlin execution to produce the root files:

```
[ VERBOSE "MyDDMarlinPandora"] PhotonReconstructionAlgorithm::CreateRegionsOfInterests no photon
[ VERBOSE "MyDDMarlinPandora"] this->CreateClustersOfInterest(clusterVector) return STATUS_CODE_I
[ VERBOSE "MyDDMarlinPandora"]     in function: Run
[ VERBOSE "MyDDMarlinPandora"]     in file:     /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-02-1
[ VERBOSE "MyDDMarlinPandora"] iter->second->Run() throw STATUS_CODE_INVALID_PARAMETER
[ VERBOSE "MyDDMarlinPandora"]     in function: RunAlgorithm
[ VERBOSE "MyDDMarlinPandora"]     in file:     /afs/cern.ch/eng/clic/work/ilcsoft/HEAD-2016-02-1
[ VERBOSE "MyDDMarlinPandora"] Failure in algorithm 0x7df5420, PhotonReconstruction, STATUS_CODE_
[ VERBOSE "MyDDMarlinPandora"] HighEnergyPhotonRecoveryAlgorithm: Failed to obtain cluster list P
[ VERBOSE "MyDDMarlinPandora"] SoftClusterMergingAlgorithm: Failed to obtain cluster list PhotonC
[ VERBOSE "MyDDMarlinPandora"] IsolatedHitMergingAlgorithm: Failed to obtain cluster list PhotonC
```

To be investigated more, but it doesn't appear to affect the results much. Probably it can be narrowed down to a particular detector region.

Errors related to geometry

The following error now appears:

```
[ VERBOSE "MyPfoAnalysis"] CalibrationHelper::GetMinNHCalLayersFromEdge: Unknown exception.
```

Probably related to the access of Gear information. The error is probably emitted from `CalibrationHelper.cc` .

Photon reconstruction retraining

Please note that according to B.Xu, one needs to also retrain the photon reconstruction (see bottom of DDMarlinPandora) when changing the ECal significantly. Presumably (to be checked with B. Xu), it should not change the conclusions of the Pandora calibration significantly (which is only suing single particles), but it doesn't hurt to repeat the calibration after the retraining is performed. Ideally, one would perform the calibration and then use the constants to perform the photon training by running pandora using the `PandoraSettingsPhotonTraining.xml` settings file. After the retraining, a new photon likelihood xml file is obtained which can be fed back to the settings of the pandora calibration procedure to be repeated.

CLIC detector reconstruction settings

Please make sure you are using the latest settings for the CLIC (or other detector). You should diff the file in the `ClicPerformance` package under `examples` with the **`Xml_Generation/CLIC_PfoAnalysis_AAAA_SN_BBBB.xml`** file and make sure that all other parameters **except the ones set by this calibration procedure, typically ending in `_XXXX`,** are the same.

-- NikiforosNikiforou - 2016-01-27

This topic: CLIC > DD4hepClicDetectorCaloCalibration
Topic revision: r13 - 2016-04-07 - NikiforosNikiforou