Efficient Iterative Calorimeter Calibration on the Grid using iLCDirac

André Sailer, Oleksandr Viazlo
CERN-EP-LCD
on behalf of the CLICdp collaborations

CHEP 2019, Adelaide

4 November 2019
Calorimeter calibrations provide coefficients, which translate the energy seen by the calorimeter system to the total energy of particles stopped in the calorimeter.

The calibration procedure provides two sets of constants:

- first set to scale the raw energy read out by the ECAL and HCAL electronics to the energy deposited by the particles.
- second set to correct the reconstructed energy of particles during the clustering step of Pandora.

When is a new calibration needed?

- change in the calorimeter geometry
- new Geant version
- new physics list
- changes in the tracking or particle flow algorithm performances

Why do we need an efficient calibration?

- An efficient way to perform calorimeter calibration is an important piece for detector development and optimization studies.
Calibration constants are obtained by iteratively updating calibration constants, until $\text{Reco_{Mean}} = \text{Truth_{Mean}}$.

To reach usual precision (2% on raw energy measurement and 0.5% on energy for Pandora cluster) calibration performs 20-30 of iterations in total.

(left) Reconstructed energy of EM cluster compared to the true photon energy.

(right) Reconstructed energy of hadronic cluster compared to the true $K_L^0$ energy.
Use case: optimization of EM calorimeter of CLD detector model for FCC-ee

- Performance studies of ECAL with different number of layers but the same total thickness of $\approx 22 X_0$
- FCC-ee centre-of-mass energies: 91.2 - 365 GeV

- 40 layer configuration provides the best photon performance
- 20+10 layer configuration provides better performance at low energies compared to 30 layers which better fits needs of FCC-ee
- 20 layer option leads to significant degradation of photon resolution
Old Pandora calorimeter calibration procedure

**Detector simulation and event reconstruction**
- full detector simulation and reconstruction is done with iLCSoft
- geometry description with DD4hep package
- reconstruction framework Marlin
- event reconstruction is done with particle-flow PandoraPFA package

**Original calibration procedure**
- developed for ILD, adopted for CLICdet
- runs on a standalone condor cluster with rather limited number of workers

**Existing issues**
- the main calibration script consists of thousand lines of BASH with many additional calls of python modules → hard to maintain
- many hardcoded geometry-dependent parameters → it’s complicated to add support for another detector model (e.g. for FCC-ee)
- data has to be copied locally to the cluster
- calibration procedure requires manual intervention → calibration takes days to be performed
Goals of the new calibration procedure

- maximum automation of the calibration procedure
- use the grid instead of a local cluster
  - more resources
  - possibility to run several calibrations simultaneously
- support for multiple detector models (CLICdet, CLD (FCC-ee), other models which use Pandora)
- to be implemented as a native iLCDirac service which allows exploiting Dirac tools and interfaces for monitoring, bookkeeping, data management, workload management, authentication and configuration.
iLCDirac is based on the **Dirac** interware originally developed for LHCb

- **Dirac** (Distributed Infrastructure with Remote Agent Control): High level interface between users and distributed resources
- **iLCDirac**: Additional functionality to provide simple interfaces for the users to the LC Software (Whizard, Whizard2, Marlin, Mokka, org.lcsim, SLIC, ROOT, ddsim)
- Central system for large scale productions
Existing calibration procedure

Calibration consists of four stages:
- Stage 1 and 3 are run on the cluster, while stage 2 is run on grid and stage 4 on local machine (since it cannot be parallelized)
- output from one stage has to be plugged in for next stage manually

New “calibration service” speeds up and simplifies the calibration process by running Stages 1-3 on the grid. It takes cares of bookkeeping of intermediate results.
Calibration service is a native iLCDirac service
- web-based grid job monitoring thanks to Dirac

Controls the whole chain of a calibration procedure
- collects results from finished iterations
- redistributes new input parameters among worker nodes

Provides more direct control over the grid resources
- reduce overhead of job initialisation by reserving grid resources for the entire calibration procedure
- download all required input files for all steps only once per worker node
Dirac calibration service

**Calibration Service**

- **Stage 1**: Calibration with simplified photon reconstruction
  - Single particle samples: gamma, muon, K0L

- **Stage 2**: Photon likelihood training
  - Zuds samples

- **Stage 3**: Calibration with full photon reconstruction
  - Single particle samples: gamma, muon, K0L

- **Stage 4**: Software compensation training
  - Single particle samples: K0L + neutrons

- Significant speedup of the calibration procedure
  - No hard limit on number of jobs
  - Possibility to run multiple calibrations simultaneously

- Multiple detector support (CLIC, CLD, other models which use Pandora)

- Simple configuration of calibration parameters and detector settings

- Bookkeeping of all done calibrations
Reliability of the Calibration Service operation

- Automatically restarts in case of any disruption in the operation.
- Contains up-to-date backup of each calibration which can be used for recovery.
- Configurable threshold on the fraction of finished jobs to start next step:
  - “stuck” jobs will not affect calibration
  - allows avoiding significant slow down of the calibration by one slow machine
- Stores results of the calibration at the server for configurable amount of time
- User can monitor online calibration status with iLCDirac web-interface

<table>
<thead>
<tr>
<th>JobId</th>
<th>Status</th>
<th>MinorStatus</th>
<th>ApplicationStatus</th>
<th>Site</th>
<th>JobName</th>
</tr>
</thead>
<tbody>
<tr>
<td>87092</td>
<td>Running</td>
<td>Application</td>
<td>PandoraCalib_291: step 19 is finished. Waiting for other jobs</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_199</td>
</tr>
<tr>
<td>87091</td>
<td>Running</td>
<td>Application</td>
<td>Running: stage: 3; phase: 3; step: 19</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_198</td>
</tr>
<tr>
<td>87090</td>
<td>Running</td>
<td>Application</td>
<td>PandoraCalib_291: step 19 is finished. Waiting for other jobs</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_197</td>
</tr>
<tr>
<td>87089</td>
<td>Running</td>
<td>Application</td>
<td>PandoraCalib_291: step 19 is finished. Waiting for other jobs</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_196</td>
</tr>
<tr>
<td>87088</td>
<td>Running</td>
<td>Application</td>
<td>PandoraCalib_291: step 19 is finished. Waiting for other jobs</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_195</td>
</tr>
<tr>
<td>87087</td>
<td>Running</td>
<td>Application</td>
<td>Running: stage: 3; phase: 3; step: 19</td>
<td>LCG.CERN...</td>
<td>PandoraCalibCalibration_calid_291_workerid_194</td>
</tr>
</tbody>
</table>

Calibration Agent

- Monitors the health of running jobs
- In case of job failure - resubmits job with input parameters from the latest step
Summary and Outlook

Summary

- New calibration service provides:
  - significant automation and increased speed of calibration procedure
  - simplicity of usage
  - possibility to use all grid resources for calibration
    (and run many calibrations simultaneously)
  - no need for “babysitting” your calibrations
  - web-based calibration monitoring (by means of Dirac)

- Calibration service was successfully used for FCC-ee EM calorimeter optimization studies

Links: Gitlab, User guide

Next steps

- Improve user experience and further automation
- Execution sequence: allow a user to specify in which sequence calibration steps have to be executed
  - allows adding extra calibration steps required in specific cases (e.g. calorimeter with different layer thickness)
- Simulate required input files automatically

Thank you for your attention!
Example of user input for calibration

*** Calibration parameters ***

- numberOfWorkJobs: 100
- fractionOfFinishedJobsNeededToStartNextStep: 0.9
- digitisationAccuracy: 0.02
- pandoraPFTracksAccuracy: 0.005
- startPhase: 0
- startStage: 1
- stopPhase: 99
- stopStage: 99
- disableSoftwareCompensation: True

*** Detector model ***

- detectorModel: FCCee_o1_v04_ecal20_10.tgz
- ecalBarrelCosThetaRange: [0.0, 0.643]
- ecalEndcapCosThetaRange: [0.766, 0.94]
- hcalBarrelCosThetaRange: [0.15, 0.485]
- hcalEndcapCosThetaRange: [0.72, 0.94]

*** ECAL parameters ***

- nEcalThickLayers: 10
- nEcalThinLayers: 20
- ecalResponseCorrectionForThickLayers: 2.0

*** SW version ***

- DDCaloDigiName: MyDDCaloDigi_10ns
- DDPandoraPFANewProcessorName: MyDDMarlinPandora_10ns

*** Input/Output ***

- outputPath: /ilc/user/o/oviazlo/fccee_caloclib/
- CERN-DST-EOS
- steeringFile: fcceeReconstruction_noSWC.xml
- gammaFiles: fccee_10GeV_gamma_FCCee_o1_v04_ecal20_10.txt
- kaonFiles: fccee_50GeV_K0L_FCCee_o1_v04_ecal20_10.txt
- muonFiles: fccee_10GeV_mu_FCCee_o1_v04_ecal20_10.txt
- zudsFiles: fccee_30GeV_Z_uds_FCCee_o1_v04_ecal20_10.txt