Kali: The framework for fine calibration of LHCb Electromagnetic Calorimeter

Ivan Belyaev, Daria Savrina
Institute for Theoretical and Experimental Physics, ITEP, Moscow

Ricardo Graciani, Albert Puig
Universitat de Barcelona

on behalf of the LHCb Collaboration

Abstract. The precise calibration of the electromagnetic calorimeter (ECAL) of the LHCb experiment is an essential task for the fulfillment of the LHCb physics program. The goal of the calibration of the ECAL is to intercalibrate the 6016 cells at an level better than 2% and provide an overall calibration below the 1% level. The final step of this task is performed with two calibration methods using the real data from the experimental setup. Both of them require processing of $O(10^8)$ events which must be selected, reconstructed and analyzed. The analysis is very CPU-consuming, since one method performs an adaptive multi-pass fitting of $O(2 \times 10^5)$ histograms and the other one the minimization of $O(1.5 \times 10^5)$ data-based functions.

In this document we present the Kali framework developed within the LHCb software framework, which implements these two final calibration methods. It is integrated with Grid and makes use of parallelism tools, such as python parallel processing modules, to provide an efficient way, both time and disk wise, for the final ECAL calibration.

The results of the fine calibration with the very first data collected by the LHCb experiment will also be presented. With the use of the Kali framework it took only 2 days of processing and allowed to achieve a miscalibration level of 2-2.5% for the different ECAL areas.

1. Introduction

LHCb experiment is one of the particle physics detector experiments built on the Large Hadron Collider accelerator at CERN. The main goal of the experiment is the precise measurement of the unitarity triangle angles and study of rare CP-violation effects in the decays of beauty particles. Moreover, studies of rare decays of charmed particles and exotic decays of $\tau$-lepton (such as lepton charge violating $\tau \rightarrow 3\mu$) are expected. A more detailed description of the LHCb physics motivation and detector structure may be found in LHCb TDR [?, ?, ?].

The Electromagnetic calorimeter (ECAL) is one of the four subsystems of the LHCb calorimeter system [3]. It is mainly designed to perform 3 tasks:

- Measure energies and positions of the photons and electrons.
- Provide the level zero trigger with high-transverse momentum electron, photon and hadron candidates.
- Take part in particle identification algorithms performing electron/hadron separation.
The reconstruction of neutral pions ($\pi^0$) and prompt photons with good accuracy is essential for the flavour tagging and for the study of B-meson decays and therefore is important for physics program.

The ECAL is formed by 6016 cells divided in 3 different areas, each with its own cell size, and is designed to provide a resolution of

$$\frac{\sigma_E}{E} = 10\% \sqrt{E} \pm 1\%$$

where $E$ is the particle energy in GeV and $\sigma_E$ is the error of this energy. The energy resolution of the ECAL modules has been determined at the test beam in agreement with the designed value.

The whole calorimeter as well as each individual cell needs to be calibrated in order to try to achieve the design resolution. The calibration procedures for the ECAL will be outlined in section 2, paying special attention to the fine calibration. The adopted software solution for this fine calibration, the Kali framework, will be introduced in section 3. With the help of the Kali framework, 2 methods for iterative calibration using $\pi^0$ have been developed. Calibration results and conclusions will be shown in section 4.

2. ECAL calibration procedure

Full calibration of the ECAL involves several steps. Initial calibration was set using the photoelectron signal detected from an LED system installed in the calorimeter, providing and intercalibration at the level of 10%. More precise calibration calibration procedures [4] can be used with real data from the experimental set. As a general rule, these methods consist in determining a multiplicative energy correction for each cell.

The ECAL is precalibrated with the energy flow method. This technique is based on the idea that the distribution of the energy over the surface of the calorimeter should be a smooth function of the coordinates. Therefore, this calibration procedure consists in attempting to smoothen the distribution of energy integrated over time, allowing to achieve a miscalibration level of $\sim 2\%$ [5], [6].

For further improvement two different calibration procedures have been devised, both based on the measurement of a well known value, namely the mass of the resolved $\pi^0$ in its decay into two photons. These procedures, known as the fine calibration, are the mass distribution method, based on mass distribution fits, and the minimization method, based on minimization of a certain function running over all selected $\pi^0$ candidates. The former relies on adjusting the $\pi^0 \rightarrow \gamma\gamma$ mass distribution and shifting the calibration constants in order to have the mass peak at its nominal value, 134.9 MeV/c$^2$. In the latter, instead of working on invariant mass distributions the mass information of all $\pi^0$ candidates (all entries in the nTuple) is used. This allows to concentrate on event-by-event variables, instead of distributions.

As the correction constants depend on the quality of the previous calorimeter reconstruction, a number of the re-reconstructions is needed to achieve convergence. That makes both fine calibration methods to have iterative nature. These two methods used together one may achieve the miscalibration of about 1%.

3. The Kali framework

Kali is the framework used for fine calorimeter calibration, and it is mainly useful for iterative calibration scenarios. It includes a C++ Gaudi-based algorithm for selecting and saving neutral pions, a base python module and two sets of python modules which implement the mass distribution and minimization calibration techniques. However, more calibration methods may be added to this package at any time by inserting the proper selection algorithm and a set of python modules into the framework.
The typical schema for an iterative calibration using Kali can be seen in figure 1. At the first step, common for both methods, data (in DST\textsuperscript{1} format) is processed with Kali selection algorithm, obtaining a fmDST (see section 3.1) and an nTuple. The latter can be processed with any of the two fine calibration methods to obtain the calibration constants. The data sample is then re-reconstructed from the fmDSTs with the same Kali selection algorithm, but applying the calibration coefficients to correct the cells energies. It results in a set of new, corrected, nTuples and fmDSTs. These ones are then reused in an iterative fashion until the method has converged.

To make the calibration procedure faster both methods use a technique of parallel processing in python by using PyROOT scripts (based on the TPySelector python extension of the ROOT TSelector). It allows to divide input data between several machines (or several cores of a multi-core machine) and process it with one common algorithm, then merging the output from different workers so that a user recieves it as a whole.

![Figure 1. Schema of a typical iterative calibration process using Kali](image)

### 3.1. The fmDST

The fmDST (femto-Data Summary Tape) is a special format for data storage, generated as output of the Kali selection algorithm. Its structure is the same as that from the typical DST used in LHCb for data storage (and hence it can be used by standard LHCb software), but each event contains only the data needed to perform calorimeter calibration tasks:

- Only relevant tracks, clusters, hits, etc.
- Only information from relevant subdetectors (calorimeter, tracking if needed...).

The result is a very compact output file, \(\sim 300\) bytes/event, which is \(\sim 100\) times less than usual DSTs.

The fmDST is fully compatible with the standard LHCb calorimeter reconstruction tools, that means that they can be reused as if it were a normal DST, allowing the application of the calibration constants. This reprocessing is much faster given the simpler nature of the fmDST, \(\sim 10\) times faster than typical DSTs.

The fmDST format proved to be very useful for calorimeter calibration scenarios since they involve processing large amounts of data (\(O(10^6)\) events, representing \(\sim 30\) GB) in an iterative fashion. Using a stripped-down data storage produces huge improvements both in storage and processing time.

\textsuperscript{1} The DST (Data Summary Tape) format is the ROOT-based data storage format used to to stored reconstructed LHCb data.
3.2. Mass distribution fit method

This method relies on fitting the \( \pi^0 \rightarrow \gamma\gamma \) mass distribution and adjusting the calibration constants of each cell in order to have the mass peak at its nominal value, 134.9 MeV/c\(^2\). Since this needs to be done for 6016 cells, this technique requires filling and fitting a large number of histograms.

First, for each of 6016 cells six histograms (signal and background for 3 different energy cuts) are filled from the nTuples obtained at the \( \pi^0 \) selection stage. This is done event by event with the help of a TSelector ROOT class.

Afterwards, a sum of a second order polynomial (for the background shape description) and gaussian function (for the peak) is used as the fit function. For a better convergency this procedure is done through several steps.

(i) First the background histograms are fitted to estimate the initial background fit parameters for the signal histograms.

(ii) Then reference histograms with big number of entries but similar properties are used to estimate initial values of the parameters (for example a histogram containing the entries from the whole calorimeter zone to which the current cell belongs to).

(iii) The initial parameters are set up to the fit function and the fit of the signal histogram is performed.

(iv) If the fit didn’t converge, reference and background histograms with a different energy cut are used for the parameters estimation. These new reference and background histograms should contain more entries, especially in the signal peak of the reference histogram. After that the previous three steps are repeated.

For the better accuracy of the obtained parameters the fit function is trained at each step by fitting the parameters one by one.

An average of \( \sim 6 \) fits per cell are performed. Several iterations are needed to achieve convergency, defined by the stabilization of the position of the mass peak around the nominal value (4-5 are usually enough, see figure 2 for the mass peak variation as a function of the number of iterations). In the end, for a full calibration procedure 200,000 histograms are filled and over 100,000 several-step fits are performed.

The calibration procedure has been performed in the LHCb-CAF\(^2\) using 7 out of 8 cores in the worker node. Speed scales nicely with the number of cores used, and therefore usage of CAF greatly increases the processing speed. For 1 iteration of \( 10^6 \) of events, processing takes \( \sim 2.5 \) hours with 7 cores, where \( \sim 1.5 \) h are needed for filling the histograms and \( \sim 1 \) h is used by the fitting procedure.

3.3. Minimization method

Instead of working on invariant mass distributions, the mass information of all \( \pi^0 \) candidates (all entries in the nTuple) is used. This allows to concentrate on event-by-event variables, instead of distributions. However, to use this method a sample that reproduces the observed background is needed. Combinatorial background (almost all background) can be generated using the opposite cell information, and thus an nTuple with only background candidates has been added to Kali.

From this nTuple containing both data and background candidates, correction constants are calculated minimizing the following function:

\[
f = \sum_{\text{data}} w_i \delta m_i(c_i) - \frac{1}{2} \sum_{\text{bkg}} w_i \delta m_j(c_j)
\]

\(^2\) The LHCb-CAF (Calibration and Alignment Facility) is a dedicated cluster providing computational resources for calibration: 4 worker nodes (2 with Scientific Linux Cern 4, 2 with Scientific Linux Cern 5) with 8 cores each.
where $m_i$ are the $\pi^0$ candidates masses. The $c_i$ are the coefficients to adjust and $w_i$ are the weights used in order to control the influence of those candidates farther from the mass peak. This function is minimized for each pair of opposite cells so 3008 minimizations have to be performed for each calibration step. Several steps (usually $\sim 10$, depending on the convergence criteria) need to be performed per iteration, after which the sample needs to be re-reconstructed again. Around 4-5 iterations are needed for full convergence (stabilization of the mass peak).

As a result, 150,000 MINUIT minimizations (running over all signal and background entries of the Kali nTuple) need to be performed. For a typical sample of $10^6$ events, using a 2.33GHz Intel Xeon machine each of these MINUIT minimizations takes about 10 seconds. Again, parallelization is called for, and in this case 2 methods have been used:

- **Parallelization via python subprocess.** Each step takes $\sim 1$ hour, so full iteration needs $\sim 10$ hours. Speed scales nicely with the number of spawned processes up to the number of cores in the machine.
- **Using local cluster.** Send up to 200 parallel jobs, but one needs to be careful that these jobs are not too short, otherwise initialization takes a substantial processing time.

4. Results and conclusions
With the arrival of the first data, and after the precalibration of the whole calorimeter system, the first fine calibration of the LHCb ECAL was performed at the end of May 2010. The sample was 80M events, and the whole process took up to 2 days. The two methods provided very good agreement (the final result can be observed in figure 3), with an estimated calibration level of $\sim 2\%$.

For this procedure Kali has proven to be very useful, allowing fast reprocessing of data while keeping storage demands very low. It has become the standard calorimeter fine calibration tool, used now for $\pi^0$ calibration but easily extendable to other types of fine calibration with similar needs.

Acknowledgments
The authors would like to say many thanks to the LHCb calorimeter group for fruitful discussions. A. Puig acknowledges financial support by the MICINN FPU grant.

References
Figure 3. $\pi^0 \rightarrow \gamma\gamma$ mass distribution plot after the first fine calibration


