Automatic data processing for prompt calibration of the CMS ECAL

Simone Pigazzini
on behalf of the CMS collaboration
ETH Zurich, Otto-Stern-Weg 5, CH-8093 Zürich
E-mail: simonepigazzini@proton.me

Abstract. The CMS ECAL has achieved an impressive performance during the LHC Run1 and Run2. In both runs, the ultimate performance has been reached after a lengthy calibration procedure required to correct ageing-induced changes in the response of the channels. The CMS ECAL will continue its operation far beyond the ongoing LHC Run 3. Its barrel section will be upgraded for the LHC Phase-2, and it will be operated for the entire duration of the High Luminosity LHC program. With the increase of instantaneous luminosity, the ageing effects will increase, and so will the required frequency of calibrations. It is therefore crucial for the CMS ECAL community to reduce the time and resources needed for this task, in order to ensure with limited person power a smooth operation and excellent performance on the long term. A new system has been developed during the LHC second long shut down to automatically execute the calibrations workflows on a daily basis during the data taking. The new system is based on industry standard tools (Openshift, Jenkins, Influxdb, and Grafana) and provides a general interface to orchestrate standalone workflows written in different programming languages. It also provides interfaces to other existing CMS systems to steer the processing of selected data streams and to upload newly computed calibrations into the database used for the data processing for physics analyses. The new system is designed with the ambitious goal of cutting the time needed to provide the best possible performance for physics analyses by one order of magnitude. The system offers an extensive suite of diagnostic tools that provide a constant monitoring of its status as well as the option to send alerts in case of problems. In this report, the general structure of the system is presented, along with the results from the first year of operation. The detail of the monitoring and alert system will also be discussed.

1. Introduction

The CMS electromagnetic calorimeter (ECAL) has provided high precision measurement of the electron and photon energy during both the LHC Run 1 and Run 2. Achieving the optimal resolution has required the work of up to 20 people, working continuously during the past 10 years, to derive new calibration for every data-taking year. The repetitive nature of the calibration task calls for the development of a system to automate the procedure as much as possible. The benefits of such a system are clear: faster and more reliable calibration delivery and liberation of human resources needed for addressing challenges lsuch as the development of new machine learning techniques for energy clustering. In this conference report, the details of the new automation system developed during LHC long shutdown 2 and deployed in 2022 are presented. The framework aims at automating the production of new calibrations during data taking. In addition, the capability to re-derive such calibrations for data after the data taking

has been completed is also available, useful for providing a legacy dataset for physics analyses targeting precision measurements.

2. The CMS ECAL calibration

The CMS ECAL is a homogenous scintillating crystal calorimeter. The $PbWO_4$ crystals used as active element undergo a loss of transparency when exposed to high radiation environments. The transparency loss translates into a degradation of the energy measurement. The response variations are tracked using a monitoring system based on the injection of laser light in the crystals. A refined calibration as function of time and position in the detector is further derived using proton-proton collision data for each single channel of the detector. An overview of the calibration procedure used for the Run 2 data-taking as well as the performance achieved can be found in [1], while a comprehensive discussion of the calibration methods can be found in [2].

The laser data have been automatically analysed since the CMS installation in 2011. The correction derived are deployed every 40 minutes and used at all stages of the CMS data processing.

The refined calibration using collision data consists of several workflows. Different workflows require a different amount of data to be recorded before producing a reliable calibration. The amount of data needed ranges between hundreds of pb^{-1} to tens of fb^{-1}. The workflows usually consist of various tasks executed in series, in most cases the first task requires running computing jobs in parallel to perform a dedicated reconstruction of the raw data acquired by the CMS detector.

3. CMS data processing and calibration infrastructure

The data acquired by the CMS data acquisition system (DAQ) is normally transferred to the head node of the CMS computing infrastructure. This node is labelled Tier 0 (T0) and is located at CERN. It connected to CMS through a dedicated high speed connection. The CMS DAQ system modularity allows collecting data for physics analyses at a rate of 1 kHz and simultaneously records several data streams with reduced content (e.g. only raw data from the ECAL rather than the full CMS detector) with a higher event rate for calibration purposes. In the case of the ECAL three other streams with a rate up to 7 kHz are currently active to assist the detector calibration.

The CMS DAQ system groups chunks of data into coherent units named *runs*. The bulk of the data collected for physics analyses is processed 48 hours after a run has been marked as completed by the DAQ system. The acquired raw events are processed in parallel in the T0 farm. Before the bulk processing starts, the calibration streams and a subset of the physics dataset are processed in a fast reconstruction, again in the T0 farm. The output of this *Express* reconstruction is fed to the prompt calibration loop (PCL) [3]. The PCL offers automatic data processing to provide per-run calibration constants. The system described in this report extends the features available in the PCL system by offering a computational model that is more flexible in the inputs available: not only data from the *Express* reconstruction, but also from the bulk reconstruction. The PCL also requires the calibration code to be compliant with the C++ based CMS software framework and dictates the format of the processing output. The automation system for ECAL calibration executes tasks written in any code (using containers). It further expands flexibility compared to the PCL allowing the developer to define any output format. It also enables workflows of arbitrary complexity in terms of number of tasks to be concatenated and grouping of runs to be processed together.

4. Automatic calibration framework

In this section, the structure of the new automation framework is discussed. The implementation details and the software used are presented as well.

4.1. Infrastructure components

The automation system uses Red Hat Openshift [4] to deploy the framework applications. A general purpose Openshift instance deployed by the CERN IT department has been used so far. The application deployed with Openshift are: Jenkins [5], Grafana [6] and InfluxDB [7]. Jenkins provides scheduled continued execution for the workflows that are part of the calibration procedure. InfluxDB is used to record the state of each task and workflow, a separate table in the database stores the status of jobs running in parallel within a task. Grafana provides a human-readable interface to monitor the status of the automated workflows. It also provides monitoring of the system resources (e.g. disk space). The configuration for each workflow is stored in a git repository, hosted on a GitLab instance deployed by CERN IT. Each workflow is defined in a separate branch in the same repository in order to track configuration changes separately for each workflow. The data processing and calibration code is versioned in separate repositories to avoid mixing code development with the configuration of the automation workflow. A python package [8] has been developed to provide the interfaces between the different applications of the framework, as well as a user interface to allow inspection and interaction with the automation framework. The python package provides base classes to build single task handler classes. The handler class implements the job submission, execution and check for completion, updating the status of the task as the processing proceeds. Another class provides the equivalent for single jobs, it is used as a wrapper when running parallel asynchronous jobs within a single task, with each job logging to the InfluxDB database its status independently of the others. The framework makes intensive use of notifications to Mattermost [9] groups to which interested users can subscribe in order to receive alerts from the system. The Mattermost integrations also allow to send information from the Mattermost application to the automation framework through webhooks. A number of automatic actions with manual triggers have been implemented, exploiting this type of communication between Mattermost, Jenkins and the automation python library.

4.2. Workflow structure

The flowchart representing the execution of a single task and of tasks within a workflow is illustrated in Fig. 1. During data-taking, the processing of a newly acquired run is triggered as soon as the CMS DAQ system marks the run as completed in the centralized CMS run registry. On the automation Jenkins instance a task runs every hour to detect completed runs and injects them in the InfluxDB database, setting the status of the workflow to be executed for that run to *new*. The set of workflows assigned to each run is determined by a set of conditions in the run metadata (e.g. amount of integrated luminosity delivered by LHC, CMS detector status, DAQ configuration). Run categories can be created through the automation python interface and are stored in the InfluxDB database in a dedicated table. Runs that fall into multiple categories will be assigned to all workflows specified by the different categories.

As mentioned in the introduction, the automation system is designed to process data both during LHC operation, and reprocess previously acquired data during re-calibration campaigns. In the latter case, chunks of data are injected manually by users in order to trigger the processing, while the following steps remain unchanged.

Each workflow is executed by Jenkins at regular intervals, loading its configuration from the GitLab repository at each execution. If the status of a task within the workflow for a run is marked as *new* the task handler class will submit the jobs and mark the status of the task as *processing*. Tasks that require data from an entire LHC fill or even larger amount of data will wait until the set of runs marked as *new* matches the requirements for the task to start. The execution of tasks along the workflow chain are subject to a locking mechanism: task N might be started only if task N-1 has been completed for a given run. The execution of a task can also be locked while waiting for payloads to be delivered by other data processing, for instance

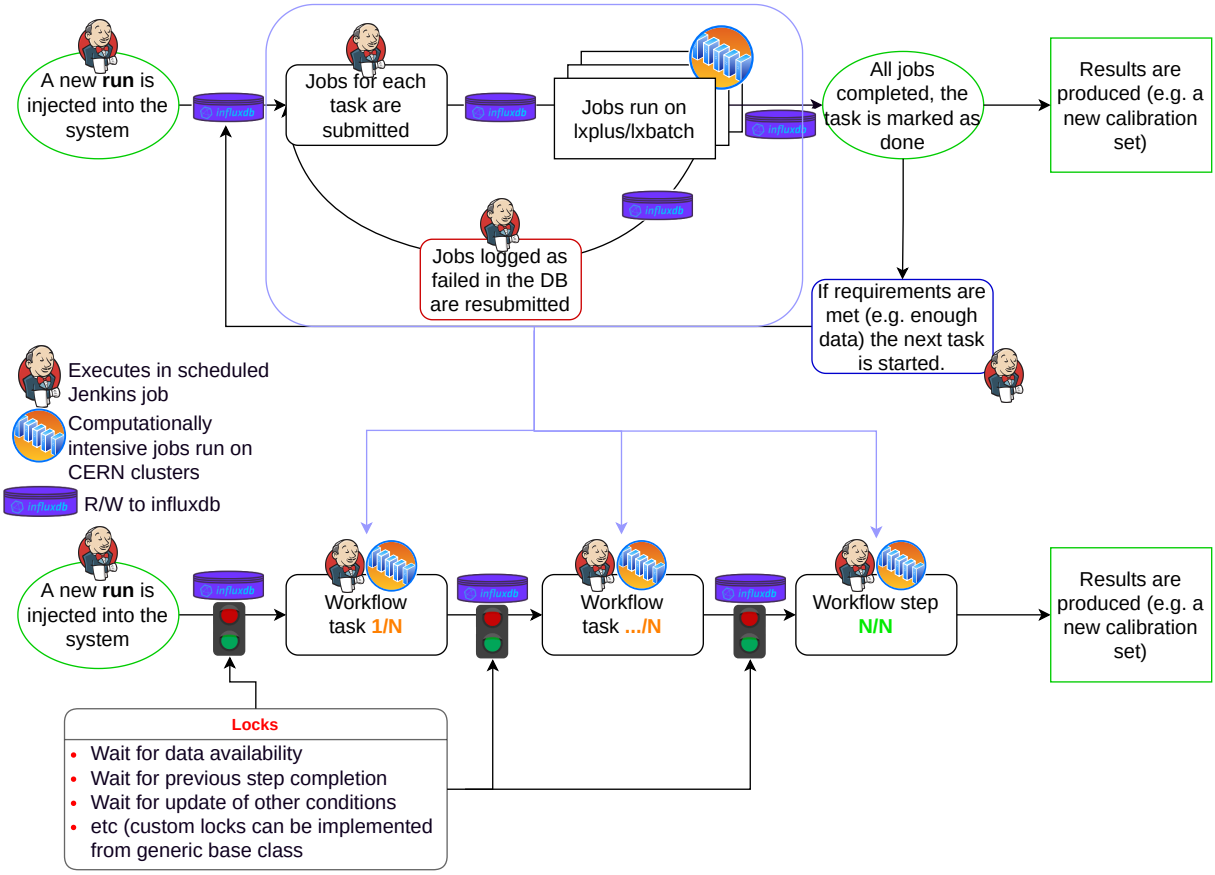

Figure 1. Task and workflow execution diagram. The flowchart representing a single task is displayed at the top, several tasks can be concatenated into a single workflow.

the computation of the ECAL energy calibration must wait for the set of correction factors derived by analysing the laser monitoring system data. These corrections are derived using a dedicated machine installed in the CMS service cavern and are not part of the automation workflow discussed here. Multiple locking conditions can be combined to steer the workflow processing. An example of a workflow definition and execution in Jenkins is given in Fig. 2. The Jenkins configuration file (Jenkinsfile) in each branch of the automation repository defines a workflow and the execution of its tasks. In the example, three tasks are executed to produce monitoring plots of the ECAL energy scale using the invariant mass of diphoton pairs identified as π^0 decay products. Note that the first task is executed for each new run and requires multiple jobs to be submitted in parallel using the CERN batch system, while the subsequent tasks are executed grouping data per LHC fill and within a single job.

Workflows can have forks in their flow, for instance the validation of a new calibration requires running two tasks in parallel on the same data, using in one task the old calibration set and in the other the new set. The automation framework supports these tasks using the locking mechanism described above. The framework also supports the possibility of merging separate workflows, again just by requiring that the merging task must wait for the completion of the relevant tasks from the different workflows to be merged, as shown in Fig 3.

The input data of each task can have different natures. An interface to access the CMS data stored on the GRID infrastructure is provided exploiting the CMS centralized systems. Products

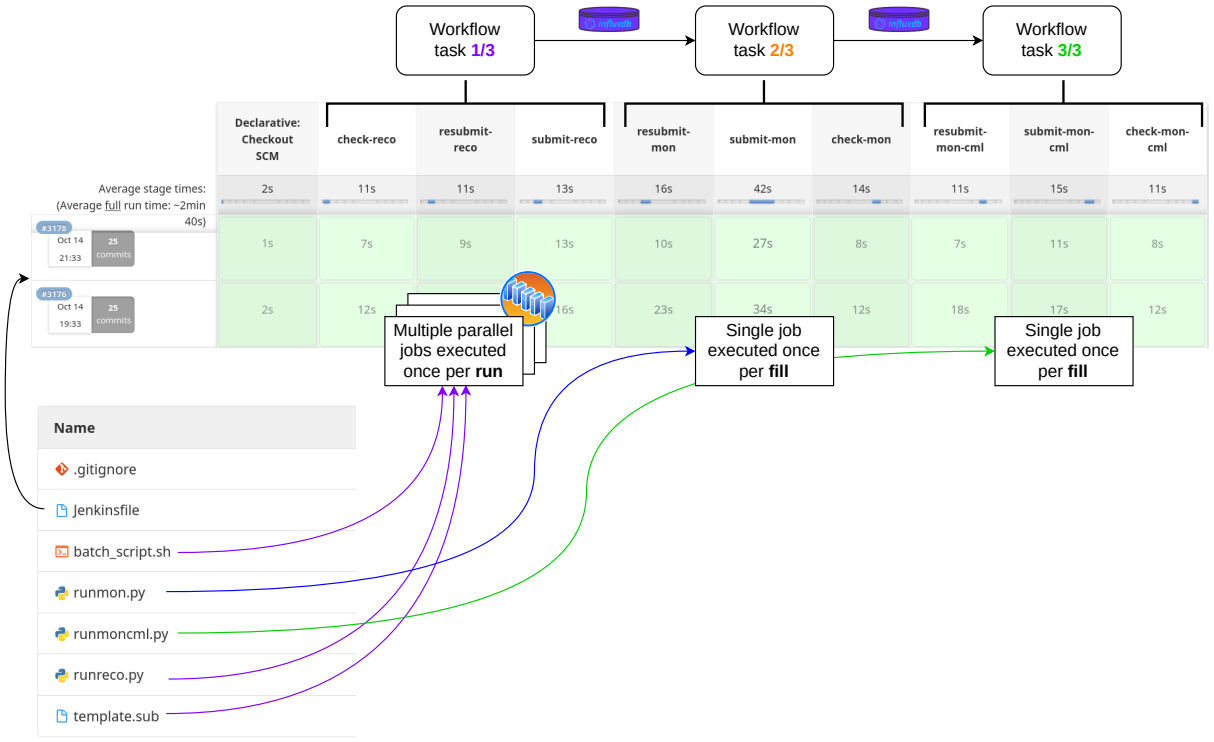

Figure 2. Relation between the GitLab repository containing the workflows configuration and the execution of a workflow with three tasks on Jenkins. Each task has three main stages: submit, resubmit and check. In the submit stage, jobs are executed for runs marked as *new* for the task, provided the locking conditions are satisfied. In the resubmit stage, failed jobs are reprocessed while in the check stage the status of the task is marked as *done* for runs in which all jobs have been successfully completed. The execution of the three stages as well as the different tasks is completely asynchronous for each run.

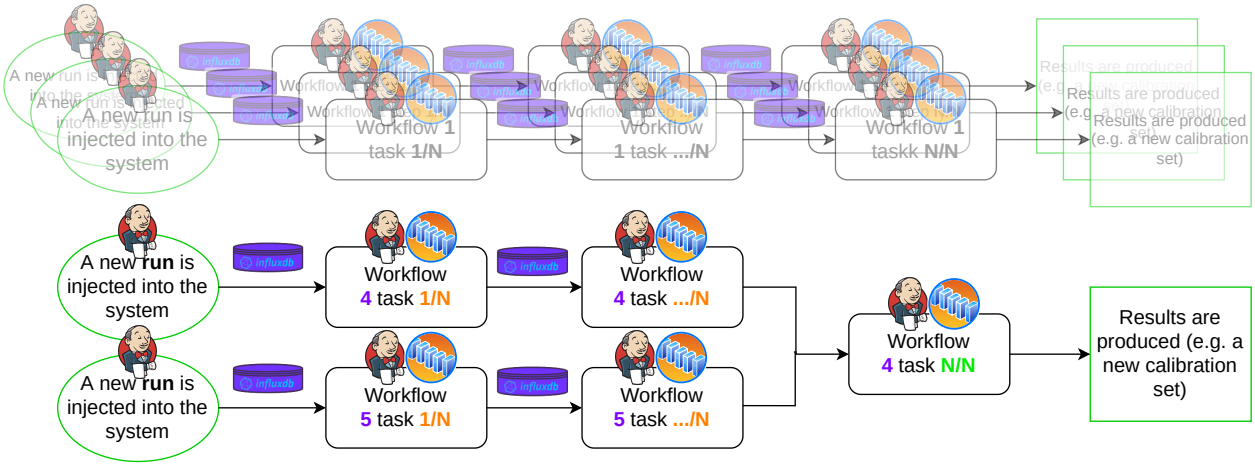

Figure 3. The automation system runs various workflows in parallel, each with their own requirement and locking mechanisms. Workflow can have forks in the processing or be merged using the locking utilities provided by the framework.

of the automation tasks are recorded in the InfluxDB database and can be easily recalled by other tasks using the python interface. The output of each job can be either a file stored on a shared filesystem or values directly stored in the database.

5. Operation in 2022

The automation framework has been operational during the first year of the LHC Run 3. Despite being planned as a commissioning year the system delivered conditions that were actually used to update the calibration for the ongoing data-taking.

The computation of the ECAL alignment with respect to the tracking system, the monitoring of the energy scale using $\pi^0 \to \gamma\gamma$ and $Z \to e^+e^-$, the relative time synchronization of the ECAL channels and the computation of the evolving pulse shape of each channel have all been deployed as workflows in the automated system. The system has also partially integrated a number of other workflows, mostly running data reconstruction tasks on dedicated calibration streams that are not currently processed at the T0 and that are then used by analysts to derive calibration factors. The system operated smoothly without interruption through the six months of data-taking, executing an average of 500 jobs per day.

6. Summary

The CMS ECAL requires a continuous calibration in order to track and correct for radiation-induced ageing effects. In the past, the calibration procedures have been mostly carried out manually, running data processing when needed. During the LHC long shutdown 2 the group in charge of the CMS ECAL calibration has developed a new automation system to allow the continuos processing of data collected by CMS while reducing the person power needed to run the calibration. The system has been in operation since June 2022, with new workflows being constantly added.

The system builds on state-of-the-art software for automatic data processing, visualization and monitoring. It is currently being extended to include the processing of the calibration for other sub-detectors in CMS. Save for the custom python library that provide the interface between the calibration tasks and the system, the framework is general enough to provide a platform that can be used by other groups or experiments to implement an automatic and continuous data processing.

References

[1] Francesca Cavallari, Chiara Rovelli, on behalf of the CMS Collaboration. Calibration and Performance of the CMS Electromagnetic Calorimeter in LHC Run2. EPJ Web Conf. 245 02027 (2020). https://doi.org=/10.1051/epjconf/202024502027

[2] The CMS collaboration. Energy calibration and resolution of the CMS electromagnetic calorimeter in pp collisions at $sqrts = 7$ TeV. (2013). Journal of Instrumentation, 8(9), P09009. https://doi.org/10.1088/1748-0221/8/09/P09009

[3] Cerminara, G., van Besien, B., on behalf of the CMS Collaboration. (2015). Automated workflows for critical time-dependent calibrations at the CMS experiment. Journal of Physics: Conference Series, 664(7), 072009. https://doi.org/10.1088/1742-6596/664/7/072009

[4] https://docs.openshift.com/

[5] https://www.jenkins.io/

[6] https://grafana.com

[7] https://www.InfluxDBdata.com/products/InfluxDBdb/

[8] https://cmsecaldocs.web.cern.ch/ecalautoctrl/

[9] https://mattermost.com/blog/mattermost-integrations-mattermost-api/