ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

EVOLUTION OF THE CERN BEAM INSTRUMENTATION OFFLINE ANALYSIS FRAMEWORK (OAF)

A. Samantas, M. Gonzalez-Berges, J-J Gras, S. Zanzottera, CERN, Geneva, Switzerland

Abstract

The CERN accelerators require a large number of instruments, measuring different beam parameters like position, losses, current etc. The instruments' associated electronics and software also produce information about their status. All these data are stored in a database for later analysis. The Beam Instrumentation group developed the Offline Analysis Framework some years ago to regularly and systematically analyze these data. The framework has been successfully used for nearly 100 different analyses that ran regularly by the end of the LHC run 2. Currently it is being updated for run 3 with modern and efficient tools to improve its usability and data analysis power. In particular, the architecture has been reviewed to have a modular design to facilitate the maintenance and the future evolution of the tool. A new web based application is being developed to facilitate the users' access both to online configuration and to results. This paper will describe all these evolutions and outline possible lines of work for further improvements.

INTRODUCTION

Several thousand instruments are installed in the CERN accelerator complex to measure beam parameters (e.g. position, losses, intensity, etc). Table 1 gives an approximate overview of the types of instrument per accelerator. Each instrument is typically composed of a monitor that can be inserted in the beam pipe or installed outside, an analogue/digital electronics system and a software layer.

The complex itself spans over several kilometres, with most of the accelerators installed underground. The regular operation of these instruments is a major challenge. They need to be available for the run periods and their performance has to be guaranteed.

Table 1: Approximate Number of Main Types of Instruments per Type and Accelerator

	LHC	SPS	PS complex	Other
Position	1300	300	300	50
Losses	4200	10	50	30
Intensity	20	10	80	10
Profile	40	10	80	30

The Offline Analysis Framework (OAF) [1,2] was developed some years ago to deal with this challenge. The instruments produce beam physics data as well as status information. Both sets of data are used to monitor the instrument performance and its evolution through time. Our final aim is to fine tune the instruments' performance and introduce predictive maintenance on their mechanics and electronics parts.

CURRENT USAGE

All these instruments measure, regularly or on demand, the different beam observables. Then, they send these values, together with relevant status and setting registers, to the CERN Control Room for the real time operation of the machines and into a centralized logging database (NXCALS) [3] for future offline analysis (Fig. 1).

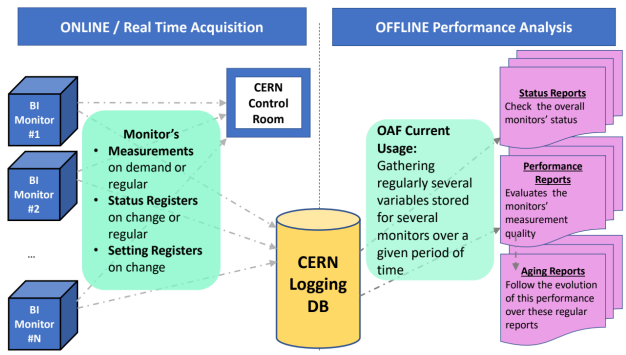

Figure 1: Standard usage workflow.

All this logged information contains too much data to be analysed properly manually. So, in 2013, we decided to develop the Offline Analysis Framework [1] in order to regularly monitor the records stored and automatically produce 3 kinds of reports:

- Status reports that focus on the state of the instrument and monitor humidity, temperatures, logging frequency and more, raising alarms whenever necessary.
- Performance reports will monitor and assess the quality (accuracy, resolution, stability) via device comparisons or regular calibration sequences.
- Aging or long term evolution reports will survey the evolution of the quality evaluations made in the daily performance reports.

Most of these analyses can be derived from the raw DB records via standard data treatment and plotting functionalities directly supported by OAF. However, some analyses require ad-hoc computations or specific plots. To cover these non-standard requirements, OAF offers the possibility, whenever necessary (i.e. when the need is not covered by the build-in OAF features), to add expert python code to this specific analysis that will be

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

executed at the end of the standard OAF process with a direct and simple access to:

- The extracted raw DB data and all subsequent data produced by the OAF associated treatments.
- The OAF programmable interface which allows the expert graphs, tables and alarms produced to be included in the reports.

ARCHITECTURE REDESIGN

Although stable and capable of running reliably for years, the original software architecture of OAF had some flaws that made working with its codebase difficult.

For example, adding new types of analyses to be run on the collected data was very complex: it required the original author to write lots of code, and it was not modular, meaning that no code could be reused from one analysis to the other. As a result, over the years the source inflated due to the code duplication, and became very hard to maintain and develop further.

An additional issue we found was that users did not find the user interface understandable enough to use it, and required constant assistance from the original author to use the system.

Recently, CERN adopted NXCALS as its central logging database, forcing OAF to adapt to this new interface. Due to the need to perform this upgrade, we took the opportunity to review the entire system, and we decided to perform a nearly complete redesign. The aim of this redesign was:

- Make OAF modular and maintainable.
- Make OAF easier to use for the end users.
- Make the creation of new analysis types easier.
- Review which of the existing analysis were still being used, and which ones could be dropped.
- Clean up and simplify the codebase heavily.
- Strongly improve the performance on some of the analysis.

As a plus, we also agreed that the output of OAF could be improved by adding more output mediums. From the original PDF-over-email report (Fig. 2), we decided to add several others: a web application, a raw data file, and we plan to add Python notebooks as well, to allow users to explore the results further (Fig. 3).

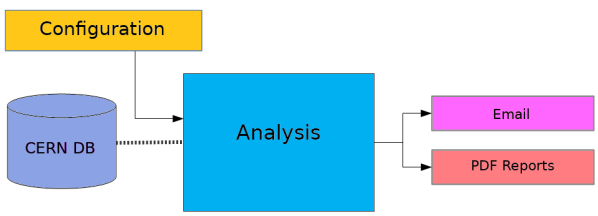

Figure 2: Original architecture of OAF.

One more incentive to the redesign was that the original OAF has been developed in Python 2. This fact meant that a full rewrite was not necessary, but large chunks of the old code could be kept in the new architecture with minimal or no changes.

During the redesign, we identified several logic units and split the codebase into meaningful modules, which were then reconnected over a more explicit API. This process proved to be very helpful to the codebase simplification goal we identified earlier.

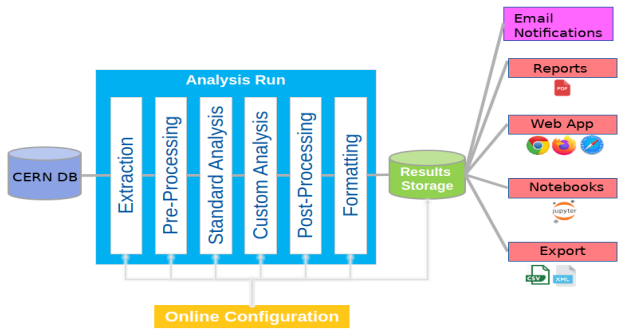

Figure 3: OAF architecture after the redesign.

In between the last step of the analysis and the output, we also introduced the concept of *results storage*, a temporary store for the processed data. This storage was critical to make our web interface capable of showing the analysis' results without having to re-run the analysis itself from the original logged data. This storage, though, also represented a challenge, as we wanted to avoid data duplication with the database as much as possible. We finally decided to use HDF5 files for storage, due to the format's properties and its capability to efficiently store complex data structures.

Along with the intermediate storage, we also developed a small utility library to act as an interface between the raw files and the Python code, called oaf-commons, which could be shared between the proper OAF pipeline and the web application without incurring in more code duplication.

As a result of the redesign the runtime architecture completely changed. This can be seen in the Figure 4. The users interact with the system through a web application that will be described in the next paragraph. For complex analysis where custom code is needed it will be placed directly in Gitlab. The OAF engine gets from Gitlab all the configuration and the custom code. It queries the logging database and produces results in a distributed file system. Then, the web application can access these results and display them.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

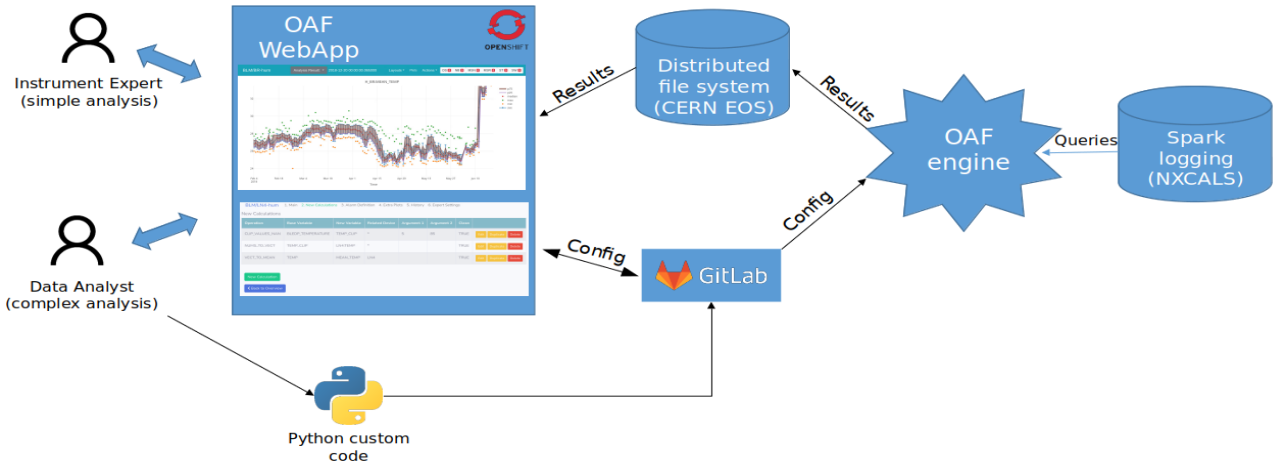

Figure 4: Runtime architecture.

OAF WEB APPLICATION

The OAF Web Application (Webapp), is a web based application in Python using Django Framework for both visualization of BI data results and analysis configuration.

Visualization of Results

All the HDF5 files that contain the different instrument analysis results are saved in a specific directory. In the Webapp, a list of different instrument analysis where we can browse between results from different dates is generated. By default one plot is visualized but there is also the option to visualize two or four plots the same time (Fig. 5).

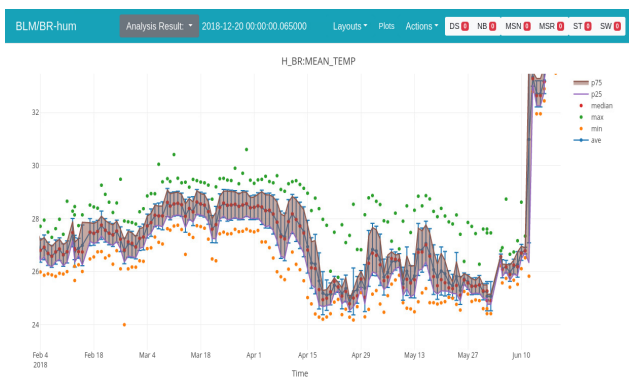

Figure 5: OAF Webapp instrument analysis example.

There are several alarms, various plot types and categories. OAF can raise an alarm in the following instances:

- MISSING: Alarm raised in case of missing entities.
- NB_REC: Alarm raised when number of records is not as expected.
- DISCRETE: Alarm raised if values do not belong to the given list of values.
- MEASURE: Alarm raised if values do not remain within given range.
- STATUS_BIT: Alarm raised if the state of status bit register does not correspond to the nominal state of each bit.
- SWITCH: Alarm is raised each time a value is changed.

The following plot categories that are already implemented are:

- Default Plots: Input data used for the analysis coming from NXCALS.
- Extra Plots: Produce more complex plots and tables combining default plots or custom ones on demand.
- Historical Plots: Show evolution of parameters across different analysis executions.
- Admin Plots: Monitoring and analysis of OAF itself.
- Info Plots: Summary of alarms and other info.

Each of these categories can produce different types of plots. These types can be value over time, value over value, Histogram, Num-box, Tables and many more.

Quite often we need to transform the displayed results in order to understand better the measurements. The most common conversion types (statistic calculations, FFTs, KDE) have been identified and are made available to the user. Different plot features are already implemented, providing like this a deeper, more specific and accurate analysis according to the users' needs.

An overview of all the alarms that were detected the last 24h in each of the available instrument analyses is provided in a table (Fig. 6).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

Instrument	Analysis	Total Alarms	Newest Result	Comment
filter data...				
BLOG	AD	0	2018-06-15 00:01:17.333000	
BLOG	PSB		2018-06-15 00:00:00.065000	
BLOG	CPS-ana	2	2018-06-15 23:40:01.900000	
BLOG	SPS		2018-06-15 00:00:03.735000	
BLOG	LEI	0	2018-06-15 00:00:00.455000	
BOFSU	LHC-survey	0	2021-04-30 00:00:00.000363275	
BCT	LHC-B1-cmp	0	2021-02-19 00:01:00	
BCT	SPS-safety	0	2018-06-15 00:00:03.735000	
BCT	LHC-fast-ldm	0	2018-05-06 23:40:00	
BCT	L2-trans	0	2018-11-11 23:00:00.065000	
BCT	L3-trans	0	2018-11-10 00:00:00.455000	
BCT	LHC-B2-cmp		2021-03-23 00:01:00	
BCT	LHC-calib	0	2018-06-24 00:00:00	

Figure 6: Overview alarm table.

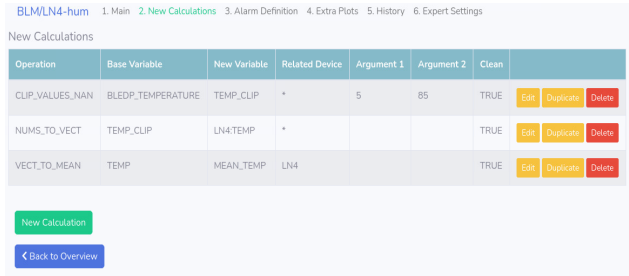

Figure 7: OAF Webapp configuration wizard.

Analysis Configuration

Apart from the visualization of results in the Webapp one can define the configuration of the analysis. Until today the user had to configure excel files without hints and error control. In the Webapp version we provide a user friendly interface guiding the user through a wizard.

Configuration is achieved by the modification of different tables (Fig. 7) that are stored in an internal database. This allows the configuration of many features of the framework. The instrumentation experts do not have to provide any code for a common usage case. The different table categories are the following:

- Main: General information.
- New Calculations: Declaration of new variables based on extracted ones.
- Alarm Definition: The criteria for the alarms.
- Extra Plots: Creation of complex plots combining default ones based on the already existing variables or the new ones that are declared in the New Calculations table.
- History: Evolution of parameters over a long time span.
- Expert Settings: The user can add expert code in order to include more functionalities than we already provide.

Different analysis management options are already available. Through these options the user can deploy the configurations to the production server, import existing configurations, compare if there is a difference between the current modifications and the production server and finally re-run the analysis. A new user can also create an analysis for an instrument by initially adding an empty one or importing an existing one. An overview table with the current status of all the analyses gives details about the production state, including inconsistencies between development and production configuration.

Technologies Involved

The Webapp is running in the local network of CERN with limited access to the CERN users only with a Single-Sign-On (SSO). For the development we use Django [4], an open source web application framework written in Python. The data analysis code is written in Python using libraries such as Plotly, NumPy, SciPy, Pandas. Using Gitlab's CI/CD tool the Webapp is deployed on Openshift after every change in the code, once the unit test have passed.

EXAMPLE USE CASES

We have selected two use cases to illustrate where the OAF is being used today. A full list would be too long to be covered in the paper. At the end of the LHC run 2 (end of 2018) there were nearly 100 use cases running daily.

The ability to measure luminosity on an absolute scale is essential for CERN's Large Hadron Collider (LHC) physics experiments [5]. The dominating contribution to the final uncertainty of the absolute luminosity calibration originates from the bunch current normalization. Thus, one important use case of OAF has been to assess and monitor our current different measurement performance. We are systematically cross-checking the measurements of the four DC BCTs (Beam Current Transformer) and the two fast BCTs to detect deviations between them. The calibration of all these instruments is compared with lab measurements to make sure that the absolute values are correct.

A second simpler example is the regular surveillance of the control of the water cooled racks temperatures hosting our Beam Position measurement electronics to prevent calibration changes caused by the building temperature variations (Fig. 8). We have to make sure that the temperature variations are below 1°C so that the orbit calculations are not affected.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

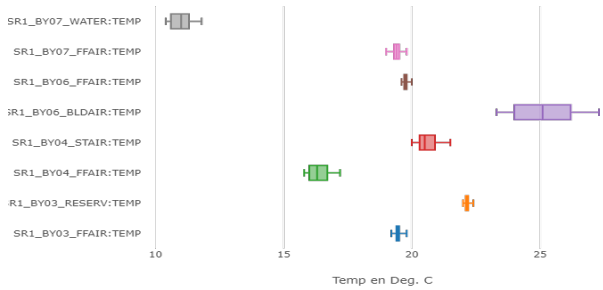

Figure 8: Temperature evolution envelop (min, max, median and a box from 1st quartile to 3rd) of the different sensors located in LHC building SR1.

EVOLUTION

We have several ideas to improve OAF in the short term. For the whole life cycle of an analysis, it is still necessary to open different tools. Our goal is to have a single entry point application that will allow to configure all the required steps (data logging, data preprocessing, data queries, analysis configuration, visualization, export).

We are also looking to have a full Python code base. Today we still need some Java functions to query the Spark based Accelerator Logging (NXCALS).

In the medium term, we want to improve and modernize the analysis capabilities of OAF to be able to cover new use cases. We are currently surveying the needs of users in the Beam Instrumentation group.

CONCLUSION

The Offline Analysis Framework was developed some years ago to systematically analyze beam instrumentation data. The tool has helped diagnose and solve numerous issues so far. In the meantime, the technologies it is based on have evolved and opened new possibilities. This paper presents a first step towards this direction by introducing a web-based self-service tool for the users to configure their analysis and visualize the results. A second future step will be to focus on the extensions of the analyses algorithms that can be used within the tool.

ACKNOWLEDGEMENT

We would like to thank our colleagues in the hardware teams of the Beam Instrumentation group for their input of new features needed for OAF. We would also like to thank our colleagues in the Controls Group for providing the infrastructure that OAF requires to run.

REFERENCES

[1] S. Jackson, C. Roderick, and C. Zamantzas, “A Framework for Off-line Verification of Beam Instrumentation Systems at CERN”, in Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13), San Francisco, CA, USA, Oct. 2013, paper MOPPC139, pp. 435-438.

[2] B. Kolad, J-J. Gras, S. Jackson, and S. B. Pedersen, “The CERN Beam Instrumentation Group Offline Analysis Framework”, in Proc. 5th Int. Beam Instrumentation Conf. (IBIC'16), Barcelona, Spain, Sep. 2016, pp. 449-452. doi:10.18429/JACoW-IBIC2016-TUPG45

[3] J. P. Wozniak and C. Roderick, “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”, presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA163.

[4] django - The web framework for perfectionists with deadlines, https://www.djangoproject.com/

[5] G. Anders et al., “LHC bunch current normalisation for the April-May 2010 luminosity calibration measurements.”, February ‘11, https://cds.cern.ch/record/1325370