Automated data quality monitoring of the LHCb Vertex Locator

L. Bel¹, A.Ch. Crocombe², M. Gersabeck³, A. Pearce¹, M. Majewski⁴ and T. Szumlak⁴

¹ Nikhef National Institute (NL),
² University of Warwick (BG),
³ University of Manchester (GB),
⁴ AGH – UST Krakow (PL)

E:mail: szumlak@agh.edu.pl

Abstract. The LHCb Vertex Locator (VELO) is a silicon strip semiconductor detector operating at just 8mm distance to the LHC beams. Its 172,000 strips are read at a frequency of 1.1 MHz and processed by off-detector FPGAs followed by a PC cluster that reduces the event rate to about 10 kHz. During the second run of the LHC, which lasts from 2015 until 2018, the detector performance will undergo continued change due to radiation damage effects. This necessitates a detailed monitoring of the data quality to avoid adverse effects on the physics analysis performance. The VELO monitoring infrastructure has been re-designed compared to the first run of the LHC when it was based on manual checks. The new system is based around an automatic analysis framework, which monitors the performance of new data as well as long-term trends and using dedicated algorithms flags issues whenever they arise. The new analysis framework then analyses the plots that are produced by these algorithms. One of its tasks is to perform custom comparisons between the newly processed data and that from reference runs. The most-likely scenario in which this analysis would identify an issue is the parameters of the readout electronics no longer being optimal and requiring retuning. The data of the monitoring plots can be reduced further, e.g. by evaluating averages, and these quantities are input to long-term trending. This is used to detect slow variation of quantities, which are not detectable by the comparison of two nearby runs. Such gradual change is what is expected due to radiation damage effects. It is essential to detect these changes early such that measures can be taken, e.g. adjustments of the operating voltage, to prevent any impact on the quality of high-level quantities and thus on physics analyses. The plots as well as the analysis results and trends are made available through graphical user interfaces (GUIs). These GUIs are dynamically configured by a single configuration that determines the choice and arrangement of plots and trends and ensures a common look and feel.

1. Introduction

The LHCb Vertex Locator (VELO) [1,2] is the most precise position sensitive detector currently operating at LHC. VELO is a critical part of the LHCb High Level Trigger (HLT), thus, its optimal performance is of the outmost importance for the whole LHCb spectrometer and has a direct impact on the quality of the physics results produced by the experiment. The VELO group has designed and implemented an innovative software platform which provides appropriate functionality to assert the best possible running conditions for the vertex detector and to monitor its performance. In this note selected features of this monitoring software are discussed. In the following section a description of the software
architecture is given. The core functionality of the monitoring part of the platform is described in Section 3. Both GUIs used to visualize and browse the monitored quantities are discussed in Sections 4 and 5 respectively. The paper is concluded by a short summary.

2. Monitoring software platform for VELO

In this Section the overall architecture of the monitoring platform is presented. For completeness, we begin with short description of the engine application, VETRA [3], that is used to process the raw data banks. The main output of VETRA consists of ROOT histograms that are subsequently fed to the monitoring application called Lovell. By splitting the data processing and monitoring we make the software more reliable and robust. The main assumption here is that the VETRA output is final and do not undergo any further processing and Lovell is used only for visualization, data quality assessment and trending.

2.1. VETRA

This part of the VELO software platform, which is integrated within the official LHCb framework, uses directly the raw experimental data. Its core functionality comprises a bit-perfect emulation of the electronic acquisition board, designated as Tell1, that performs data compression and hit reconstruction. The resulting hit (or cluster) bank is subsequently used by the HLT for track reconstruction. In order to produce the emulated cluster bank non-zero suppressed (NZS) data are needed. These data are written out as a part of the calibration stream with a small frequency of approximately 0.1 kHz. Each stage of the NZS data processing (such pedestal subtraction or common mode suppression) can be monitored individually and appropriate key quantities can be extracted. The NZS data processing capability allows to evaluate the noise and perform the full calibration procedure. In addition, the zero suppressed (ZS) data that contains reconstructed hits can also be processed by the VETRA. This gives us access to such quantities as cluster rates or energy deposits in silicon. The calibration data are taken continuously during the normal operation of the LHCb detector. In order to process them efficiently a dedicated automatic infrastructure, based on cron utility, has been created. By using a time-based job scheduler we assert that all of the newly taken calibration data are processed and respective monitoring plots are created.

2.2. Lovell

The monitoring histograms produced by VETRA make the main input stream for the Lovell package. As mentioned in Section 2.1 these histograms are final and will not be processed in any way at this stage. Lovell is implemented in Python which is particularly well suited for building involved systems which must integrate diverse functionality such large data sets handling, building flexible graphical interfaces, interaction with relational data bases or providing convenient tools for web application design. A huge number of high quality modules available for Python allows a rapid development and do not require writing large amounts of code. Also, it aids in testing and debugging cycle considerably thanks to built-in utilities for agile programming.

The architecture of Lovell is schematically presented in Figure 1. The main functionality can be logically splitted into four parts. The first one is responsible for handling the input ROOT file and using its content it can dynamically configure the main GUI. Since the input histograms are ROOT objects they are difficult to handle and must be reformatted into matplotlib objects using tools from utilities module of Lovell. The core functionality of Lovell is implemented in the analysis module (see Section 3 for more detailed description) that performs VELO data quality assessment using features obtained by reducing the histogram data stream. These reduced data can also be used to create various trending plots that are vital for tracking the condition of the VELO detector. The trending information as well as

---

1 Because of the historical reasons the part of the experimental Data Acquisition system (DAQ) responsible for compressing the raw data stream and buffering is called Tell1 (Trigger Electronics Level 1).
2 The HLT output rate during the regular data taking amounts to approximately 12.5 kHz.
bookkeeping is stored in a dedicated database. Interaction with a user is handled via two specialized graphical interfaces which can be called detailed and simplified respectively. The first one is optimized for providing all available information obtained in the monitoring and calibration procedures. This feature is necessary for daily operation of the vertex detector and is the main tool for understanding potential problems with the data quality. The second one is a web browser application that allows to display selected histograms by a remote user without a necessity of connecting to the online processing cluster.

The choice of a particular technological stack used for Lovell development was mainly driven by features like stability, robustness and popularity (documentation). Using proven technologies decreased significantly risks related to the design and implementation of such novel system as Lovell. Visualization is facilitated by PyQT and matplotlib. The former is used for the Lovell GUI design and implementation whilst the latter is employed for graphic objects formatting and plotting. SQLite management system was used to handle the database needed for storing data that evolves with time and bookkeeping. Since this is not a typical client-server engine it can be embedded in the application and simplifies the overall architecture. Finally, for the simplified GUI implementation the Flask web micro framework has been chosen.

3. Analysis Framework
The analysis framework is the central part of the VELO offline monitoring infrastructure. Its purpose is to read in monitoring data, analyse their quality, and pass on the data themselves as well as the analysis outcome to the two GUIs via a common interface. In addition, the analysis outcome is stored in a central database, which serves as input for displaying trends of the various quantities.

The monitoring data are read in from ROOT files containing one and two-dimensional histograms, profiles, and graphs. Their content is translated into a python data format, which is provided to the GUIs as a return value of a central data access method. The GUIs request this input by providing the path and name of the object they want to display.

The graphics objects are then analysed in two ways. First, higher-level quantities, like mean or width of a distribution, are assessed. The quantities to be extracted from these objects are configured in a

Apart from the monitoring plots that are produced using the collision data it is possible to display the results from, so called, special analyses such as IV characteristic scan that is performed to check the properties of the VELO sensors.
central configuration file. Second, these objects are compared to the corresponding ones in a reference file. This comparison is quantified by means of statistical tests, the details of which are again defined in a configuration file. Both the output of the individual analysis and that of the comparison are stored in an analysis database as well as provided to the GUIs as additional information.

In addition to displaying the analysis information in the GUIs it is used as input for long-term trends, which are obtained via database queries. These trends are available both for the quantities obtained from individual plots, e.g. the average noise level as a function of time, and for the comparison output, e.g. the evolution of the matching quality with the reference for a particular monitoring output. These trends are provided to the GUIs on request via a similar interface to the other graphics objects.

Finally, in addition to the regular monitoring of offline data, the analysis framework also handles data from special analyses such as current-voltage scans of the sensors that are carried out periodically. These are handled on an individual basis as their input data structure varies significantly depending on the type of analysis, but the interface to the GUIs is very similar to that for offline monitoring data to minimise the need to customisation in the GUIs.

4. Graphical User Interface (GUI) and selected views
A user interaction with Lovell is handled via two GUIs. In this section the one that can be run locally on the online LHCb computing cluster is discussed whilst the web GUI is described in Section 5. The main purpose of providing a GUI is to allow an operator to browse the results of respective monitoring and analysis algorithms in an efficient and convenient way. These results are provided as histograms representing a single monitored quantity, summary plots or trending plots. In this way any problem flagged by central shifters can quickly be followed and understood. In the following text a selected analyses that are presented by the local GUI are shortly discussed.

4.1. Pedestal subtracted data monitoring
Pedestal subtraction is one of the most critical steps of the NZS data processing and hit reconstruction. If this procedure is not performed properly the output cluster bank may either be overflown with fake hits (if the pedestals are underestimated) or the reconstruction efficiency may significantly drop (if the pedestal values are too high). Pedestal subtraction is monitored for each sensor individually and both one and two dimension plots are created. The former contains the mean values of the signal measured in each channel after pedestal correction plotted against respective channel number whilst the latter one saves the distribution of pedestal subtracted data for each readout channel. Also, a summary plot of the mean values for the whole VELO is prepared which gives immediately information on the number of channels where the correction is not effective. An example view regarding the pedestal correction available in GUI is presented in Figure 2.

4.2. Tell1 processing parameters
The quality of the VELO data is directly linked with its calibration which, in turn, is performed using the NZS data stream taken at the end of each fill (when proton beams no longer collide). Such data samples are often referred to as noise runs. The calibration procedure, implemented in VETRA, is then executed and respective calibration constants are calculated. Since we require approximately one million parameters to operate the VELO a task to monitor and analyse them is quite challenging. A dedicated module for calibration data visualization is a part of Lovell and allows to create graphical objects representing the most important calibration parameters such pedestals or clusterisation thresholds used for hit reconstruction algorithm. It is also foreseen that analysis framework will be extended to add a module for automatic calibration data assessment. This new functionality would check if the currently used calibration is still valid and issue a request for evaluation of a new set of calibration constants if necessary.
4.3. Clusters monitoring

The main output data (ZS stream) produced by the vertex detector contains the cluster bank. Clusters are objects representing the effects of the charged particles interactions with the VELO silicon sensors. They contain information on particle’s position (expressed in terms of strip number), deposited energy and number of readout channels that contribute to a given cluster (cluster size). Detailed monitoring of clusters is vital since they are used for track reconstruction by the LHCb high level trigger. Lovell offers access to a diverse information pertaining to cluster reconstruction performance including among others occupancy plots (evaluated per sensor and per readout channel), Landau distributions of energy deposits and cluster multiplicity distributions. These plots are very important for daily operation of the VELO during collision data taking and allow a quick and robust assessment of the detector performance.

4.4. IV scans

Proton-proton collisions at LHC energies result in a very harsh hadronic environment which induces significant radiation damage in silicon that leads to a drop in charge collection efficiency and larger noise. Both effects impact severely the performance of the VELO and need to be carefully measured and monitored. One of such analyses rely on measuring the leakage current as a function of reverse bias voltage. The results can be used to assess the general condition of the sensors and tune the bias voltage that is applied for each sensor during data taking. Lovell is instrumented with a proper module that can access the IV scan data and display them in the GUI.

5. Web GUI

The Qt-based GUI discussed in the previous section is accessible to users working in the LHCb control room, where the PCs are connected to the online network, and to external users via X11 forwarding over...
an SSH connection. The latter access pattern can be frustrating to use, due to latency over the Internet causing lag in GUI interactions, or intermittent connections prematurely stopping the SSH session entirely. To overcome these problems, a second GUI is made available over HTTP. It is accessible to authenticated users on all systems, including mobile devices such as smartphones and tablets, and has the added benefit of allowing users to share URLs to specific pages and plots, in case a detector problem is found and requires discussion.

The architecture of the web GUI consists of two relatively well-separated components: a frontend, comprised of webpages that the user (referred to here as the ‘client’) interacts with, and a backend, which the frontend communicates with asynchronously via AJAX requests. The layout of each page in the frontend, such as the section structure and what plots to display under what sections, is defined in a GUI-agnostic configuration, such that a visual similarity is maintained with the Qt-based GUI.

The backend presents a plot retrieval API through which any authenticated client can request plots. The frontend is then designed as ‘just another client’, rather than being tightly coupled to the workings of the backend. This allows for the backend implementation to be changed without considering whether the frontend is affected, given that the API isn’t modified.

To reduce the workload on the web server process, plot retrieval and manipulation is offloaded to a set of worker processes. When a client requests a plot from a specific data-taking period, the request is pushed onto a queue which is monitored by the workers. An available worker pops a job off of the queue, and is then responsible for loading the requested ROOT file, retrieving a specific plot from it, and pushing a GUI-agnostic format of the plot onto an in-memory cache. (Both the queue and the result cache are backed by Redis, with the worker queue logic handled by the rq Python package.) This decoupling of the plot retrieval service from the web server increases the robustness of the latter, as a failure in the plot retrieval only affects worker processes, which can easily be respawned. In addition, such a decoupling allows for workers to be distributed across several computing nodes, and allows for the number of workers to be simply scaled to cope with the demand on the web server. As the typical monitoring page can contain five to ten plots, setting the number of workers to a similar magnitude allows for all plots on a page to be retrieved at once, improving the perceived responsiveness of the frontend.

6. Summary
A novel monitoring and analysis software platform, Lovell, employed by the LHCb VELO group for daily operation of the vertex detector has been presented. Lovell constitute a central part of the automatized infrastructure that is used to assert the proper calibration and highest possible data quality produced by the detector. User interaction with the application is provided via two GUIs, one of which is a web service that can be run outside of the LHCb online cluster. This is quite innovative approach for the monitoring of the data collected by a high energy physics experiment.

Acknowledgements

We acknowledge support from CERN and LHCb and from the national agencies: FOM and NWO (The Netherlands); MNiSW and NCN (Poland); STFC (United Kingdom).

References
[1] LHCb Collaboration, LHCb Detector Performance, Int. J. Mod. Phys. A 30 (2015)
[2] LHCb Velo Group, Performance of the LHCb Vetrax Locator, J. Instrum. 9 (2014) P09007
[3] Szumlak T., VETRA: Offline analysis and monitoring software platform for the LHCb vertex locator, J.Phys.Conf.Ser. 219 (2010)