LHCb data quality monitoring

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 092027 doi:10.1088/1742-6596/898/9/092027

LHCb data quality monitoring

M Adinolfi[3], F Archilli[2], W Baldini [4], A Baranov[1], D Derkach[1,6], A Panin[1], A Pearce[5] and A Ustyuzhanin[1,6]

[1] Yandex School of Data Analysis, Russia
[2] Nikhef National institute for subatomic physics, Netherlands
[3] University of Bristol, UK
[4] Universita di Ferrara & INFN, Italy
[5] CERN, Switzerland
[6] Higher School of Economics, Russia

E-mail: a.baranov@cern.ch

Abstract. Data quality monitoring, DQM, is crucial in a high-energy physics experiment to ensure the correct functioning of the experimental apparatus during the data taking. DQM at LHCb is carried out in two phases. The first one is performed on-site, in real time, using unprocessed data directly from the LHCb detector, while the second, also performed on-site, requires the reconstruction of the data selected by the LHCb trigger system and occurs later. For the LHC Run II data taking the LHCb collaboration has re-engineered the DQM protocols and the DQM graphical interface, moving the latter to a web-based monitoring system, called Monet, thus allowing researchers to perform the second phase off-site. In order to support the operator's task, Monet is also equipped with an automated, fully configurable alarm system, thus allowing its use not only for DQM purposes, but also to track and assess the quality of LHCb software and simulation over time.

1. Introduction

LHCb [4] is an experiment running at the LHC designed for the study of particles containing the charm and beauty quarks. Its detector consists of several sub-systems (sub-detectors). To achieve good detector performance, continuous detector monitoring and data quality assessment is performed. The most valuable assessment of data is visual inspection, performed by research scientists operating on a rotating shifts schedule (Data Quality, or DQ Shifter). It is therefore needed to have a clear and efficient tool to present the data to be assessed to the operator

2. LHCb DQ workflow

Data from consecutive interactions collected at LHCb are grouped together in runs. Each run is first processed by the LHCb trigger system [6]. A small subset of the data selected by the trigger is fully-reconstructed on the LHCb Online computing farm. The reconstruction produces sets of histograms which allow the (sub-)detector performance to be assessed. These histograms are presented by the Data Quality Monitoring(DQM) software to the DQ shifter who decides whether the run is suitable for physics analysis or not, by comparing it to a reference run previously set by experts. The same software allows the histograms to be posted with comments to an electronic logbook if a decision cannot be reached. This workflow is shown in Figure 1. The Presenter, a software package previously used in DQM shifts, was based on dedicated custom

C++ code and X Window System. Here we present "Monet" which is a python based web application that supersedes the Presenter.

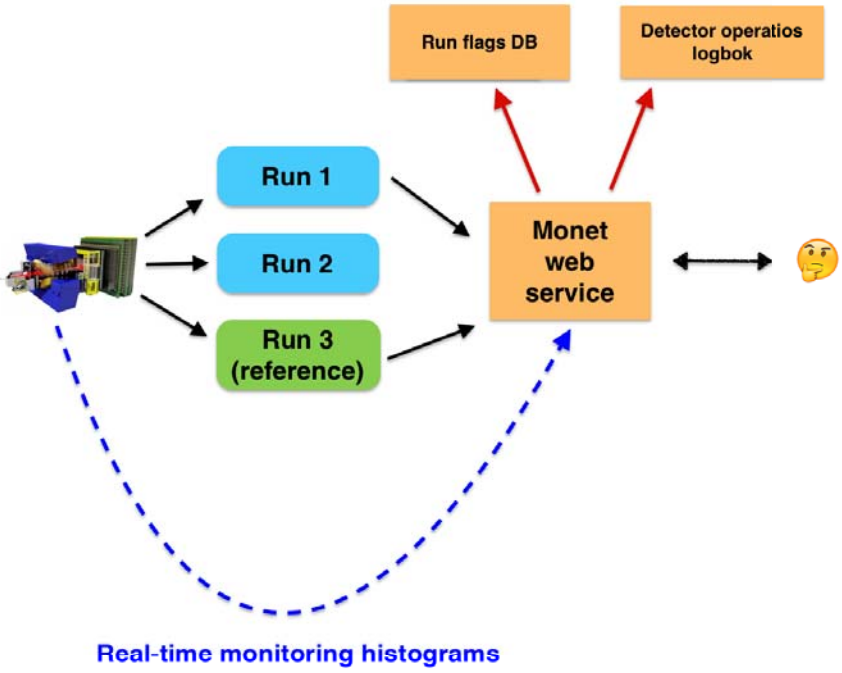

Figure 1. Schematic Data Quality workflow

3. DQM software upgrade

The presenter, being based on X Widnow System, works slowly from outside the LHCb control room. Its C++ implementation is also problematic to support and extend. The upgrade to a web application has provided a much faster responsiveness outside the control room. The same software can be applied in other areas such as verification of simulated data. Using Python as the primary language allows the usage of rich set of libraries provided in the large ecosystem of third-party Python packages. This simplifies both development, as common functionality has already been implemented elsewhere, and maintainability, as the size of the required LHCb-specific code is reduced. Using Python in particular allows for simple integration of machine learning libraries, which can be used for automatic anomaly detection, which will be discussed in Section 4.

The software stack used in transition is shown in Figure 2. Web application logic is implemented using the Flask [2] framework. Database interfacing is based on the SQLAlchemy [3] library. Both Flask and SQLAlchemy were chosen due to their simplicity and wide adoption. For plotting we evaluated the D3 [1] and Bokeh [7] libraries. Both libraries provide interactive plots in web browsers. Bokeh was preferred due to its pythonic interface, which does not require writing any additional javascript code.

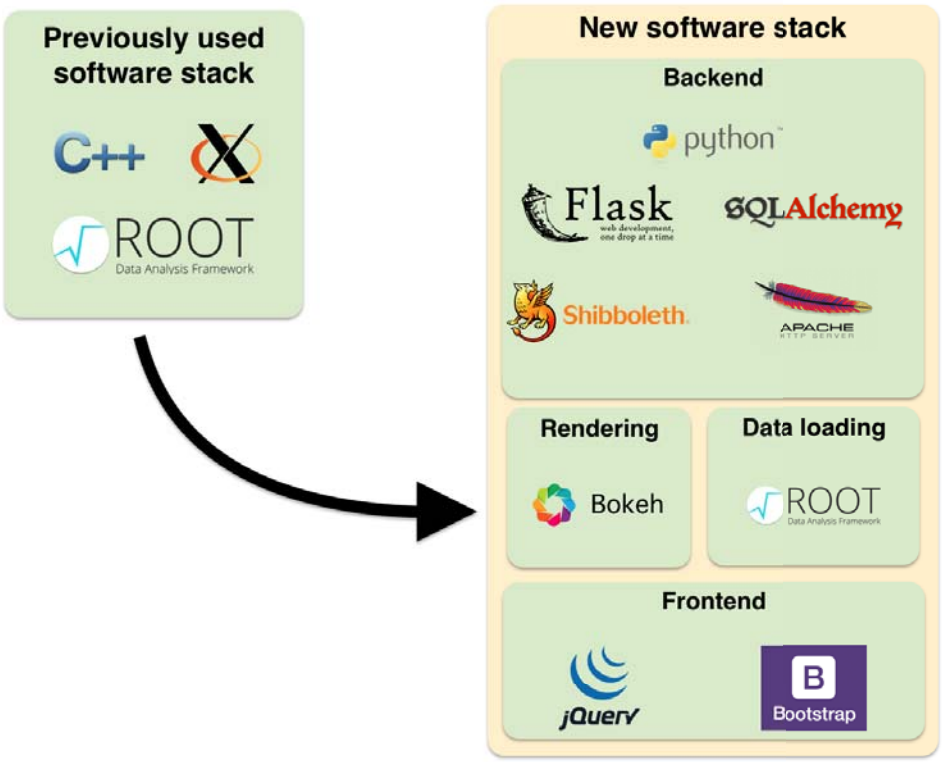

Figure 2. DQM software upgrade

4. Roboshifter - automatic problem detection

Together with its histograms the DQ shifter decision is stored for each run. With such history of shifter decisions one can construct datasets of (run histograms, run flag) pairs. One can construct a vector of Kolmogorov-Smirnov(KS) distances between the histograms and their references. With such a vector and flag for each run one can use machine learning to build a model which predicts the run flag based on its KS-distances vector.

This is being achieved with AdaBoost BDT [8] with shallow trees of depth 1, so that each tree makes a decision based on one KS-distance. This algorithm can be used to predict the probability of each run to be good or bad.

In AdaBoost BDT, decisions made by each tree are summed with weights, representing the importance of each tree, to form a final decision - a probability that a run is bad. As each tree corresponds to a single histogram, it is possible to compute, for each histogram, its contribution to the probability of the run being bad. Histograms with the highest contributions can be presented to DQ shifter as potentially problematic ones.

This algorithm is implemented in Monet using scikit-learn [5] and the list of potentially problematic histograms is shown to the DQ shifter through the web interface, as shown in Figure 3.

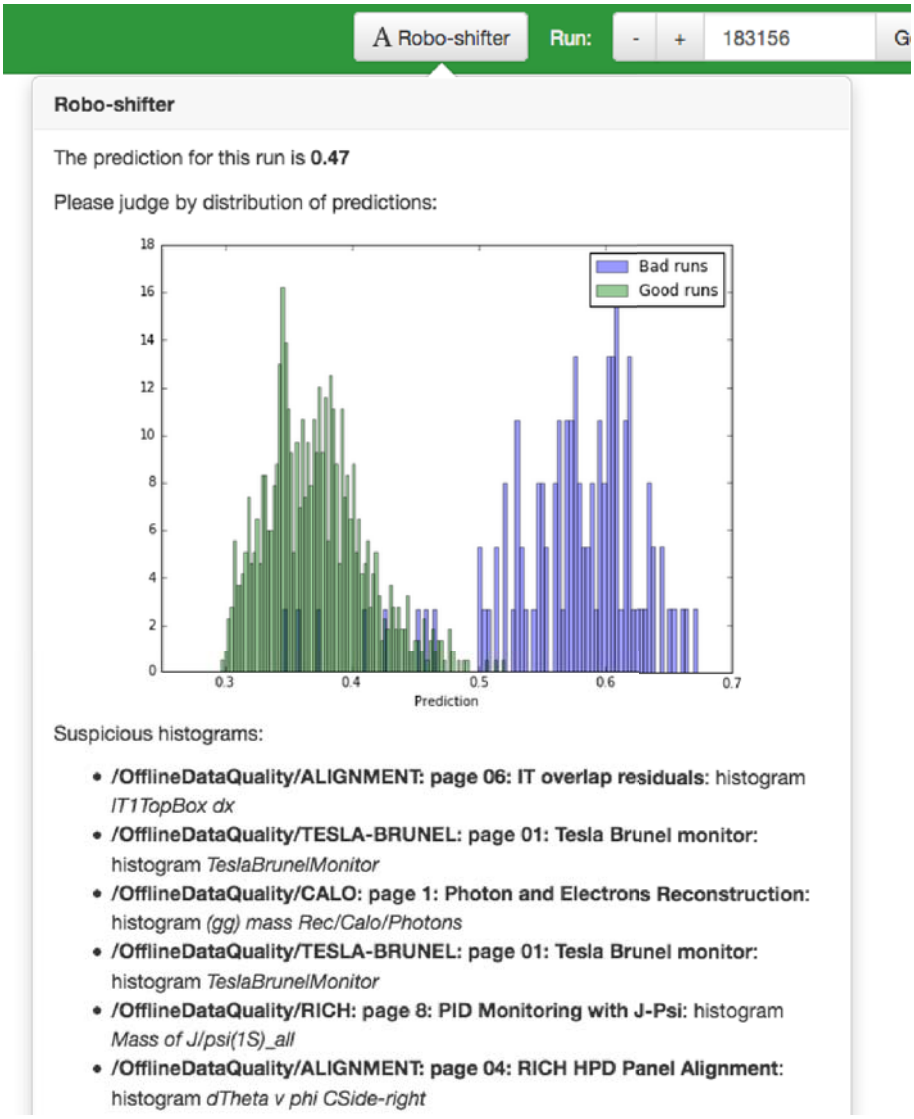

Figure 3. Robo-shifter interface

5. Summary

An upgraded DQ monitoring workflow for the LHCb experiment in Run-II is presented. This supersedes the old, X Window System based, user interface. As a result, the usability of the application outside of the LHCb control room is improved. The new DQM software can also be applied to the verification of simulated data. A user-friendly interface allows users to interact with different local storages and flag runs in the grid-based run-quality data base. In addition, an automatic problem detection system, which predicts the likelihood of a run to be accepted as good, is provided to the DQ shifter as a useful tool in reaching his conclusion.

Monet future roadmap consists of two major milestones. First, it is planned to commission Monet for LHCb-wide data monitoring and to expand data monitoring itself(i.e. by adding simulation data quality). Second, Monet is gradually evolving to become less dependent on LHCb software stack and becoming more general-purpose tool which will make open-sourcing it in the future possible.

References

[1] *D3 library https://d3js.org/.*
[2] *Flask microframework http://flask.pocoo.org/.*
[3] *SQLAlchemy http://www.sqlalchemy.org/.*
[4] Alves A et al. *The LHCb detector at the LHC*, 2008.
[5] Pedregosa F et al. *Scikit-learn: Machine Learning in Python*, 2011.
[6] Aaij R et al. *The LHCb trigger and its performance in 2011*, 2013.
[7] Bokeh Development Team. *Bokeh: Python library for interactive visualization*, 2014.
[8] Freund Y and Schapire R. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.