A Historic Data Quality Monitor (HDQM) tool for the CMS Tracker Detector

Daniel Duggan[a], Francesco Fiori[b], Hassan Halil[c], Tomas Hreus[d], Aristotelis Kyriakis[e], Dimitrios Loukas[e], Alkis Papadopoulos[e], Alessandro Rosi[f], Ahmed Sami Uridah[c]

[a]Fermi National Accelarator Laboratory (USA), Scuola Normale Superiore and INFN of Pisa (IT), [c]National University of Sciences and Technology (PK), [d]Universitaet Zuerich (CH), [e]National Center for Scientific Research Demokritos (GR), [f]Universita e INFN, Perugia (IT)

Introduction

Monitoring the time evolution of sensitive quantities is fundamental to LHC experiments. It permits keeping control on data quality during LHC running and also effectively checking the influence on data of any detector calibration performed during the year. The Historic Data Quality Monitor (HDQM) of the CMS experiment is a framework developed by the Tracker group of the CMS collaboration that permits a web-based monitoring of the evolution of measurements (i.e. S/N ratio, cluster size) in the Tracker Silicon micro-strip and pixel detectors.

Objectives-Architecture

The main goal of the tool is to provide web-based dynamic and interactive trend plots to facilitate the experts in monitoring the evolution of various Tracker DQM quantities (i.e. cluster size, S/N ratio). The visualization is performed through a distributed three-tier architectural model. The model consists of the lower database tier where the data are stored in the form of ROOT [1] objects, the middle logic tier that is based on a set of python [2] scripts (data harvesting) that read the data and generate JSON [3] type files for the trend of each input quantity, and the higher presentation client tier that is a web portal which consists of a front-end HTML [4] Module and a back-end JavaScript [5] Module (Figure 1). This last tier is responsible for fetching JSON data from the middle logic tier to dynamically create the trend plots.

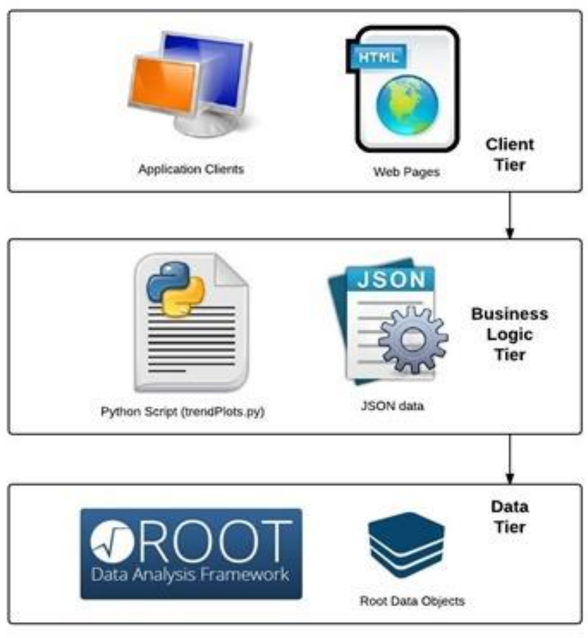

Figure 1: The three-tier architectural model of HDQM that consists of the higher presentation tier, the middle logic tier and a lower database tier.

Data Harvesting

There is a main python script, which performs the data harvesting and generates the trend values as a function of the run number. This script retrieves the requested information from the central Data Quality Monitor (DQM) system of the CMS experiment and saves a set of JSON type files that are used for the trend visualization. Figure 2 gives a example of such JSON type file.

The harvesting script accepts a set of options:


```
{
   "clusterStoN_ONTk_TIB_mpv": [
        {
            "run": 295315,
            "ymax": 0,
            "y": 18.117590797125597,
            "x": 1.0,
            "ymin": 0,
            "hTitle": "S/N (mpv)",
            "yErr": 0.003118155942316216,
            "yTitle": "S/N (mpv)"
        },
        {
            "run": 295318,
            "ymax": 0,
            "y": 18.22742050318757,
            "x": 2.0,
            "ymin": 0,
            "hTitle": "S/N (mpv)",
            "yErr": 0.0005641553197393178,
            "yTitle": "S/N (mpv)"
        },
.............
}
```


Figure 2: Example of a JSON type file containing the trend information.

- Data type (collision data or cosmic data),
- Data reconstruction type (raw data or reconstructed data),
- Data quality filter (data that fulfill specific quality specifications) and
- Configuration files defining the information need to be extracted from the data (calling predefined metrics methods) and the names of the plots to be created.

More than one configuration files can be passed as argument to the script. The available metrics are defined within two python files: the ”basic” metrics (i.e. mean value, bin content, integral) and the more complex metrics with fitting capabilities (i.e. Gaussian, Landau). The most common metrics methods and their description are presented in Table 1.

Table 1: Description of the most frequently used methods for extracting information from the DQM root files

File Name	Method Name	Method Description
basic.py	basic.Mean()	Mean value of 1D histogram
	basic.Count()	Total number of entries
	basic.BinCount(i_{th})	Single bin content
	basic.BinsCount(i_{th})	Sum of all the bins from the i_{th} to the last
	basic.MaxBin()	Bin with the highest value
	basic.MeanY()	Mean value in vertical axis
	basic.FractionInBinArray([x],[y])	Ratio between x′th and y′th bin
fits.py	fits.FlatLine(0,0.,192.,(30.))	flat line fit
	fits.Gaussian(0,15.,30.,(25.,5.))	Gaussian fit
	fits.Landau(Out,Min,Max,(MPV_ini, Sigma_ini))	Landau fit (Out: 0 → MPV, 1 → Sigma)

When the script is executed, the configuration files are parsed and from them a list of trend objects is created. Then a query is sent to the central DQM in order to check if all the runs are present satisfying the requested filter and a final list of runs is created. Once this is done the harvesting can start. For each run the corresponding ROOT file on the central DQM server is opened (via web protocol, without download) and a loop over the different trend objects is executed. For each trend object the relative histogram is loaded in the memory and the corresponding metric is executed. When all the runs have been processed, the script builds the output files and for each trend object the results are dumped in a JSON type file.

Web Interface

The JSON type files that contain the trend object information are fed to the web interface tool for visualization. This tools provides controls for dataset selection, filtering of data and navigation. It uses drop down menus where the user can select: the year of the data taking period, e.g. 2017, the Data type e.g. Collision data or Cosmics data, the CMS tracker data taking mode and the Tracker Subsystem e.g. Pixel, Silicon Strips. A set of check boxes help the user to apply various filters concerning the number of runs that will be displayed, e.g. all runs from the beginning of the run period or the latest runs or specific run sets. Figure 3 shows a possible user selection.

Figure 3: Main page of the HDQM Web Interface with a possible user selection related to the Silicon Strip Tracker Detector.

The graphical representation of the data is done using the Highcharts [6] API. Figure 4 shows an example of a plot displaying the Signal over Noise of the Tracker Inner Barrel layers. The white and grey zones of the runs that belong to the same LHC Fill number and the Run duration in the right vertical axis are also displayed. For each point a pop up box with all the relevant information is also available. All the relevant plot names can be also seen. Figure 5 shows two trend plots from the Pixel subsystem, one of which is a 2D plot of the number of cluster in the Tracker Subsystem as a function of the corresponding number of clusters in the Pixel subsystem for each run.

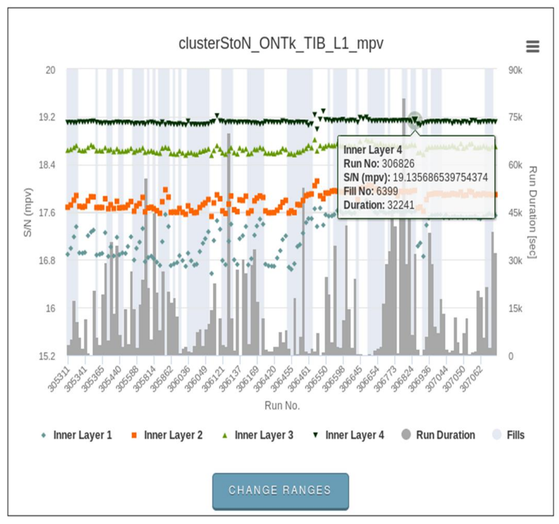

Figure 4: Example of a plot displaying the Signal over Noise of the Tracker Inner Barrel layers. Runs that belong to the same LHC Fill are shown as while and grey zones while the Run duration in seconds is displayed in the right vertical axis.

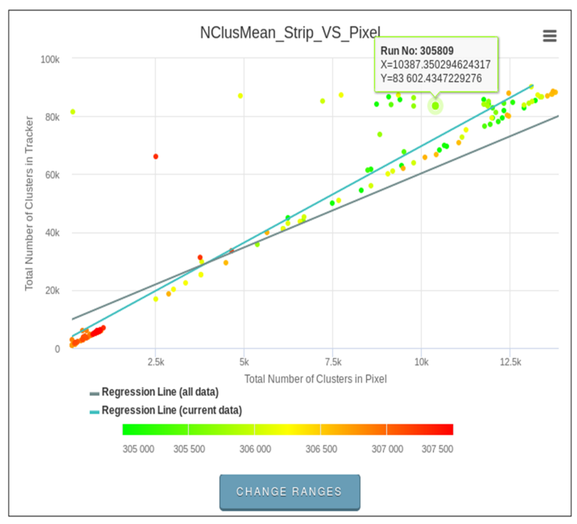

Figure 5: Example of two plots from Pixel detector displaying a 2D plot of the Total Number o f Clusters in Silicon Strip Tracker versus the Total Number o f Clusters in Pixel.

The Web interface tool is based on a static HTML document structure while all the dynamic contents are created with JavaScript. Due to the amount of data to be displayed on a single page and to avoid freezing, data loading and rendering is done asynchronously. Figure 6 shows the Flow Chart of the HDQM Web Interface tool explaining the navigation through the various classes.

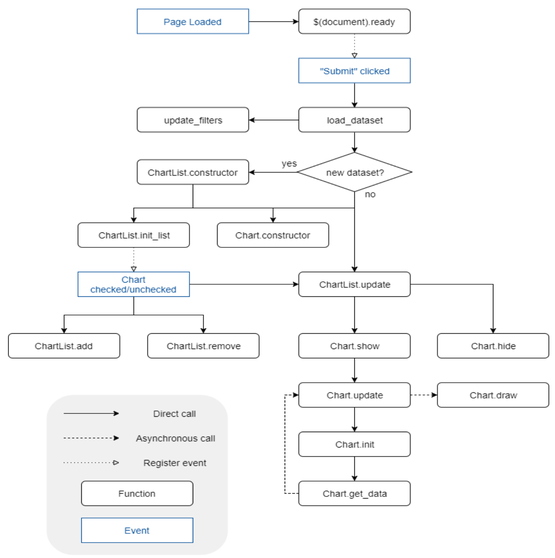

Figure 6: Flow Chart graph of the HDQM Web Interface explaining the navigation through the various classes.

Future Plans

The tool is used centrally in CMS and is under continuous development in order to fulfill any possible new requirement like:

- Use of a DB scheme instead of a collection of JSON files,
- Bin width to be proportional to the run duration instead to have this information in the right vertical axis,
- Possibility to fit interactively the trends,
- Extent its use to the other CMS sub-detector systems like Electromagnetic Calorimeter (ECAL), Hadronic Calorimeter (HCAL) or Muon system.

References

[1] R. Brun, F. Rademakers, “ROOT An object oriented data analysis framework”, Nuclear Instruments and Methods in Physics **A389 (1997) 81–86,** doi:{10.1016/S0168-9002(97)00048-X}.
[2] P. S. Foundation, “Python developers guide”, (https://www.python.org/dev/).
[3] H. Andrews, A. Wright, “JSON schema: A media type for describing JSON documents”, (http://json-schema.org/latest/json-schema-core.html).
[4] L. Hunt, “Html5 reference: The syntax, vocabulary and apis of html5”, (https://dev.w3.org/html5/html-AUTHOR/).
[5] J. Duckett, “Javascript and jquery: Interactive front-end web development”, (http://javascriptbook.com/).
[6] Highsoft AS, “Highcharts, highstock and highmaps documentation”, (https://www.highcharts.com/).