

Report on DESY Summer Student Program

Alexander Chaushev

2011-08-23

Abstract

Abstract: The data quality monitoring (DQM) system and data harvesting procedures used by the DESY CMS group are outlined in this report. The purpose and mechanism of a script designed to aide the offline shifter in identifying runs which are ready to be certified are described. The method of data harvesting and histogram production is described and well as the role and function of a script used in the process. The usefulness and quality of both scripts is assessed and areas of potential improvement are discussed.

1 Introduction

The role of Data Quality Monitoring (DQM) is to validate the data coming from the CMS experiment. It is a complex and involved process which is required to process the large amount of data coming from the CMS experiment both in realtime as data is recorded and afterwards for detailed testing.

DQM has been setup to assure the quality of the data being distributed to the scientific community for analysis. From the recording and reconstruction of real data to certify the validity of the data being used by the wider physics community. The DQM process involves writing and developing of tools and procedures used to check the various detector components and maintaining them as well as managing the people who certify the data.

2 DQM Procedures

Data Quality Monitoring, or DQM, consists of a number of import procedures designed to check and test if all the elements of the detector are functioning correctly so that data used for physical analysis can be certified as reliable. Furthermore DQM is useful for ensuring that the detector is run as efficiently as possible by reducing the amount of data lost to detector errors. Part of a DQM workflow is discussed which begins with histogram creation and ends with the data certification.



Figure 1: DQM GUI - Data Quality Monitoring Graphical User Interface - This webserver is used to store the histograms which are obtained from data harvesting and are used to check the validity of the data. These histograms are available to anyone with CMS credentials.

2.1 Online DQM

The DQM process is roughly split into 'online' and 'offline' components. The 'online' components consist of high level filtering via the use of triggers as well as applications which generate histograms for the purpose of DQM in real time. Due to the large amount of data generated by the detector the data is split into groups and the histograms are calculated in different jobs. Once the jobs have been completed, by parallel processing over a large computer network, the resulting histograms are sent to 'storage manager proxy' server. This server then produces a stream of data which can be used by various DQM applications for further analyses and histogram generation.

This stream serves the raw data from which online DQM is based. The algorithms run on the stream typically check for a wide range of possible problems with the detector as well as providing basic information about the run. The data on the DQM GUI can be viewed in realtime so as to provide an instant check of the function of the detector and the quality of the data being produced. Once this process has been completed the data is backed up to tape. Part of this information is extracted and uploaded to 'Run Registry' for an 'automated certification summary'.

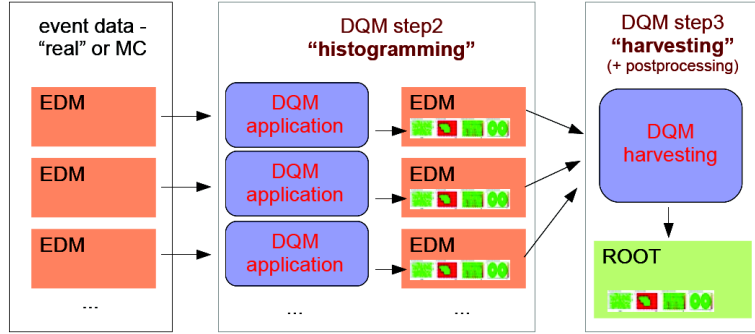


Figure 2: Data Harvesting Workflow - Detailed view of the histogram production process.

2.2 Data Harvesting

The offline process is based around histograms which are generated by jobs run on parallel on the data coming from the detector. Once the histograms have been created and uploaded to the 'storage manager' server they have to be added up together so that information from all the events is used to double check the data. This can be a complex process as not all histograms can be created by simple addition of the individual histogram components. A simple example of a quantity like this is an efficiency which requires a division as well.

2.3 Offline DQM

The offline component of the DQM process involves further processing and analysis of the information coming from the detector. This is typically done on a time scale of a few days after the data is initially recorded. Due to the asynchronous and parallel nature of the data processing it is not known when the processing will be completed. Once the various histograms have been uploaded to the 'DQM GUI' they are used by the offline shifter to certify the data as ready for 'sign off'. On the DQM GUI the histograms can be viewed by anyone in the physics community in possession of a computer and internet connection (and CMS credentials). This is important in order to ensure that the data is as well validated as is possible.

3 Scripts

Two scripts are detailed which are designed to be used in the some parts of the DQM process. The aim of the first script called the 'auto-run-checker' script is to check whether are new runs are available for certification. The second script is used in the harvesting process and is designed to replace a bash script whose

Figure 3: Run Registry - A webserver where runs coming from the CMS detector are registered. Basic information is kept such as the run number as well as the number of events. On the right hand side of the display there is a list of flags which are set by hand. These flags are basic indications of the operation of different components of the detector.

purpose is to run the data harvester on the correct data sets and to upload the histogram out to the DQM GUI.

3.1 Assisting the DQM Shifter in identifying runs

3.1.1 Aim

The role of the offline data shifter involves the certification of runs to verify their validity. The end goal of this task is to form a 'good-run list'. This list is used by those producing data sets further on down the line of physical analyses.

3.1.2 Requirements

In order to certify that the run is ready for 'sign-off' the following information about the run must be available to the shifter:

1. The name of the run, date taken and various other data along with alarm state 'flags' are present on run registry
2. A full set of DQM histograms coming from the 'offline' DQM, typically within a few days of the time of collection. These histograms comprise a complete analysis of the validity of the data required for offline certification.

Once these criteria have been met the run can be certified by the current shifter.

A script was developed by myself and Sarah Lindner in order to automate the process of checking that the various criteria have been met for a particular run.

The advantage of an automated procedure is that a script can check whether all data is present on the GUI and on Run Registry quickly and with little error - thus identifying very efficiently which runs are ready for certification. Doing this by hand is an iterative and time consuming process as it requires checking that a variety of conditions have been met for every run. An additional advantage of the script is the systematic way in which it checks all the databases thus ensuring that no data slips through the process.

3.1.3 Structure

The program queries the run registry server to obtain a list of all the run's which have two dataset entries. The double dataset entry implies that the run is at a stage where data has begun to be uploaded for it in the DQM GUI.

The program then queries the DQM GUI to verify that data corresponding to that entry has in fact been uploaded. An additional check is done to make sure the complete data set corresponding to each sub-system has been uploaded - each complete run has a total of . This is done by checking that the correct number of entries are present for each run; a total of no less than 27 entries.

Finally the data from each query is parsed and compared. The resulting list from the comparison is a list of run numbers which are ready to be certified.

3.1.4 How to Run

1. The run checking script has been setup to be run from any directory.
2. The first major point is to check that the proxy certificate has been correctly configured. The location of the proxy certificate is hard coded into the script and should be changed to reflect that the certificate may be elsewhere. A small script has been bundled together with the program which sets up the GRID Proxy via the 'voms-proxy-init' command.
3. The resulting output can either be in the form of a graphical window or as a file whose output directory can be specified. If the output is stored in a webserver it would be possible to access the list of available runs from any computer with a web browser.
4. Initially the script has been configured to run every 5 minutes. This is also hardcoded in the script and should be changed by hand.

Note that cmsenv must be setup so that the script can find the correct python libraries.

3.1.5 Functioning and Further Improvements

There are a few issues which affect the functioning of the script:

- It has been noted by Sarah and myself that the run registry server which is queried often does not respond to queries. Sufficient safety checks have

been done to ensure that the program does not crash but continues to run smoothly, however it's functioning can be made more reliable if additional run registry servers are added (if such exist) and queried in parallel. This can also be used as a check for the reliability of the server, though this is very unlikely to be an issue.

- Using a proxy to automate the GRID authentication procedure was an issue and I believe there may still be some bugs which need to be ironed out with the authentication procedure. Due to the lack of new runs in the last few weeks prior to the writing of this report it has been difficult to test that this is working as expected.

The script was thoroughly tested in earlier iterations when data was still available. This was done by producing a list of runs and checking this list with the offline shifter who used the manual procedure to identify runs which are ready. At this stage there is no evidence to suggest an issue with the fundamentals of the scripts - it produces a correct and complete list of runs which are ready for certification.

3.2 Running the harvester

3.2.1 Aim

The role of the CMS Harvesting script is to take data from the DBS (data book keeping system) and to produce the correct histograms which can then be uploaded to the DQM GUI. These histograms are then used to certify data for offline DQM. In the following section I detail a script which was written to automate the process identifying new data, running the CMS Harvesting script and uploading the resulting histograms to the GUI.

3.2.2 Requirements

1. The script must correctly identify which data sets have already been uploaded to the server. For this person a book keeping file is kept with the list of data set names which have been submitted.
2. New datasets must be submitted to the CMS Harvester which configures the files for CRAB. The script must then submit the CRAB jobs.
3. Once the CRAB jobs have completed the resulting histograms are then uploaded to the DQM GUI for use of the shifters. An external upload script is used for this.

3.2.3 Structure

The harvester script consists of a number of elements:

- First the DBS is queried to produce a list of runs which have been added.

- This data is parsed to select the relevant information - a list of dataset names, global tags, data types and CMSSW versions.
- Once this list is produced it is further parsed to separate all the relevant parts into a useful lists for the CMS Harvester script which is then compared with the book keeping file.
- The CMSSW version is configured for the data set. The Harvester is then called and the crab configuration files are produced.
- Crab is then run to submit all the data for processing.
- This process is then repeated for each CMSSW version which is present in the base directory (hard-coded into the script)
- Once the data is processed it is uploaded to the DQM GUI server.

3.2.4 How to Run

1. Change the base directory to point the directory where all the CMSSW versions are stored.
2. A book keeping text file should be added in the directory to keep track of which datasets have been processed. IF such a file has not been generated before run the script once without calling the harvester and by using the appropriate function calls to generate such a text file. Otherwise the script will attempt to submit all the datasets which match the query (currently hardcoded to check everything added to the database from the 1st of August 2011.
3. Once the script is run it will start the CMS Harvester which will then save the resulting histograms in a directory which is coded in the Harvester script. From there they can be uploaded using the 'Upload.sh' script or using a tool such as 'VisDQMUpload' or similar code.
4. Note the upload functionality has not been implemented yet. To fully automate the upload process it may be necessary to produce an upload book keeping text file for 'Upload.sh'. The tools in the script should allow one to do so quite easily. For 'VisDQMUpload' a list of names of files should be supplied. In addition it may be wise to remove the files once they have been uploaded.

3.2.5 Functioning and Further Improvements

There are two main ways in which the script needs to be improved:

- At the moment the script does not automatically upload data to the DQM GUI.

- The script can employ multithreading so that data is processed simultaneously on different cores and so that threads can be switched around while a particular process is waiting to receive data from the internet. An interesting extension of this which can be done is to check how the speed of the program varies, over a constant data set based, as a function of the maximum number of allowed threads. It is anticipated that the function will have a definite peak at a number of threads which is greater than the number of cores and from there will decrease.

At present the script is able to keep track of which datasets have been uploaded, start the harvester and run CRAB to successfully process the data.

4 Summary

The CMS Data Quality Monitoring system consists of several distinct systems. One particular workflow which begins with histogram creation (for online and offline DQM), proceeds through data harvesting and finally ends with data 'signoff' by the offline shifter is discussed.

Two scripts were built to function at various the certification and data harvesting stages of the DQM process. These scripts were designed to automate various aspects of the system which were iterative and time consuming to do manually. The scripts functioned successfully and produced the correct output. However, there is still some scope to improve them, to make them easier to use and more efficient.

References

- [1] L Tuura, A Meyer, I Segoni, G Della Ricca, "CMS data quality monitoring: systems and experiences", 2010 J. Phys.: Conf. Ser. 219 072020