European Organization for Nuclear Research

Development of a monitoring system for the DQMGUI in ElasticSearch and Kibana

Adrian Diaz

September 13, 2016

Duration:	June 27th - September 2nd 2016
First supervisor:	Broen van Besien
Second supervisor:	Marco Rovere
Department:	EP-CMG-CO

Abstract

The Data Quality Monitoring Graphical User Interface (DQMGUI) is the heart of the process of monitoring the quality of data in CMS. The health-status of the DQMGUI and its performances are constantly being monitored and stored in log files, which are subsequently parsed for errors and warnings. This very process allows human operators to act in case of problems. However, the monitoring infrastructure has been migrated to a CERN-hosted ElasticSearch engine in the last year. As a consequence, it is necessary to refactor the old monitoring system, adapting and extending it to be compliant with the new ElasticSearch-based one.

1 Introduction

1.1 Background

The DQMGUI is a web-based tool which allows users to access the data generated by the CMS detector. This process is possible due to the *ROOT* files created by CMS Software (CMSSW). These are the files that contain the histograms to be shown.. As a consequence of the new information generated by the CMS detector, new ROOT files are constantly being loaded in the GUI's database (also known as index), giving the user access to an enormous amount of histograms. As a result, the DQMGUI allows every user to consult any histogram, from whatever run, always and everywhere.

1.2 Objectives

The main objective of this project is the development and delivery of a monitoring system for the DQMGUI. In order to accomplish such goal, the following tasks have been defined:

1 **Analysis of the problem**: This task is focused on learning the most relevant tools that are expected to be used during the project as well as the context in which they are used.

2 **Analysis of the log message generation process**: Understanding the log message generation process carried out by the *Server Monitor*.

3 **Use of ElasticSearch and Kibana**: This task is focused on displaying the first streams of data collected from the host machines in a *Kibana* dashboard by using the metric system provided by LEMON and the technology of *ElasticSearch*.

4 **Publishing the dashboard**: As a result of the previous task, it is required to make available the information plotted for any CERN user, so the previously generated dashboard must be published.

5 **Documenting:** This project represents a new feature never implemented before by the DQM group. As a result, it is mandatory to document not only every step followed, but also every problem found, so this new technology could be easily used and improved in the future.

2 The maintaining log files generation process

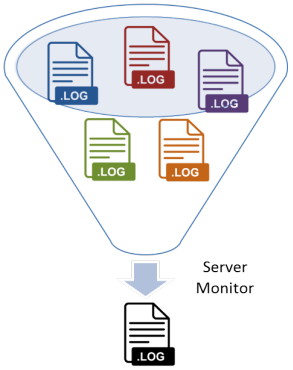
2.1 Overview

The first step to develop a monitoring system is to know where the information desired to be monitored is stored. This has been my first task. In the case of the DQMGUI, the data, which have been generated by the services (agents) of the DQMGUI, are stored in log files. These source log files contains all the information to be monitored: services queue size, errors generated, histograms rendering time, etc.

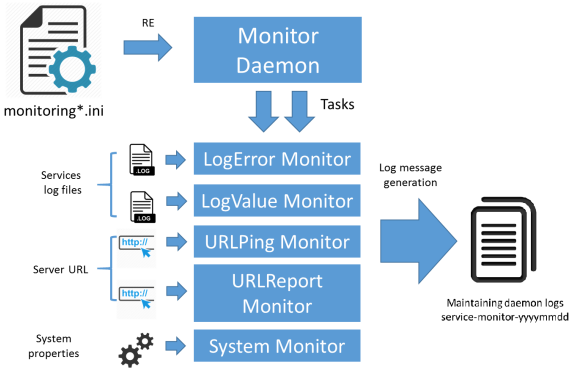
2.2 The Server Monitor

The Server Monitor is a *daemon* program. It is constantly checking for new changes in the directory where the source log files are stored, so every time a log file is modified, deleted, or created, the Server Monitor starts parsing and collecting the information included in such files.

The parsing process needs regular expressions (RE) to be defined. For every piece of information desired to be collected a RE must be specified, so the Server Monitor can gather every instance of such RE for later processing. Since there are different pieces of information contained in the original log files (i.e. they have different message structure) there are also different types of RE according to the semantics of the information they match: ERROR, REPORT or INFO. In addition, the Server Monitor is composed of several classes according to the source of the information it gathers. Some of this classes are the ones presented in Figure 1 (b). All the RE used by the Server Monitor are included in the configuration files *monitoring-dqm-[flavor]-web.ini* and *monitoring-dqm-[flavor]-agents.ini*, where *[flavor]* might be *dev*, *offline* or *relval* depending on the DQMGUI production server in which the RE are running.



(a) Black-box representation of the log files generation process



(b) White-box representantion of the log files generation process within the Server Monitor.

Figure 1: Log files generation process

Every message from the source log files that matches a certain RE is processed by the Server Monitor and written in a standardized version in a new log file. This final log file, also known as maintaining daemon log file, is the one which contains all the information that is going to be subsequently sent to ElasticSearch. Despite the different semantics of each message in the source log files, the ones stored in the maintaining daemon log have a well-defined structure. As a result, it is very simple to classify the messages according to their semantics. Furthermore, it is also very simple to extract the most important pieces of information from them, so they can be processed later by ElasticSearch and Kibana.

Maintaining daemon log files are updated every 30 seconds. As a consequence, the information cannot been sent by using a time step shorter than 30 seconds since no new data would be available to be monitored. Notice that if the time step chosen to collect data from the maintaining log files is higher than 60 seconds, then it is possible to lose some information depending on the treatment of the data. This issue will be explained in the following sections thoroughly.

2.3 ElasticSearch and Kibana

2.3.1 ElasticSearch

An official definition for ElasticSearch has never been provided by its development group. However, according to its creators a possible definition for ElasticSearch may be presented in the following lines.

"It is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine. Accessible through an extensive and elaborate API, Elasticsearch can power extremely fast searches that support your data discovery applications."

2.3.2 Kibana

According to the official documentation of ElasticSearch, Kibana is considered as an open source analytics and visualization platform designed to work with Elasticsearch. Kibana provides the tools needed not only for importing the data from ElasticSearch, but also for displaying such data in completely customizable and intuitive dashboards.

3 Sending data from hosts to ElasticSearch and Kiabana

3.1 Overview

After understanding how the Server Monitor carries out the log generation process, my second task was to figure out how data can be sent from hosts machines to ElasticSearch and Kibana. In this section I provide an explanation of the process.

3.2 The Lemon Metric Manager

The data collection process and sending procedure to ElasticSearch is carried out by the Lemon Metric Manager. The Lemon Metric Manager is a software tool developed at CERN capable of sending the information from the maintaining log files to the ElasticSearch database by using *sensors*, *metric classes* and *metrics*. A definition for the concepts previously mentioned is provided in the following lines:

- **Sensor:** Software object capable of collecting data from the host machines by using the REs provided to it as arguments.
- **Metric Class:** A software object composed of a single sensor and several fields. Once the information is gathered by the sensor, this very information is stored in the fields of the metric class. In the fields the type of the data stored is provided as well as a brief description and the units of the information stored.
- **Metric:** An instance of a metric class which contains the values/samples sent every 120 seconds (minimum time step) from the host.

Data from host machines is collected every two minutes. This time step is the minimum period of time that can be set up for data collection. As a consequence, a new problem is raised. Lets suppose that a collection of 100 numerical entries have been gathered in two minutes and they want to be displayed. Then, since two minutes is the minimum time step, just a single value can be displayed and, therefore, a loss of information is involved. Calculating the average value or selecting the minimum or the maximum value of the whole collection are possibilities that anyway leads to an inevitable loss of information. Finally, once the information has been stored in the metrics, the Lemon Metric Manager sends this information to ElasticSearch, so it can be easily indexed and accessed by any other user or service such as the Dashboard Manager.

3.3 The Dashboard Manager

The Dashboard manager is a web-based software tool that displays the information collected by metrics by using Kibana. A dashboard is composed of two main components: *filters* and *queries*. Filters provides the user a mechanism to import the metrics and the data they include into the dashboards. On the other hand, queries allows the user to select which metric to use from the imported ones (among other possible uses).

4 Results

4.1 A functional monitoring system

The main objective of this project has been achieved successfully. The delivered monitoring dashboard includes the properties mentioned below:

- A total of 24 histograms are currently being displayed in almost near real-time.
- Information from multiple sources (DQMGUI measurements, CPU usage, Network IO, etc) is currently being monitored and plotted.

The monitoring system for the DQMGUI is currently available at the following link:

`https://meter.cern.ch/public/_plugin/kibana/#/dashboard/temp/CMS::DQMMONITORINGSYSTEM`

4.2 An in-detail technical documentation

As a consequence of developing a monitoring system for the DQMGUI, an technical documentation about my project has been written. This documentation includes an in-detail description of how the Server Monitor works as well as how to upload data from a host machine to ElasticSearch and Kibana and how to plot such data in a dashboard, so that can be accessible to every CERN user. The documentation has been written in a Twiki webpage and it can be accessed by using the following link:

`https://twiki.cern.ch/twiki/bin/viewauth/CMS/DQMAdrian`

4.3 An automatic test suite for the DQMGUI

In addition to the main goal of this project, a test suite for the DQMGUI has been developed. This automatic test suite is meant to checking that the *import agent*, one of the most relevant DQMGUI agents, works as expected. This test suite has been programmed in *Python* and it includes several test mainly focused on the indexing process.

5 Conclusions

5.1 Overview

This chapter presents a general view of the project in which I have been working on. Although the following lines are mainly focused on giving a general explanation the monitoring process, the problems found are also explained in the last section of this chapter.

5.2 Review of the process

As shown in the beginning of this report, the main objective of this project was to delivery a completely functional monitoring system for the DQMGUI. According to the section of results it is possible to state that the main goal has been accomplished successfully. Therefore, in order to make this process repeatable the steps followed in the monitoring process are presented in the following points:

1. Target the lines or pieces of information desired to be monitored in the original log files.
2. Define the REs Design the REs that match such line/s and include them in the proper configuration files.
3. Create the metric/s or metric class/es required for gathering data from host machines.
4. Activate and install such metric/s and metric class/es. For this step it is recommendable to contact to a member of the *http group* (http group Bruno Moreira).
5. Include them in the DQMGUI monitoring system dashboard by using filters and queries.

The previous steps summarize the process I followed in the development of a monitoring system for the DQMGUI. If the reader desires to know more about this process, it is highly recommended to take a look to the documentation written about this project. The link can be found on the results section of this document.

5.3 Problems found

As in any technical project, some problems have been found during the development of the monitoring system. However, in this section it is only included the most important one which is still unsolved nowadays.

We have been experiencing some troubles while plotting data from DQMGUI services in the monitoring system. These troubles do not allow the users to see most of histograms displayed for a certain production server (*vocms0138*). After studying this issue thoroughly, one hypothesis has been considered. The problem seems to be in the communication among the Lemon Metric Manager and the ElasticSearch server, since Lemon might be not prepared to handle heavy loads of data in short periods of time (as it is done in the DQMGUI monitoring system). As it was mentioned before, this problem is currently unsolved since it involves several teams from different departments as the IT Group.

6 Acknowledgements

Finally, I would like to express my appreciation to all those who have helped me along this summer project. First of all, I would like to express my gratitude to my supervisors: Marco Rovere and Broen van Besien. They have provided me all the resources and answers I needed in this project, being available for me almost all the time. Finally, I would like to acknowledge with much appreciation the inestimable help of Bruno Moreira from the *http group*. Thank you all for your help and guidance during this project.