FROM DATA STORAGE TOWARDS DECISION MAKING: LHC TECHNICAL DATA INTEGRATION AND ANALYSIS

A. Marsili*, E.B. Holzer, A. Nordt, M. Sapinski, CERN, Geneva, Switzerland

Abstract

The monitoring of the beam conditions, equipment conditions and measurements from the beam instrumentation devices in CERN's Large Hadron Collider (LHC) produce more than 100 Gb/day of data. Such a big quantity of data is unprecedented in accelerator monitoring and new developments are needed to access, process, combine and analyse data from different equipments.

The Beam Loss Monitoring (BLM) system has been one of the most reliable equipments in the LHC during its 2010 run, issuing beam dumps when the detected losses were above the defined abort thresholds. Furthermore, the BLM system was able to detect and study unexpected losses, requiring intensive offline analysis. This article describes the techniques developed to: access the data produced ($\simeq$ 50000 values/s); access relevant system layout information; access, combine and display different machine data.

INTRODUCTION

Beam Losses

More than 3600 *Beam Loss Monitors (BLMs)* were installed around the LHC ring at expected loss locations, in order to measure the *beam losses* in the LHC. Losses are measured every 40 μs, and these basic integration windows are continuously combined into sliding windows called *integration intervals*, ranging from 40 μs to 84 s.

All the data are used in real time: if the losses in one integration interval exceed the predefined *abort thresholds*, the beam is removed from the LHC ring. Some of these data are stored in the LHC *databases (DB)*: this article will only describe the processing of these "offline" data.

The layout of the machine where the loss was produced is just as important as the measured loss itself, and must be understood.

Beam Loss Analysis Toolbox

This article describes the techniques used to access the different databases, to combine the values of losses with the positions of the BLMs and magnets and display them. Each of these functionalities corresponds to a *module*, and they are gathered in the Beam Loss Analysis *toolbox*.

Databases

The *Layout database* holds all the information about the positions of magnets, BLMs and collimators. Its structure is the one of a standard SQL database: several tables hold all related information in different columns.

Every second, for each monitor, one value per integration interval is saved in the *Measurement database* during one week. Some values are stored permanently in the *Logging database* after applying a filter: if the difference between two subsequent values is higher than a delta set per integration interval, the value is saved; if not, one value per minute is saved. The thresholds are logged "on change", and once per day if they don't change.

Both databases have the same structure: a *variable name* `BLM_EXPERT_NAME:LOSS_RSXX` is associated to each integration interval (`"XX"` is its number). It holds two SQL columns: *timestamps* and *values* (see Fig. 1). The structure for the thresholds is the same.

Languages

The database access tool was designed to allow the access, processing and display of data in a fast, user-friendly and easy way. The language used for database access at the time of development was PL/SQL. The Layout database is still accessed with PL/SQL. The main programing and data analysis framework at CERN is ROOT [3], a C++ framework providing classes and a C++ interpretor called CINT.

The choice was made to use Python (2.4) as the developing language (and not CINT), since a true interpreter was desirable. It links easily with ROOT (5.28) objects thanks to the pyROOT project.

DATABASE ACCESS TECHNIQUES

The `database` module accesses data by PL/SQL and Java, and returns them in a format compatible with the other modules, described in Fig. 1. A direct display of the data structure is shown in Fig. 4.

PL/SQL Access

The core of the SQL access class is a ROOT object called `TSQLServer` [3]. Queries such as the time window or variable names are automatically constructed and sent as strings. The data corresponding to the required variables are returned also as strings so the user does not have to change method depending on the data type. SQL access is still used for the Layout database access.

Java Access

The current way to access the measurement databases is through an *Application Programming Interface (API)* written in Java and provided by the data management section.

* amarsili@cern.ch

$$\left\{\begin{array}{cccc} \text{‘BLM\_1’} & \text{‘BLM\_2’} & \cdots & \text{‘BLM\_m’} \\ \left[\begin{array}{c} \\ (\text{‘}t_1\text{’}, v_1) \\ \\ (\text{‘}t_2\text{’}, v_2) \\ \\ \end{array}\right] & \left[\begin{array}{c} (\text{‘}t_3\text{’}, v_3) \\ \\ (\text{‘}t_4\text{’}, v_4) \\ \\ (\text{‘}t_5\text{’}, v_5) \end{array}\right] & \cdots & \left[\begin{array}{c} (\text{‘}t_6\text{’}, v_6) \\ \\ (\text{‘}t_7\text{’}, v_7) \\ \\ \end{array}\right] \end{array}\right\}$$

Figure 1: Original structure of the object for the Logging database, reflecting the DB structure. Dictionaries are represented between curly brackets “{ }”, lists between square brackets “[ ]” and tuples between parentheses “( )” The vertical spaces between 2-tuples indicate the absence of data between two times and show that all times are different.

It is able to be called from the command line, thus being more portable: the data are not returned inside one specific application. They are written to the shell standard output.

The API provides the user names and passwords for the Logging and Measurement databases, so the user does not have to. Most of the connection settings are written in a file called `configuration.properties` [4]. Rather than having the user edit this file separately (and for backward compatibility), this file is generated automatically.

The default Python `os` module provides the interaction with the unix shell from Python. The `os.popen` function executes a string as a command line, which allows easy manipulation. It returns a “file iterator”, the default object for reading text files. The use of this command line is further described in [4]. The data flow is summarized in Fig. 2.

Data Structure

The object holding the data reflects the structure of the database: dynamic *lists* of immutable *2–tuples* holding the timestamp as a string and the signal as a Python float (see Fig. 1). The mapping is provided by a *dictionary* associating a key (the variable name as a string) to the corresponding value (a list).

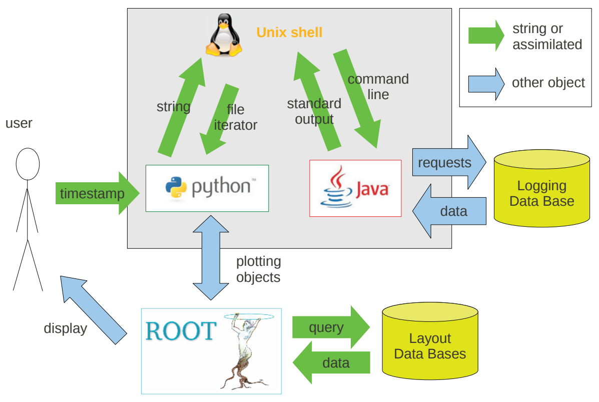

Figure 2: Data flow in the toolbox, from the user to the databases and back.

$$\left\{\begin{array}{ccc} \text{‘name\_of\_graph\_1’}, & \text{‘name\_of\_graph\_2’}, & \cdots \\ \left[\begin{array}{c} (dcum_1, v_1) \\ (dcum_2, v_2) \\ \vdots \end{array}\right], & \left[\begin{array}{c} (dcum_3, v_3) \\ (dcum_4, v_4) \\ \vdots \end{array}\right], & \cdots \end{array}\right\}$$

Figure 3: Final structure of “plotdict” object after data combining. $dcum_1$ is the position of BLM$_1$, and v_1 the signal of BLM$_1$ at the plotting time. See Fig. 5 for the corresponding graph.

DATA PROCESSING

Sorting BLMs

Currently, one of the main applications of the toolbox is to display a longitudinal loss profile of a section of the LHC at a requested timestamp (see Fig. 5). The names of the BLMs are not displayed, but they contain all relevant information: type of monitor (*Ionisation Chamber (IC)* or *Secondary Emission Monitor (SEM)*), associated beam (B1 or B2), injection or extraction line. The `process` module sorts the BLMs according to these criteria. The names are then mapped with the longitudinal position of the BLM on the LHC, called *DCUM*, which was obtained from the Layout database. The resulting structure is shown in Fig. 3.

Linear Interpolation

One of the features of the Logging database is that no BLM signal is recorded every second (see Fig. 1). A specific second requested by the user may not have been recorded. The choice was made to calculate a linear interpolation between the previous and next entries to estimate the signal at this second. The algorithm in the `process` module progressively checks the requested timestamp against a list of tuples. If the same time is found, the value (2^{nd} element of the tuple) is returned. If not, a linear interpolation is calculated using the next entry in the list.

The algorithm has to do $m \times n$ operations, where m is the number of monitors and n the number of tuples before the requested second. The processing time is dominated by the data access time in the case of the Logging database, when only one second is requested in a time range composed of a few tuples. It is not the case for the measurement database (one tuple per second). If all seconds in the time range are requested, the number of operations becomes $m \times \frac{N \cdot (N+1)}{2}$ where N is the total number of tuples (seconds): the processing time is dominating the access time. In this case, a different algorithm is used with only one index for all BLMs. The number of operations is then only m.

DISPLAY

The whole `display` module relies on ROOT objects, using `TGraph` for the plots and `TCanvas` for the display on screen [3]. Each type of plot has its own class, and they all

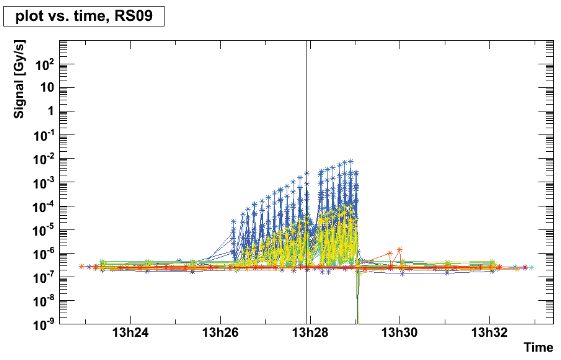

Figure 4: Plot of the data as received from DB, showing the time development of a loss (collimator scraping). Each of the requested variables are given versus time. Note that the values are not synchronised. The vertical black line corresponds to the requested second. The requested plot is shown in Fig. 5.

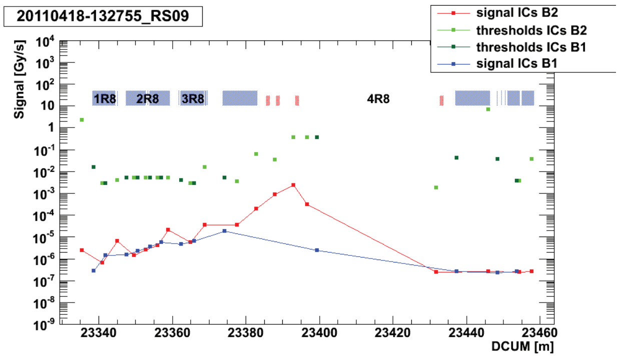

Figure 5: Final display, showing the longitudinal development of a loss (collimator scraping), and the positions of the magnets (blue boxes) and the collimators (red boxes) The vertical axis is the value of loss or threshold of one BLM, and the horizontal axis is its longitudinal position.

share methods to save the plot as seen on screen in a file with an automatically generated and meaningful name.

Time Plot

The time plots reflect the structure of the DB: each graph corresponds to one variable, with points only at recorded times (see Fig. 4). The colours are automatically selected to be as far away as possible from each other. The names are ordered so that similar variables will have similar colours.

Longitudinal Plot

For the longitudinal plot of the LHC, the positions of magnets and collimators are obtained from the Layout database. The value of the threshold is obtained directly from the Logging DB, and is mapped to the longitudinal position similarly to the BLM signals. The final result is shown in Fig. 5.

WRAPPING MODULE

All functions described previously are automatically executed when passing a timestamp and two values of longitudinal position to the loss analysis class in the wrapping module. In its basic behaviour, it:

- finds the corresponding BLMs, collimators and magnets with information from the Layout DB;
- gets the losses and thresholds for the relevant BLMs from the measurement or Logging DB;
- displays the losses versus time (see Fig. 4);
- combines all data to display the loss in the LHC at the requested time (see Fig. 5).

The wrapping module also provides additional methods to:

- save the data set;
- print all the names of the LHC elements in longitudinal order;
- search for a string in the BLM names;
- plot a vertical line at a given longitudinal position;
- plot the ratio signal/threshold for each BLM;
- order BLMs by “closeness to threshold”;
- plot the beam intensity at the time;
- plot the position of the jaws of the displayed collimators;
- plot the signals of the Beam Position Monitors;
- plot the optics in the LHC.

CONCLUSION AND FUTURE IMPROVEMENTS

In this paper, a tool allowing the access, processing and display of data in a fast, user-friendly and efficient way was presented. Data can be requested automatically with a few simple arguments such as timestamp and position, and are immediately available for display or further processing. This tool is now used as a standard database access in the CERN Beam Loss section.

The future improvements include parallel downloading and processing of data, and adaptation to the new vector format for BLM-specific databases.

REFERENCES

[1] All available views, C. Roderick, BE-CO-DM, `http://lhc-logging.web.cern.ch/lhc-logging/software/public_data_access_views.jpg`

[2] LHC Logging Service, CERN-AB-Note-2006-046, `http://lhc-logging.web.cern.ch/lhc-logging/software/default.htm`

[3] ROOT class index, `http://root.cern.ch/root/html528/ClassIndex.html`

[4] Logging Data Extraction Client, Command Line API, `https://espace.cern.ch/be-dep/CO/DM/CALS/other/CommandLineManual.pdf`