A FRAMEWORK FOR OFF-LINE VERIFICATION OF BEAM INSTRUMENTATION SYSTEMS AT CERN

S. Jackson, C. Zamantzas, C. Roderick, CERN, Geneva, Switzerland

Abstract

Many beam instrumentation systems require checks to confirm their beam readiness, detect any deterioration in performance and to identify physical problems or anomalies. Such tests have already been developed for several LHC instruments using the LHC sequencer [1], but the scope of this framework doesn't extend to all systems; notably absent in the pre-LHC injector chain. Furthermore, the operator-centric nature of the LHC sequencer means that sequencer tasks aren't accessible by hardware and software experts who are required to execute similar tests on a regular basis. As a consequence, ad-hoc solutions involving code sharing and in extreme cases code duplication have evolved to satisfy the various use-cases. In terms of long term maintenance, this is undesirable due to the often short-term nature of developers at CERN alongside the importance of the uninterrupted stability of CERN's accelerators. This paper will outline the first results of an investigation into the existing analysis software, and provide proposals for the future of such software.

INTRODUCTION

Large quantities of data were produced by LHC's various instrumentation systems during the machine's more than three years of operation. Much of this data was logged and made available for offline observation via rudimentary data extraction tools such as the Timber web interface, but early in the LHC's operation it became clear that detailed analysis of this data would be necessary to detect equipment deterioration and data quality issues. It soon became apparent that examining the huge quantities of raw data manually was impossible, and ad-hoc analysis software began appearing within the BI (Beam Instrumentation) group.

With no standard framework or tools in place, this analysis software was created on a case-by-case basis, with most of the analysis tools being written by CERN students. Many of the students have since left CERN, leaving a legacy of undocumented, yet vital scripts which will be required to be operational when the LHC restarts. During the LHC shut-down, a project has been set up to analyse the contents of these scripts and to propose a framework into which the scripts can be managed. The scope of this framework should also be flexible enough to allow analysis scripts to be made for other LHC instrumentation systems as well as for instrumentation systems in the LHC injector chain.

This paper will detail the results of an investigation of the current analysis software which has been running prior to the LHC shut-down. It will highlight the strengths and weaknesses of the current software and conclude with some recommendations for the migration of existing and future analysis software.

ANALYSIS DATA SOURCES

Most of the existing software is driven by data extracted from the LHC measurement database. Direct access to this Oracle database is forbidden, and all data extraction must be performed via a Java API supplied by the database team. This allows the imposition of 'fair-use' policies and limits the possibility of runaway clients taking down the database, but it also restricts which languages can be used in the analysis.

The data analysed by the current software is generally machine triggered data. There are however other sources of data generated by Java applications written for equipment experts' use when analysing the state of their equipment. These so called 'expert applications' read data from instrumentation on-demand and often create raw data files for immediate or later analysis. One such application, for the readout of BLM (Beam Loss Measurement) modulation tests, not only creates these files when an expert presses a button in the GUI (Graphical User Interface), but the analysis code is written in such a way that the readout-and-dump-to-file functionality can also be triggered by the LHC sequencer. In this case, a set of data files is generated by the sequencer before each LHC fill – files which then form another data source for analysis.

CHOICE OF ANALYSIS LANGUAGE

The three officially sanctioned development languages in CERN's Beams department are C, C++ and Java. C and C++ are used for most of the accelerators' real-time software, and Java can be found predominantly in the GUIs and middle tiers. Given that the API of the main data source (the measurement database) is in Java, it would seem natural that Java would be used in the analysis software as well. However, analysts decided against the use of Java or C++ for the following reasons:

- The steep learning curve for object-oriented languages such as Java and C++ (many analysts didn't come from a computing background).
- Although the scripts were well designed regarding their purpose and function, little or no software design was made before the analysis scripts where written. An edit and run scripting language was preferred as it allowed rapid prototyping whereas the design, implement, compile, deploy and run approach of Java applications is very heavy when you are making frequent changes.

ISBN 978-3-95450-139-7

CERN also has a mature data analysis framework, ROOT, which is used extensively for analysis in the LHC experiments. This framework provides many C++ libraries for common data processing tasks, but the use of C++ as an analysis language had already been ruled out due its complexity. ROOT also provides a scripting language which satisfied the rapid prototyping requirements, but interfacing this with the Java API for accessing the database isn't trivial. The ROOT project however has a parallel project, PyROOT, which allows the ROOT libraries to be accessed by the Python scripting language. Also, the analysts found a way in Python to allow integration of the Java API which meant that their Python scripts could access both the data sources as well as the rich code-base of ROOT from a rapid scripting language.

It is clear why Python was chosen as the preferred language. However, the responsibility for the analysis environment is moving to the software section of the Beam Instrumentation group which means that there is a strong argument to move the new framework to Java (the department's standard development platform for data analysis), insisting that the lack of object-oriented programming expertise and lack of software design are addressed by basic training. Even with this training however, significant effort would be required to write or source Java packages which will replicate the algorithms provided by PyROOT and NumPy (a numerical package used in existing Python scripts).

EXECUTION AND MONITORING OF SCRIPTS

The existing Python scripts are executed from an SLC6 (Scientific Linux CERN) based machine at staggered intervals to avoid excessive instantaneous load on the database. The standard UNIX cron daemon is used to schedule and execute the scripts early in the morning to minimize interference with other users. Each of the Python scripts has an accompanying shell script which sources a common environment (setting Java class paths for the data extraction, setting paths for ROOT modules and Python distributions, etc.), then executes the script piping any output from stdout or stderr to a per-script log file.

Although not critical for machine operations, it is considered imperative that the scripts always run. Some of the analysis scripts act as an early warning, signalling an imminent problem, or the approaching of some predefined limit of the hardware. The cron mechanism ensures that the tasks are always executed (presuming the hosting machine is running), but on several occasions during the last LHC run the scripts blocked during their execution. For scripts that generate a daily report, it is likely that somebody will notice the problem, but for other scripts that only report on problems, it was sometimes days before the blocked script was noticed. In order to address this problem, two main ideas have been proposed:

- An additional monitoring cron task should be executed at a time when all other tasks should not be running. If this task sees a blocked process, it will send an email to signal the problem.
- As all the scripts output is sent to a log file, the final line in the file should contain a predefined message and time-stamp. A monitoring task can ensure that this line is present, confirming the task didn't exit prematurely.

TYPICAL ANALYSIS

Currently, most of the scripts analyse measurement, configuration and status data from the LHC's BLM system. It is beyond the scope of this paper to detail all of these scripts, but the following three scripts highlight the current main use-cases.

ThresholdChanges Script

BLM thresholds (logged at 1Hz intervals in the measurement database) are retrieved between two dates. A comparison is performed on a per monitor basis resulting in a report highlighting any changes (grouped by the monitor's family). A detailed PDF with plots and summary tables (showing thresholds at the two dates for the highlighted energy levels as well as the ratio of all the 12X32 thresholds per BLM) is then emailed to a list of people on a daily basis. This is an example of a script which is alerting the equipment expert when something critical (in this case the thresholds) has been changed and can highlight a setting corruption or even an unauthorized change, see Fig. 1.

CardTemperature Script

This analysis script is intended to give a full overview of status information without limiting it to anomalous data. The script produces daily graphs showing the temperature of all the beam loss acquisition cards. At a glance, the viewer of the graph can quickly see a trend indicating a potential hardware problem or even a cooling ventilation failure before the problem becomes critical.

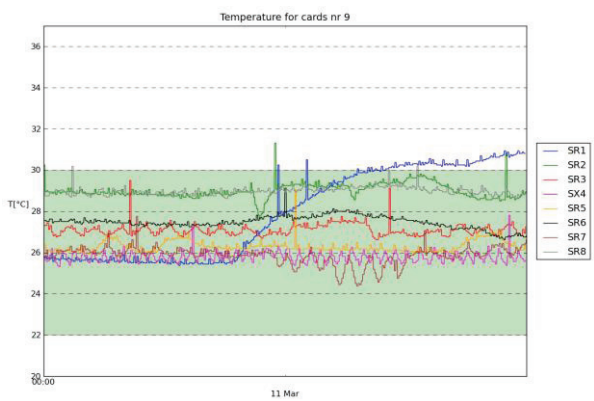

Figure 1: An example image showing BLM cards in 8 of LHC's surface buildings, generated using matplotlib.

ISBN 978-3-95450-139-7

HVStatus Script

As well as analysing the status of the high voltage measurements, this script also reports on unexpected measurements from the monitors where the loss was zero. This could highlight a serious failure somewhere in the transmission of the data, or a problem in the high voltage supply for the affected monitor. Without analysing all the measurement data, a glitch in the system could be missed.

Other Scripts

The analysis scripts aren't only limited to monitoring data from the BLM system. Scripts have also been made for producing regular measurement graphs for BGI (Beam Gas Ionization) monitors, and tabular reports for the LHC wire-scanners. In the case of the wire-scanners, analysis of how many scans have been made, along with acquisition error rates are used to help the hardware experts predict when the wires will have to be replaced due to their degradation by the beam.

EMAIL VERSUS WEBSITE REPORT DISTRIBUTION

The current Python scripts send their results via an email, sometimes with PNG or PDF attachments. This system works well when the report contains an important indication of potential failure, but can be considered intrusive if the report is for informative purposes. Furthermore, a list of email recipients has to be defined, and people who have an interest to see the reports weekly or monthly can't do so without being on this list (and thus subjected to daily emails). The proposed solution is to firstly classify the reports as either informative, or action reports. Then, we can publish analysis results considered as informative via a public website, and publish results which potentially require some action via the current email/attachment approach. See Fig. 2. Public publishing of data is widely used for other operational systems, in the form of a dashboard. This dashboard approach can be extended to handle historical data, thus providing all visitors access to all current and previous reports if and when required.

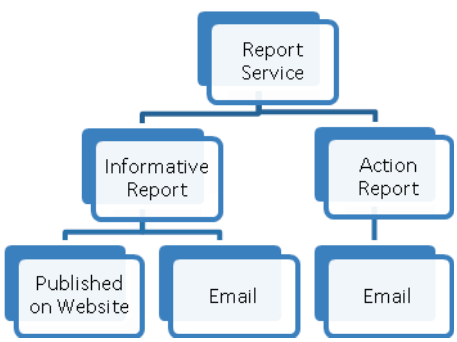

Figure 2: Proposed delivery of reports.

RESTRUCTURING / CREATION OF A DEVELOPMENT FRAMEWORK

The existing analysis scripts don't follow any framework, and little effort has been made to modularize common code to avoid code-duplication. For example, the code used to wrap the access to the measurement database's Java API into a Python module has been written once, and then duplicated for each of the modules which use it with very minor changes. Examination of the code immediately highlights that most of the code for initializing the connection to the database can easily be extracted to a common module. As well as code duplication, there are numerous examples of difficult to understand code, such as the creation of LaTeX files using hundreds of lines of code which simply append line after line of text to a file. While modularizing this kind of code would add little or no value to the execution of the code itself, it would certainly help in the long term readability and maintenance of the code.

Some effort has been made however to separate configuration within the scripts. Many scripts have an accompanying file which defines anything deemed to be configurable such as email recipients, start and stop times for data extraction, and other script-specific parameters. Presently, any modification of these configuration files must be done by hand in a text editor, but it has been requested that any new proposed framework should define a common format for these files along with accompanying graphical user interfaces to edit their contents.

The outcome of an analysis of the existing scripts indicated that any new framework would have to provide the following functionality to allow the existing scripts to be ported.

- A single module to access the database. While some custom code might be needed to access analysis specific data, most of the code for connecting to the database and collecting data can be modularised.
- Production of reports which contain text and tables. This is currently achieved by producing intermediate LaTeX files passed through the pdflatex UNIX command to produce PDF files. This should be abstracted to hide the implementation details, so that the framework can ultimately produce the reports as PDF files, web pages, rows in a database, etc.
- Production of graphs. Graphs are created using the matplotlib Python library. The graphs are exported to image files (PNG) and sent as attachments to email recipients or embedded in LaTex scripts. If possible, the proposed framework should also provide an abstracted API to hide the implementation details, see Fig. 3.

ISBN 978-3-95450-139-7

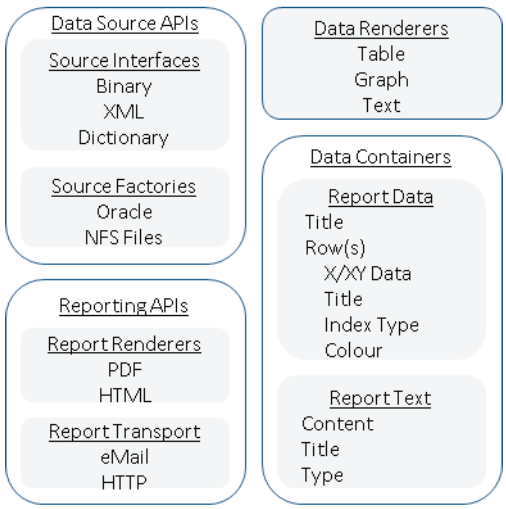

Figure 3: Proposed APIs (not all detail shown).

AN ALTERNATIVE MEANS TO ANALYSE

During LHC's long shut-down, the Beams Controls group has launched an initiative to facilitate data analysis. Key objectives include being able to perform analysis as close to the relevant data sources as possible, and being able to store the analysis results using the Accelerator Logging Service [2]. Concerning data analysis - the current idea is to have a means to describe the analysis to be performed using a DSL (Domain Specific Language), with the option to schedule the analysis based on time or beam related events, and describe how to persist the results for subsequent access. This would mean that instead of performing the analysis in Python, the analysis could be described and executed for example inside the Logging Service Oracle database using PL/SQL or Oracle Enterprise R. Thanks to the DSL, the actual implementation details would be hidden from the end user.

The aforementioned initiative aims to allow analysis of data from other sources than just the Logging Service, such as the LHC Post Mortem system, and the LSA Settings database. It is clear that this could be an interesting alternative means to perform analysis, and there will be an obvious benefit from using the standardized Logging Service interface to store analysis results in addition to (or even in place of) the current reports dispatched by email.

CONCLUSION AND OUTLOOK

There are still many decisions to be made for the future analysis framework, some of which will depend on the future roadmap of the Accelerator Logging Service. In the meantime, new and existing scripts should be categorized as either *informative* (publishing public results on internal CERN websites), *action* (generating reports when problems are detected and sending the reports directly to interested recipients), or as both *informative and action*.

There is a strong case to migrate away from Python to Java, but further analysis needs to be done to ensure that Java can provide an equivalent set of libraries as is currently provided in PyROOT, NumPy and matplotlib. If a decision is made to keep the framework in Python, efforts must be made to remove code duplication from the existing scripts and impose APIs to abstract the generation of tabular and graph reports. Once the new initiative proposed by the Controls group takes shape, many existing and future scripts should be migrated to this new framework as it makes sense to have the analysis as close to the data source as possible. This proposed Controls framework however will only cover scripts which are sourcing data from the databases and LHC Post Mortem system.

A standard file format for script configuration will be devised, meaning graphical user interfaces can be made to edit and view scripts' parameters. Equipment experts can then change the behaviour of the analysis scripts without touching the source code. Issues arising from the malfunctioning of analysis scripts should be addressed by an additional cron task which periodically checks for hanging processes and monitors the log files from scripts.

ACKNOWLEDGMENTS

The authors would like to thank the Beam Loss section for their help in understanding the many scripts which formed the basis for this analysis. Also, the insights into the future initiatives proposed by the Beams Controls group regarding the use of alternative techniques like DSL have highlighted that in some cases, the future of some data driven analysis may be better placed in the database itself rather than in external scripts.

REFERENCES

[1] R. Alemany-Fernandez et al, “The LHC Sequencer”, ICALEPCS’11.

[2] C. Roderick et al., “The LHC Logging Service: Handling Terabytes of On-line Data”, ICALEPCS’09.

ISBN 978-3-95450-139-7