The ALICE Data Quality Monitoring System

Barthélémy von Haller, Sylvain Chapeland, Valerio Altini, Franco Carena, Wisla Carena, Vasco Chibante Barroso, Filippo Costa, Roberto Divià, Marc Frauman, Ulrich Fuchs, Irina Maklyueva, Ornella Rademakers-di Rosa, David Rodriguez Navarro, Filimon Roukoutakis, Klaus Schossmaier, Csaba Soós, Adriana Telesca and Pierre Vande Vyvre for the ALICE collaboration

Abstract—ALICE is one of the four experiments installed at the CERN Large Hadron Collider (LHC), especially designed for the study of heavy-ion collisions. The online Data Quality Monitoring (DQM) is an important part of the data acquisition (DAQ) software. It involves the online gathering, the analysis by user-defined algorithms and the visualization of monitored data. This paper presents the final design, as well as the latest and coming features, of the ALICE’s specific DQM software called AMORE (Automatic MonitoRing Environment). It describes the challenges we faced during its implementation, including the performances issues, and how we tested and handled them, in particular by using a scalable and robust publish-subscribe architecture. We also review the on-going and increasing adoption of this tool amongst the ALICE collaboration and the measures taken to develop, in synergy with their respective teams, efficient monitoring modules for the sub-detectors. The related packaging and release procedure needed by such a distributed framework is also described. We finally overview the wide range of usages people make of this framework, and we review our own experience, before and during the LHC start-up, when monitoring the data quality on both the sub-detectors and the DAQ side in a real-world and challenging environment.

I. INTRODUCTION

A. The ALICE experiment

ALICE (A Large Ion Collider Experiment) [1] is the LHC experiment dedicated to the study of heavy-ion collisions at CERN. It focuses on the study of the quark-gluon plasma in heavy ion collisions, but will also carry out extensive measurements of proton-proton interactions. The experiment consists of several detectors of different types and is designed to cope with very high particle multiplicities ($dN_{ch}/dy$ up to 8000). Commissioning has been carried out during the year 2008 in the underground experimental pit at the Swiss-French border. Detectors, along with the required support services, were ready for the LHC startup in September 2008 and are now getting prepared for the LHC restart.

B. Data Quality Monitoring

Data Quality Monitoring (DQM) is an important aspect of every High-Energy Physics experiment, especially in the era of LHC where the detectors are extremely sophisticated devices. To avoid recording low quality data, one needs an online feedback on the quality of the data being actually recorded for offline analysis. DQM software provides this feedback and helps shifters and experts to identify early potential issues. DQM involves the online gathering of data, their analysis by user-defined algorithms and the storage and visualization of the produced monitoring information.

C. ALICE Data Acquisition

The Data Quality Monitoring is part of the ALICE Data Acquisition (DAQ) system [2][3], whose dataflow and architecture are described in figure 1. Event fragments come from the Front End Read Out (FERO) electronic of the detectors, through optical links, to Local Data Concentrators (LDC). Sub-events are then shipped through a Gigabit Ethernet network to Global Data Concentrators (GDC) where the full events are built. DQM software runs on dedicated servers connected to the event building network. The sample data that feed the DQM nodes are intercepted on either the LDC’s or the GDC’s depending on the needs.

The Data Acquisition and Test Environment (DATE) [4] is the software framework that has been developed as a coherent environment for all operation of the ALICE DAQ. DATE is composed of packages that perform the different functionalities needed by the DAQ system. These include low-level functionalities such as memory handling or process synchronization and higher-level functionalities like event building and data recording. DATE provides a low-level monitoring package which forms the basis of any high-level monitoring framework for ALICE. It exposes a uniform Application Programming Interface (API) to access on-line raw data on DAQ nodes as well as data written in files. This API also gives the possibility of selecting the event sampling strategy for on-line streams in order to balance the needed computing resources.

II. AMORE : A DQM FRAMEWORK FOR ALICE

A. Design and architecture

A DQM software system, named AMORE (Automatic MOnitoRing Environment) [5]-[7], has been developed for
the ALICE experiment. It is founded on the widely-used data analysis framework ROOT [8][9] and uses the DATE monitoring library. In case the same analysis is needed online and offline, the use of the ALICE Off-line framework for simulation, AliRoot [10], is encouraged.

AMORE is based on a publisher-subscriber paradigm (see figure 2) where a large number of processes, called agents, execute detector-specific decoding and analysis on raw data samples and publish their results in a pool. Clients can then connect to the pool and visualize the monitoring results through a dedicated user interface. The serialization of the published objects, which occurs on the publisher side before the actual storage in the database, is handled by the facilities provided by ROOT. The only direct communication between publishers and clients consists of notifications by means of DIM [11]. The notifications coming from the outside world, especially from the Experiment Control System (ECS), use the same technology.

As shown in figure 3, the data samples feeding the agents come either from the DAQ dataflow nodes or directly from other agents. The resulting physical quantities are published encapsulated in MonitorObject’s that essentially contain additional housekeeping information allowing a proper and coherent handling by the framework. Published objects are often histograms but there is no limitation on their type.

B. The AMORE data pool

The pool is implemented as a database. The open-source MySQL system [12] was chosen as it proved to be reliable, performant and light-weight. The database contains a table with a list of all the agents, specifying on which machine they are allowed to run and to which detector they belong (see figure 4). Another table contains configuration files. These files are optional and each detector can define several of them corresponding per instance to different run types. Finally, a data table is created for each agent where the published objects are stored.

C. Archiving

Keeping former versions of the published objects is a considerable asset to observe their evolution over time and to determine when a change occurred. A short-term history is available, implemented with a First In First Out policy (see Fig. 5). Monitor objects can also be archived for a longer time either at shifter’s request or automatically on a regular basis.
D. Pluggable architecture

AMORE uses a plug-in architecture to avoid any framework’s dependency on users’ code. The plug-in mechanism is implemented through the ROOT reflection feature. Users, usually detector teams, develop specific code that is built into dynamic libraries called modules that are loaded at runtime by the framework if, and when, it is needed. Modules are typically split into two parts corresponding to the publishing and the subscribing sides of the framework (see figure 6). The module’s publisher can be instantiated several times, to collect more statistics for instance, each instance corresponding to an agent. The same is true for the subscriber part of the module.

III. Visualization

The subscriber part of the users’ modules mainly consists in a Graphical User Interface (GUI) capable of handling the objects produced by the publishing part. As the basic needs of most of the detectors teams were very similar, a generic GUI has been developed in order to avoid code duplication and to ease the life of the users. It can be used to browse and visualize any object of any running agent. The only requirement is that the objects follow a very basic interface to allow them to be drawn. To ease the browsing, the thousands of histograms published by the detector agents are displayed as a tree (see left panel in figure 7). The right panel of the window displays the MonitorObjects selected by the user, automatically fitting them on the available space by splitting the tab in a grid. Finally, the layout can be saved as XML files for future reuse.

Sometimes, it is preferable that users develop their own user interface that would better fit their needs than the generic GUI. It is the case if they want to have a detector-specific display (see figure 8) or if they wish to use custom objects type that could not be handled in a meaningful way by the generic browser.

IV. Packaging and release procedure

About 15 teams have started the development of one or several AMORE modules, each one with its own schedule and its own working environment. It became therefore necessary to have a unique and strict release procedure as well as a
Fig. 7. The generic GUI

Fig. 8. Custom GUI of the MUON tracking detector
centralized code repository. Subversion [13] has been chosen for the source code management. The framework and the module code are in two different repositories with different access rights policies, allowing everyone to share its work easily while making sure no code gets lost.

The AMORE build system is based on GNU Autotools, while users use simple Makefiles to create their modules libraries. The framework, as well as the modules, are distributed as binary RPM’s, which is the format of choice for all the software that is installed on ALICE DAQ machines.

Before having its module deployed on the production DQM nodes, one must go through a strict release procedure to ascertain good software quality. It consists in mainly three steps: building the code, producing the RPM and running the agent on a set of data files provided by the detectors. The validation is done on a test machine, in a controlled and isolated environment. This procedure is run every night to identify early the potential issues introduced during the day, either in the framework or in the users’ code.

V. Benchmarks

A. General framework performances

In an online environment, where heavy calculation is required, one must ensure performances and scalability. To identify and handle performance issues, it is necessary to define metrics, to collect statistics and to have reproducible tests. To a large extent, these requirements correspond to the modules’ validation procedure described in the previous section. These could therefore be used as a benchmark after a few modifications were made in the framework, essentially to log the execution time and to make the resulting benchmark easily usable on any machine.

This benchmark gives important and useful information in various area. First, it helped us better estimate the needs of the users, in terms of CPU, memory and software architecture. Moreover, as it runs every night, it gives the opportunity to identify variations of performance over time. Finally, several comparisons of machines, compilers and software architecture have been carried out. For instance, tests were conducted on new hardware, such as the latest generation of Intel processors, and compared with our current DQM servers. Figure 9 shows the performance increase provided by the Nehalem-based PCs for some typical tests. Although our current servers are modern machines, an improvement of almost 50% of the execution time was observed. These tests were carried out in a mono-threaded mode, and one can expect even larger gain with multi-threaded software. More details about the different tests used as benchmarks can be found in [14].

B. Database benchmark

As the data pool is the backbone of the system where all data transit, it is important to guarantee good performances and reliability even at peak time. The database benchmark consists of several tests; some check extreme conditions where, for instance, a large number of agents continuously publish data, and other tests correspond to more standard use cases. The benchmark proved that the pool can sustain high loads and thus that the database within the current design is capable of handling the increasing number of agents and clients. This benchmark also improved the way the database is accessed. For example, experience showed that the results were far better
when small queries were concatenated into very large queries. Findings also concerned the configuration of the database itself. The difference between MySQL storage engines proved to be very important. InnoDB engine supports transactions and row-level locks whereas MyISAM is much simpler and only have table-level locks and no transactions. Even in simple cases with a single running agent (see Fig. 10), InnoDB is twice slower than MyISAM. When running 10 agents concurrently, the execution time with InnoDB becomes 10 times larger than the one with MyISAM (see Fig. 11).

VI. STATUS AND EXPERIENCE DURING COMMISSIONING

AMORE has been in production since spring 2008 and was used successfully from that time. It proved to be very useful to test detectors and to identify potential problems during the commissioning. Figure 12 shows one example where AMORE was successfully employed to spot a potential problem. The event size distribution of a detector is plotted and appears to be split into two peaks. This was a clear indication of a possible glitch in the detector zero-suppression and it triggered an early investigation that would not have occurred so quickly otherwise.

Of course, no monitoring would be possible without the detectors teams being involved and developing their modules. Most of the 18 ALICE detectors have started working on their monitoring module and a few already passed successfully the validation procedure and are now used in production. Continuous interactions between the users and the framework developers have been at the center of our attention since the beginning of AMORE’s development. It led to a quicker implementation of the modules and to a framework fitting well the needs of the detectors’ teams.

VII. FUTURE DEVELOPMENTS

One future direction is towards a wider access to the MonitorObjects published by the agents. For the time being, only shifters in the Control Room can run an AMORE client and have access to the latest produced objects. An access through the web, thanks to the ALICE electronic logbook [15], will provide experts with an easy way to look at the monitoring histograms from outside the Control Room.

The plan is also to further automate the monitoring by making comparisons with reference data and automatically take action if some criteria are met. Finally, to fully exploit the coming generation of machines, AMORE will have to be parallelized: it will then take advantage of multiple cores and CPU’s.

VIII. CONCLUSION

Since its release in 2008, AMORE has seen an increasing adoption amongst ALICE detectors. The framework proved to fit their needs and to provide useful feedback on the data being recorded during their commissioning and during first beam period.
Fig. 12. Event size distribution of the ALICE SSD, as displayed on the day before the first beam

REFERENCES

[1] The ALICE Collaboration, ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC, CERN, Tech. Rep. CERN/LHCC/95-71, LHCC/P3, 1995.
[2] V. Chibante Barroso et al., “Commissioning and First Experience of the ALICE Data Acquisition System”, presented at IEEE NPSS Real Time Conference, Beijing, China, 2009, to be published
[3] The ALICE DAQ Project website [Online]. Available: http://ph-dep-aid.web.cern.ch/ph-dep-aid/
[4] The ALICE DAQ Project, ALICE DAQ and ECS User’s Guide, 2006, release 5.0.
[5] F. Roukoutakis, S. Chapeland and O. Çobanoglu, “The ALICE-LHC Online Data Quality Monitoring Framework: Present and Future,” IEEE Trans. Nucl. Sci., vol 55, no. 1, pp. 379–385, Feb. 2008.
[6] F. Roukoutakis and S. Chapeland, “The ALICE-LHC Online Data Quality Monitoring Framework,” J. of Physics: Conf. Series, vol 119, Jun. 2008.
[7] F. Roukoutakis and B. von Haller, “Commissioning of the ALICE-LHC Online Data Quality Monitoring Framework,” Nuclear Instruments and Methods in Physics Research, Section A, Feb. 2009
[8] R. Brun et F. Rademakers, “ROOT - An Object Oriented Data Analysis Framework,” Nuclear Instruments and Methods in Physics Research, vol. A389, pp. 8186, 1997.
[9] The ROOT website [Online]. Available: http://root.cern.ch
[10] The ALICE Offline Project website [Online]. Available: http://alice-info.cern.ch/Offline
[11] C. Gaspar, M. Dönzelmann and P. Charpentier, “DIM, a Portable, Lightweight Package for Information Publishing, Data Transfer and Inter-process Communication,” in International Conference on Computing in High Energy and Nuclear Physics, Padova, Italy, Feb. 2000.
[12] The MySQL website [Online]. Available: http://mysql.org
[13] The Subversion website [Online]. Available: http://subversion.tigris.org/
[14] S. Chapeland and B. von Haller, “Evaluation of computers performance for the ALICE DAQ” ALICE-INT-2009-009, CERN, 2009.
[15] V. Chibante Barroso et al., “The ALICE Electronic Logbook”, CHEP’09, Prague, Czech Republic, 2009