The ALICE Online Data Quality Monitoring

B. von Haller[*]**, A. Telesca, S. Chapeland, F. Carena, W. Carena, V. Chibante Barroso, F. Costa, R. Divià, U. Fuchs, I. Makhlyueva, O. Rademakers-Di Rosa, G. Simonetti, C. Soós and P. Vande Vyvre for the ALICE collaboration**
CERN, Physics Department, Geneva, Switzerland
E-mail: barthelemy.von.haller@cern.ch, adriana.telesca@cern.ch

ALICE (A Large Ion Collider Experiment) is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN Large Hadron Collider (LHC). The online Data Quality Monitoring (DQM) is a critical part of the Data Acquisition's software chain. It intends to provide shifters with precise and complete information to quickly identify and overcome problems, and as a consequence to ensure acquisition of high quality data. DQM typically involves the online gathering, the analysis by user-defined algorithms and the visualization of monitored data.
This paper describes the final design of ALICE's DQM framework called AMORE (Automatic MOnitoRing Environment), as well as its latest and coming features like the integration with the offline analysis and reconstruction framework, a better use of multi-core processors by a parallelization effort, and its interface with the eLogBook. The concurrent collection and analysis of data in an online environment requires the framework to be highly efficient, robust and scalable. We will describe what has been implemented to achieve these goals and the procedures we follow to ensure appropriate robustness and performance.
We finally review the wide range of usages people make of this framework, from the basic monitoring of a single sub-detector to the most complex ones within the High Level Trigger farm or using the Prompt Reconstruction and we describe the various ways of accessing the monitoring results. We conclude with our experience, before and after the LHC restart, when monitoring the data quality in a real-world and challenging environment.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research
February 22-27, 2010
Jaipur, India

[*]Speaker.

1. Introduction

1.1 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 quark-gluon plasma formation signature, but it will also be able to observe 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 years 2008 and 2009 in the underground experimental pit at the Swiss-French border. Detectors, along with the required support services, were ready for the LHC restart in November 2009 and are successfully taking data since then.

1.2 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 algorithm and the storage and visualization of the produced monitoring information.

1.3 ALICE Data Acquisition

The Data Quality Monitoring is part of the ALICE Data Acquisition (DAQ) system [2][3], whose dataflow and architecture is described in Figure 1. Event fragments come from the Front End Read Out (FERO) electronics 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 data samples 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 the ALICE DAQ operations. DATE is composed of packages that perform the different functionalities needed by the DAQ system. Amongst others, it provides a low-level monitoring package which exposes a uniform Application Programming Interface (API) to access on-line raw data on DAQ nodes as well as data written in files. This package forms the basis of any high-level monitoring framework for ALICE, including the framework we are going to present.

2. AMORE: A DQM framework for ALICE

2.1 Design and architecture

A DQM software system, named AMORE (Automatic MOnitoRing Environment) [5][6][7][8], 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

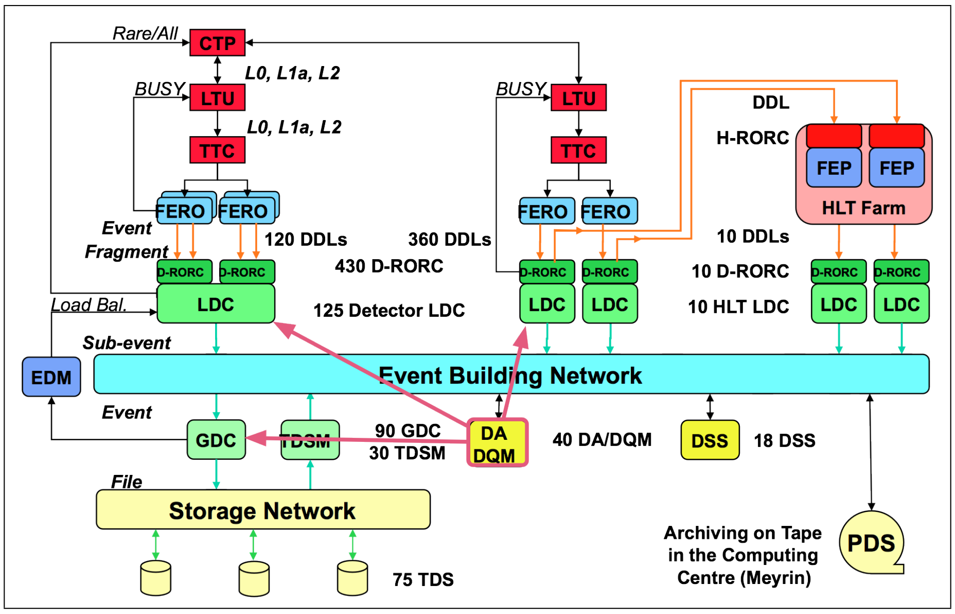

Figure 1: The ALICE Data Acquisition architecture. DQM nodes are depicted by the yellow and red box

online and offline, the use of the ALICE Offline framework for simulation, AliRoot [11], 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 (Distribution Information Management system) [12]. The notifications coming from the outside world, especially from the Experiment Control System (ECS), use the same technology.

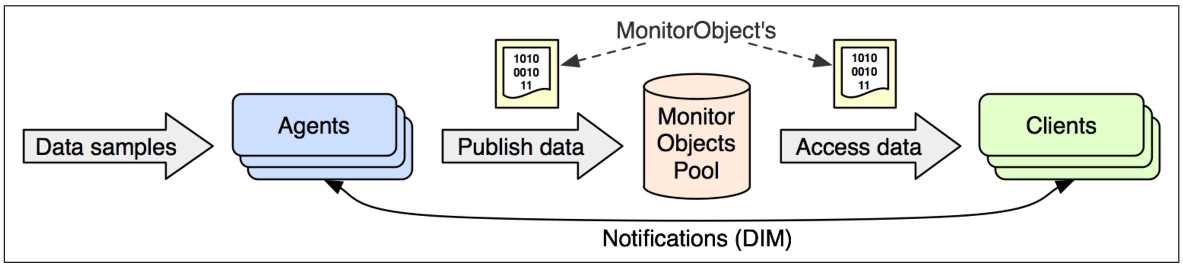

Figure 2: The publisher-subscriber paradigm in AMORE

The data samples feeding the agents might come from the DAQ nodes or from other agents. The resulting physical quantities are published encapsulated in *MonitorObject's* that essentially contain metadata allowing a proper and coherent handling by the framework. Published objects are often histograms but there is no restriction on their type.

2.2 The AMORE data pool

The pool is implemented as a database. The open-source MySQL system 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 3). Another table contains configuration files. These files are optional and each detector can define several of them corresponding for instance to different run types. Finally, a data table is created for each agent where the published objects are stored.

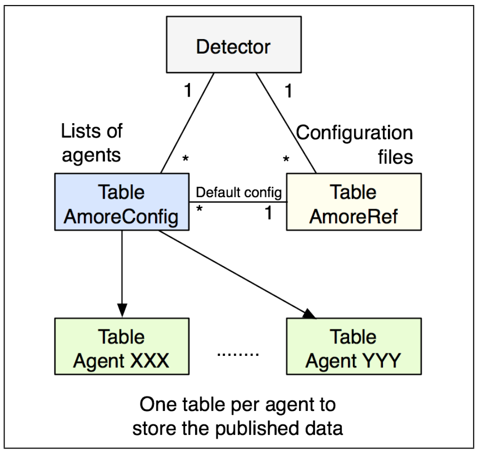

Figure 3: AMORE's database architecture

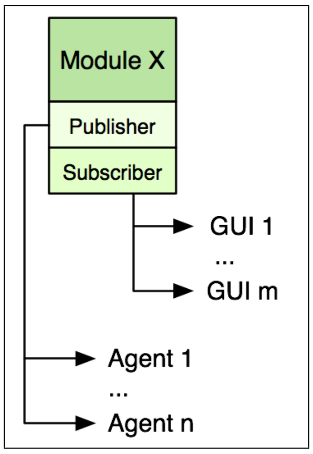

Figure 4: Schema of a module

2.3 Archiving

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

2.4 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 detectors 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 4). The module's publisher can be instantiated several times, to collect more statistics per instance, each instance corresponding to an agent. The same is true for the subscriber part of the module.

3. Visualization

The subscriber part of the users' modules consists mainly in Graphical User Interfaces (GUI's) 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 minimize users development. 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 objects published by the detector agents are displayed as a tree (see left panel in Figure 5). 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 in the database or in the local file system for future reuse.

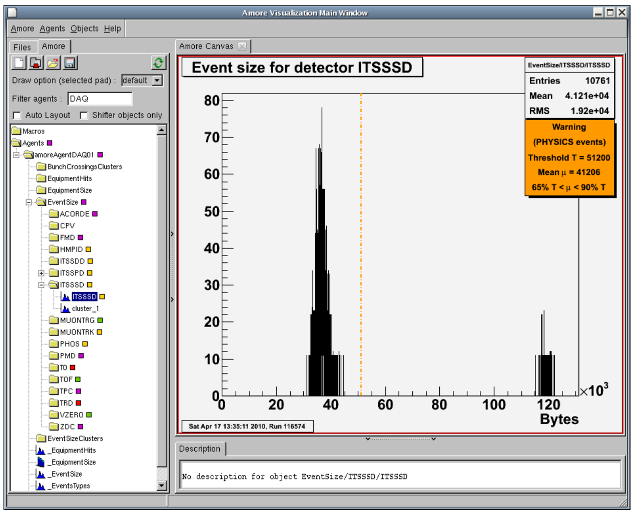

Figure 5: The generic GUI

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's specific display or if they wish to use custom objects types that could not be handled in a meaningful way by the generic browser. An example of such a custom GUI is shown in Figure 6.

Finally, we also provide an access to the monitoring data through the web. This is done by

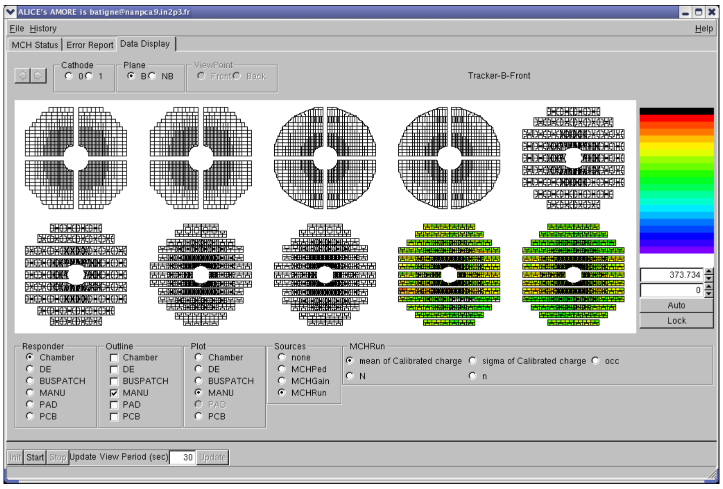

Figure 6: Custom GUI of the MUON tracking detector

means of the ALICE eLogbook [15] which offers all kind of information about runs and amongst them the monitoring information. Given the fact that the Generic Gui and the custom Gui's can only access data within the ALICE control room, the eLogbook provides an essential and world-wide way of checking the monitoring results.

4. Packaging and release procedure

About 15 teams 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 centralized code repository. *Subversion* has been chosen for the source code management.

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. The validation is done on a test machine, in a controlled and isolated environment. This procedure is also run every night to identify early the potential issues introduced during the day, either in the framework or in the users' code.

5. Benchmarks

5.1 General framework performance

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 mentioned in the previous section. Therefore, after a few modifications were made in the framework, the validation procedure can now be used as a benchmark, to check the framework performance or new hardware solutions.

5.2 Multithreading

Today's computers are getting more powerful by means of multiplying the number of cores. To take advantage of this architecture, AMORE must be able to run on multiple threads. Our first step in this direction has been to create a dedicated thread for the creation of images needed by the ALICE eLogbook. The first tests are very promising and are discussed in the next section. Given the good results, we are going to create yet another thread for the database operations.

The next stage will consist in running not only with several threads but also in multiple processes. By creating slave agents, one can process more data in a way which might be transparent for the users and without major change in the framework.

5.3 Results

Extensive testing of AMORE in multithreading mode has been carried out to make sure that it does not disrupt the normal behaviour and that it brings the expected benefits. Results are shown in

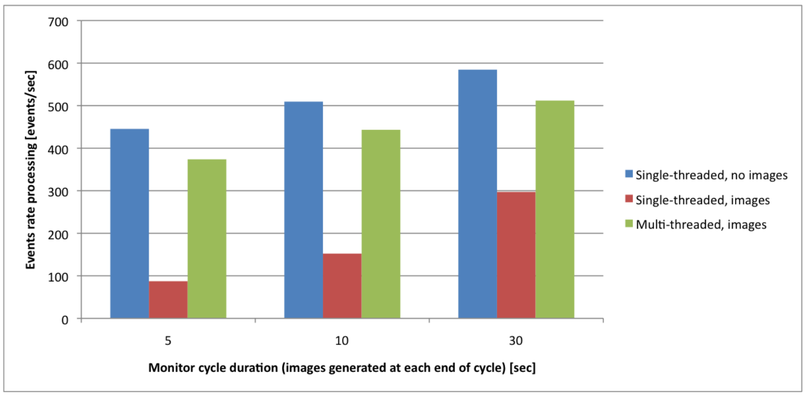

Figure 7: Comparison between single-threaded and multi-threaded AMORE (higher is better)

Figure 7 where the events' processing rate is plotted vertically for various monitor cycle lengths; a monitor cycle is the period during which the events are monitored and is terminated with the image generation (if any) and with the publication into the database.

The results clearly show that the images generation dramatically affects the event processing rate (blue versus red bars) whereas having a dedicated thread produces performance similar to the case when no images are created (green bar).

6. Status and experience since the LHC startup

Since the LHC startup in November 2009, AMORE has been extensively used in production. The system saw up to 35 agents running concurrently and publishing a maximum of 3400 objects per second; these objects added to a total of 115 MB per second.

As time passed and experience was gained, the usages of AMORE evolved. We realized that having the production of data quality objects done only by the agents running on the DQM nodes was not always enough. Therefore, we added the possibility to publish monitoring objects within other systems and software. Amongst others, this is the case for the High Level Trigger (HLT) which runs on a dedicated farm. Along with its normal activity, it calculates monitoring data and ships it to the AMORE pool and thus making it available to agents for post-processing and clients for display.

Another interesting usage is the *prompt reconstruction* which is the online execution on a DQM node of the online reconstruction part of the offline analysis framework reconstructing sample events immediately after they have been acquired. Another process calculates 3-dimensional views of the events, generates an image and publishes it in AMORE. By doing so, collaborators can access them in the eLogbook, and a public website makes them available to anyone interested in what is currently happening in our experiment (see [16]).

7. Future developments

Future plans include the further automation of the monitoring process by making comparisons to reference data and by identifying automatically possible issues. As discussed in section 5.2, we would also like to take full advantage of multi-core architecture by mean of making the monitoring framework running on multiple threads and processes.

8. Conclusion

Since the LHC startup, AMORE has been successfully used by all the ALICE detectors and systems teams. It proved to fit their needs and to provide a valuable feedback on the data being recorded. The framework and its infrastructure have been able to adapt to the wide range of usages that appeared and to cope with very large number of agents, clients and objects.

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. 379Ð385, 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] B. von Haller, F. Roukoutakis and S. Chapeland, *The ALICE data quality monitoring*, J. of Physics: Conf. Series, Manuscript approved but not yet published

[9] R. Brun et F. Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Nuclear Instruments and Methods in Physics Research, vol. A389, pp. 8186, 1997.

[10] The ROOT website [Online]. Available: http://root.cern.ch

[11] The ALICE Offline Project website [Online]. Available: http://alice-info.cern.ch/Offline

[12] C.Gaspar,M.DönszelmannandP.Charpentier, *DIM, a Portable, Light Weight 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.

[13] The MySQL website [Online]. Available: http://mysql.org 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

[16] ALICE live feed [Online]. Available: http://alice-logbook.cern.ch/aliceOnline/alice_online.html