The ALICE Data Quality Control

Barthélémy von Haller[1,*] and *Piotr* Konopka[1,**] for the ALICE Collaboration

[1]CERN, Experimental Physics Department, Geneva, Switzerland

Abstract. ALICE (A Large Ion Collider Experiment) has undertaken a major upgrade during the Long Shutdown 2. The increase in the detector data rates, and in particular the continuous readout of the TPC, led to a hundredfold increase in the input raw data, up to 3.5 TB/s. To cope with it, a new common Online and Offline computing system, called O^2, has been developed and put in production.

The online Data Quality Monitoring (DQM) and the offline Quality Assurance (QA) are critical aspects of the data acquisition and reconstruction software chains. The former intends to provide shifters with precise and complete information to quickly identify and overcome problems while the latter aims at selecting good quality data for physics analyses. Both DQM and QA typically involve the gathering of data, its distributed analysis by user-defined algorithms, the merging of the resulting objects and their visualization.

This paper discusses the final architecture and design of the Quality Control (QC), which runs synchronously to data taking and asynchronously on the Worldwide LHC Computing Grid. Following the successful first year of data taking with beam, we will present our experience and the lessons we learned, before and after the LHC restart, when monitoring the data quality in a real-world and challenging environment. We will finally illustrate the wide range of usages people make of this system by presenting a few, carefully picked, use cases.

1 Introduction

Modern particle physics experiments produce vast volumes of data. Due to the complexity of particle detectors, as well as the complexity of their associated data acquisition and processing software, the quality of gathered information has to be thoroughly controlled. Data Quality Monitoring (DQM) systems are used to quickly spot issues that appear during data taking. During months or years after, Quality Assurance (QA) systems allow for a more complete assessment of the suitability of physics data for analyses, which leads to new physics results.

The ALICE experiment recently underwent a major update involving the detector, the data acquisition system [1] and the data processing framework [2]. It enabled the experiment to acquire up to 3.5 TB/s of raw data produced by the detector. The new Quality Control (QC) software framework combines two previously separate systems, DQM and QA, to improve the code reusability between the two domains and to facilitate the access and the understanding of the results in a consistent way. During data acquisition, it is able to sustain two orders

*e-mail: barthelemy.von.haller@cern.ch
**e-mail: piotr.konopka@cern.ch

of magnitude greater data rates than the previously used ALICE DQM. The QC framework is based on the common ALICE O^2 software framework [2]. As a consequence, it makes it the first of its kind to thoroughly rely on message passing and actor model.

The Quality Control is used in two substantially different environments. During synchronous data acquisition, it's distributed across approximately 500 nodes at the experimental site, alongside particle collision reconstruction. Asynchronously, during the second phase of data preparation for physics analyses, it operates within a computing grid, overseeing thousands of data reconstruction tasks. Moreover, the Quality Control plays a role in validating processing and physics results by managing simulated data production and reconstruction, also on the computing grid.

In this paper, we describe the new ALICE Quality Control framework, focusing on allowing it to run in two vastly distinct environments. Also, we share experiences from its first year of operation.

2 Design and Architecture

The system is designed to provide operators and experts with a high-level quality assessment of the 3.5 TB/s data produced by the detector. Through a multi-step process (summarized in Fig. 1), the system extracts information and knowledge about the quality of the data being acquired and the underlying processing.

The initial step consists in sampling the data, usually at a rate of 1%. *QC Tasks* will then execute user-defined algorithms to process it and generate a *QC Object*, often a histogram. Data types available in the ROOT framework [3] are typically used as QC objects. Given the parallel nature of this processing, with a copy of the task running on each of the hundreds of nodes, these histograms are then merged. Finally, the merged results are evaluated by a series of *Checks* to determine one or several *Qualities*, which can themselves be aggregated to give a general assessment of the health of the data. Both QC Objects and Qualities are stored in a repository based on the ALICE Calibration and Conditions Database [4].

The QC system is based on a message passing paradigm where data flows asynchronously through a set of *devices* connected via buffered *channels*. The channels use ZeroMQ [5] by passing either the whole message payloads or just pointers to the shared memory region. The connections can be blocking or non-blocking. The QC software is part of the Online-Offline (O^2) computing system [6] and it shares a common message-passing framework with it [2].

This design allows us to split computations across thousands of nodes during synchronous processing. If one Merger process cannot sustain the output data throughput of all the parallel QC tasks, data reduction can be performed in several steps, as shown in Fig. 2. In the first layer, input data throughput is shared among multiple Mergers and then further reduced until a complete set of objects is obtained. Each actor publishes its most complete result in regular intervals, thus, with each new layer an additional delay is introduced.

When running the software in an asynchronous environment, combining the output of hundreds of subjobs sequentially may take a substantial amount of time. Thus, the processing can be drastically sped up by dividing the merging into several layers, as in the synchronous context, but with intermediate results stored as files. Additionally, since grid jobs sometimes suffer from challenging running environment (temporary lack of memory, processing node faults), they are more likely to fail. Restarting a smaller batch of operations takes a smaller toll on the total latency and the computing resources usage.

The Data Sampling component brings a convenient way to reduce the load on the system when running with limited resources and quick data quality feedback is preferred over having complete statistics. During data taking, only a few lightweight tasks which require 100% of the data stream do not use sampling, while all the rest receive between 1 and 10%

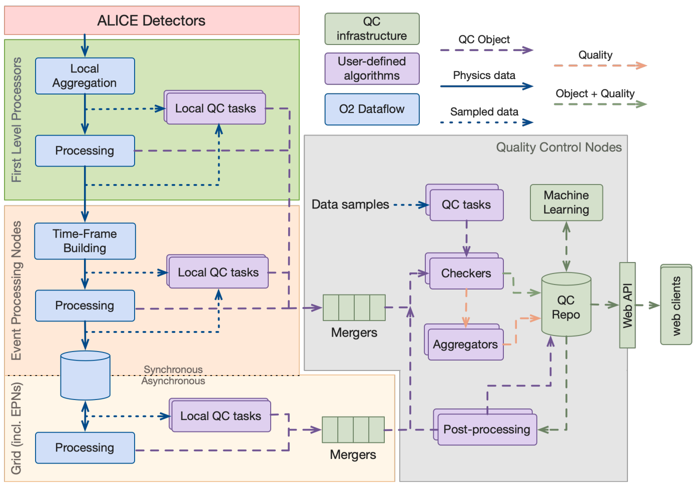


Figure 1. Overview of the Quality Control framework.

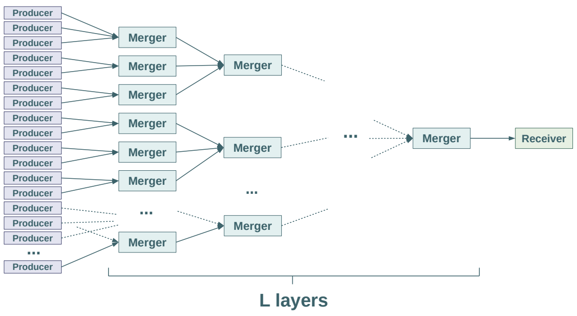


Figure 2. A multi-layer topology of Mergers.

of messages. If a given task becomes too resource-hungry, its sampling rate can be easily reduced at the cost of obtaining a lower statistical significance of the results. During asynchronous reconstruction, most of these tasks can process the complete statistics, as long as their computational resource usage is within the expected budget.

3 Experience during the first year of operations

The Quality Control has been used throughout the commissioning and, subsequently, the first year of operations. Every subdetector in ALICE has developed one or several tasks, both for synchronous and asynchronous processing. After a year, almost 180 QC tasks have published 34'000 objects with thousands of versions each (see Table 1 for details).

3.1 Performance

In the O^2 system, the maximum data throughput of the processing topology can be treated as the throughput of the slowest processing step. Detailed benchmarks of each potential bottleneck were presented in [7]. It was shown that the Dispatcher can sustain around 100,000

	Sync data	Async data	Simulation
Tasks	121	35	23
Checks	69	8	2
Objects paths	17817	8263	7812
Objects versions	10 million	2 million	2.75 million

Table 1. Number of tasks, checks, and objects after 1 year of data taking (April 2023).

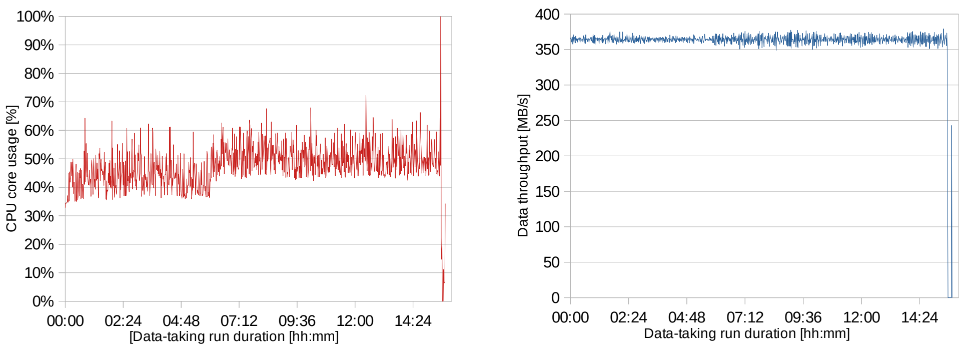


Figure 3. Performance of Mergers for the Time Projection Chamber's QC Tasks. On the left side the total CPU usage is shown, on the right one can find the total input data throughput.

messages per second when rejecting all, regardless of their size. When sampling, it can reach over 2 GB/s of data throughput for payloads in the range of [256 kB, 1 GB]. A single QC Task instance can be provided with 5 GB/s of data split into 2 MB messages, excluding the time needed for actual processing done in the user code. One Merger can process more than 2600 histograms per second with 62500 bins and a size of 250 kB.

In Fig. 3, we show an example of Mergers' performance observed in the production system. In this case, the merging topology is split into two layers, the first consisting of 3 Merger instances, which split the load equally. Such a configuration guarantees that the processing chain is sufficiently efficient without the risk of inflating input buffers with too many messages.

The new ALICE computing system read out 2000 PB of test and physics data during 2022 [8]. The data Quality Control system took part in the majority of test runs and all physics runs. Until April 2023, QC took part in asynchronous reconstruction of 192 physics runs, where most were processed a few times. In total, QC was enabled in 551,399 jobs.

We expect further increase in the number of tasks and objects. Since computations of separate tasks run independently, adding new tasks is trivial if CPU and network resources are available. Tuning the merger process topology allows for an increased number of objects. However, one has to monitor the number and size of objects stored in the repository, which we do by periodic detector code reviews and automated obsolete object cleanups.

3.2 Objects merging

Reducing large portions of data into merged objects presents several challenges during synchronous processing. On the one hand, minimizing the number of Merger processes is preferred due to the additional overhead each creates in terms of CPU and memory usage. On the other hand, when input messages reach the worker simultaneously (as shown in Fig. 4),

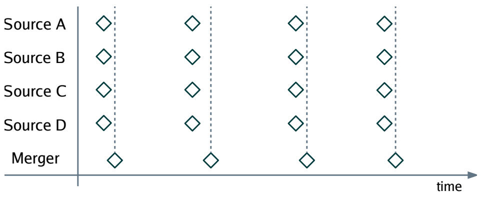


Figure 4. Data sources publish objects at the same time.

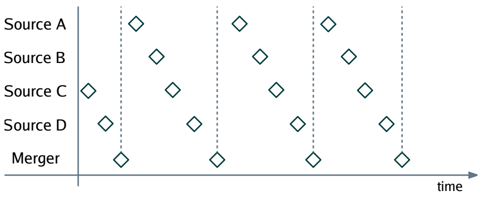


Figure 5. Data sources publish objects in shifted intervals.

input buffers grow larger, requiring temporary storage of all the messages that have not been processed yet. As a result, a computing node might run out of memory and might have to resort to killing processes.

In the QC framework, this problem is avoided in two ways. Primarily, input buffer sizes are configured to store the minimal possible number of messages allowed by the message queue library [5], which is one per connection. Still, despite this, additional messages might be stored by the underlying transport layer (TCP in this case), contributing to the total memory usage. When serving hundreds of nodes, each hosting a few tasks publishing payloads of dozens of MB, the peak usage may reach hundreds of GB. To mitigate this effect, data sources publish messages in regular intervals, but shifted in phase (Fig. 5). This way, the load on Mergers is distributed more evenly in time, reducing the need to queue the messages in input buffers before they are processed. However, since the data sources are purposefully not synchronized with the Mergers, the complete plots may contain uneven numbers of entries in different parts. An end-user, like a shifter in the experimental control room, may interpret this as if some parts of the particle detector saw less activity than the rest.

3.3 System monitoring

A single computing node hosts a processing topology, which can be imagined as a directed acyclic graph consisting of up to a hundred processes exchanging data messages. The whole system is made of hundreds of these nodes. Operating such a highly distributed processing system makes it a challenge to understand the dataflow and finding the root cause of issues. For example, one can struggle to determine which graph node blocks the processing by being too slow or in an unhealthy state when the whole system is stuck. In such case a shifter usually can only restart the data taking, while an expert has to carefully inspect the logs and metrics or use a debugger to track down an issue. Moreover, when running such jobs on a computing grid site (asynchronously), one usually cannot inspect the process topology while running, but can only investigate it post-mortem, which brings an additional difficulty. Thus, the system operators and developers must be provided with enough, but not overwhelming, amount of logs and performance metrics which can be browsed with accessible tools in order to understand an issue. We are developing an interface to visualize the dataflow between the processes in real-time in large production setups.

3.4 Dependencies

In the LHC Run 1 and Run 2 the dependencies of the DQM were few and well under our control. With the complexity and size of the O^2 framework, as well as the inter-dependency of its components, the number of libraries we rely on directly and indirectly has increased by a factor of four. Although some dependencies are under our control, many more are not. To

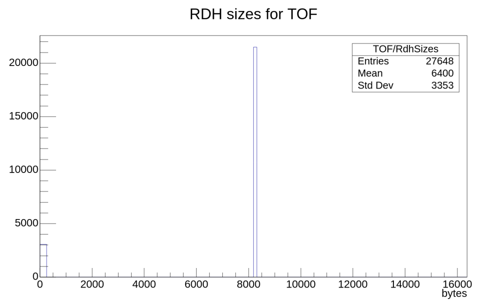
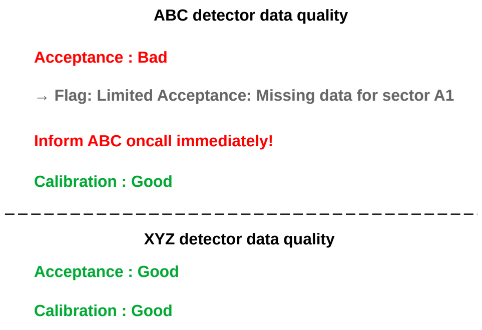


Figure 6. A plot produced by the general-use Data Acquisition Task, displaying distribution of Raw Data Header size in received data payloads.



ABC detector data quality

Acceptance : Bad

→ Flag: Limited Acceptance: Missing data for sector A1

Inform ABC oncall immediately!

Calibration : Good

XYZ detector data quality

Acceptance : Good

Calibration : Good

Figure 7. Quality summary page generated by the Quality Task.

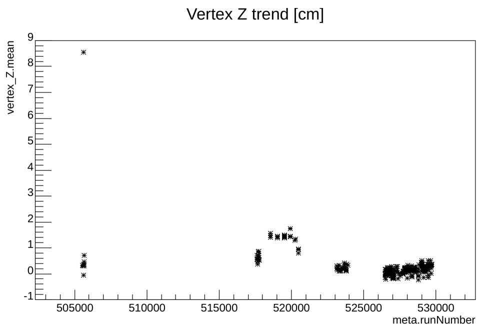


Figure 8. A trend of the collision vertex in the beam axis generated by the Trending Task.

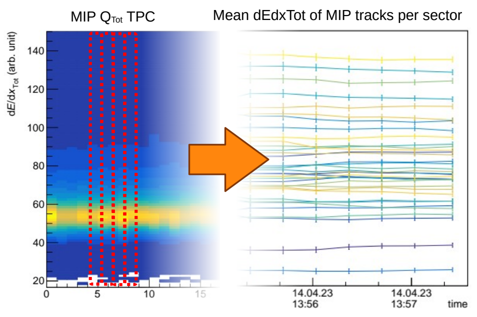


Figure 9. A trending plot where each line is derived from a separate slice of an input histogram. Courtesy of Marcel Lesch and Berkin Ulukutlu.

ensure the stability of our software despite this situation, we extensively utilize testing and Continuous Integration. Each Pull Request is tested on several platforms, and nightly builds are carried out along with functional and full system tests on our staging setup. This way, a version change in a dependency is tested extensively as soon as it is introduced.

3.5 Code reuse

Currently, there are 19 subsystem modules which correspond to either an ALICE sub-detector or a processing system component. They are developed by teams from institutes around the globe consisting of just one up to a dozen people, who might have a lot or limited experience in software development. To even out manpower inequalities of the contributing teams, several tasks are available for general use.

The Data Acquisition Task processes information available in raw data headers of any of the ALICE sub-detectors and displays the statistics in histograms, such as in Fig. 6. It is particularly useful as a plug-and-play task to use during early commissioning when there are no custom low-level tasks developed for a given sub-detector.

The Quality Task (Fig. 7) generates a canvas summarizing the selected Check results and adds a configurable message for the shift crew according to particular data quality. It was developed in collaboration with Andrea Ferrero (a Muon Chambers expert) and Marcel Lesch.

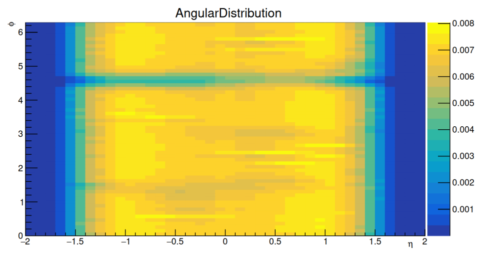


Figure 10. An acceptance hole in the layer 3 of the Inner Tracking System. Courtesy of Rik Spijkers and Ivan Ravasenga.

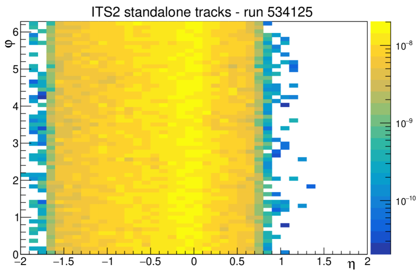


Figure 11. Vertex point shift seen in the Inner Tracking System. Courtesy of Rik Spijkers and Ivan Ravasenga.

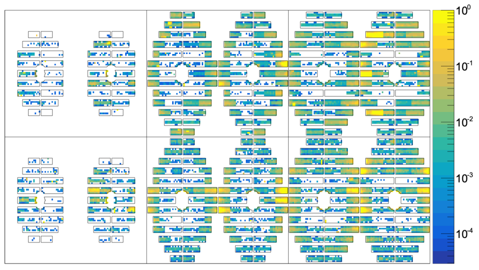
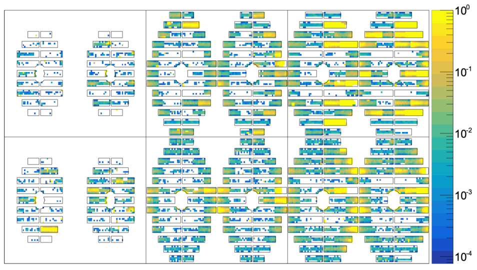



Figure 12. Visualization of hit rate (kHz) in the whole tracking system of the ALICE Muon Spectrometer. Each rectangle represents one muon tracking detector. Color code is proportional to rate (max. 1 kHz). The left plot shows the standard noise levels, the right plot demonstrates an increased noise. Courtesy of Andrea Ferrero.

The Trending Task (Fig. 8) allows users to create trends and correlations of any other objects generated within the QC framework. It reduces the input objects into a set of commonly used statistics, stored in a ROOT's `TTree` and visualized with the `TTree::Draw` interface. Applying it does not require developing any additional code; preparing a configuration file is sufficient.

One of the Time Projection Chamber module developers, Marcel Lesch, contributed an extended version of this task (Fig. 9). It lets users slice input plots into several parts and compute relevant statistics for each of those. Thus, it allows to trend observables of each detector sector independently.

In a typical physics data-taking run there are 19 instances of Trending Task, 4 instances of Slice Trending Task and 5 instances of Quality Task running.

3.6 Usage examples

The Inner Tracking System (ITS) is the vertex and tracking detector in the ALICE experiment. Their QC team runs 5 different tasks during data taking. One of the plots produced by the Tracks task is shown in Fig. 10. It visualizes the angular distribution of tracks seen by ITS. In this case, one of the staves was not providing data, which appeared as a dip in a small range in the ϕ axis. Fig. 11 illustrates the standalone track distribution of the ITS. During the concerned data-taking run, the shift crew spotted an asymmetry in η due to the vertex point shifted by 36 cm in the z-axis.

In Fig. 12, one can find plots visualizing the hit rate in the Muon Spectrometer. The Task which generates these histograms processes 100% of acquired data, which is essential for efficient detector diagnostics and allows to spot issues that normally go unnoticed when observing raw data rates or calibration data. In this case, the detector experts could discover a low-frequency, high-amplitude electronics noise, as seen in the plot on the right.

4 Future developments

Having the necessary core functionalities implemented, the future developments are driven by the continuous feedback from the users, the modules developers and the Run Coordination.

One key feature we plan on introducing next year is the possibility of storing reference data and comparing the QC objects with them. Moreover, we plan on leveraging the QC data we have produced until now by using Machine Learning to detect abnormal situations.

Major research efforts will be put into developing automatic procedures to let the user analyse only the parts of data acquisition runs which consist of good-quality data. Thus, instead of discarding runs which consist of both good and bad data, one will select the good and bad time intervals. Tagging good-quality data will require additional post-processing steps to combine quality flags issued automatically and manually, and then derive a global quality for a given data set.

5 Conclusion

The new ALICE data Quality Control system brings together two previously distinct worlds: one involves monitoring data quality during acquisition, while the other offers comprehensive feedback during the subsequent reconstruction process. Its viability, efficiency and usefulness had been confirmed during the detector commissioning in the years 2020-2021 and then while taking physics data.

To our knowledge, this is one of the largest data quality control systems worldwide. It is also the first in the high energy physics community to leverage the message passing technique and the actor model to such an extent. We hope that the experience shared in this paper will be useful for designers and developers of similar distributed processing systems.

References

[1] V. Barroso et al., *The new ALICE Data Acquisition system (O2/FLP) for LHC Run 3*, in *Proceedings of the CHEP 2023 conference (to be published)* (2023)

[2] G. Eulisse, D. Rohr, *The O2 software framework and GPU usage in ALICE online and offline reconstruction in Run 3*, in *Proceedings of the CHEP 2023 conference (to be published)* (2023)

[3] R. Brun, F. Rademakers, Nuclear Instruments and Methods in Physics Research A **389**, 81 (1997)

[4] C. Grigoras, *Calibration and Conditions Database of the ALICE experiment in Run 3*, in *Proceedings of the CHEP 2023 conference (to be published)* (2023)

[5] ZeroMQ, *ZeroMQ website* (2023), `https://zeromq.org/`

[6] The ALICE Collaboration, Tech. Rep. CERN-LHCC-2015-006. ALICE-TDR-019 (2015), `https://cds.cern.ch/record/2011297`

[7] P. Konopka, Ph.D. thesis, AGH University of Science and Technology, Cracow, Poland (2022)

[8] S. Chapeland, *Commissioning of the ALICE readout software for LHC Run 3*, in *Proceedings of the CHEP 2023 conference (to be published)* (2023)