Design of the data quality control system for the ALICE O²

Barthélémy von Haller¹, Patryk Lesiak² and Jacek Otwinowski³ for the ALICE Collaboration

¹ CERN, Physics Department, Geneva, Switzerland
² AGH University & IFJ PAN, Cracow, Poland
³ IFJ PAN, Cracow, Poland

E-mails: barthelemy.von.haller@cern.ch, patryk.lesiak@cern.ch, jacek.otwinowski@cern.ch

Abstract. 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). A major upgrade of the experiment is planned for 2019-20. In order to cope with a 100 times higher data rate and with the continuous readout of the Time Projection Chamber (TPC), it is necessary to upgrade the Online and Offline computing to a new common system called O². 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 in order to quickly identify and overcome problems while the latter aims at providing good quality data for physics analyses. 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 architecture and the design of the data Quality Control system that regroups the DQM and QA. In addition it presents the main design requirements and early results of a working prototype. A special focus is put on the merging of monitoring objects generated by the QC tasks. The merging is a crucial and challenging step of the O² system, not only for QC but also for the calibration. Various scenarios and implementations have been made and large-scale tests carried out. This document presents the final results of this extensive work on merging. We conclude with the plan of work for the coming years that will bring the QC to production by 2019.

1. ALICE
ALICE [1] is the heavy-ion detector designed to cope with very high particle multiplicities to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN LHC. It is optimized to study the properties of the deconfined state of quarks and gluons produced in such collisions known as quark-gluon plasma [2]. It is in addition well-suited to study elementary collisions such as proton-proton and proton-nucleus interactions.

2. O² Project
After Long Shutdown 2 (LS2), the LHC will deliver to ALICE Pb-Pb interactions at 50kHz. ALICE will therefore get 100 times more statistics than during Run 1. Moreover, the physics topics addressed by the ALICE upgrade [3] are characterized by a low signal to noise ratio, requiring large statistics, as well as a large background, making triggering techniques very inefficient if not impossible. Finally,
the TPC will require the implementation of a continuous readout to keep up with the 50kHz interaction rate in order to deal with the events pile-up and the trigger generated dead time.

For all these reasons, it is necessary to implement a new computing system for Run 3, called O\(^2\)[4], whose data flow is described in Fig. 1 and that is characterized by:

- The readout of all interactions.
- The intelligent compression of this large amount of data by reconstructing and calibrating it online.
- A common online and offline computing system.

![Diagram of O\(^2\) dataflow](image)

**Figure 1.** The O\(^2\) Dataflow. Processing is synchronous down to the storage. Rate is not decreased as opposed to the data volume.

### 3. Data Quality Control and Assessment

#### 3.1. Definition

Given the integration of the online and offline computing systems into the common O\(^2\) project, the offline Quality Assurance (QA) and the online Data Quality Monitoring (DQM) are combined into a single system called “Data Quality Control and Assessment” (QC). The QC is critical to quickly identify and overcome problems during data taking and to provide good quality data for physics analyses. It is also necessary to ensure that the calibration and reconstruction behave as expected, especially when running synchronously with the data taking.
3.2. Requirements
The key requirements of the QC are:

- 25000 QC objects produced globally, updated every minute (~400 objects per second).
- More than 100 tasks, most running in parallel (up to 1500 instances per task).
- QC objects: scalars, ROOT histograms and trees.
- QC objects size: 1kB – 50MB.

3.3. Architecture
The architecture of the QC is summarized in Fig. 2.

The QC tasks can run on the FLPs, on the EPNs or on dedicated QC servers. In the last case, they can receive only a sample of the whole data flow. The input of the QC tasks varies depending on the stage but covers the whole range of data types in the system from raw to analysis-ready objects. Their output is usually an histogram but there are no technical limitations. Mergers are necessary as the tasks will mostly run in parallel, for example on all the FLPs of a detector. Checkers receive a merged object and evaluate its quality by running a user defined algorithm. Its output, the object and its quality, is stored in the QC repository. A post-processing loop gives the possibility to trend and correlate the output of the QC tasks.

A detailed description of the architecture and design of the system can be found in [4].
3.4. Status
A first simple QC prototype was implemented in 2016 made of the QC framework and infrastructure key classes covering the concepts of Tasks, Checkers and Repository. Moreover, a first implementation of the merging system was developed in parallel. Both have been evaluated and tested using dedicated benchmarks that are described in the following sections.

4. Framework prototype benchmark

4.1. Description
The goals of the benchmark presented here were to assess the framework code developed so far, to identify possible bottlenecks as early as possible and to evaluate whether the performance meets the requirements. The mergers were not included in this test. The benchmark is described in Fig. 3.

![Figure 3. The framework benchmark topology and parameters. The connections are of type publish-subscribe. The checkers do not block the tasks; they drop objects if they cannot cope. Eventually all objects are checked but not all versions of all objects.](image)

The following metrics were collected from 1 randomly chosen task and 1 randomly chosen checker in each run of the test: CPU, RAM, network bandwidth, rate of histograms produced and rate of histograms treated by the checker. The benchmark was performed on a cluster of 40 machines equipped with 16 cores and 24 GB RAM connected via 1GbE. The database machine had a 40Gb/s Ethernet port.

4.2. Results

4.2.1. Code assessment. The benchamrk proved that the code is able to run and to fulfil the requirements in terms of features and stability.

4.2.2. Bottleneck(s) identification. As it was expected, the QC repository appears to be a potential bottleneck since it is a single component acting as a sink. It was particularly visible in this benchmark as the database ran on a machine with a limited disk IO. By asking the tasks to output the maximum possible number of histograms (about 600 per second) we can provoke a saturation of the database as shown in Fig. 4.
4.2.3. Performance comparison with requirements. As shown in section 4.2 the number of estimated QC objects published globally in the system is about 400 per second after merging. The benchmark showed that the system scales up to 500 histograms per second (see Fig. 5). Although the performance of the final system will depend on the users’ code, it is important at this early stage to make sure that the framework is able to perform at the required performance level.

Concerning the CPU and memory, the framework uses less than 10% of 1 core (see Fig. 6) for the load we expect (400 histograms per second). The memory scales linearly with the size of the objects.

Figure 4. Rate of histograms treated by a checker as a function of the number of checkers. The number of tasks equals the number of checkers in each test and thus the rate should be constant. The drop is due to the database getting saturated and blocking the checkers.

Figure 5. Comparison of the rate of histograms produced and treated in the system. The ratio represents the capacity of the system to handle the produced histograms. It drops above 500 histograms per second flowing through the system. This plot group results from tests using different number of checkers and producers.
Figure 6. A checker's CPU and memory usage as a function of the input rate (histograms per second). The dashed vertical line represents the expected rate in the production system. The wide spread of the data points is due to the aggregation of many different tests projected on a single dimension. The CPU and memory usage is expressed as a percentage of a full core and the total available memory respectively.

5. Merging infrastructure benchmark

5.1. Description
The merging infrastructure consists of producers and mergers (state-machine devices), which can run asynchronously as independent processes. The communication between devices is based on the messaging library ZeroMQ, using the push-pull socket pattern. The QC objects (ROOT histograms or trees) created by the producers are serialized and placed in the ZeroMQ socket buffers to be sent over the network. These messages are collected by the mergers, which perform the deserialization and the actual merging. The serialization, deserialization and merging of the QC objects are based on the ROOT functionality. The producers will be replaced in the future by the tasks running on the FLP and EPN nodes (Fig. 2).

The goal of the presented benchmark was to compare the performance of the merging infrastructure for various configuration parameters and to establish a baseline for future benchmarks. The following parameters were considered: the number of producers and mergers, the size and type of the QC objects and the size of the socket buffer. The choice of the parameters is dictated by the ALICE analysis requirements. The tests were performed varying one parameter at the time. For each test, the metrics CPU usage, memory usage, average merging time and number of merged QC objects per second, were collected.

The tests were performed on the PL-Grid cluster (“Prometheus”) [5] with the following hardware configuration for each computing node: 2 x Intel Xeon E5-2680v3, 24 cores, 2.5 GHz, 128 GB memory. In order to deploy the merging system on the PL-Grid cluster the Dynamic Deployment System (DDS) [6] was used. This system allows to deploy programs using a topology configuration file. DDS can be used on the local systems or on grid solutions with batch systems such as PBS and SLURM. The DDS intercom API was used to exchange control information between devices of the system.
The prototype of the merging infrastructure and its performance is discussed in detail in [7]. The selected results based on the collected metrics are presented in the next section.

5.2. Results

5.2.1. Number of producers and mergers. The system was tested varying the number of producers (2-500) and the number of mergers (1-4). During the first set of tests, each producer was sending QC objects (TH1F histograms) to one merger at a rate of 100 Hz. The socket buffer size was set to $10^5$ QC objects. The number of merged objects per second as a function of time is shown in Fig. 7. This number reaches a maximum for ~375 producers. The second set of tests was performed for the system consisting of 500 producers and a variable number of mergers (1-4). As indicated in Fig. 8, all objects can be merged when at least 2 mergers are included in the system.

![Figure 7. Number of merged objects per second by a single merger [9]. Color lines indicate the number of producers in the system.](image)

5.2.2. Size and type of the QC objects. Two sets of tests were performed varying the size (1kB – 50 MB) and the type (TH1, TH2, TH3, THn and TTree) of the QC objects. In the first set of tests, the system consisted of a variable number of producers (2-8) sending QC objects (TH1F histograms) to one merger. The socket buffer size was set to 1000 QC objects. Each of the producers generated the QC objects with rates from 0.3 Hz to 400 Hz, depending on the object size. The obtained results show that the efficiency of the CPU usage for the merging process is decreasing with the size of the QC objects. It is connected with the long time spent on serialization and deserialization of these objects. In the second set of tests, 20 producers were sending 0.5 MB objects of different type to one merger. The socket buffer size was set to $10^5$ QC objects. The results indicate that the memory usage during merging of THn and TTree objects dramatically increases in time, which might be related to non-optimised implementation of their ROOT merge functions.
5.2.3. Buffer size. Four producers were sending 1kB QC objects (TH1F histograms) to a single merger at a rate of 100 Hz. The size of the socket buffer varied from 1 to 1000 objects. The results show that the merger’s CPU was not fully used for any given configuration. However, the CPU usage is two times larger for a buffer of size 1. We consider that it is due to an increase of the number of operations that the processor has to perform to handle the buffer unloading.

6. Future work
The path to a full QC system ready for production is still long but well marked out. The limitations observed in the mergers benchmark concerning certain objects types will be further investigated and if possible alleviated. The framework will be integrated with the merging infrastructure as well as with the Experiment Control System and attached to a real data source. The correlation and trending post-processing will be developed, in addition to the web GUIs and clients able to display and manipulate ROOT objects. Finally, and in parallel, commissioning with detectors will be carried out.

References
[1] The ALICE Collaboration et al. 2008 The ALICE experiment at the CERN LHC JINST 3 S08002
[2] Cabibbo N and Parisi G 1975 Phys. Lett. B 59 pp 67-69
[3] The ALICE Collaboration et al. 2014 Upgrade of the ALICE Experiment: Letter of Intent J. Phys. G 41 087001
[4] Buncic P, Krzewicki M and Vande Vyvre P 2015 Technical Design Report for the Upgrade of the Online-Offline Computing System Technical Design Report ALICE 19
[5] Pl-Grid, http://www.plgrid.pl/en, accessed on 23rd November 2016
[6] The DDS User Manual v1.2, http://dds.gsi.de/doc/nightly/dds.pdf, accessed on 23rd November 2016
[7] Patryk Lesiak 2016 Development of the data quality assurance and visualization system for the Time Projection Chamber in ALICE experiment at the LHC Master Thesis AGH University of Science and Technology