Commissioning of the ALICE readout software for LHC Run 3

Sylvain Chapeland[1*], *for the ALICE collaboration*

[1]CERN – European Organisation for Nuclear Research, Geneva, Switzerland

Abstract. ALICE (A Large Ion Collider Experiment) is a heavy-ion detector studying the physics of strongly interacting matter and the quark-gluon plasma at the CERN LHC (Large Hadron Collider). During the second long shut-down of the LHC, the ALICE detector was upgraded to cope with an interaction rate of 50 kHz in Pb-Pb collisions, producing in the online computing system (O^2) a sustained input throughput of 3.5 TB/s.

In the past years, the O^2/FLP project built the new data-acquisition system capable of handling this load. It consists of 200 readout nodes, collecting the data transferred from over 8000 detector links to PCs memory by dedicated PCI boards. The readout software manages the hardware and software memory buffers used for DMA and inter-process communication. It initiates the data flow, performs on-the-fly consistency checks, formats the data, reports performance, and finally pushes the data to the local processing pipeline. The output is then sent by the data distribution software over 100Gb/s links to a dedicated event processing farm.

The readout software modular design allowed to address the manifold needs faced during the prototyping, installation and commissioning phases, which proved essential from the lab tests to physics production, like file replay and recording, or online multi-threaded LZ4 compression.

We will describe the hardware and software implementation of the O^2 readout system, and review the challenges met during the commissioning and first months of operation with LHC collisions in 2022.

1 Introduction

1.1 ALICE and the O^2 project

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

The detector has been upgraded [3] in the latest LHC long shutdown (2019–2020), and started physics operations in spring 2022. The detector data throughput increased by a factor ~100 compared to LHC Run 2, reaching 3.5 TB/s in LHC Run 3.

The Online-Offline system, named O^2 [4], is in charge of the streaming readout, processing the data on-the-fly in order to reduce the volume down to 130 GB/s initially recorded to storage. These demanding data-acquisition and processing steps are handled by a computing farm consisting (in summer 2023) of 200 nodes for readout (named FLPs –

* Corresponding author: sylvain.chapeland@cern.ch

First Level Processors) and 350 nodes for online reconstruction (named EPNs – Event Processing Nodes).

1.2 Readout

The new ALICE Data Acquisition system (DAQ) is presented in a separate paper [5], and the readout software [6] architecture has been described previously [7].

For convenience, the primary data flow is illustrated in Figure 1, and the main readout features are summarized below.

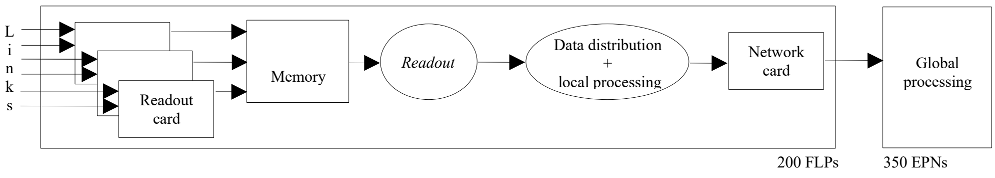

Fig. 1. Overview of the online data flow

- A high-density readout system receives the data from 8000 optical links. They connect the detector front-end electronics to 500 custom DAQ boards hosted in 200 standard servers (2x 10 cores CPUs with 96 GB RAM). There are two types of DAQ boards connected to the servers PCIe (Peripheral Component Interconnect Express) bus: a legacy PCIe gen 2 x8 (used in ALICE Run 1 and Run 2) and a new PCIe gen 3 x16 (developed for ALICE Run 3). They write the incoming raw data to the host memory by DMA (Direct memory access). The 15 sub-detectors may produce data with various payload formatting flavours (data packing options in detector, compression done in DAQ board FPGA), but shall all comply with one common data format and a well-defined RawDataHeader structure (RDH). There are different running modes (detailed later), producing a wide range of throughput and content. The data throughput at the input (links from the detector) is 3.5 TB/s, and reduced down to below 900 GB/s (aggregate network output of FLPs), mainly on the FPGAs of the DAQ boards. The maximum output throughput of the top FLPs is around 35 Gb/s.
- The readout software, a stand-alone running process (thereafter *readout*), controls the DAQ boards and provides memory buffers to them (detailed later). It checks the incoming data headers consistency and adapts the formatting to the O^2 software stack requirements. In particular, the data are sliced by time intervals called timeframes, corresponding usually to 32 LHC orbits (~3 ms) used for processing. *Readout* needs to detect and be robust against payload issues.
- The output of *readout* is the O^2 Data Distribution software [8], which organizes network transport (Infiniband) to EPN processing nodes. The local inter-process communication is done by means of a message-based format and shared memory. Some local processing tasks may also take place on the FLP readout nodes.
- *Readout* is controlled and synchronized with the other runtime processes by the ALICE Experiment Control System [9], which also provides key runtime parameters (like the ones needed for inter-process communication).

All in all, *readout* needs to cope with its input diversity and its output constraints, while fulfilling its primary goal of achieving the target throughput performance within the fixed resources allocated (CPU and memory).

We focus in this paper on the features added and used during the commissioning period, and performance results obtained during the first physics operations.

1.3 Memory buffers

The *readout* process is in charge of creating and managing the readout memory buffers in the FLP. A large memory chunk (typically, 32 GB in shared memory) is used to create pools (one per DMA channel) of data pages (typically, ~0.1-1 MB each), with following life cycle: 1) Free pages are taken from a pool and provided to the corresponding readout card. 2) Incoming detector link data are written to the pages by DMA. 3) Completed pages are pushed to *readout.* 4) After checking/grouping/formatting, they are forwarded to Data distribution (DD) and finally to EPNs. 5) Pages are given back to *readout* after use by DD, and made available again in the initial pool.

Each device is readout independently and data are aggregated together by a separate thread collecting all inputs.

The buffers are sized to accommodate the in-flight time between DMA and final network transfer, which may involve some processing steps with significant latency (transient peaks up to a couple of seconds if some component gets slow downstream). Also, finding the optimal page size is a matter of compromise: it should not be too big (a new page is used at the beginning of each timeframe, so the last page is usually not filled completely) nor too small (handling of too many pages creates an overhead both in software and hardware).

Each buffer can be pined to a specific Non-Uniform Memory Access (NUMA) zone of the host computer, to optimize the physical memory data flow between readout cards, memory, processors and network card.

2 Operations

2.1 Running modes

The readout system accommodates different running scenarios, each having its own specificities for the data taking and *readout* software:

- Physics runs: taken with p-p or Pb-Pb beam, they provide the valuable collision data for physics studies. The amount of data gradually increased with the accelerator commissioning to reach the design values of 3.5 TB/s. Especially for Pb-Pb (not yet seen with the current detector, expected autumn 2023), data taking efficiency is critical due to the short production time available in this mode and to maximize the statistics recorded.

- Cosmics runs are regularly taken when no beam is available, and provide useful data for detector calibration and alignment. Despite the low event rate, they still produce a fair amount of data because of the detector continuous data readout.

- Calibration runs, taken in commissioning and production, are some detector-specific measurements used for physics data calibration. In this situation there might be different trigger mechanisms, implying special handling of the data slicing by *readout* and distribution for processing. For example, it might be needed to group the data by channel (for one link, all the data of the run to be processed by a single host) instead of by time (full timeframe distributed round-robin across EPNs). This represent a marginal fraction of the total data.

- Commissioning runs: taken usually at high readout rates, to push the detectors and online components to their limits and understand the system under high load, like the *readout* memory buffers, in order to prepare for high occupancy Pb-Pb collisions. Readout rate can exceed the baseline bandwidth by a factor 2 or more in such tests.

- Synthetic runs: used to exercise the system when there is no beam. This includes the ability to emulate realistic data when the detector is not available. This is done by injecting data from synthetic files (generated from simulation), or actual recorded data which is replayed in the system. The *readout* software reads data from local files instead of the readout cards, and replaces some headers when necessary. This emulates realistic data patterns in the full online software chain.
- Technical runs: used to be a general category for test and debugging runs, now mostly split in the Synthetic and Commissioning runs.

In all these modes, it happens that some some issues are spotted and need further debugging. This possibly involves the ability of *readout* to record data locally, both for routine calibrations (this was already used in test facilities before commissioning), but also to exhaustively check the content of the raw data (before processing / reduction) in specific situations. It occurred that at the beginning of the run we had to investigate synchronisation, header, or buffering issues in the hardware, software, or firmware. There were also cases of errors detected in the payload. In this case, local recording can be enabled to save only the data pages with errors. Real-time compression of the raw detector data is also sometime necessary to allow recording on the local resources (the FLPs don’t have significant disk space, as it’s not their main purpose to record data). An option to enable the LZ4 library [10] packing allows to save efficiently the Inner Tracking System payload, with a pool of processing threads internal to *readout* making use of the CPU cores available. Some data monitoring features detailed later are also handy for runtime debugging.

2.2 Configuration

A key achievement of the commissioning was to tune and validate the parameters to be used for *readout*. Because of the many running modes and evolving requirements, *readout* requires a highly adjustable configuration, with 200 nodes doing different things at different times.

The critical settings for the data flow are related to the memory buffers. This is an iterative process requiring in-depth monitoring of the resources and buffer status at runtime, and to minimize possible back-pressure situations where detector data would be discarded. Once defined, these parameters do not change unless significant modifications are made to the data pattern. This was the case for example when changing the timeframe duration (it was reduced by a factor 4 at some point for processing needs).

During the commissioning phase, many features were used which are not strictly necessary for production. Both to handle ad-hoc situations and tests, or implemented as a temporary solution while waiting for final solution provided by other components of the O^2 system. It required flexibility to adapt to users needs.

At the moment, the main readout process accepts 164 different option keywords in the configuration syntax, and the list continuously increases to implement new features.

From the programming point of view, this requires some care to define, document, and maintain all the parameters. The simple approach used is as follows: a specifically formatted comment is inserted in the code wherever a parameter is loaded from the configuration. This includes a parameter name, its type, the default value, and a description. These fields are extracted from the code to automatically generate reference documentation (a table in a dedicated Markdown file) and a general-purpose interactive configuration editor uses as input the same list to generate the graphical user interface (i.e. when a new parameter is added, the editor does not require any code change). This ensures no syntax error on keywords in the generated file, and provides online hints on parameters while editing.

Although the main parameters are mostly static since they have been optimized (e.g. for the memory buffers), many configuration files have to be generated regularly (because of new features or new data sets). This is done by scripts, taking as input a set of reference settings (like the list of hardware cards and their location) kept in a versioning system, and producing ad-hoc configuration files for each readout instance and running mode. The main production files are stored in a central Consul [11] instance, whereas some test and reference files are distributed (together with the data replay files) locally on the FLPs by means of NFS, rsync over SSH, or by RPM packages depending on the needs.

Between 05/2022 and 08/2023, 83800 raw data files and 259000 *readout* configuration files have been generated and distributed on the FLPs for the replay mode in 17 releases, whereas there were 8 releases (200 files each, one per node) for the production reference configurations.

2.3 Monitoring

Appropriate monitoring tools are essential to understand the behaviour of the online data flow, both in real time and for later post-mortem analysis. The key components which have been useful (and sometime specially added) for this purpose are listed below:

- Time series plots of various metrics: amount of data readout and passed to O^2, associated rates (global, per host, and down to the per-link information), memory buffers usage, downstream latency and in-flight time of data pages, packets dropped upstream (i.e. in the readout cards, when host memory is full). They are collected, archived, and accessible online. Typical sampling period is 5s. The data injection and web-based GUI is provided by the O^2 monitoring system [12].
- Real-time view of all FLPs readout status. i.e. showing transition phases of each *readout* process instance on startup and runtime counters. A publish/subscribe mechanism exports a raw C structure with all relevant counters. They can be displayed both on the command line and in a specific GUI, showing all nodes at the same time and useful to identify blockers at a glance.
- Real-time status view of all memory pages for a given FLP. A sub-second refresh rate allows to see where the data pages are in use, and to possibly identify bottlenecks. It helps understanding issues like memory fragmentation, and is extremely useful to optimize memory allocation where needed. A text command-line tool is available for usage summary, and a GUI version displays all pages at once, animated with each status update. This is launched on demand for on-the-fly debugging, and data is not collected.
- Online data sampling, with access to RAW data headers and payload. This is launched manually as needed, with console dump and decoding of main fields, displayed in a human readable format. Some basic filtering rules allow to select the samples (e.g. a few consecutive pages from time to time, or all the data from a given link, etc).
- Archival of run statistics in an internal readout database and in the experiment logbook. This is used to check online counters discrepancies (e.g. number of triggers) and to get long-term reports on data taking.
- Log message are used for online issues reporting, and of course for later debugging. This can represent up to 10% of the FLP logs (which are in total ~7 M messages per day on average), based on *infoLogger* [13, 14]. Some mechanisms prevent messages storms, e.g. some error messages repeated with each new incoming data. The low-level output can be verbose when corresponding setting enabled, but messages can easily be filtered by

audience (shifter, support, expert). In addition to typical fields (timestamp, hostname), they are tagged with an integer identifier (as an error code, but not only for errors) associated to each readout message category. This allows to easily collect statistics when needed (e.g. to check some counters reported at end of run or to analyse frequency of some specific errors) and efficiently filter the target information in the large logging database.

3 Performance

An initial target of 70 Gb/s was used to validate hardware and software when the O^2 FLP system was implemented. This corresponds to twice the expected DAQ boards output throughput on one of the FLP nodes with maximum input data rate in production. The DAQ boards produce in average over all nodes ~2 GB/s per FLP, which corresponds for most nodes to a margin factor of 3x to 4x with respect to the validation criteria.

The commissioning rates have however escalated up to the usable FLP network bandwidth (around 80 Gb/s) to cope with increased detector needs. The readout system handles these new requirements flawlessly, and routinely runs at rates exceeding twice the initial design specification for production. An example run in Figure 2 shows a view of the monitoring interface, with a selected FLP rate on top (10.5 GB/s) and the aggregated rate of all FLPs below (1.26 TB/s), both with a smooth behaviour on the 10 minute window queried.

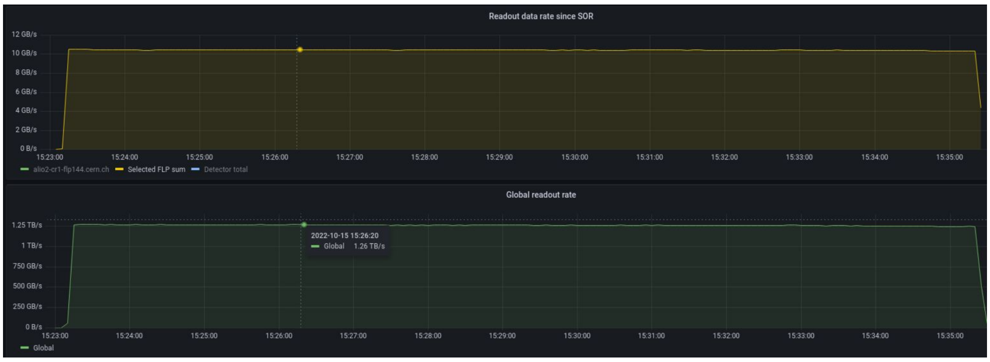

Fig. 2. Example plot of high readout data rates in production

The use of threads is generalized in *readout* and prevents single-CPU bottlenecks (as was seen for example on some memory repacking operations when parallel multi-core processing is not used). It is also useful to handle data compression (when enabled) and asynchronous calls to external components.

The *readout* CPU usage is minimal considering the rates (~5% of FLP system resources, i.e. 1 core out of 20, mostly used for data scanning and formatting), given that most data transfers are copy-less thanks to the DMA and shared memory. We have not observed significant impact of the NUMA settings, as there is plenty of headroom on the cross-socket memory links connecting CPUs and the different memory domains. These links, known as QuickPath Interconnect (QPI) on the Intel platforms, were measured to be used at less than 10-20%, even when enforcing memory and cores on opposite domains.

When testing the software on the development system, before releasing, start and stop cycles proved to be useful to trace possible issues in resources clean-up and interlock mechanisms. Such cycles are not used in production yet, but will also be handy to optimize setup speed: allocating and committing the memory may take time for large blocks, up to several tens of seconds when the process configures a block of 60 GB.

Figure 3 shows the accumulated data over time produced by *readout* during the commissioning and early ALICE Run 3 production. Thanks to the continuous efforts of the Run Coordination team and the experiment shift crew, the readout and full online system have been operated and exercised continuously since its release, providing valuable feedback for tuning and enhancement. Over 4000 Petabytes have been readout from the hardware and software sources into the memory of the FLPs. First physics runs started in summer 2022. The short LHC winter shutdown period is visible in the December to March period. First heavy-ion collisions are expected in fall 2023. The calibration data is a marginal fraction too thin to be clearly visible on the plot.

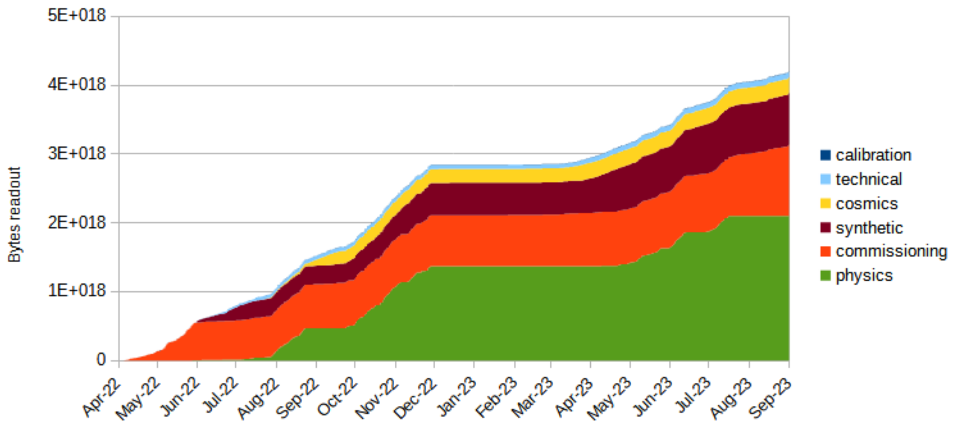

Fig. 3. ALICE data readout accumulated over time, by run type

The readout system is heavily used on a continuous and daily basis, averaging 500 MB/s/host for a 16 month period, and up to 1 GB/s/host for 1 month during the peak production seasons so far.

4 Conclusion

The *readout* software architecture has not changed since its design, and could cope with all the needs which emerged in the meantime. It internally makes use of a flexible thread-based pipeline with buffers and memory pools.

It was extensively tested in all possible data taking scenarios, and augmented with a number of tools to provide detailed feedback and insights about the data flow status. This helps spotting the source of possible issues at runtime (detector hardware, processing software) and react accordingly.

A long campaign of commissioning allowed to take data successfully since the start of LHC Run 3, with streaming readout running up to twice the design speed in production.

References

1. The ALICE Collaboration et al. "*The ALICE experiment at the CERN LHC*" JINST **3** S08002 (2008)

2. N. Cabibbo, G. Parisi Phys. Lett. B **59** pp 67-69 (1975)
3. The ALICE Collaboration et al. "*Upgrade of the ALICE Experiment: Letter Of Intent"* J. Phys. **G 41** 087001 (2014)
4. P. Buncic, M. Krzewicki, P. Vande Vyvre et al. *"Technical Design Report for the Upgrade of the Online-Offline Computing System"* Technical Design Report ALICE **19** (2015)
5. V.Barroso et al. *"The new ALICE Data Acquisition system (O^2/FLP) for LHC Run 3"* Proceedings of the CHEP 2023 conference (to be published)
6. The O^2 project, *"Readout"* URL https://github.com/AliceO2Group/Readout [accessed 11-08-2023]
7. S.Chapeland, F. Costa et al. "*Readout Software for the Alice Integrated Online-Offline (O^2) System"* CHEP 2018, EPJ Web of Conferences **214**, 01041 (2019)
8. The O^2 project, *"Data Distribution"* URL https://github.com/AliceO2Group/DataDistribution [accessed 11-08-2023]
9. T Mrnjavac et al. "*The ALICE Experiment Control System in LHC Run 3"* Proceedings of the CHEP 2023 conference (to be published)
10. The LZ4 library, URL https://github.com/lz4/lz4 [accessed 11-08-2023]
11. Consul, URL https://www.hashicorp.com/products/consul [accessed 11-08-2023]
12. The O^2 project, *"Monitoring"* URL https://github.com/AliceO2Group/Monitoring/ [accessed 11-08-2023]
13. S Chapeland et al. *"The ALICE DAQ infoLogger"* CHEP 2013, J. Phys. Conf. Ser. **513,** 012005 (2014)
14. The O^2 project, *"infoLogger"* URL https://github.com/AliceO2Group/InfoLogger/ [accessed 11-08-2023]