The evolution of the ALICE O^2 monitoring system

Adam Wegrzynek[1,*], and *Gioacchino Vino*[2]

[1]CERN, Geneva, Switzerland
[2]INFN, Bari, Italy

Abstract. The ALICE Experiment was designed to study the physics of strongly interacting matter with heavy-ion collisions at the CERN LHC. A major upgrade of the detector and computing model (O^2, Offline-Online) is currently ongoing. The ALICE O^2 farm will consist of almost 1000 nodes enabled to read out and process on-the-fly about 27 Tb/s of raw data. To efficiently operate the experiment and the O^2 facility a new monitoring system was developed. It will provide a complete overview of the overall health, detect performance degradation and component failures by collecting, processing, storing and visualising data from hardware and software sensors and probes. The core of the system is based on Apache Kafka ensuring high throughput, fault-tolerant and metric aggregation, processing with the help of Kafka Streams. In addition, Telegraf provides operating system sensors, InfluxDB is used as a time-series database, Grafana as a visualisation tool. The above tool selection evolved from the initial version where collectD was used instead of Telegraf, and Apache Flume together with Apache Spark instead of Apache Kafka.

1 Introduction

1.1 The ALICE Experiment

ALICE (A Large Ion Collider Experiment) [1] is a detector designed to study the physics of strongly interacting matter (the Quark–Gluon Plasma), produced in heavy-ion collisions at the CERN Large Hadron Collider (LHC). ALICE consists of a central barrel and a forward muon spectrometer, allowing for a comprehensive study of hadrons, electrons, muons and photons produced in the collisions of heavy ions. The ALICE collaboration also has an ambitious physics program for proton–proton and proton–ion collisions. After the successful Run 1 (2010-2013) and Run 2 (2015-2018) data taking periods, the LHC entered into a consolidation phase (Long Shutdown 2) and ALICE started its upgrade to fully exploit the increase in luminosity expected in Run 3. The upgrade foresees a complete replacement of the computing systems (Data Acquisition, High-Level Trigger and Offline) by a single, common O^2 (Online-Offline) system.

*Corresponding author: adam.wegrzynek@cern.ch

1.2 ALICE O^2

The ALICE O^2 computing system [2] will allow the recording of Pb–Pb collisions at a 50 kHz interaction rate. Some detectors will be read out continuously, without physics triggers. Instead of rejecting events, the O^2 system will compress the data by online calibration and partial reconstruction. The first part of this process will be done in dedicated FPGA cards that will receive the raw data from the detectors. The cards will perform baseline correction, zero suppression and inject the data into the memory of the FLPs (First Level Processors) to create a sub-timeframe. Then, the data will be distributed over EPNs (Event Processing Node) for aggregation and additional processing. The O^2 facility will consist of 198 FLPs and 500 EPNs. Each FLP will be logically connected to each EPN through high throughput links. The O^2 farm will receive data from the detectors at 27 Tb/s, which will be reduced to 720 Gb/s after processing.

2 Monitoring subsystem

The O^2 monitoring subsystem provides a complete overview of the overall health, detects performance degradation and component failures by collecting, processing, storing and visualising values from hardware and software sensors and probes.

As presented in Fig. 1, metrics are fed to the system from both monitoring library [3] (via Telegraf) and Telegraf [4] itself. The monitoring library provides a convenient C++ interface to inject metrics from other O^2 subsystems. The Telegraf agent monitors the operating system, services and fetches the status of O^2 specific hardware. All these metrics, converted to the InfluxDB Line Protocol format [5], are pushed over Kafka [6] protocol to the Kafka cluster. The selected metrics are aggregated over time and processed using Kafka Streams [7]. Eventually, they all reach a consumer which outputs them to InfluxDB [8] time-series database for permanent storage.

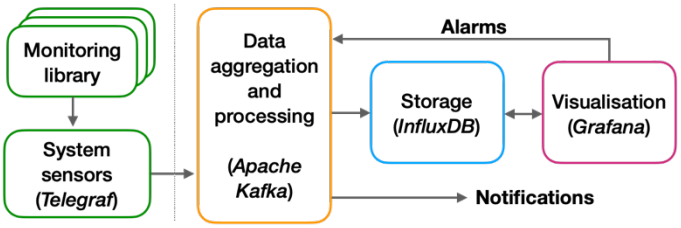

Fig. 1. Monitoring subsystem architecture and message flow

The metrics can be visualised in Grafana [9] which serves rich historical record dashboards. Grafana also generates alarms which are fed back to the Kafka cluster where they are translated into notifications.

2.1 Monitoring library

The monitoring library collects the metrics from other O^2 subsystems, eg: Readout [10], Quality Control [11] and DPL (Data Processing Layer) [12] framework. The library can push the metrics into any stage of the monitoring chain: Telegraf, Kafka or InfluxDB. It can also deliver a benchmark that can produce metrics at a given rate, or specially formatted values in order to measure latency between creation and storage times.

2.2 Telegraf

Telegraf probes the operating system and hardware to provide its overall health. It replaces the initially selected CollectD [13] due to the following limitations: protocol message format not flexible enough, strings values not supported, partial IPMI support, fewer plugins and difficulty with writing custom ones.

Telegraf scrapes metrics from Kafka exposed by Jolokia bridge [14] and provides cluster performance status, eg.: message rate or throughput per topic and partition (see section 2.3). It was also extended with custom plugins to monitor the O^2 PCI-e readout card called CRUs (Common Readout Units) [15] and the CCDB (Condition and Calibration Data Base) database. In addition, it receives metrics from C++ monitoring library instances over Unix sockets and compacts them into 1-second batches. Then, it passes all the metrics to a Kafka cluster through a built-in output plugin.

2.3 Kafka

Apache Kafka performs central metric aggregation and processing. It replaces the initially used Apache Flume and Apache Spark [13], mostly due to limitations identified in Apache Flume: manual cluster management (via configuration file), lack of scalability, lack of fault-tolerance and small user base.

Kafka uses several concepts and terms which are crucial to understand the following subsections:

- Broker: single Kafka server that is part of the cluster;
- Producer: entity producing messages (metrics) and publishing them to the cluster;
- Consumer: entity consuming messages (metrics) from the cluster in order to pass them to a non-Kafka endpoint or database;
- Topic: stream of messages that can be distributed over several partitions to provide scalability;
- Partition: part of topic running on a given broker, it can also be replicated to ensure fault tolerance (replication factor).

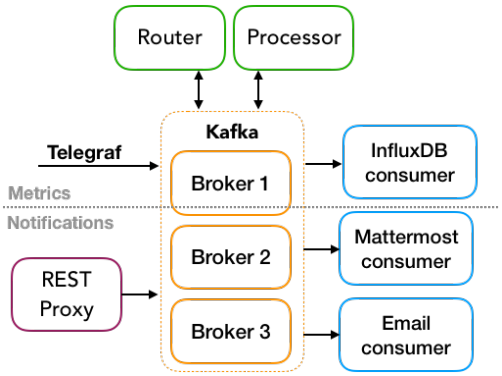

Fig. 2. Kafka cluster environment

As presented in Fig. 2, the Kafka cluster is composed of 3 brokers. Each topic is handled by multiple partitions and is also replicated. The cluster is divided into two logical parts which are isolated from each other: metrics and notifications.

Incoming metrics, from Telegraf (and monitoring library), are passed to the Router that redirects them either to one of the Processors or directly to the InfluxDB consumer.

The notifications originate in Grafana as alarms (see section 2.3.5) and they are injected to the cluster via REST Proxy [16] and then passed to one of the available notification consumers.

2.3.1 Metric router

The metric router is responsible for routing the metrics around the cluster by forwarding them to a desired topic. As InfluxDB Line Protocol allows a message to carry multiple fields. The measurements that are meant for processing are split in order to contain only one field per measurement.

2.3.2 Metric processor

A metric processor aggregates values over a configurable time window and applies one of supported functions: average, sum, minimum, maximum. In addition, it provides non-standard aggregation called "onoff". The "onoff" was introduced to decrease the rate of slowly changing binary values such as: state of the link (up/down), state of hardware (on/off), etc. The "onoff" keeps the last received metric of a given name in the cache and passes it to a consumer only if its value changes, otherwise the metric is immediately dropped, except for a minimum heartbeat of 1 value per 15 minutes.

2.3.3 InfluxDB consumer

The InfluxDB consumer writes metrics from a specific topic to the database over UDP (User Datagram Protocol). It uses multiple server-side ports in order to increase the writing performance. It chooses a destination port using a round-robin system.

The metrics could also be pushed to the database using Confluent InfluxDB Sink Connector [17], but this was ruled out as the connector introduces additional latency and does not provide optimal performance due to the usage of HTTP [13].

2.3.4 Alarms and notifications

The alarms are generated within Grafana by setting thresholds via the user interface. Grafana passes the alarms through Kafka REST Proxy to the cluster. The second source of alarms is Kafka itself, which may trigger alarms while processing values. The alarms are translated into notifications and pushed to one of the dispatchers:

- Mattermost [18]: online chat service commonly used at CERN;
- Email: email message including also the concerned plot;
- Web Notification API [19]: via dedicated topic to O^2 WebUI framework [20], and then over WebSocket protocol to user web browser which triggers native browser notifications.

2.3.5 Performance

To ensure that the Kafka cluster is capable of handling the desired amount of data a performance test was performed to plot the latency as a function of the metric percentile for different metric rates. The latency is defined as timestamp difference between the moment in which the metric is produced and the one in which it is stored in the database.

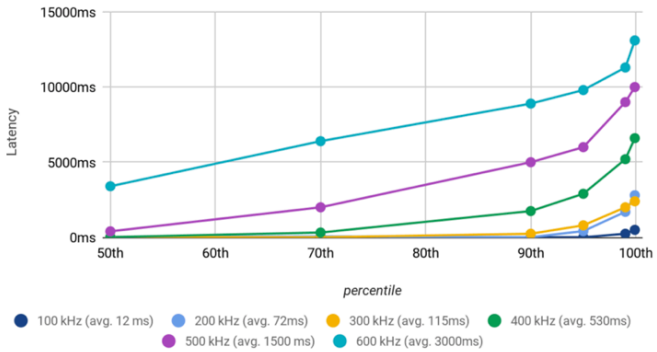

Fig. 3. Latency, defined as the difference between the production and storage times, as a function of the metric percentile for different metric rates

Figure 3 represents the scenario in which Kafka does not process any value and each measurement contains only one field (in a real system most of the measurements will carry multiple fields), which translates into a high storage metric rate. It is foreseen that the processing will reduce the metric storage rate by factor of 2. During the test the percentage of dropped messages between the InfluxDB consumer and the database never reached 0.1%. The test confirms that the performance of Kafka cluster is satisfactory. Additional details are available in [21].

2.4 InfluxDB

The InfluxDB time-series database supports downsampling which decreases the value resolution over time by bringing down the total database size. It is planned to keep the raw metrics for 7 days. After this time the metrics will be downsampled to one value per minute, which will decrease their volume by a factor of 5 and allow the storage of the metrics until the end of the calendar year.

2.5 Grafana

Grafana serves as a data visualisation tool. Currently, it supports rich historical record dashboards. After the Grafana 7 release, which enables backend data source plugins [22], it is foreseen to add real-time data source delivering metrics directly from Kafka to Grafana. This is necessary for providing robust dashboards for the shift crew in the ALICE Control Room and decrease the database query rate. Figure 4 presents one of the available dashboards that reports the status of a CRU readout card.

Fig. 4. Status of single CRU readout card: temperature, dropped packets and status of its 24 links

3 Outcome

The final system description and performance results presented above confirm that the monitoring subsystem will provide a robust and intuitive overview of the O^2 system status. The monitoring subsystem is nearly complete. The next steps mostly concern creating top-level, real-time dashboards and further integration with O^2.

References

1. ALICE Collaboration, *The ALICE experiment at the CERN LHC,* JINST **3** S08002, (2008)
2. ALICE Collaboration, *Technical Design Report for the Upgrade of the Online–Offline Computing System*, CERN-LHCC-2015-006 (2015)
3. *ALICE O^2 monitoring library*, https://github.com/AliceO2Group/Monitoring, accessed 2020-01-19
4. *Telegraf*, https://www.influxdata.com/time-series-platform/telegraf/, accessed 2020-01-23
5. *InfluxDB line protocol tutorial – InfluxData Documentation*, https://docs.influxdata.com/influxdb/v1.7/write_protocols/line_protocol_tutorial/, accessed 2020-01-23
6. *Apache Kafka*, https://kafka.apache.org/, accessed 2020-01-22
7. *Kafka Streams*, https://kafka.apache.org/documentation/streams/, accessed 2020-01-22
8. *InfluxDB – Downsampling and data retention*, https://docs.influxdata.com/influxdb/v1.6/guides/downsampling_and_retention/, accessed 2020-01-23
9. *Grafana - The open platform for analytics and monitoring*, https://grafana.com, accessed 2020-01-12
10. F. Costa, S. Chapeland, *Readout software for the ALICE integrated Online-Offline (O2) system*, EPJ Web Conf. **214** 03043 (2019)
11. B. von Haller, P. Lesiak, J. Otwinowski, *Design of the data quality control system for the ALICE O^2*, J. Phys. Conf. Ser. **898** 032001 (2017)
12. G. Eulisse, P. Konopka, M. Krzewicki, M. Richter, D. Rohr, S. Wenzel, *Evolution of the ALICE Software Framework for LHC Run 3*, EPJ Web Conf. **214** 03043 (2019)
13. A. Wegrzynek, G. Vino, V.C. Barroso, D. Elia, C. Grigoras, A.G. Ramirez, *Towards the integrated ALICE Online-Offline (O2) monitoring subsystem*, EPJ Web Conf. **214** 03043 (2019)
14. Jolokia, https://jolokia.org/, accessed 2020-01-23
15. J. Mitra et al, *Common Readout Unit (CRU) - A new readout architecture for the ALICE experiment*, JINST **11** C03021 (2016)
16. *Kafka REST Proxy*, https://docs.confluent.io/3.0.0/kafka-rest/docs/intro.html, accessed 2020-01-22
17. *InfluxDB Sink Connector*, https://docs.confluent.io/current/connect/kafka-connect-influxdb/influx-db-sink-connector/index.html, accessed 2020-01-23
18. *Mattermost*, https://mattermost.com/, accessed 2020-01-22

19. *Notifications – Web API*, https://developer.mozilla.org/en-US/docs/Web/API/notification/, accessed 2020-01-22
20. *ALICE O^2 Web UI Framework*, https://github.com/AliceO2Group/WebUi/, accessed 2020-01-23
21. G. Vino, D. Elia, V.C. Barroso, A. Wegrzynek, *A Monitoring System for the New ALICE O2 Farm*, ICALEPCS'19 (to be published)
22. *Grafana's Backend Plugin System*, https://grafana.com/docs/grafana/latest/plugins/developing/backend-plugins-guide, accessed: 2020-01-23