Journal of Physics: Conference Series **664** (2015) 082003 doi:10.1088/1742-6596/664/8/082003

MAD – Monitoring ALICE Dataflow

V Chibante Barroso[1], F Costa[1], C Grigoras[1] and A Wegrzynek[1,2] for the ALICE Collaboration

[1] CERN, Geneva, Switzerland
[2] Warsaw University of Technology, Warsaw, Poland

E-mail: Vasco.Chibante.Barroso@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). Following a successful Run 1, which ended in February 2013, the ALICE data acquisition (DAQ) entered a consolidation phase to prepare for Run 2 which will start in the beginning of 2015. A new software tool has been developed by the data acquisition project to improve the monitoring of the experiment's dataflow, from the data readout in the DAQ farm up to its shipment to CERN's main computer centre. This software, called ALICE MAD (Monitoring ALICE Dataflow), uses the MonALISA framework as core module to gather, process, aggregate and distribute monitoring values from the different processes running in the distributed DAQ farm. Data are not only pulled from the data sources to MAD but can also be pushed by dedicated data collectors or the data source processes. A large set of monitored metrics (from the backpressure status on the readout links to event counters in each of the DAQ nodes and aggregated data rates for the whole data acquisition) is needed to provide a comprehensive view of the DAQ status. MAD also injects alarms in the Orthos alarm system whenever abnormal conditions are detected. The MAD web-based GUI uses WebSockets to provide dynamic and on-time status displays for the ALICE shift crew. Designed as a widget-based system, MAD supports an easy integration of new visualization blocks and also customization of the information displayed to the shift crew based on the ALICE activities.

1. Introduction

1.1. The ALICE Experiment

ALICE (A Large Ion Collider Experiment) [1] 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). 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 ion. The ALICE collaboration also has an ambitious physics program for proton-proton and proton-ion collisions.

After 15 years of design and installation, the ALICE experiment successfully finished Run 1 in February 2013. The collaboration has then entered in a consolidation phase (denominated Long Shutdown 1 – LS1) to prepare for Run 2 which started in the beginning of 2015.

1.2. The ALICE DAQ

The ALICE Data Acquisition (DAQ) system [2], [3] is responsible for handling dataflow from the detector to permanent data storage in CERN's computer centre. Data generated by the many sub-detectors is sent via optical fiber Detector Data Links (DDL) to Local Data Concentrators (LDC),

computers which perform the readout of these event fragments. After validation, the subevents are sent from the LDCs to Global Data Collectors (GDC) for event building and temporary storage in the Transient Data Storage (TDS) system located in the experimental cavern. Finally, the data is transferred to the computer centre to be stored on tape.

During LS1, one of the points identified for improvement was the monitoring of the experiment dataflow - from the data arrival on the DAQ farm via the readout links up to its shipment to CERN's main computer centre. To address this requirement, the ALICE MAD (Monitoring ALICE Dataflow) system was developed.

2. Motivation and requirements

The goal of MAD is to provide the shift crew, the on-call experts and the Run Coordination with a clear view of the ALICE dataflow status. An initial set of monitoring metrics has been identified, with more foreseen for the future.

Following discussions both inside the DAQ team and with the ALICE Run Coordination, the following requirements have been identified:

- Collection of monitoring data should have a negligible impact on the existing DAQ machinery. Whenever possible, data collection should be handled by new and dedicated programs, therefore minimizing changes and introduction of new dependencies on existing software.
- The system should be able to handle metrics with frequencies up to 1 Hz.
- The system should allow for aggregation of lower level metrics into higher level metrics.
- Given the criticality of the tasks performed by the shift crew, the MAD displays available in the ALICE RunControl Center should present the information in a clear and intuitive way (i.e. current status should be easily understandable in a few seconds) and be highly dynamic.

3. Monitoring metrics

An initial set of base monitoring metrics was identified as representative of both the dataflow and the data acquisition operations status. Later, additional metrics were added once operational experience with MAD was acquired. The current list of base metrics is available in table 1.

Table 1. List of base monitoring metrics.

Metric	Entity (#)	Frequency (Hz)
Link backpressure status	DDL (1042)	1.0
Link active	DDL (1042)	1.0
Link connected	DDL (1042)	0.1
Bytes injected rate	LDC (220)	0.2
HLT pending decisions	LDC (220)	0.2
Events recorded rate	GDC (32)	0.2
Events recorded rate trigger cluster #1	GDC (32)	0.2
Events recorded rate trigger cluster #2	GDC (32)	0.2
Events recorded rate trigger cluster #3	GDC (32)	0.2
Events recorded rate trigger cluster #4	GDC (32)	0.2
Events recorded rate trigger cluster #5	GDC (32)	0.2
Events recorded rate trigger cluster #6	GDC (32)	0.2
Events recorded rate trigger cluster #7	GDC (32)	0.2
Events recorded rate trigger cluster #8	GDC (32)	0.2
Bytes recorded rate	GDC (32)	0.2

These base monitoring metrics are then aggregated into composed metrics that represent the status of either a sub-detector or the full experiment. The current list of composed metrics is available in table 2.

Table 2. List of composed monitoring metrics.

Metric	Entity (#)	Frequency (Hz)	Aggregation Type
Bytes injected rate	Sub-Detector (18) / ALICE (1)	0.2	SUM
Bytes recorded rate	Sub-Detector (18) / ALICE (1)	0.2	SUM
HLT pending decisions	ALICE (1)	0.2	MAX
Events recorded rate	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #1	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #2	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #3	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #4	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #5	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #6	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #7	ALICE (1)	0.2	SUM
Events recorded rate trigger cluster #8	ALICE (1)	0.2	SUM

4. General architecture

Figure 1 shows MAD's general architecture. At its core, MAD uses the *MonALISA* [4] (Monitoring Agents in A Large Integrated Services Architecture) tool to gather, process and distribute the different monitoring values.

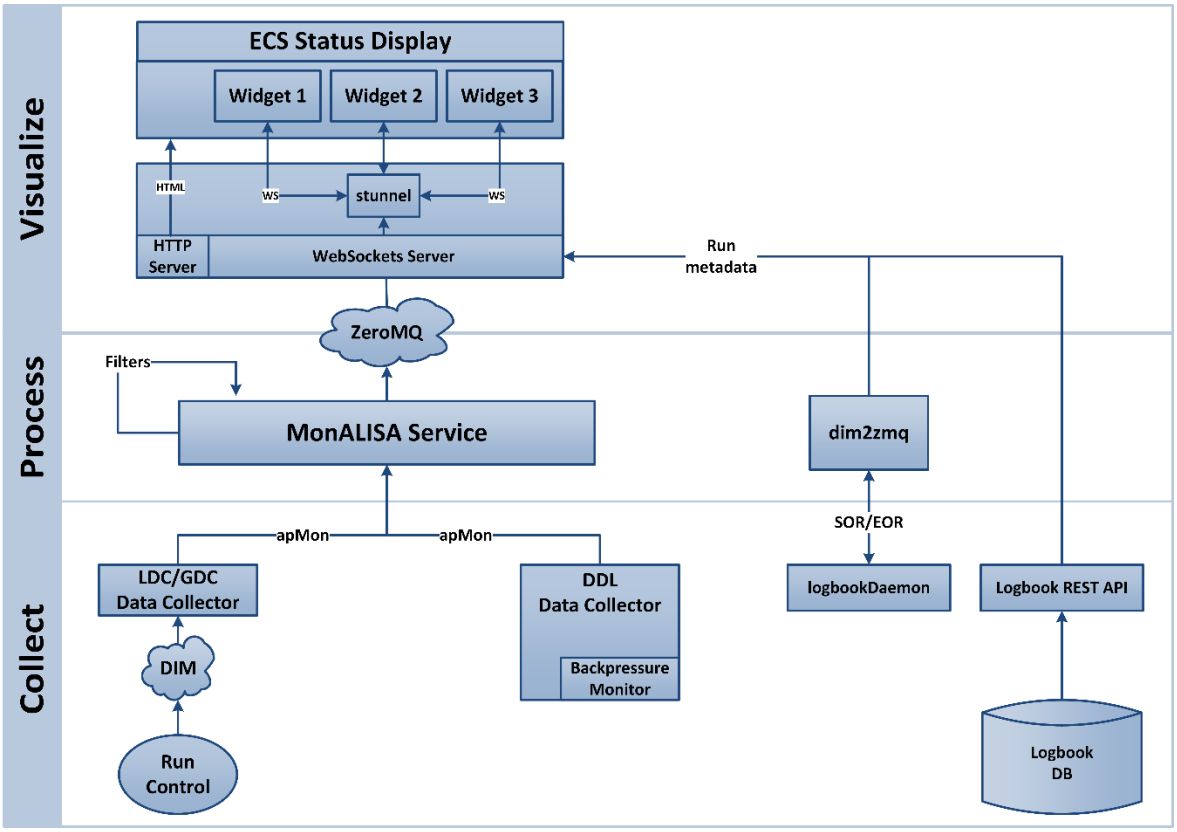

Figure 1. MAD general architecture. In the Collect phase, data collectors connect to existing data publication services, extract the relevant values and send them to a *MonALISA Service* via the *ApMon* library. In the Process phase, values for high-level metrics are generated. In the Visualize phase, the data is presented in highly intuitive and dynamic widgets that can be combined to create thematic dashboards.

Data collection is performed by dedicated C++ daemon processes that connect to existing data publications, extract the monitoring data and push it, via the *ApMon* library, to a MonALISA Service. The *logbookDaemon* [5] process provides notification of run start and stop events. The *Logbook REST API* [5] is queried at start of run to extract static run metadata such as participating sub-detectors, trigger configuration or recording mode.

Higher level metrics are generated by *MonALISA Data Filters*, which subscribe to the relevant base metrics, perform the aggregation and inject the new derived values back into the same *MonALISA Service*.

Visualization by the shift crew is implemented by a custom solution where initial content is provided by a standard Apache web server and monitoring values are then pushed by a WebSockets server to achieve a highly dynamic display. The content is organized in reusable widgets which can be combined to build thematic dashboards.

5. Data collectors

MAD *Data Collectors*, developed in C++ and executed as daemon processes, are responsible for gathering the raw monitoring values of the base metrics and pushing them to the *MonALISA Service*.

Two types of *Data Collectors* exist: "*LDC/GDC Data Collector*" and "*DDL Data Collector*". The first is responsible for reading the different counters published by the DAQ *runControl* servers (control processes that run on each of the DAQ nodes and publish several counters every 5 seconds) via the Distributed Information Management System (DIM) [6]. The second is responsible for reading the status of the readout links (using the readout card's driver, which accesses a specific register of the card via the PCIe bus).

Once the raw values are collected, they are sent to the *MonALISA Service* using the *ApMon* library (part of the *MonALISA* software), which sends the monitoring values as small UDP datagrams. To minimize the load, monitoring values are only generated when data taking activities are ongoing.

One of the big advantages of having dedicated data collectors that plug into existing data publications is that it avoids the need to change existing software, thus preserving the stability of ALICE critical data taking activities.

6. Monitoring values identification

In MonALISA, monitoring values are identified by the (Farm Name, Cluster Name, Node Name, Parameter Name) hierarchical tuple (one farm can have many clusters, one cluster can have many nodes and so on) and a timestamp. Although envisaging the common organization of computer farms, the fields of these tuples are free text and can therefore be used to represent any hierarchical organization.

In MAD, the monitoring values are represented in the following way:

- Farm Name: "daqP2" is the name of the *MonALISA Service* collecting and aggregating the data.
- Cluster Name: run number as defined by the DAQ software. For certain metrics which are not related to specific runs (e.g. "Link Connected"), special labels are used.
- Node Name: DDL equipment ID (e.g. 4096), DAQ role name (e.g. TPC-A-1), sub-detector name (e.g. TPC) or "ALICE" for metrics representing the whole experiment. To facilitate data filtering, prefixes (e.g. LDC) are added to certain values.
- Parameter Name: metric name (e.g. *hltPendingDecisions*).

The motivation to use the Cluster Name to represent the run numbers comes from the access pattern normally followed by the shifters and experts, which will almost always consult several metrics of the same run instead of a single metric across several runs (which will anyway be possible).

Since the Cluster Name field is used to store the run number, it cannot be used to aggregate nodes. Nevertheless, an implicit hierarchy exists between the different nodes, as shown in figure 2.

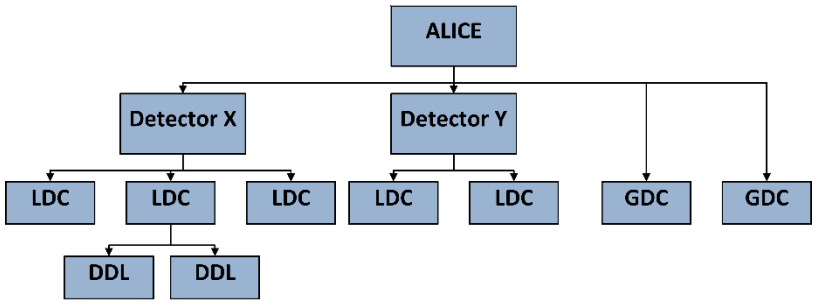

Figure 2. MAD node's hierarchy. The DDL roles are grouped into LDCs, the LDC roles are grouped into detector nodes. On the top, an ALICE node is used to represent ALICE-wide monitoring values such as total recorded data or event rate.

7. Data filters

MonALISA provides the possibility to have custom data filters that subscribe to a given set of monitoring values (based on a list of predicates that define which values should be received), perform any desired aggregation and re-inject new values to the *MonALISA Service*. These filters are developed in Java and executed by a pool of threads managed by the service.

MAD uses this feature to perform the aggregation of base monitoring metrics and the subsequent creation of higher-level metrics. The following aggregation filters are currently defined in MAD:

- *DDL Data Filter*: aggregates DDL link monitoring values into LDC, sub-detector and ALICE values.
- *LDC Data Filter*: aggregates LDC monitoring values into sub-detector and ALICE values.
- *GDC Data Filter*: aggregates GDC monitoring values into ALICE values.

Additionally, a special *ALICE Data Filter* subscribes to the top-level ALICE values and publishes them using the *ZeroMQ* [7] library (publish-subscribe pattern).

8. Visualization

In order to maximize the usefulness of the monitoring values, good and intuitive visualization tools are essential. Of special importance is how the data is presented to the shift crews, normally operating on an environment where decisions need to be taken in a time critical manner. With this in mind, MAD provides the Experiment Control System (ECS) Status Display, a custom made visualization platform that provides ALICE shifters with a clear and highly dynamic set of displays.

8.1. ECS Status Display overview

The *ECS Status Display* is a web-based application that uses modern web technologies to create intuitive displays for the ALICE shift crew. As shown in figure 3, the ECS Status Display connects to different data sources to gather the information to display. The PHP-based ECS Status Display WebSockets Server (*WSServer*) allows for data to be server-pushed to the widget-based client GUI.

8.2. ECS Status Display data sources

Data collection is performed by a dedicated *WSServer* thread. Local cache limits the impact on the data sources. Currently, the *WSServer* connects to three different data sources:

- *logbookDaemon*: provides run start/stop notification and list of ongoing runs via DIM. It is used for display reset and triggers data fetching from other sources. Given the lack of PHP support in DIM, a DIM-to-ZeroMQ C++ bridge ensures the communication between the DIM notification and the *Data Connector*.
- *Logbook REST API*: provides run configuration defined when the run starts via a simple HTTP request performed using the *libcurl* library.

- *MonALISA Service*: provides dataflow monitoring values. The communication is done via *ZeroMQ* (publish-subscribe pattern).

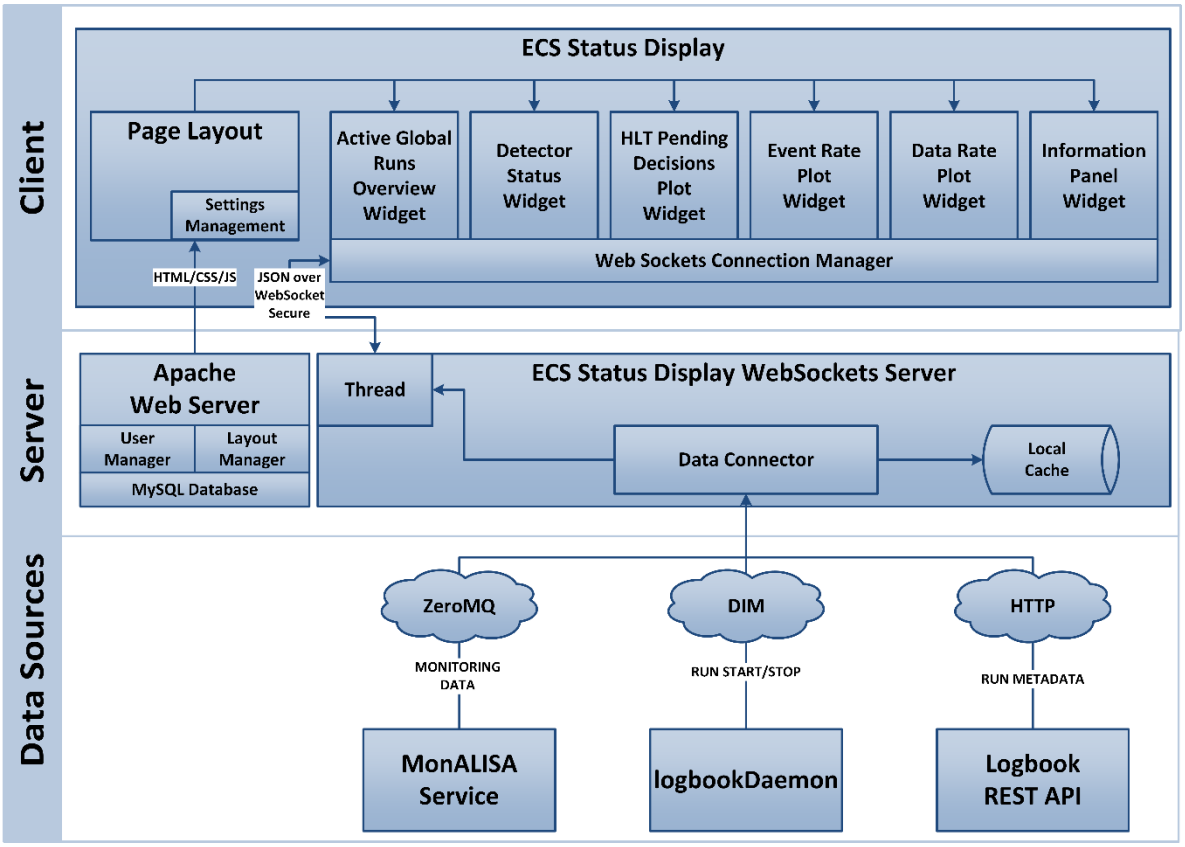

Figure 3. ECS Status Display general architecture. Different interfaces (DIM, HTTP, and ZeroMQ) provide access to existing data sources. The collected data is gathered by the *ECS Status Display WebSockets Server* (with local caching) and sent to the registered clients via the *WebSocket Secure* protocol in JSON format. The widget-based client layer provides high modularity and flexibility, with users being able to organize widgets in their own specific layouts.

8.3. ECS Status Display Server

The *ECS Status Display Server* layer is responsible for collecting the data from the data sources and pushing it to the Client layer via the WebSocket protocol (RFC 6455). The PHP-based *WSServer*, running as a daemon process, handles client connections via PHP *pthreads* that push data to the connected clients in JSON format.

Also in this layer, a traditional Apache webserver provides the initial client content (HTML, Javascript and CSS) and handles the user preferences client features.

8.4. ECS Status Display Client

The *ECS Status Display Client* layer is based on a widget system that allows for high modularity and flexibility. Widgets subscribe to event streams (normally dataflow monitoring metrics) and the arrival of a new value on one of these streams triggers the execution of a Javascript callback function which updates the display. Handling of event subscription, JSON distribution and callback execution are performed by a common *Web Sockets Connection Manager* component shared between the different active widgets. Plot-based widgets use the jQuery-based *Flot* [8] Javascript library, supporting multiple data series in a single plot and zoom window.

A MySQL database stores widget (name and three code files: HTML, CSS and Javascript) and layout (widget positions and sizes) definition. Users can organize widgets in layouts by defining their position (drag-and-drop interface) and size, thus allowing for a richer user experience.

Integration with CERN Single Sign On (SSO) services provides Authentication and Authorization which is ensured via Apache configuration settings and the *Shibboleth* middleware.

Based on the requirements collected from the ALICE Run Coordination, the following widgets are currently available:

- *Active Global Runs Overview Widget*: lists ongoing global runs, displaying high-level run configuration settings (e.g. run number or recording on/off), current run duration and number of recorded events.
- *Detector Status Widget*: displays in a matrix, for each ALICE detector, their role in each of the active trigger clusters. It also displays each detector's backpressure status, the recently executed calibration runs and the ongoing standalone test runs. Finally, it displays the number of recorded events for each trigger cluster.
- *HLT Pending Decisions Plot Widget*: displays a chart with the number of events waiting for a decision from the High Level Trigger subsystem.
- *Event Rate Plot Widget*: displays an event rate chart for ALICE as a whole and for each trigger cluster.
- *Data Rate Plot Widget*: displays a chart with the readout and event building data rates.
- *Information Panel Widget*: generic information panel for ad-hoc notifications.

Figure 4 shows a snapshot of the default layout as displayed in the ALICE Run Control Center.

Figure 4. Snapshot of default layout. On the left column, the following widgets are displayed (from top to bottom): Active Global Runs Overview Widget and Detector Status Widget. On the right column, the following widgets are displayed (from top to bottom): HLT Pending Decisions Plot Widget, Event Rate Plot Widget, and Data Rate Plot Widget.

8.5. MonALISA interactive client

As an alternative to the *ECS Status Display*, users can access the MAD monitoring data with the *MonALISA* interactive client. As shown in figure 5, this java-based GUI allows experts to access all values currently available in the *MonALISA Service*, which is useful for drill-down investigations by experts.

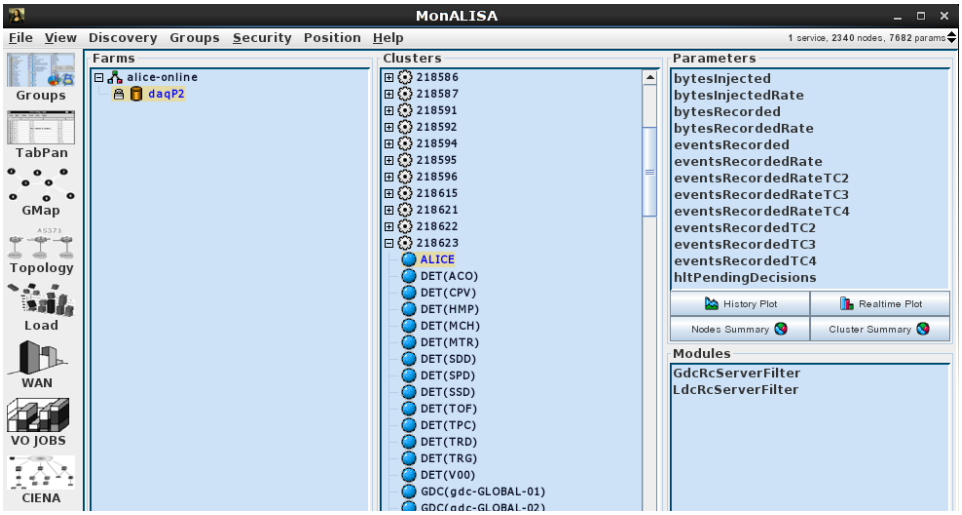

Figure 5. MonALISA interactive client. Run numbers are displayed as Clusters, dataflow entities (detectors, LDCs, GDCs) as nodes.

9. Future plans

Extension to other base monitoring metrics and the corresponding derived metrics will increase the scope of MAD. In particular, the extension to the Trigger and High Level Trigger ALICE subsystems will provide the shift crew with a better global view of the ALICE data taking activities.

The creation of an historical record using a *MonALISA Repository* will allow for *post-mortem* analysis of abnormal conditions.

10. Conclusion

Monitoring of the ALICE dataflow during Run 2 has greatly improved with the deployment of MAD, a MonALISA-based facility that will provide shifters and experts near real-time access to key metrics and statuses of the ALICE data taking operations. Special care has been put in the ALICE RunControl Center visualization, where shifters operate the experiment uninterruptedly.

References

[1] The ALICE Collaboration 2008 The ALICE experiment at the CERN LHC 2008 JINST 3 S08002
[2] ALICE Collaboration 2004 The technical design report of the trigger, data-acquisition, high level trigger, and control system CERN-LHCC-2003-062
[3] Carena F *et al* 2014 The ALICE data acquisition system *Nucl. Instr. Meth. Phys. Res. A* **741** 130–62
[4] Legrand I *et al* 2009 MonALISA: an agent based, dynamic service system to monitor, control and optimize grid based applications *Comput. Phys. Commun.* **180** 2472–98
[5] Chibante Barroso V *et al* 2010 The ALICE electronic logbook *J. Phys.: Conf. Ser.* **219** 022027
[6] Gaspar C *et al* 2001 DIM, a portable, light weight package for information publishing, data transfer and inter-process communication *Comput. Phys. Commun.* **140** 102-9
[7] http://www.zeromq.org, accessed on 06/May/2015
[8] http://www.flotcharts.org, accessed on 06/May/2015