THE CONTROLS AND CONFIGURATION SOFTWARE OF THE ATLAS DATA ACQUISITION SYSTEM FOR LHC RUN 2

A Kazarov[5], I Aleksandrov[1], G Avolio[2], M Caprini[3], A Chitan[3], A Corso Radu[4], A Kazymov[1], G Lehmann Miotto[2], M Mineev[1], A Santos[2], I Soloviev[4], M Vasile[3] and G Unel[4]

[1]JINR, Dubna, Russian Federation
[2]CERN, Geneva, Switzerland
[3]National Institute for Physics and Nuclear Engineering, Bukharest, Romania
[4]University of California, Irvine, USA
[5]NRC "Kurchatov Institute" - PNPI, St. Petersburg, Russian Federation

E-mail: Andrei.Kazarov@cern.ch[1]

Abstract. The ATLAS experiment at the Large Hadron Collider (LHC) operated very successfully in the years 2008 to 2013, a period identified as Run 1. ATLAS achieved an overall data-taking efficiency of 94%, largely constrained by the irreducible dead-time introduced to accommodate the limitations of the detector read-out electronics. Out of the 6% dead-time only about 15% could be attributed to the central trigger and DAQ system, and out of these, a negligible fraction was due to the Control and Configuration sub-system. Despite these achievements, and in order to improve the efficiency of the whole DAQ system in Run 2 (2015-2018), the first long LHC shutdown (2013-2014) was used to carry out a complete revision of the control and configuration software. The goals were three-fold: properly accommodate additional requirements that could not be seamlessly included during steady operation of the system; re-factor software that had been repeatedly modified to include new features, thus becoming less maintainable; and seize the opportunity of modernizing software written even before Run 1, thus profiting from the rapid evolution in IT technologies. This upgrade was carried out retaining the important constraint of minimally impacting the mode of operation of the system and public APIs, in order to maximize the acceptance of the changes by the large user community. This paper presents, using a few selected examples, how the work was approached and which new technologies were introduced into the ATLAS DAQ system, and how they were performing in course of Run 2. Despite these being specific to this system, many solutions can be considered and adapted to different distributed DAQ systems.

ATL-DAQ-PROC-2019-007
17 May 2019

1. Introduction

The ATLAS experiment [1] at the Large Hadron Collider at CERN relies on a complex Trigger and Data Acquisition (TDAQ) system [2] to gather and select particle collision data at unprecedented energy and rate. The TDAQ system is composed of a large number of distributed hardware and software components (about 3000 machines and more than 50000 concurrent processes) which, in a coordinated manner, provide the data-taking functionality

[1] © 2019 CERN for the benefit of the ATLAS Collaboration. CC-BY-4.0 license.

of the overall system. The Control and Configuration (CC) system [3] is responsible for all the software required to configure and control the ATLAS data taking; it provides essentially the glue that holds the various sub-systems together. The CC software ranges from high level applications to low level packages and it is designed following a layered component model (Figure 1). At the very bottom are common libraries and packages to deal, for instance, with threads, in house developed object persistency system and the libraries for the CORBA based inter process communication. Higher up are a set of services, like the configuration service or the process management. Above these layers is the so-called application layer, with the run control, the online recovery system, and a set of graphical user interfaces (GUIs) allowing the operator to act on the system.

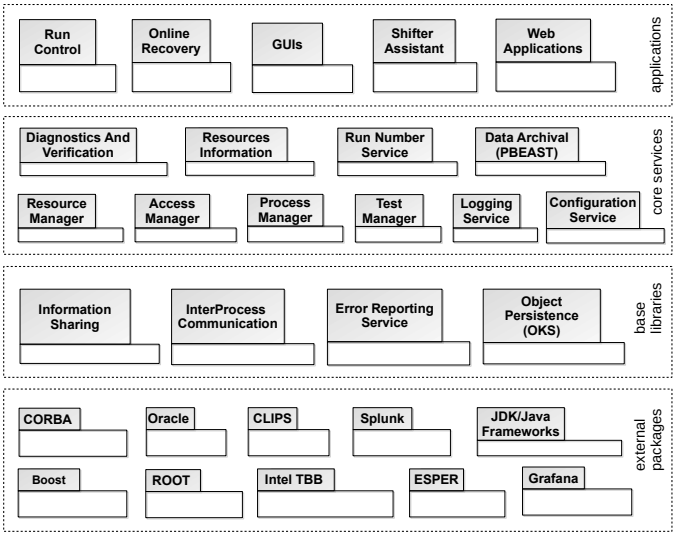

Figure 1. Control and Configuration software: high-level architecture.

2. From Run 1 to Run 2

Control and Configuration services and applications played an important role in successful TDAQ operations during data taking period in Run 1, allowing a high level of the data taking efficiency. Given the complexity and heterogeneity of the system, failures of hardware and software components were not unlikely. The CC system handled this quite well, mostly thanks to recently developed intelligent components, namely:

- the Expert System [4] for automation of routine actions and recovery scenarios, minimizing interruptions in data-taking, and
- the Shifter Assistant [5] for analysis of operational parameters and for shift crew alerts, helping operators to promptly undertake proper actions.

The intelligent components allowed experts to automate many routine operational tasks and to substantially reduce load on the operations team, whether in the control room and on-call experts.

Code quality, maintainability and evolution

CC software components are a part of the TDAQ software suite which is organized in more then 200 individual packages, including about 7000 source files and more then 1.6 M lines of C/C++, Java, Python and other code (not counting 3rd party packages). Code maintainability and quality is an important aspect of the stability and robustness of the whole TDAQ system.

Many packages were designed in late 1990's and developed in the first decade of 2000, and those packages have been maintained by a small group of developers through decades of ATLAS development and operations. Rapid development of open-source software projects, toolkits and appearance of new software technologies, standards, compilers and so on shall be also taken into account.

In these conditions, gradual software and code evolution is a process which must coincide with ongoing software development. The evolution should aim for simplification of the software architecture and clarity of the code, at the same time taking into account new functional and scalability requirements which come along with operational experience and evolution of the TDAQ system.

Lessons learned
Operation-wise, the architecture of the system should aim to guarantee robust and stop-less operations. In particular, the system should:

- focus on operational automation and intelligence;
- provide real-time monitoring of important operational parameters and shifter alerting;
- allow browsing the history of the operations, for better understanding the system and spotting potential and hidden problems;
- use of components testing widely to anticipate and to diagnose the problems; and
- involve detector experts and developers in formalizing and sharing the operational knowledge specific to the subsystem.

Software-wise, scalability and performance of the software and also simplicity, quality and maintainability of the code should be better addressed. The number of existing packages, including quite some old and historical code, and the time scale (15 or more years) for the system to be maintained in production suggest that software quality shall not be underestimated. The following goals need to be addressed:

- use 3rd party software toolkits and libraries where and when possible;
- improve performance and scalability of the software by exploiting modern C++11 features and utilizing modern threading techniques and multy-core CPUs;
- revise the old and historical code, and re-implement pieces of it where possible - even if all functional requirements are fulfilled, for the sake of better and simpler architecture, code clarity and maintainability; and
- follow the trends in software technologies, like web-based applications.

3. Upgrades for Run 2

The following sections give a description of the strategic choices and approach adopted for the development of some of the major components of the CC system.

3.1. Software refactoring: The Run Control and the Central Hint and Information Processor

The Run Control (RC) system steers the data acquisition by starting and stopping processes and by carrying all data-taking elements through well-defined states in a coherent way. Given the size and complexity of the TDAQ system, errors and failures are bound to happen and must be dealt with. In ATLAS this task is carried out by an expert system. The system was originally based on an embedded rule-based forward-chaining Expert System (CLIPS [6]), which was deeply integrated with the RC system. For Run2́, these two components were completely redesigned, using a distributed and hierarchical control tree and a separate central expert system

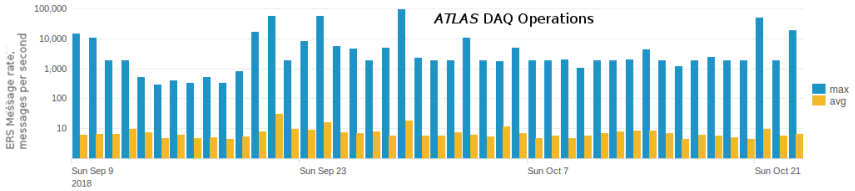

Figure 2. Rates of messages handled by MTS during data taking runs in 2018 [12].

application, the Central Hint and Information Processor (CHIP). The new design has been based on two pillars: (1) a pluggable modular schema in order to face the new requirements eventually emerging during LHC Run 2; (2) clear separation of responsibilities for what concerns pure application management, automation of procedures and error recovery and management.

The new RC is now assisted by the CHIP that can be truly considered as its "brain". CHIP is executed as a separate service with respect to the RC system and handles abnormal conditions, automates complex procedures, performs advanced recoveries and supervises the ATLAS data taking.

Additionally, it was perceived as a useful gain to use in CHIP the same CEP [7] technology (ESPER [8]) as in the Shifter Assistant implementation.

3.2. Requirements extension: Test Management

In a large, heterogeneous system such as the ATLAS TDAQ, it is essential to be able to verify the correct functioning of hardware and software components. Therefore, a TDAQ functional testing framework was developed and used already throughout Run 1. Additional requirements were identified with the experience gained during data taking, making it possible for experts to configure:

- the order in which tests should be executed for a component (the test sequence may dynamically change based on the result of completed tests);
- the order in which inter-related components shall be tested (the test sequence may change depending on the result obtained for the components); and
- which actions should be executed upon failure of a test or a component to further diagnose the issue or recover.

In addition, the test management functionality was required to be implemented in C++ and Java.

The details about new implementation are presented in [9].

3.3. Code modernization:

Message Reporting and Transfer System (MTS) MTS [10] underwent a review of the requirements that led to a complete redesign and new implementation to match its actual role (fast and reliable transport layer for TDAQ Error Reporting System [11]). The redesigned system is reliable and scalable, and its performance has been improved. The plot in Figure 2 shows the rates of messages reported in MTS during ATLAS operations in Sept-Oct 2018. In these conditions MTS handled a maximum rate of up to 100 kHz of delivered messages.

Resource Manager After an initial review and simplification of the requirements, this component underwent partial changes with the introduction of Boost multi-index containers. As a result the code base has been reduced by 40% compared to the previous implementation, thus leading to a more maintainable system [13].

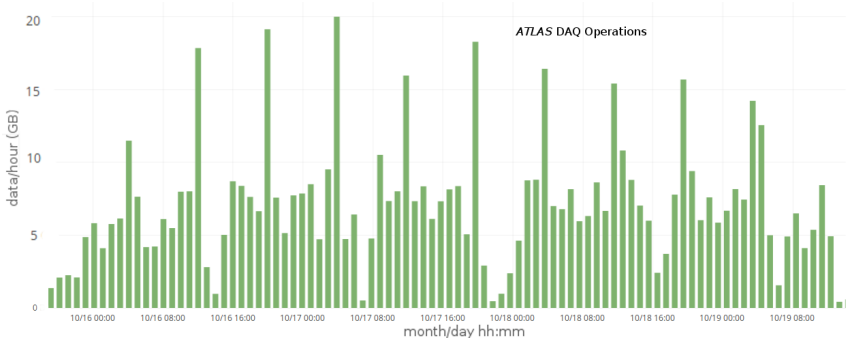

Figure 3. Performance of P-BEAST in physics runs in 2018 [12].

3.4. New developments: Information Archival System P-BEAST (a Persistent Back-End for the ATLAS TDAQ)
P-BEAST [14] is a new component designed and implemented to archive operational monitoring information for analysis by experts. It provides CORBA and REST interfaces for data access. Its implementation is based on Google protobuf (data persistence), CORBA (internal protocol and user programming interface) and libmicrohttpd (Web server). The plot in Figure 3 demonstrates how much data was archived per day by P-BEAST during operations in Sept-Oct 2018.

3.5. Web applications
P-BEAST Dashboard This web application offers an interface to visualize any operational monitoring data published by the TDAQ system through configurable and customizable dashboards. The data is provided by P-BEAST and the application is based on the Grafana project [15], adapted to support a custom data source within the AngularJS framework. An example of an operational dashboard is shown in Figure 4.

ELisA The ATLAS electronic logbook (ELisA) [16] is a web application used to record and share messages about ATLAS data taking activities by system operators, experts and automated services (Figure 5). The information is stored in an Oracle database. The adoption of an MVC-driven architecture has allowed to focus code development on specific features of the project, while profiting from the reliability of established third-party technologies such as the Spring framework. The tool also provides an HTTP-based REST API [17], such that other programs can access its features.

4. Conclusions and Outlook
The Control and Configuration software has contributed to the physics results obtained by the ATLAS experiment during Run 1 by ensuring smooth and efficient data taking. It was completely revised during 2013-2014 in order to accommodate additional requirements, improve maintainability and profit from advances in IT technologies. All this was done applying minimal changes to APIs, such that the large amount of client code would not need significant adaptations. The Control and Configuration software has proved to be stable and well performing in LHC Run 2 (2015-2018). It is designed to face the new challenges that will arise in Run 3 operations, after further modernization in different components foreseen during Long Shutdown 2. This experience has also demonstrated that the overall modular architecture of the control and configuration system is flexible and supports partial upgrades, as well as step-wise modernization of its components. This is fundamental for a system that is foreseen to run for the next 20 to 30 years and that will undergo several more upgrade iterations.

Figure 4. Screenshot of a P-Beast dashboard with TDAQ operations plots.

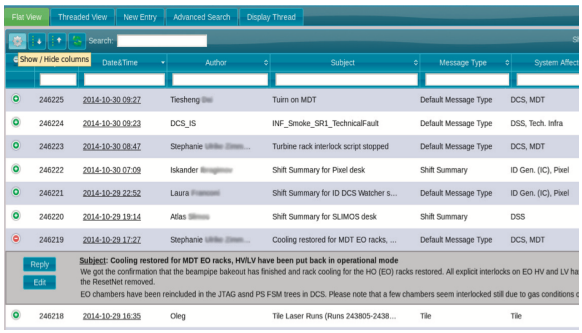

Figure 5. Screenshot of ELisA web page with log entries.

References

[1] The ATLAS Collaboration (ATLAS) 2008 *JINST* **3** S08003
[2] Abolins M *et al.* (ATLAS TDAQ) 2016 *JINST* **11** P06008
[3] Lehmann Miotto G *et al.* 2010 *Nucl. Instrum. Meth.* **A623** 549–551
[4] Anders G, Avolio G, Miotto G L and Magnoni L 2015 *J. Phys. Conf. Ser.* **608** 012007
[5] Kazarov A, Lehmann Miotto G and Magnoni L 2012 *J. Phys. Conf. Ser.* **368** 012004
[6] Giarratano J C and Riley G 1989 *Expert Systems: Principles and Programming* (Pacific Grove, CA, USA: Brooks/Cole Publishing Co.) ISBN 0878353356
[7] Complex Event Processing `https://en.wikipedia.org/wiki/Complex_event_processing` accessed: 2019-04-29
[8] ESPER `http://www.espertech.com` accessed: 2019-04-29
[9] Kazarov A, Radu A C, Avolio G, Miotto G L, Soloviev I and Unel G 2018 *J. Phys. Conf. Ser.* **1085** 032054
[10] Kazarov A, Caprini M, Kolos S, Lehmann Miotto G and Soloviev I 2014 *Proceedings, 19th Real Time Conference (RT2014): Nara, Japan, May 26-30, 2014* p 7097447
[11] Kolos S, Kazarov A and Papaevgeniou L 2015 *J. Phys. Conf. Ser.* **608** 012004
[12] Public DAQ Operations plots `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ` accessed: 2019-04-29
[13] Aleksandrov I, Avolio G, Miotto G L and Soloviev I *J. Phys. Conf. Ser.* **898** 032016
[14] Soloviev I and Sicoe A 2014 *Proceedings, 19th Real Time Conference (RT2014): Nara, Japan, May 26-30, 2014*
[15] Avolio G, D'Ascanio M, Lehmann-Miotto G and Soloviev I 2017 *J. Phys. Conf. Ser.* **898** 032010
[16] Corso Radu A, Lehmann Miotto G and Magnoni L 2012 *J. Phys. Conf. Ser.* **396** 012014
[17] Corso-Radu A, Magnoni L and Garcia R M 2013 *2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC)* pp 1–4 ISSN 1082-3654