PROCEEDINGS OF SCIENCE

Recent updates of the Control and Configuration of the ATLAS Trigger and Data Acquisition System

Riccardo Maria BIANCHI*
CERN
On behalf of the ATLAS TDAQ Collaboration†
E-mail: rbianchi@cern.ch

The ATLAS experiment at the Large Hadron Collider at CERN relies on a complex and highly distributed Trigger and Data Acquisition (TDAQ) system [3] to gather and select particle collision data at unprecedented energy and rates. The Control and Configuration (CC) system is responsible for all the software required to configure and control the ATLAS data taking. This ranges from high level applications, such as the graphical user interfaces and the desktops used within the ATLAS control room, to low level packages, such as access, process and resource management. Currently the CC system is required to supervise more than 30000 processes running on more than 2000 computers. At these scales, issues such as access, process and resource management, distribution of configuration data and access to them, run control, diagnostic and especially error recovery become predominant to guarantee a high availability of the TDAQ system and minimize the dead time of the experiment. And it is indeed during the data taking activities that the CC system has shown its strength and maturity, featuring a great scalability against the always increasing number of software processes in the TDAQ system and implementing several automatic error recovery procedures in complex and sophisticated scenarios.

This paper gives an overview of the new functionalities and recent upgrades of several CC system components, with special emphasis on speed and reliability improvements and on optimization of the user experience during operations.

The 2011 Europhysics Conference on High Energy Physics, EPS-HEP 2011,
July 21-27, 2011
Grenoble, Rhône-Alpes, France

*Speaker.

†The ATLAS Trigger/DAQ Authorlist, version 4.1, ATL-DAQ-PUB-2011-002, CERN, Geneva, 2011, http://cdsweb.cern.ch/record/1386334.

ATL-DAQ-PROC-2011-031
30 October 2011

PoS(EPS-HEP2011)401

1. Optimization and New Technologies

The ATLAS experiment is up and running, taking data successfully since more than one year. The first goal of the TDAQ achieved — that is, recording data from the experiment in a safe and reliable way — it was time to improve the core packages to optimize the usage of the available resources; and to explore new technologies, developing ancillary tools to get better system monitoring and to help the human operators during the configuration and data-taking operations. Recent improvements of the ATLAS TDAQ Control and Configuration system (CC in the following) and additions to it are presented in this paper.

2. Improving Operation timings through parallelization and network usage optimization

To setup ATLAS in order to take data, we have to go through a Finite-State-Machine (FSM), up to the "RUNNING" state, configuring all necessary services and applications. Each step is now optimized to efficiently use CPU and Network resources.

Figure 1 shows the outcome of the recent improvements and optimizations of the mechanism which configures and runs the ATLAS experiment. In Figure 1(a) one can see the trend of the total "INITIAL" time. This is the time spent interacting with the underlain operating system, starting all the services and applications involved in the data acquisition system of the experiment. Once all applications received the signal, they start; and the whole system goes from the "BOOTED" to the "INITIAL" state of the FSM. The computer racks of the TDAQ system contain ~30 nodes each, with a total of ~450 applications running on each rack. Starting all the applications took a lot of time with the old implementation (blue dots on the graph), showing a quadratic dependence on the number of instances per node ($\sim x^2$). The optimization introduced with the new implementation (red dots on the graph) reduced the starting time, which is now linearly dependent on the number of instances per node ($\sim x$).

Figure 1(b) shows the CPU cores occupancy, with the blue line. The first, fourth, fifth and eight peaks show the CPU load of normal run configuration (the other peaks are from test runs and so they are not indicative of the normal behaviour). The plot shows that the system is now capable of using most of the available CPU (more than 50 %) while leaving a safety margin for the extra work, despite the huge load due to the large size of the full ATLAS configuration. The red line shows the total memory usage, which is stable over the run, once the configuration is loaded in memory.

Figure 1(c) shows the total time spent by ATLAS during the various transitions of the FSM, resulting in a total "cold start" time of ~440 seconds (from booting all the applications, up to running) and a "stop/start" time of ~200 seconds (from a running state, down to a "STOPPED" state and then up again). Those are very good values for a so complex system like ATLAS, involving tens of thousands of applications running on several thousands of machines.

During operations, configuration data and commands have to be sent to many systems and applications through a dedicated 1Gbit network link, and a recent upgrade of the code let the system effectively use all the bandwidth available. Figure 1(d) shows the result in term of network usage. As one can see from the plot, the network is only used during state transitions by the applications

involved in the configuration of the data acquisition; with a peak during the "CONFIG" transition, when the whole ATLAS configuration is served to all clients. Between transitions the resources are left wholly available to data transfer services.

3. Optimization of the Configuration DataBase access

The run configuration of the ATLAS detector is stored in a custom distributed database. Configuration data have to be promptly accessible, when needed, to all applications which are responsible to run the detector. Bottlenecks have been spotted and a more effective code has been recently implemented, involving a multi-threaded parallelization of the DB loading, a complete rewriting of the slowest algorithms and a more effective memory usage. Now the ATLAS configuration can be concurrently modified by many users at the same time. The reload of the configuration data after any modification is then centrally managed, to ensure a safe and stable data-taking. Configuration database Read / Write operations are now ~6 times faster.

4. Automation via Artificial Intelligence: the Shifter Assistant

The ATLAS detector system is operated by a non-expert shift crew, assisted by a set of on-call experts providing knowledge and assistance for specific components. Operational tasks include operational procedures to run the system, periodic checks and controls and procedures to notify experts in case of problems. But computers are better than humans in automation. Hence checks and controls have been recently automated in order to reduce and simplify shifters tasks with pertinent messages and suggestions, providing more detailed information at the same time. Thus — avoiding repetition and formalizing and storing the knowledge from experts — checks can be more effective, minimizing system down-time and dealing faster and more effectively with errors and failures.

The Shifter Assistant uses an open-source Complex Event Processing (CEP) engine from "EsperTech"[1] to process streams of information from several sources in real time, in order to detect patterns in time windows, and reacting to them producing alerts and messages, served to clients through the ActiveMQ dispatcher[2]. The knowledge base is a list of directives and suggestions whose target is the person sitting at the shifter desk. Figure 6 shows the architecture of the Shifter Assistant.

5. ELisA, the new interface to the ATLAS operation log messages DB

During operations, when the detector is running, a lot of messages and logs are exchanged among shifters, experts and automatic services. And a real-time analysis of those messages helps to spot and understand problems. That's why the ATLAS experiment needs a reliable fast interface to the log messages database.

The current implementation, called "ATLOG", has many major problems, mainly due to its monolithic architecture, where the same application acts both as server and as web client; this

[1] http://www.espertech.com

[2] http://activemq.apache.org

makes the package difficult to maintain and deploy. It also lacks of a good Oracle interface and multi-threading support, resulting to a poor scalability with the number of messages to handle, and in a quite slow and unstable behaviour.

The new implementation, called "ELisA" (or "Electronic Logbook for the information storage of ATLAS") uses the Spring Java[3] web application framework: a mature, stable framework with an optimized interface to Oracle. Using such a framework allowed us to adopt a new client-server architecture implementing a Model-View-Controller design pattern, with web pages dynamically created using JSP, and the Oracle interface developed through the JDBC package. All this results in a modern web application, which scales well with the number of messages to handle.

In Figure 3 the comparison between the old and the new implementation is presented. Speed and scalability tests show that the old application takes a lot of time to fetch and display e-log data ($\sim x^2$ in time, where x is the number of messages), putting a huge overhead on top of Oracle operation time; and it can not handle more than few thousands of entries. While the new application linearly scales with the number of messages, and it only puts a little overhead on top of the pure Oracle operation time, letting the whole system to cleanly and easily handle tens of thousands of entries.

6. DBE: the new Configuration Database editor

The old editor used to edit the ATLAS Configuration database so far, had been conceived and implemented many years ago upon the Motif graphical libraries[4], and it lacks of many features which one expects to find in modern graphical user interfaces.

The new Configuration editor, called "DBE", is based on the "Qt" graphical libraries[5], and it has all the features that one could expect from a modern editor, like multi-threaded DB access for faster page rendering, bunch editing of multiple objects at the same time, Drag&Drop with type check capability, UNDO/REDO actions and customizable user views with user settings storage. Moreover it allows Read/Write operations both to remote DB servers and to local database files, and it has C++ and Python interfaces to let the user develop plugins and scripts to make tests and to operate on the DB.

References

[1] Giovanna Lehmann Miotto, et al.,*"Present & Future of the Configuration Control of the Atlas TDAQ"*, TIPP 2009, Tsukuba, Japan. Nuclear Instrumentation and Methods in Physics Research, Section A: Volume 623, Issue1, 1 Nov. 2010, Pages 549-551, http://dx.doi.org/10.1016/j.nima.2010.03.066

[2] ATLAS Collaboration, J. Instr. 3 (2008) S08003, 437pp.

[3] ATLAS Collaboration, *"ATLAS High-Level Trigger, Data Acquisition and Controls Technical Design Report"*, CERN/LHCC/2003-022, June 2003. See http://cdsweb.cern.ch/record/616089/.

[3] http://www.springsource.org

[4] http://www.opengroup.org/openmotif/datasheet.html

[5] http://qt.nokia.com

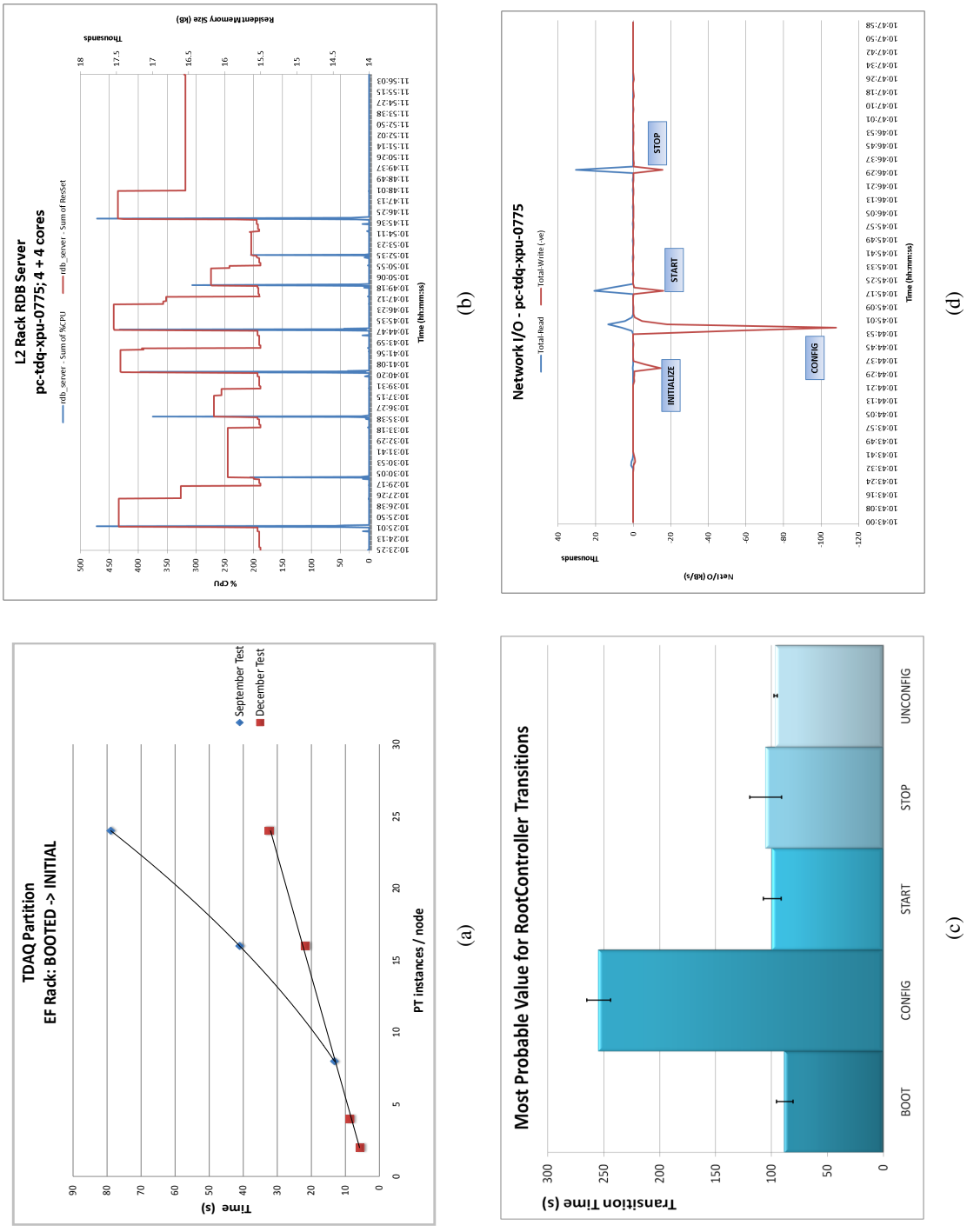
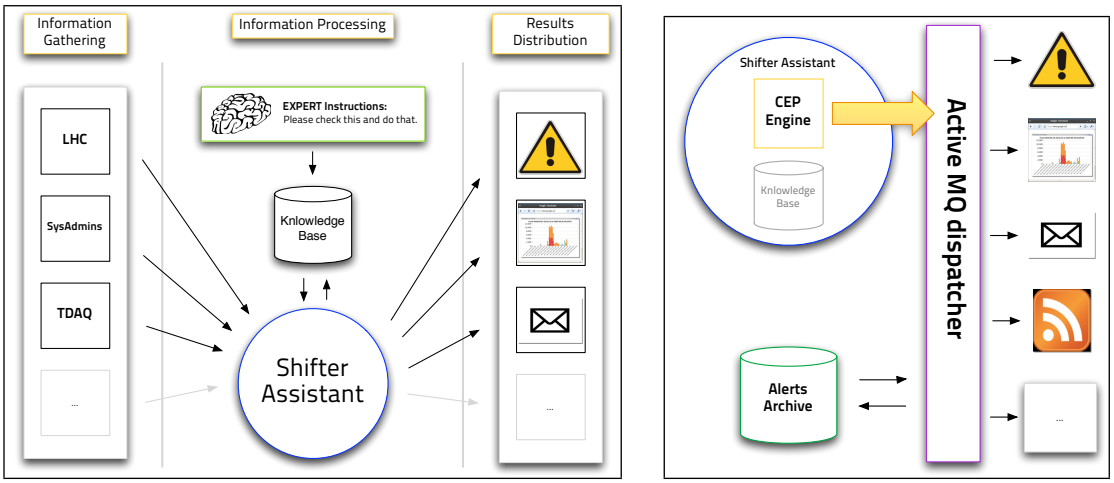


Figure 1: Improving Operation performances. (a): "BOOTED"→"INITIAL" transition time, as a function of instances running per node. (b): CPU load and memory usage of the CC system. Please notice that the total available CPU would be represented by the value of "800" on the Y-axis, i.e. 100 per each core, times 8 cores. (c): FSM transition times. (d): network usage of the CC system.



(a) (b)

Figure 2: Shifter Assistant architecture. (a): Shifter Assistant gathers information streams from various systems and returns alerts based on patterns found in data. (b): Shifter Assistant uses ActiveMQ dispatcher system to send notifications and alerts.

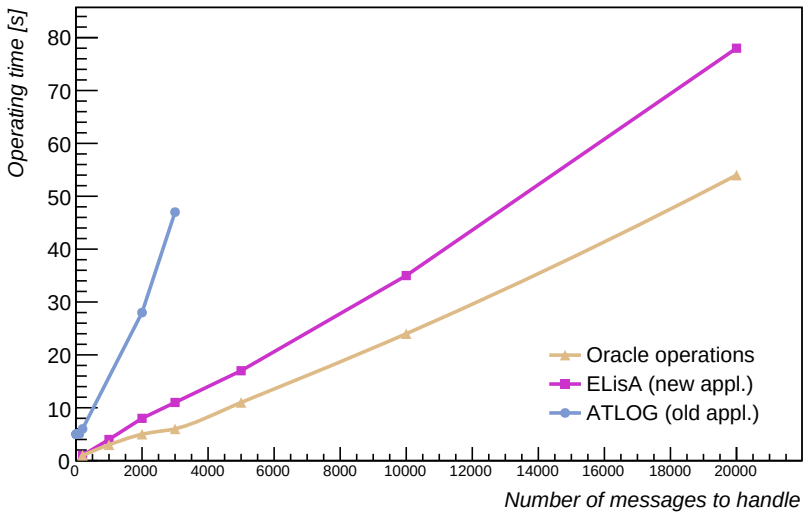


Figure 3: Speed and scalability tests of the ATLAS log message user interface