The TDAQ Analytics Dashboard: a real-time web application for the ATLAS TDAQ control infrastructure

Giovanna Lehmann Miotto[†], **Luca Magnoni**[†], **John Erik Sloper**[†]

[†]European Laboratory for Particle Physics (CERN), Geneva, Switzerland

E-mail: {giovanna.lehmann, luca.magnoni, jhon.erik.sloper}@cern.ch

Abstract.

The ATLAS Trigger and Data Acquisition (TDAQ) infrastructure is responsible for filtering and transferring ATLAS experimental data from detectors to mass storage systems. It relies on a large, distributed computing system composed by thousands of software applications running concurrently. In such a complex environment, information sharing is fundamental for controlling applications behavior, error reporting and operational monitoring. During data taking, the streams of messages sent by applications and data published via information services are constantly monitored by experts to verify the correctness of running operations and to understand problematic situations. To simplify and improve system analysis and errors detection tasks, we developed the TDAQ Analytics Dashboard, a web application that aims to collect, correlate and visualize effectively this real time flow of information. The TDAQ Analytics Dashboard is composed by two main entities, that reflect the twofold scope of the application. The first is the engine, a Java service that performs aggregation, processing and filtering of real time data stream and computes statistical correlation on sliding windows of time. The results are made available to clients via a simple web interface supporting SQL-like query syntax. The second is the visualization, provided by an Ajax-based web application that runs on client's browser. The dashboard approach allows to present information in a clear and customizable structure. Several types of interactive graphs are proposed as widget, that can be dynamically added and removed from visualization panels. Each widget acts as a client for the engine, querying the web interface to retrieve data with desired criteria. In this paper we present the design, development and evolution of the TDAQ Analytics Dashboard. We also present the statistical analysis computed by the application in this first period of high energy data taking operations for the ATLAS experiment.

ATL-DAQ-PROC-2011-004
14 January 2011

1. Introduction

The ATLAS Trigger and Data Acquisition (TDAQ) [1] computing infrastructure is made of thousands of running applications interacting with each other. During data taking, the streams of messages sent by applications and data published via information services are constantly monitored by experts to verify correctness of running operations and to understand problematic situations. The TDAQ Analytics Dashboard is a new tool meant to improve and simplify the monitoring of the flow of messages for a better and more effective understanding of the overall system status. In this paper we present the design and development of the analytics dashboard. Section 2 introduces the importance of message analysis, section 3 explains dashboard main concepts and section 4 presents in detail the architecture and the main components of the

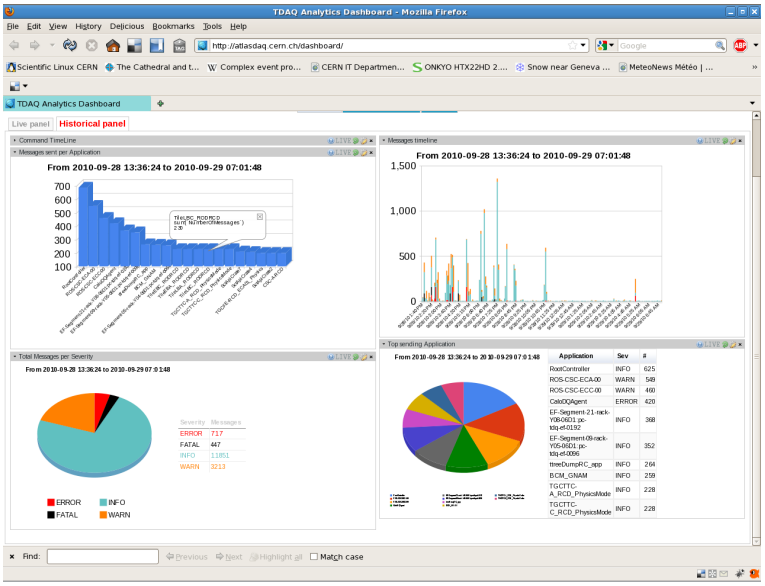

Figure 1. The TDAQ Analytics Dashboard

dashboard project as well as the technologies and framework we used to implement it. Finally, section 5 presents the conclusion and a look at the future.

2. Messages analysis in the TDAQ infrastructure

In the ATLAS TDAQ infrastructure message analysis is fundamental for controlling applications behavior, error reporting and operational monitoring.

2.1. Message oriented communication

Software applications in the TDAQ system adopt a message oriented communication to report error and information events to other components of the distributed infrastructure. The Error Reporting Service (ERS) provides a common format to errors, a fixed range of severity levels and a uniform way of reporting and transporting the errors. Message communication is based on the Message Reporting System (MRS) [2], that provides the functionality to transport, filter and route messages. Thanks to this well defined structure of information, messages are not used only for logging purpose, but also to trigger reaction in other applications, such as the expert system component [6].

2.2. The importance of message analysis

The flow of messages produced by applications during data taking operations is one of the main source of information to monitor system behavior and to detect problems and errors. Experts inspect the flow to discover which events has been reported by applications during a certain time interval. The flow can be composed by hundreds of messages per minutes in case of problematic conditions. The TDAQ Log Service [3] provide a permanent message archive in a Oracle database server to allow backward analysis. It provides a Java-based graphical interface to browse among messages with filtering capabilities and it is a perfect tool for depth analysis of problems. But getting information inspecting the raw flow of messages is not an easy and immediate job, in particular for non expert users. With this in mind, we developed the analytics dashboard to complement and improve the task of message monitoring.

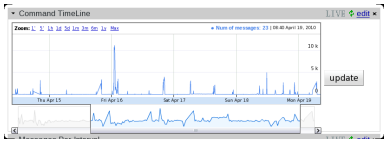

Figure 2. The timeline selector chart

3. The TDAQ Analytics Dashboard

The TDAQ Analytics Dashboard (`http://atlasdaq.cern.ch/dashboard`) , in Fig.1, is a new tool meant to help system analysis and errors detection in the TDAQ infrastructure. It can be used by experts to track unexpected behaviour as well by operators to detect global system status. The dashboard provides an aggregate and effective view on the flow of messages sent in the system, summarizing meaningful information in easy-to-read analytics graphs. It allows users to customize layouts and specify analysis criteria, for real time information as well as for historical data. Being a web based application, the dashboard provides an easy and user friendly interface without any software and platform constraint.

3.1. Interactive analytics charts

The dashboard presents summarized and effective views on the messages sent in the TDAQ system. These views are organized in easy to read analytics graphs, in order to give an immediate feedback on system status and behaviour, as shown in Fig:1. There are two main graph categories:

- **Analysis graphs**: to present aggregated information about the message flow.
- **Selector graphs**: to allow users to select the time interval to visualize.

The main analytics graphs aggregate information on the number and the types of messages sent in the system, with emphasis on detecting top offending applications and showing the overall system conditions. Each graphs is interactive, that means users can click on a specific section to visualize more information on demand. As well, each graph provides a configuration panel for various options, filtering capabilities, etc. Selectors charts can be interactive timeline showing the overall number of message sent in the system, as shown in in Fig. 2, or can provide views on specific time frame with a special meaning for the TDAQ infrastructure, as the case of the Run number selector.

3.2. Challenges

During the design and implementation of the dashboard we faced three main challenges, that actually characterize the architecture of the tool:

- **Producing analytics data**: to collect and correlate messages sent in the system and produce analytics summary in windows of time.
- **Distributing analytics results**: to make analytics results available to client-specific requests.
- **Visualizing data**:to aggregate the analytics results in easy to read, interactive and immediate views.

4. Architecture

The main architecture of the project is shown in Fig. 3. The dashboard is composed by three main components addressing the main challenges: processing data, distributing results and visualization.

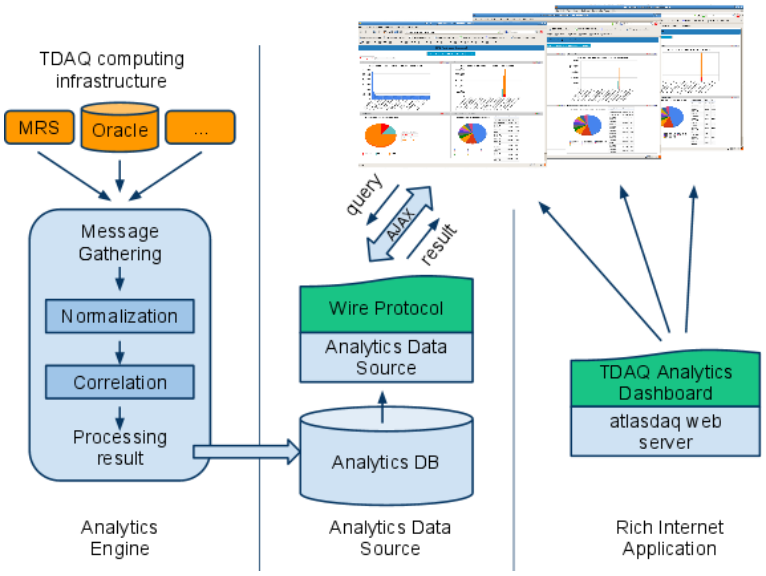

Figure 3. The Dashboard Architecture

4.1. The Analytics Engine
On the left side of the Fig.3 is shown the analytics engine.It is responsible of collecting messages from the TDAQ infrastructure and producing analytics correlations. It is able to gather the flow of messages from a MRS stream or from the Oracle archive, then it performs correlations in windows of time and save the results in a database.

4.1.1. Normalization and correlation of messages The engine is a standalone Java service able to perform a 2-phase correlation analysis of the flow of messages and to produce aggregated information in time windows. The normalization phase is mainly needed to extract some hidden information from messages (such as the un-throttling of message spikes automatically compressed in one single message). The correlation phase is a simple processing phase grouping events by message properties (message type, sender application, message text, etc.) and aggregating information in time interval (5 minutes). The results are archived in a MySQL database. Initially archiving was based on CSV files, but we moved to MySQL to improve performance for our data distribution approach.

4.2. Web application
The main requirements for the dashboard user interface is to provide an easy to use and easy to access tool. Nowadays, modern web technologies allow for rich and interactive user interfaces, and this convinced us to go for a rich internet application. The dashboard interface runs in the client's browser, allowing for an easy and standard access without requiring software installation or platform constraints.

4.2.1. Dashboard For the user interface we chose a dashboard approach for its flexible layout, easy to configure and customize. It is structured in panels, tabs like area, populated by widgets, as shown in Fig:1. Widgets are the main information containers and can be moved, closed and minimized by users. Users can also decide to add more widgets to panels, as well as create custom panels with the desired layout. The main information contained by widget are the analytics and selector graphs.

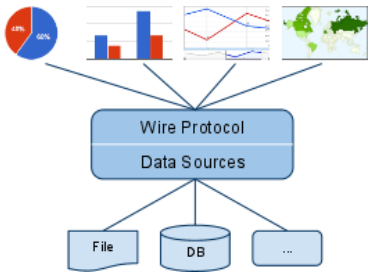

Figure 4. Data distribution

4.2.2. Google Web Toolkit The Google Web Toolkit (GWT) [5] is an open source framework to build complex web based application. It translates Java code in high performance, deeply optimized and browsers compatible Javascripts and HTML pages. It fits very well our use case to build a complex and effective rich internet application. Indeed, we developed the dashboard as a GWT project, modelling the dashboard structure and interactivity in Java, integrating Ajax techniques for communication and relying on external projects, such as Google Charts [4], to build the visualization layer.

4.2.3. Google Charts The Google Charts project [4] is composed by a javascript charts library providing several graphs options and by a data distribution mechanism. We decided to follow this approach for two main reasons: javascript based charts give us the interactivity we need to provide information on demand. Charts are rendered on client's browser, and users can easily discover more information with few mouse clicks on the graphs itself. The second advantage is that client-side rendering give us better customization options for user requirements. This is related with the data distribution mechanism, we are presenting in the next sections.

4.3. Analytics data distribution
The need for a data distribution capability is bound with the visualization strategy we adopted. To better fulfill users's requirements and use cases, the dashboard user interfaces have a direct connection with the analytics data, allowing to specify analysis criteria, with filtering, aggregating and grouping capabilities.

4.3.1. Analytics result as data source: JSON data over HTTP Relying on Google Chart for the visualization, we inherited also a smart and effective data distribution mechanism. Google Charts is aiming at building a selection of visualization charts library able to gather data from several different data sources (file, spreadsheet, databases, etc.), as shown in Fig. 4. A data provider is seen as data source, and they defined a simple HTTP based web interface (the Google Charts wire protocol) a data source have to implement to distribute data. How it works: 1) The data source will expose a URL, to which user send an HTTP GET request; 2) The client makes an HTTP GET request with parameters that describe what format to use for returned dat and an optional SQL-like query string. 3) The data source receives the request, prepares the data in the format requested (JSON, CSV and HTML are supported). The query string allows to specify filtering, sorting, and other data manipulation; 4) The data source creates an HTTP response that includes the serialized data and sends it back to the client.

Following this approach, we developed the simple web interface on top of the analytics database, as shown in the central part of Fig. 3, publishing the analytics data as a Google Charts compliant data source. In this way every chart in the dashboard is bound to a specific query URL used to retrieve the data to show, as shown in Fig: 5. Every visualization element in

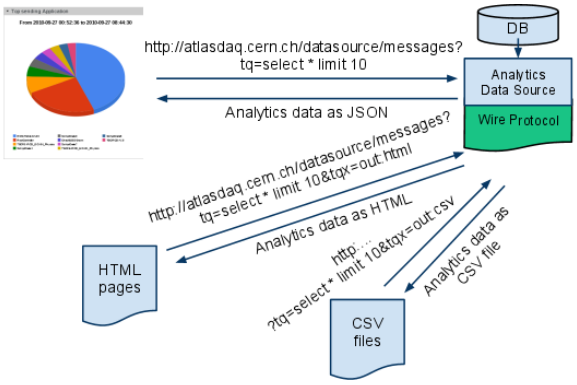

Figure 5. The Google Charts wire protocol in action

the dashboard retrieve the meaningful data in an asynchronous fashion, with an Ajax approach, and users customization become as easy as changing parameters in the query URL. Thanks to the standard format (JSON) returned by the Google Chart wire protocol, we are not bound to any Google-specific visualization library and it will be easy to integrate external charts libraries to add other visualization options.

5. Conclusion

The TDAQ analytics dashboard is currently used in production by ATLAS to monitor the flow of messages and the overall TDAQ system status. Thanks to its simplicity of usage and to the customization capabilities, the dashboard can be adopted by the various TDAQ communities, helping in increasing the quality and decreasing the overall amount of messages sent in the system. The dashboard approach works, new widgets can be easily created to fulfill users requirements and to provide new features. Finally, the Google Charts approach of defining a simple and common web interface on top of several data sources can be of inspiration for the TDAQ framework, where many different information providers share fundamental data in various way and format. A common interface on top of them can be the foundation for a global monitoring tool able to gather and correlate data coming from several heterogeneous data sources.

References

[1] ATLAS Collaboration. Atlas high-level trigger, data acquisition and controls technical design report. http://cdsweb.cern.ch/record/616089/, Jun 2003.

[2] Ivan Fedorko. The Message Reporting System of the ATLAS DAQ System. *ICATPP Conference on Astroparticle, Particle, Space Physics Detectors and Medical Physics Applications*, 2007.

[3] Raul Murillo Garcia and Giovanna Lehmann Miotto. A log service package for the ATLAS TDAQ/DCS group. *Nuclear Science, IEEE Transactions*, 54(4):202–207, 2007.

[4] Google. Google Charts. http://code.google.com/apis/chart/, Sep 2010.

[5] Google. Google Web Toolkit. http://code.google.com/webtoolkit, Sep 2010.

[6] John Erik Sloper and et al. Dynamic Error Recovery in the ATLAS TDAQ System. *Nuclear Science, IEEE Transactions*, 55(4):405–410, 2008.