The ADAM project: a generic web interface for retrieval and display of ATLAS TDAQ information.

A Harwood[1], G Lehmann Miotto[2], L Magnoni[2], W Vandelli[2] and D Savu[2]

[1] University of the West of England

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

E-mail: adam2.harwood@uwe.ac.uk, {giovanna.lehmann, luca.magnoni, wainer.vandelli, dan.savu}@cern.ch

Abstract. This paper describes a new approach to the visualization of information about the operation of the ATLAS Trigger and Data Acquisition system. ATLAS is one of the two general purpose detectors positioned along the Large Hadron Collider at CERN. Its data acquisition system consists of several thousand computers interconnected via multiple gigabit Ethernet networks, that are constantly monitored via different tools. Operational parameters ranging from the temperature of the computers to the network utilization are stored in several databases for later analysis. Although the ability to view these data-sets individually is already in place, currently there is no way to view this data together, in a uniform format, from one location. The ADAM project has been launched in order to overcome this limitation. It defines a uniform web interface to collect data from multiple providers that have different structures. It is capable of aggregating and correlating the data according to user defined criteria. Finally, it visualizes the collected data using a flexible and interactive front-end web system. Structurally, the project comprises of 3 main levels of the data collection cycle: The Level 0 represents the information sources within ATLAS. These providers do not store information in a uniform fashion. The first step of the project was to define a common interface with which to expose stored data. The interface designed for the project originates from the Google Data Protocol API. The idea is to allow read-only access to data providers, through HTTP requests similar in format to the SQL query structure. This provides a standardized way to access this different information sources within ATLAS. The Level 1 can be considered the engine of the system. The primary task of the Level 1 is to gather data from multiple data sources via the common interface, to correlate this data together, or over a defined time series, and expose the combined data as a whole to the Level 2 web interface. The Level 2 is designed to present the data in a similar style and aesthetic, despite the different data sources. Pages can be constructed, edited and personalized by users to suit the specific data being shown. Pages can show a collection of graphs displaying data potentially coming from multiple sources. The project as a whole has a great amount of scope thanks to the uniform approach chosen for exposing data, and the flexibility of the Level 2 in presenting results. The paper will describe in detail the design and implementation of this new tool. In particular we will go through the project architecture, the implementation choices and the examples of usage of the system in place within the ATLAS TDAQ infrastructure.

ATL-DAQ-PROC-2011-050
30 November 2011

1. Introduction

This paper presents the ADAM project, a new approach for aggregating and visualizing ATLAS operational data. In the first section we describe the project architecture, then we present the details of the ADAM interface, a lightweight HTTP interface for data collection from multiple

providers, and finally we describe the ADAM visualization dashboard, a web portal to gather and visualize data from different data sources.

1.1. ATLAS TDAQ infrastructure

ATLAS [1] (A Toroidal LHC Apparatus) is a particle physics experiment at the Large Hadron Collider at CERN. The Trigger and Data Acquisition (TDAQ) system of the ATLAS experiment is responsible for selecting and transferring ATLAS experimental data from the detector to the mass storage system. To cope with the very high data rate produced by the detector a complex and distributed computing infrastructure has been built. More than 200 switches and routers interconnect around 2000 hosts to build a real-time filtering system for particle collision events [2]. On top of the hardware infrastructure, over 20.000 applications are responsible for analyzing, filtering and moving event data to permanent storage [3].

1.2. Motivation

The maximization of the ATLAS data-taking efficiency requires operators and experts to precisely know the TDAQ system status in order to be able to understand and solve any unexpected behavior. The system status is defined by a large set of operational parameters, collected by different tools and provided via different means, such as web pages, graphical user interfaces or ad-hoc APIs. Moreover, the retrieved data have to be correlated, on the basis of desired criteria, and finally visualized.

The main objectives of the ADAM project are:

- the definition of a standard way of accessing data gathered by various TDAQ information storage systems;
- the correlation of pieces of information provided by different sources;
- the creation of a single point for data aggregation and visualization.

2. The ADAM project

The goal of the ADAM project (ATLAS Data API Mechanism) is to provide a uniform interface for information retrieval from several applications (data providers) and to expose a unique access point for data querying and collection. Data providers are heterogeneous data sources that offer several kind of information, potentially very different in type and structure. The complexity of each data provider implementation is however hidden, since their capabilities are exposed through the ATLAS Data API (ADAM interface), a SQL-like web interface. Leveraging on the common interface, a unified web portal can gather and visualize data from different providers. By using the web portal, users and experts have access to summary graphs that can be dynamically customized.

2.1. Architecture

The ADAM architecture is structured as a 3 layer application, as shown in Figure 1.

- **Level 0** includes all the ADAM data providers.
- **Level 1** is an intermediate component able to gather and correlate data from different data sources.
- **Level 2** is a visualization layer implemented as a web dashboard where users can query and visualize data coming from multiple data providers.

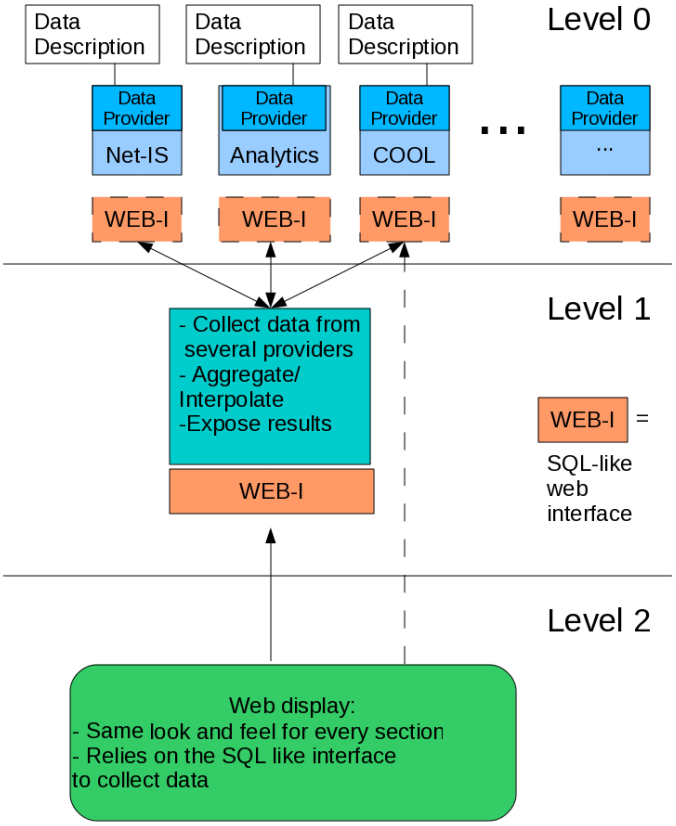

Figure 1. The ADAM project architecture

2.2. Data providers

An application displaying ad-hoc or historical statistics about a system has either to monitor all the parameters by itself or to rely on the information gathered by other applications (data providers). Usually these data providers use custom methods for data collection and a variety of technologies for data storage and data retrieval (RRD files [4], relational databases, custom user interfaces, APIs etc). This is an expected situation given the complexity of the system and the long development time-line. For efficient log-back analysis, system wide statistic generation or problem investigation operators and experts need a unified visualization system. The ADAM project defines a common web-based SQL-like interface that every data provider has to implement. As a result, the common interface hides the complexity of the provider implementations giving users a simple but very powerful language to collect the desired information. The language, described in section 3, allows data providers to expose their data in various formats. The default format is XML (Extensible Markup Language), where results are represented as structured data suitable for machine-based processing. Optionally, data provider can return CSV (Comma Separated Value) documents or JSON (JavaScript Object Notation) data, useful for web technologies integration. Through the web-based approach a data collection exchange is composed of the following steps, as shown in Figure 2:

(i) The data provider exposes a URL, called the data source URL;

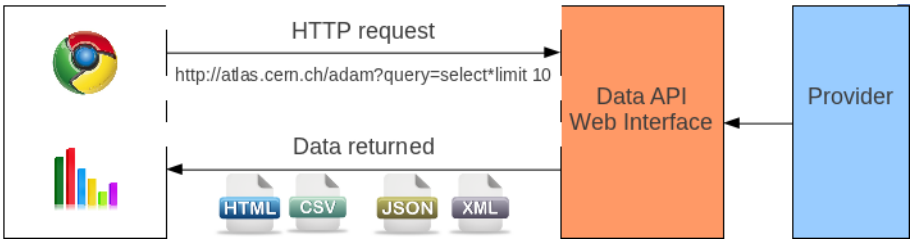

Figure 2. Client request via the ADAM interface with query criteria

(ii) The client makes a HTTP request with optional parameters that describe the query string and the required result format;

(iii) The data provider receives and parses the request, as described in section 3.1;

(iv) The data provider prepares the data in the requested format (XML, CSV, JSON);

(v) The data provider creates a HTTP response that includes the serialized data and other response parameters, and sends it back to the client.

Currently, the following data providers expose a ADAM-compliant interface.

COOL COOL [5] is the condition database of the ATLAS experiment. Operational data are stored indexed by interval of validity. Information are retrieved via a programmatic API available in C/C++ and Python or via an ad-hoc web interface.

Net-IS The Net-IS (Networking Integrated System) [6] is the main tool used for collecting statistics of the network infrastructure. Net-IS provides system metrics (e.g. CPU load, system temperature), environmental conditions (e.g. rack power, ambient temperature) and a reduced set of data taking parameters allowing the correlation of network related problems with external factors. The tool is designed to efficiently handle time-series data and uses RRD files for raw data and a relational database for metadata storage.

MRS Message analytics The MRS analytics [7] data source provides an aggregate and effective view on the flow of messages sent by applications during data-taking runs. Messages are sent via the Message Reporting facility (MRS) [8].

In order to facilitate and stimulate the development of ADAM-compliant providers, a Java-based library targeting relation databases has been developed. The library allows to expose any table (relying on the JDBC abstraction) through a web interface compatible with the ADAM interface.

2.3. Level 1

The subsequent step in the system status analysis requires not only to collect information but also to correlate and unify data access across different providers. Because of the ADAM unified interface an intermediate aggregation and analysis engine, so called Level 1, has been developed. The Level 1 is placed between clients and data providers and is able to gather data from multiple sources and to perform basic correlation functions. For example, the Level 1 is able to correlate data sets over the desired criteria, such as time, or to extrapolate values in order to unify time resolution. The Level 1 capabilities are exposed via the same ADAM interface, and users can query the Level 1 asking a desired set of data independently from the data provider offering the information. Handling a data request in the Level 1 requires:

(i) Parse the HTTP request identifying the data sources involved;
(ii) Forward the request and retrieve data from each provider;
(iii) Perform aggregation, correlation and combination functions on the data;
(iv) Return the combined data in the required format.

3. ADAM interface

The ADAM interface is the core of the ADAM project. It is a HTTP-based interface that unifies the way information is exposed from different data providers. It has a simple request format based on HTTP GET requests and it offers a powerful SQL-like protocol to express query on data with desired criteria, inspired by the Google Charts project [9].

3.1. Request format

A client can send a HTTP GET request with several parameters, including custom elements and an optional query string. Representing requests as HTTP URIs, rather than as HTTP headers or as part of the payload, has several benefits such as direct linking and persistence.

Table 1. ADAM interface request parameters.

Parameter name	Key:Value	Description
op	-	A set of colon-delimited key/value pairs for standard or custom properties. Pairs are separated by semicolons.
-	**reqId***:value*	A numeric identifier for the request, to be sent back in the response. It allows multiple requests to be sent in parallel.
-	**hash***:value*	A hash of the query results sent to the client in any previous requests to this data source. This is an optimization to avoid sending identical data to a client twice.
-	**out:**[**xml**|**json**|**csv**]	A string describing the format of the returned data.
query	-	A query written in ADAM Query Language, described in section 3.2, specifying how to filter, sort, or otherwise manipulate the returned data.

The request parameters shown in Table 1 are composed to build the desired data query. Some examples of possible requests are given below. A simple request with no parameters:

- `http://dataprovider.cern.ch/resource`

This is a request with the `op` parameter and two properties:

- `http://dataprovider.cern.ch/resource?op=reqId:0;hash:4641982796834063168`

Request with a minimal query string:

- `http://dataprovider.cern.ch/resource?query=select time, CPU limit 10`

3.2. The Query language
The query language used when communicating with data providers can be defined as a dilute dialect of the SQL query language. As presented in Table 2, the simple query style of the SQL language is preserved as well as the capability of expressing basic data manipulation functions (e.g. MIN, MAX, AVG, COUNT). On the other hand, the SQL statements are meant to address one database. In the ADAM project the concept is therefore extended to allow one SQL-like statement to query multiple data providers. This allows the Level 1 to parse a multi provider query, retrieve data from every provider and aggregate the result.

Table 2. ADAM query language clauses

Clause	Description
select	Select which information to return and in which order.
from	Select which data provider to query.
where	Returns only the information that matches the condition.
group by	Aggregate values across rows.
order by	Sort the result in the specific order.
limit	Limit the number of returned rows.
correlate by	Only for the Level 1. Request to correlate result from multiple data provider on the specified criteria, such as time.

The following is an example of a multiple data sources query:

- `SELECT datasource1:table2:field1, datasource2:table2:field2 FROM datasource1:table2:field1, datasource2:table2:field2 WHERE globalTime < 20-10-2011 16:20 AND globalTime > 20-10-2011 15:20`

In this way multiple tables from differing sources can be selected, using a common time format.

3.3. Describe capabilities
A dedicated feature is built into the ADAM data provider interface to allow users to introspect the data source architectures, internal structures and supported data formats. Very similar to the `SQL DESCRIBE` command, the feature has two basic variants:

- `describe table`
- `describe provider`

The behaviour of the `describe table` command matches the equivalent SQL statement (for MySQL databases). It returns the list of field properties within the specified table.

The `describe provider` returns information on the data source capabilities in terms of supported ADAM query functionalities. For example, a data source can expose the capability of performing `MAX` and `MIN` calculations.

4. The ADAM visualization dashboard
A web-based visualization dashboard, that gathers and aggregates information from ADAM-capable data sources, has been implemented with the aim of providing a unique entry point to query and visualize operational data. The dashboard, which can be easily customized by users both in the layout as well as in the information displayed, has been developed on top of

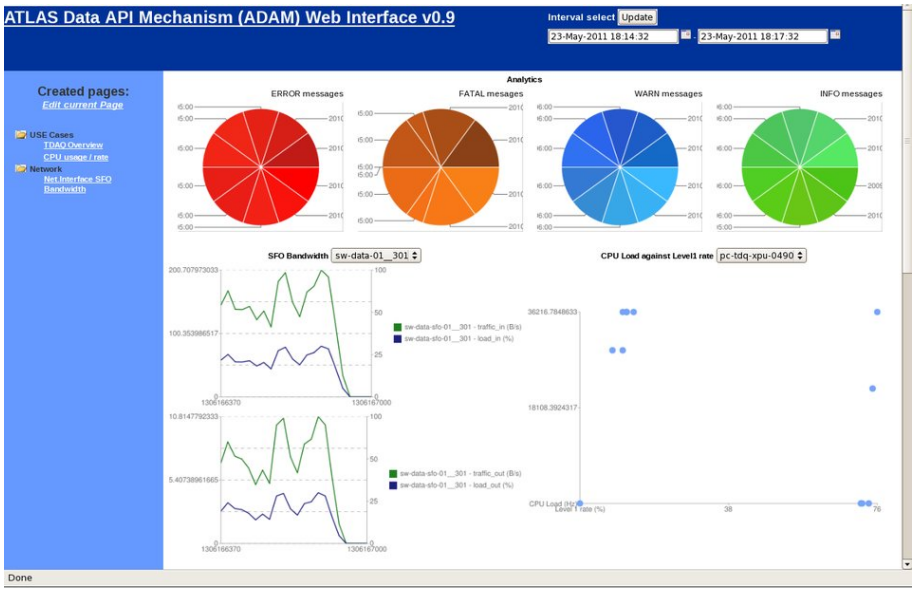

Figure 3. The ADAM visualization dashboard

the Django framework [10] and is written in the Python scripting language. The web-site is based on a series of customizable HTML components and tables representing the set of basic visualization elements and it is completely dynamic with respect to the layout. Every HTML component can be mapped with an ADAM query and a visualization option. The query can be directed to individual providers or to Level 1, for more advanced aggregation, normalization or analysis features.

Graph to query mapping The dashboard, as shown in Figure 3, presents information aggregated in tables and charts of different types. Users can customize visualization elements both in type and content. The supported charts are the ones provided by the Google Charts Image tool [11], such as pie charts, column/bar charts or more advanced options such as heat maps or composite graphs. Thanks to the common ADAM interface the content of visualization elements can be associated to data via HTTP queries. The latter are executed only at rendering time on user browsers. The dashboard allows customization of the query, supporting all the ADAM language features.

User customizable The dashboard is structured in pages and tabs whose layouts can be dynamically composed using the basic visualization elements. The dashboard layout is therefore user dependent: dedicated views can be created with the aim of tackling specific area of the TDAQ system.

4.1. Command line shell

Similarly to the command line interfaces provided by several database vendors, an ADAM command line shell has been developed. The shell allows to interact with ADAM-capable data providers. It is based on the ADAM query language, offering auto completion and syntax checking functionalities. It is a fast and effective means to build and validate queries and to access ADAM capabilities in a programmatic way.

5. Conclusions

The main goal of the ADAM project is to propose a new and effective approach to gather and correlate information from many different data providers. The ADAM interface is a lightweight HTTP-based interface with a very powerful SQL-like query language. The uniform interface allows for loose coupling of clients and data providers. In fact, different data-provider back-end technologies are abstracted using the same SQL-like paradigm. The ADAM visualization dashboard is a unique entry point to collect and visualize data of different nature. The main areas of research for the future are the finalization of the web-based visualization, offering dedicated layout for specific use cases, and the expansion of the gathering and processing capabilities, including more ADAM-compatible data providers.

References

[1] ATLAS Collaboration 2008 *The ATLAS Experiment at the CERN Large Hadron Collider.* Journal of Instrumentation, S08003.

[2] Ciobotaru M, Leahu L, Martin B, Meirosu C and Stancu S 2006 *Networks for ATLAS trigger and data acquisition* Conference on Computing in High Energy and Nuclear Physics, Mumbai, India.

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

[4] Oetiker T RRD Tool http://oss.oetiker.ch/rrdtool

[5] Andrea Valassi et al 2010 *LCG Persistency Framework (CORAL, COOL, POOL): Status and Outlook* Conference on Computing in High Energy and Nuclear Physics, Taipei, Taiwan.

[6] Savu D, Al-Shabibi A, Martin B, Sjoen R, Batraneanu S and Stancu S 2010 *Efficient Network Monitoring for Large Data Acquisition Systems* Proceedings of the ICAPELCS 2011, Grenoble, France.

[7] Magnoni L, Miotto G L and Sloper J 2010 *The TDAQ Analytics Dashboard: a real-time web application for the ATLAS TDAQ control infrastructure* Conference on Computing in High Energy and Nuclear Physics, Taipei, Taiwan.

[8] Fedorko I 2007 *The Message Reporting System of the ATLAS DAQ System* Conference on Astroparticle, Particle, Space Physics Detectors and Medical Physics Applications, Como, Italy.

[9] Google 2010 Google Charts http://code.google.com/apis/chart/

[10] The Django software foundation Django framework https://www.djangoproject.com

[11] Google 2010 Google Charts Image http://code.google.com/apis/chart/image