GNAM and OHP: Monitoring Tools for ATLAS experiment at LHC.

P. Adragna[a], D. Cimino[b], M. Della Pietra[c], A. Dotti[b], R. Ferrari[d], G. Gaudio[d], C. Roda[b], D. Salvatore[e], W. Vandelli[f],P. F. Zema[g].

[a]Queen Mary, University of London, London, UK
[b]University and INFN of Pisa, Pisa, Italy
[c]INFN of Napoli, Napoli, Italy
[d]INFN of Pavia, Pavia, Italy
[e]University of Calabria and INFN of Cosenza, Cosenza, Italy
[f]University and INFN of Pavia, Pavia, Italy
[g]CERN, Geneva, Switzerland, on leave from University of Calabria and INFN of Cosenza, Cosenza, Italy

ATL-DAQ-CONF-2007-023
31 August 2007

***Abstract*— ATLAS is one of the four experiments under construction along the Large Hadron Collider (LHC) ring at CERN. The LHC will produce interactions at a center-of-mass energy equal to $\sqrt{s} = 14$ TeV at 40 MHz rate. The detector consists of more than 140 million electronic channels. The challenging experimental environment and the extreme detector complexity impose the necessity of a common scalable distributed monitoring framework, which can be tuned for the optimal use by different ATLAS sub-detectors at the various levels of the ATLAS data flow. This note presents two monitoring tools that have been developed for this aim within the architecture ATLAS Monitoring Framework and the Data Acquisition System: GNAM and OHP. The first one is a framework for online histogram production; the second one is graphical application for histogram presentation. This tools are now widely used during the ATLAS commissioning and their performances are reported in this note.**

ATLAS, DAQ, online monitoring, GNAM, OHP. ***Index Terms*—**

I. Introduction

ATLAS [1] is one of the four experiments being installed at the LHC. The experiment includes several challenging detection technologies and is supported by a large, distributed data acquisition and trigger system (TDAQ). LHC will provide collisions at a center-of-mass energy of 14 TeV with a frequency of 40 MHz. The output of the ATLAS first level trigger will be about 100 kHz. This frequency will be further reduced by the higher trigger levels and finally some hundreds of events will be selected and stored every second. The whole ATLAS detector consists of about 140 million electronic channels and the expected average event size is about 1-2 MB. Considering the huge number of channels and the high event rate, a monitoring system is an essential tool to assess the status of the hardware and the quality of the data while they are being acquired. Such monitoring system should cope with the challenging experimental conditions providing a flexible, scalable and tunable framework, in order to be useful for the different ATLAS sub-detectors and sub-systems.

Corresponding author: M. Della Pietra.
Tel.: +39 081676125. *E-mail: massimo.dellapietra@na.infn.it*

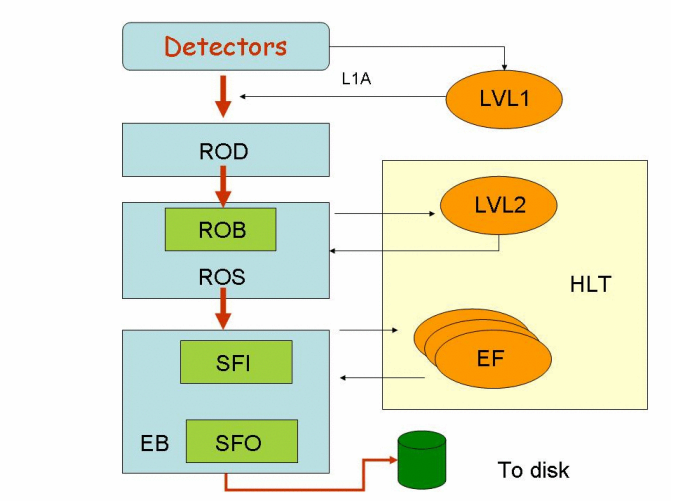

Fig. 1. Simplified schema of the ATLAS dataflow and trigger system. All the acronyms are explicated in the text

II. ATLAS TDAQ and Monitoring Framework

In ATLAS there are several levels of data flow [2]. Data are acquired by the front-end electronics (FE), located next to the detectors, and are collected, step by step, until the full event is assembled (see Fig. 1). The level one trigger electronics (LVL1) is devoted to the first selection level; if an event is accepted, all the FE boards send the data to the Read Out Drivers (RODs), which are detector-specific custom modules. Each ROD is point-to-point connected to one Read Out Buffer

(ROB), that is a PCI custom board, to which it sends detector-specific data. The event is then assigned to one processing node of the second level trigger (LVL2) farms, which collects from the Read Out Systems (ROSs) the data fragments belonging to the detector regions selected by the LVL1 and starts its filtering algorithms. Accepted events are assigned to a Sub-Farm Input (SFI), which collects all the data fragments from the ROSs and assembles the complete event. The last filtering stage is the Event Filter (EF), the second component, together with the LVL2, of the High-Level Trigger (HLT) sub-system. Built events are sent to EF farm processing nodes, in which a Processing Task (PT) completely reconstructs and analyzes the data with high-precision algorithms taken from the ATLAS offline analysis framework (Athena). Events accepted by the EF are passed to the Sub-Farm Output (SFO) for the transmission to the mass storage. The ATLAS TDAQ infrastructure is a large distributed environment, including thousands of computing nodes and custom modules. In order to verify the good quality of the data sent to the permanent storage, the whole triggering system, the DAQ system and the ATLAS sub-detectors should be constantly monitored in terms of functionality and results.

To understand the complexity of the monitoring framework mandate, one has to consider that more than 3000 sources of monitoring information and up to 300 event sampling points are foreseen in the final ATLAS environment. This will lead to a O(10 GB) of monitoring data produced each run. To fulfill this mandate, the ATLAS monitoring system is organized as a distributed framework and includes several applications, ranging from low-level information-sharing components up to high-level graphical interfaces. This separation permits to isolate the problems and to optimize each application for the specific needs.

The ATLAS TDAQ software provides a number of services, collectively known as Information Sharing Services, that can be employed for building a monitoring system. Their main task is to carry requests, about monitoring data, from monitoring destinations to monitoring sources, and then the actual monitored data back from sources to destinations. The main services supplied for Monitoring purposes are the following [3]:

- *Event Monitoring* (Emon) provides a framework to enable event sampling and distribution. User programs may request event fragments with selected properties, like trigger or sub-detector type, from a specific sampling point. Since an event is transported as a sequence of bytes, Emon is neutral to the event format and can handle events coming from any level of the data flow. In order to minimize the load on the sampling application, requesting programs with the same selection criteria are arranged in a tree. Hence the sampling application forwards the events only to the first requester in any tree. The distribution of the data along the tree is done transparently to the users in a peer-to-peer approach.
- *Information Service* (IS) is responsible for archiving and sharing any kind of information among the different processes running. It supports three main types of interactions: information providers can create, update or delete information, while information readers can get the value of the information. Moreover information receivers can subscribe to the repository to be notified about changes. In addition, through IS any application is able to send commands to any of the running providers. This is useful to control the IS information flow: for example an application may ask a particular provider to increase the frequency of information updates or to republish a particular information.
- *Online Histogram Service* (OHS) manages histograms, providing a transient storage between histogram producers and displays. Histograms are published in the OHS server, where they are available to the entire system. The OHS also allows commands to be routed. Actually, any application can issue a command related to a particular histogram: the OHS takes care of sending the command to the appropriate histogram provider. OHS is essentially based on IS and extends its functionalities to handle histogram objects, in particular raw and ROOT [4] histograms are supported.
- *Message Reporting System* (MRS) transports messages among TDAQ applications. Messages may be used to report debug information, warnings or error conditions. MRS allows to associate qualifiers and parameters to each message. Message receivers can moreover subscribe the service to be notified about incoming messages, also using filtering criteria.

The ATLAS TDAQ software services are built on top of a common Inter Process Communication service, whose implementation is based on CORBA [5].

III. GNAM

GNAM [6] is a light-weight configurable framework optimized for detector functionality monitoring. It can be used to perform many sorts of jobs, thanks to a plug-in design that separates common actions and analysis algorithms, which are stored in dynamic libraries loaded at run-time. The GNAM application has been designed in order to fulfill the following requirements:

- The application architecture must be detector independent in order to manage data produced by different subdetectors. Fragment processing must rely only on its structure and not on its content.
- The application must be modular in order to separate the common actions from the detector dependent ones. Common actions consist, for example, in unpacking up to the subdetector specific part or managing histograms. Detector dependent actions include simple raw data decoding and analysis, booking and filling of histograms;
- The application must avoid duplication of code. All the functionalities available in the ATLAS TDAQ software framework must be available in the monitoring program. The underlying technology for histogram management should rely on well-known tools.

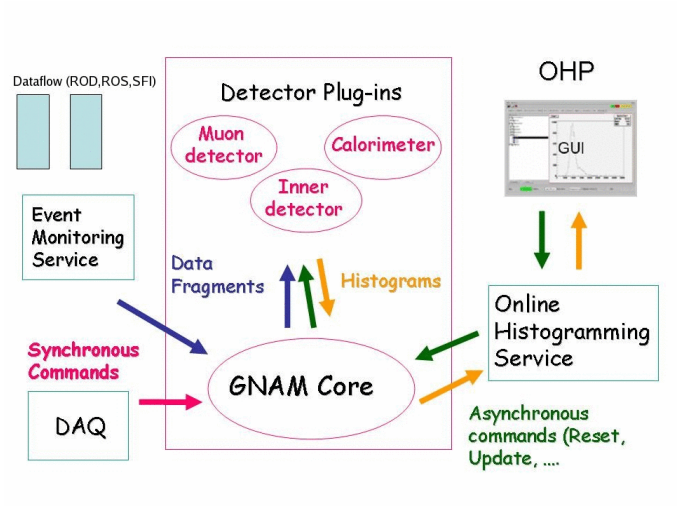

Fig. 2. GNAM framework schema and its relation with the ATLAS Online Monitoring environment.

- The application must be controllable either by the TDAQ or by the users. In standard conditions, the monitor is controlled by the TDAQ run control system. However, for debugging and testing purposes, the user must be able to control the monitoring process independently from the overall TDAQ state.
- Histogram production and visualization must be independent of one another. Histograms have to be published in a common server and made available for all the processes that may require them, their further processing (online display, comparison with reference histograms, fitting) must be carried out by a separate application.

These requirements drove the design of the monitoring architecture. The core, see Fig. 2, responsible for common actions, has been modeled as a finite state machine synchronous with the TDAQ. Detector dependent actions have been implemented in dynamic libraries. The data source can be either an online sampler from Emon (ROD, ROS, or SFI) or a file sampler. For the latter case, an application was developed to read a raw data file and feed the Event Monitoring Service, emulating an online data source. The core application, identifies all the ROD data fragments and arranges them in a list. No further unpacking can be done centrally, as from this point on the decoding is detector dependent. GNAM typically loads one decoding library per subdetector and one or more histogramming libraries. ROOT [4] is the underlying technology for histogramming. The implementation of the decoding and histogramming libraries is responsibility of the subdetector groups. The decoding library receives from the core the list of the detector data fragments and, among them, identifies and decodes the pertinent data. The decoded data are then transiently stored by the core. The histogramming library retrieves the data related to one or more detectors, carries out simple analysis and fills histograms. The same library books the histograms and returns a list of them to the core for central management (publication on the server, dump on file, etc.). Any histogramming library has access to all the decoded data stored inside the core. This allows to produce histograms of correlations among different detectors without duplication of decoding. The state transitions are managed by the core and are completely transparent to the detector libraries. The finite state machine architecture of GNAM also guarantees the synchronization with the overall TDAQ system. For debugging and testing purposes, the user can execute the monitoring application in asynchronous mode, controlling the state transitions manually. As previously mentioned, histograms are centrally handled by the core, that takes care of publishing them to the OHS server, saving them into a file and executing user commands on request. In order to extend the ROOT histogram functionalities, a wrapper for the ROOT histogram base class has been developed, adding meta-data and other properties to each histogram. GNAM can moreover handle asynchronous commands coming through the OHS to modify at run-time histogram properties or to execute custom functions defined in the analysis libraries. Special GNAM subdetector libraries can also be used to to feed the Offline Event Display software with decoded and formatted data, in order to display the event online. This is achieved writing the decode data into a circular buffer, resident on the RAM of the CPU that runs the process.

IV. OHP

The Online Histogram Presenter (OHP) is a general purpose, highly congurable, interactive presenter developed in the context of the Online Monitoring System of the ATLAS experiment. As described in the section II, all the histogram producers in the Monitoring framework publish their results to the OHS server that collects all this histograms. From this point of view, the OHP can be considered as client of the OHS. OHP can operate in two different modes: it can browse the OHS and/or show a configurable set of online or reference histogram in a series of tabs. The two modes allow both the detector experts and the standard shifters to have all the needed functionalities within the same application.

OHP is structured as a multi-thread application and as been designed in order to allow all the users (expert or non expert) to interact with the displayed histograms, in order to perform operations like fitting, zooming and changing the graphical appearance of the plot. Moreover is possible for the user to interact with the histogram producer, asking for a reset or turning on/off the filling of the histogram; it is also possible to compare the online histograms with reference ones.

The most important requirement for this application is to get the histograms available to the user as soon as they have been produced, minimizing the network traffic, as much as possible. OHP receive histograms from the OHS server through a subscription mechanism: this allows the process to be informed whenever the subscribed histogram change status, e.g. when the histogram is published or updated.

OHP is composed of the following subsystems:

- the Core Subsystem which provides access to the histograms from the Graphical User Interface. It also controls

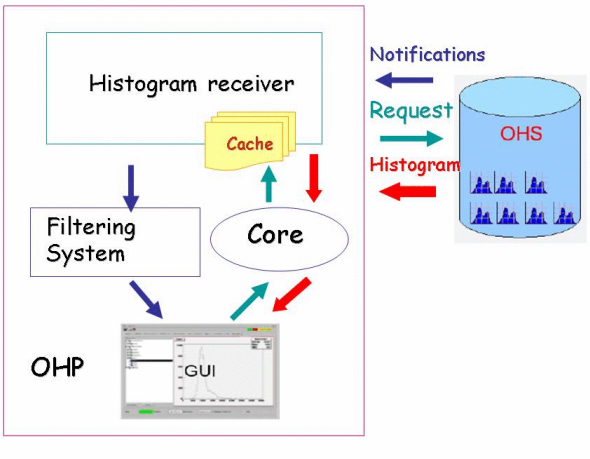

Fig. 3. Schema of the OHP architecture.

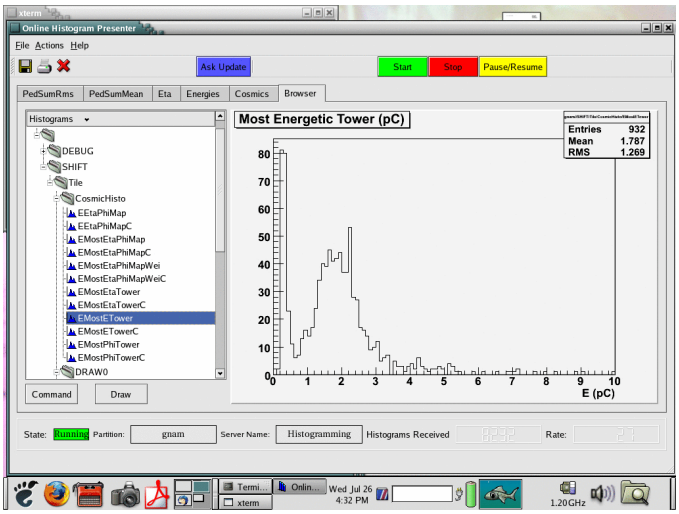

Fig. 4. A screenshot of the OHP GUI.

all the other components. the Histogram Receiver which is responsible for receiving notifications and retrieving histograms from OHS on request. Multiple subscriptions are possible: each subscription creates a dedicated receiver.

- the Cache which is used to store the received notification and histogram, each Histogram Receiver owns a cache. The aim of the cache. The cache is a key component to minimize the network traffic, and the description of its usage is described below.
- the Filtering Subsystem which filters the notifications that are related to the active canvas. The active canvas is the one on focus, i.e. the one that the user actually sees.
- the Graphical User Interface (GUI) which is based on the Qt ®(Trolltech Inc.) [7] framework. The underlying technology for histogramming is ROOT, which provides the user with all the functionalities to interact with the histograms, such as zooming, fitting, changing the graphic aspect etc. Figure 4 shows a screenshot of the GUI as it appears when used to browse the OHS content.

In order to better understand how the different subsystems operate together, let us examine the OHP works in more detail. Whenever a histogram is published or updated, the OHS notifies the event to the OHP, which will decide whether to retrieve the plot and draw it on the screen. This notification received by OHP is sent to the Filtering Subsystem and the cache is updated with the histogram name and its time-stamp Tn. If the histogram must be drawn in the active window, the notification is also propagated to the GUI; otherwise it is discarded. Note that no histogram is transferred over the network during this operation. After receiving the notification, the GUI asks the Core for the histogram. The histogram is searched for in the cache. If it is not found, the histogram is downloaded from OHS, stored in the cache together with the time-stamp Tr (where r stands for retrieval) and sent back to the GUI that displays it on the screen. On the contrary, if the histogram is found in the cache, the notification time-stamp Tn is compared with the retrieval time-stamp Tr. If Tr ¡ Tn the cached histogram is replaced with the more recent one available in the OHS. Otherwise the cached histogram is displayed immediately. When the user changes the active window, the histograms to be drawn are searched in the caches and, only if needed, retrieved from OHS. The use of the notification mechanism together with a caching system effectively contributes to reducing the network traffic, since histograms are transferred only when actually needed. The histograms in the active windows are continuously updated with the same frequency they are received. In order to be able to interact with a particular histogram, for example to perform a fit, when a histogram is double-clicked the histogram is copied and displayed in a pop-up window. This allows both to work on the selected histogram and to keep the active window continuously updating. The Online Histogram Presenter can be fully configured (subscriptions, graphical aspects, predefined tabs, reference histograms) either through an ASCII or XML file or via a graphical panel. Finally, commands to the histogram producers (i.e., GNAM) can be sent through the OHS in order to reset, rebin an histogram or force the upgrade of the histogram itself.

V. Conclusion

GNAM and OHP have been developed within the ATLAS Online Monitoring framework to perform the data quality and detector performances monitoring. This two tools fulfill the requirements of scalability and flexibility imposed by the huge amount and rate of data to be checked. They are already widely used during the commissioning phase of the detector that is ongoing at CERN, representing a very useful tool to understand the detector performances. This commissioning phase will be also a benchmark for the performances of these application and for the whole Online Monitoring framework.

References

[1] ATLAS Collaboration, *ATLAS Muon Spectrometer Technical Design Report*, CERN/LHCC/97-22, ATLAS TDR 10 (1997).

[2] ATLAS Collaboration, *ATLAS High-Level Trigger, Data Acquisition and Control*, CERN/LHCC/2003-22, Geneva, CERN, 2003.
[3] S. Kolos et al., *Online Monitoring software framework in the ATLAS experiment*, CHEP 2003. La Jolla, CA, USA, 2003
[4] R. Brun, F. Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst.& Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/.ROOT."
[5] A. Amorim et al., *Use of CORBA in the ATLAS prototype DAQ*, IEEE Transaction on Nuclear Science, Vol. 45, No. 4, August 1998.
[6] A. Dotti *et.al*, IEEE Transaction on Nuclear Science, Vol 53 N. 3 June 2006.
[7] J. Blanchette, M. Summerfield, *C++ Programming with Qt 3* (ISBN 0-13-124072-2). :Prentice Hall, 2004.