Portable Gathering System for Monitoring and Online Calibration at ATLAS.

P. Conde Muíño on behalf of the ATLAS TDAQ HLT group *[1]

Abstract

During the runtime of any experiment, a central monitoring system that detects problems as soon as they appear has an essential role. In a large experiment, like ATLAS, the online data acquisition system is distributed across the nodes of large farms, each of them running several processes that analyse a fraction of the events. In this architecture, it is necessary to have a central process that collects all the monitoring data from the different nodes, produces full statistics histograms and analyses them. In this paper we present the design of such a system, called the *gatherer*. It allows to collect any monitoring object, such as histograms, from the farm nodes, from any process in the DAQ, trigger and reconstruction chain. It also adds up the statistics, if required, and processes user defined algorithms in order to analyse the monitoring data. The results are sent to a centralized display, that shows the information online, and to the archiving system, triggering alarms in case of problems.

The innovation of this system is that conceptually it abstracts several underlying communication protocols, being able to talk with different processes using different protocols at the same time and, therefore, providing maximum flexibility. The software is easily adaptable to any trigger-DAQ system.

The first prototype of the gathering system has been implemented for ATLAS and has been running during this year's combined test beam. An evaluation of this first prototype will also be presented.

*S. Armstrong[a], A. dosAnjos[r], J.T.M. Baines[c], C.P. Bee[d], M. Biglietti[e], J.A.Bogaerts[f], V. Boisvert[f], M. Bosman[g], B. Caron[h], P. Casado[g], G. Cataldi[i], D. Cavalli[j], M. Cervetto[k], G. Comune[l], P. Conde Muíño[f], A. De Santo[m], M. Díaz Gómez[n], M. Dosil[g], N. Ellis[f], D. Emeliyanov[c], B. Epp[o], S. Falciano[p], A. Farilla[q], S. George[m], V. Ghete[o], S. Gonzålez[r],M. Grothe[f], S. Kabana[l], A. Khomich[s], G. Kilvington[m], N. Konstantinidis[s], A. Kootz[t], A. Lowe[m], L. Luminari[p], T. Maeno[f],J. Masik[v], A. Di Mattia[p], C. Meessen[d], A.G. Mello[b], G. Merino[g], R. Moore[h], P. Morettini[k], A. Negri[w], N. Nikitin[x],A.Nisati[p], C. Padilla[f], N. Panikashvili[y], F. Parodi[k], V. Pérez Reale[l], J.L. Pinfold[h], P. Pinto[f], Z. Qian[d], S.Resconi[j], S. Rosati[f], C. Sánchez[g], C. Santamarina[f], D.A. Scannicchio[w], C. Schiavi[k], E. Segura[g], J.M. de Seixas[b], S. Sivoklokov[x], Soluk[h], E. Stefanidis[t], R. S. Sushkov[g], M. Sutton[t], S. Tapprogge[z], E. Thomas[l], F. Touchard[d], B. Venda Pinto[aa], V. Vercesi[w], P.Werner[f], S. Wheeler[h bb], F.J.Wickens[c], W. Wiedenmann[r], M. Wielers[cc], G.Zobernig[r].[a]Brookhaven National Laboratory (BNL), Upton, New York, USA. [b]Universidade Federal do Rio de Janeiro, COPPE/EE, Rio de Janeiro, Brazil. [c]Rutherford Appleton Laboratory, Chilton, Didcot, UK. [d] Centre de Physique des Particules de Marseille, IN2P3-CNRS-Université d´ Aix-Marseille 2, France [e]University of Michigan, Ann Arbor, Michigan, USA. [f]CERN, Geneva, Switzerland. [g] Institut de Física d´ Altes Energies (IFAE), Universidad Autónoma de Barcelona, Barcelona, Spain. [h]University of Alberta, Edmonton, Canada. [i]Dipartimento di Fisica dell´ Università di Lecce e I.N.F.N., Lecce, Italy. [j]Dipartimento di Fisica dell´ Università di Milano e I.N.F.N., Milan, Italy. [k]Dipartimento di Fisica dell´ Università di Genova e I.N.F.N., Genova, Italy. [l]Laboratory for High Energy Physics, University of Bern, Switzerland. [m]Department of Physics, Royal Holloway, University of London, Egham, UK. [n]Section de Physique, Université de Genève, Switzerland. [o]Institut für Experimentalphysik der Leopald-Franzens Universität, Innsbruck, Austria. [p]Dipartimento di Fisica dell´ Universit di Roma ´ La Sapienza´ e I.N.F.N., Rome, Italy. [q]Dipartimento di Fisica dell´ Università di Roma ´ Roma Tre´ e I.N.F.N., Rome, Italy. [r]Department of Physics, University of Wisconsin, Madison, Wisconsin, USA. [s]Lehrstuhl für Informatik V, Universität Mannheim, Mannheim, Germany. [t]Department of Physics and Astronomy, University College London, London, UK. [u]Fachbereich Physik, Bergische Universitat Wuppertal, Germany. [v]Institute of Physics, Academy of Sciences of the Czech Republic, Prague, Czech Republic. [w]Dipartimento di Fisica Nucleare e Teorica dell´ Università di Pavia e INFN, Pavia, Italy. [x]Institute of Nuclear Physics, Moscow State University, Moscow, Russia. [y]Department of Physics, Technion, Haifa, Israel. [z]Institut für Physik, Universität Mainz, Mainz, Germany. [aa]CFNUL - Universidade de Lisboa, Faculdade de Ciências, Lisbon, Portugal.[bb]University of California at Irvine, Irvine, USA. [cc]University of Victoria, Victoria, Canada.

REQUIREMENTS

The monitoring system for the ATLAS experiment has very strong requirements, focusing on user friendliness and flexibility. A scheme showing the general idea of the system in depicted in figure 1.

From the user[1] point of view, the gathering system should be completely transparent. Their aim is to produce the histograms or monitoring quantities using their standard software environment and tools, without worrying about the data transfer or the communication protocol.

From the point of view of the gatherer itself, the requirements can be summarized in the following way:

- It should be able to work in push or pull mode. This means that either the gatherer is waiting until the different processes send the monitoring data to it, or it requests the data from each one of the processes that are producing monitoring information.
- For each of the monitoring variables, it should be possible to choose if the gatherer has to sum up statistics or just publish single event quantities.
- The gatherer must be capable of using different communication protocols, being able to receive data using one protocol and send it using a different one. It may also be possible to receive data from different subsystems using different protocols at the same time or change the protocol used for the communications in a new run.

[1]Users here are any developer that wants to monitor something on the system, either software of hardware. The majority of users, however, come from the software developers community.

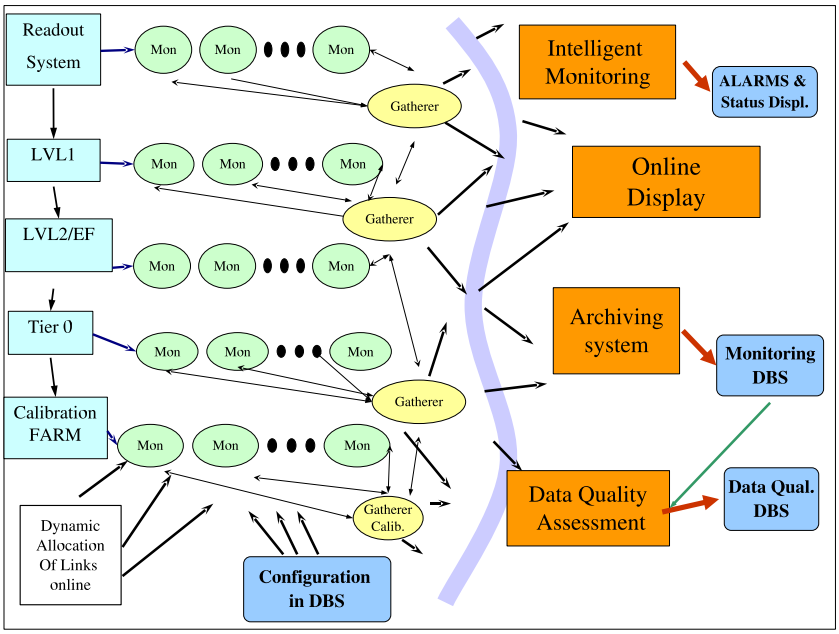

Figure 1: Scheme of the monitoring system for ATLAS. Any process in the Trigger DAQ chain can produce monitoring information that is distributed between the nodes of the corresponding farm. A central process, called gatherer, collects all the monitoring information, adds up statistics and sends the full statistics monitoring data to the online display and archiving. It provides also the infrastructure for the intelligent monitoring, that produces alarms and data quality assessments.

- The gatherer should react to state transitions (start/stop of run, configure, unconfigure, or any other one) since it should take certain actions at given times, like saving the histograms at the end of the run.

- It should be possible to send/receive not only histograms but also any user defined data structure.

- The gatherer should have a dynamic configuration, in such a way that new histograms or monitoring quantities are automatically taken into account, without the need of modifying the system configuration.

- The monitoring system should provide the framework for a more intelligent monitoring, executing user-defined algorithms that further analyse the monitoring data to produce alarms and data quality statements or new monitoring quantities, more meaningful and easier to understand for the shift crew.

DESIGN

The monitoring system can be divided in two very different parts: the first one corresponds to the infrastructure that makes the gatherer an online application to be run at the DAQ system of ATLAS. The other one consist of all the software that deals with the communications and the manipulation of the monitoring data. In the two following subsections the design of each of this parts will be explained in more detail.

Data transfer and communications

The design of the software to deal with the monitoring data and the communications has been done taking advantage of object oriented techniques, using the UML modeling language, since the implementation of the monitoring system had to be done in C++.

In order to fulfill all the requirements, each monitoring object should be handled in an independent way. Therefore, it is logical to define an object whose main purpose is to keep a copy of the monitoring data and deal with it. This object is called *MonObject* and it plays a central role in the monitoring system. According to the second and third requirements, it must be possible to configure each MonObject in a different way so each piece of monitoring data is treated according to its specifications. Thus, each MonObject configures itself at creation time.

To perform the data transfer two other objects were defined: a communications client object and a server object. They are tools that the MonObject uses to send or receive data whenever it is necessary. They have been designed as abstract interfaces, that can be implemented in many different ways according to the different protocols, but are always handled in the same way. Since the MonObject does not know about communication protocols, the server and client objects are created through a factory class pattern[3]. In this way, all the code dealing with communication protocols and data transfer is hidden inside the implementation of the server and client objects, and in the factory class. This allows to easily introduce new protocols without mod-

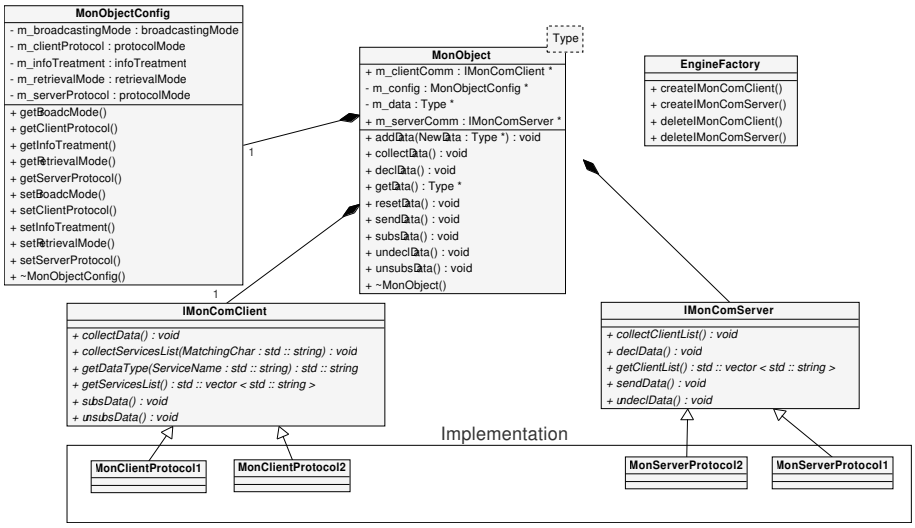

Figure 2: UML scheme showing the classes defined to handle the monitoring data and the communications.

ifications in the existing code.

The UML scheme representing these objects and the relations between them is shown in figure 2. The central object is the MonObject that has a configuration object containing all the information about how to deal with the monitoring data and two communication objects, one server and one client. The MonObject is implemented as a template, allowing to reuse the same code for different data types.

Infrastructure

The gatherer infrastructure was designed as an application belonging to the Data Flow software[4] of ATLAS. A scheme showing how the gatherer works is shown in figure 3.

At configuration time, the gatherer reads its configuration from a data base. The configuration is written in terms of groups of MonObjects that have common characteristics: client and server communication protocol, information treatment (sum up or substitute), retrieval mode (pull/push), broadcasting mode (wait for request or send after a certain time) and frequency to broadcast. The name of the monitoring variables is specified with wildcards, in such a way that the gatherer has to find out (usually contacting the name server) which monitoring variables are available that match the wildcard. In this way, the gatherer can automatically get new monitoring data that has been added, without the need of changing the data base for any single histogram that is included in a certain monitoring package. The gatherer has, in this sense, dynamic configuration.

Once the MonObjects have been created, the gatherer performs actions on them: configure, send data, collect data, ... The monitoring information is collected with a certain frequency, it is sum up (if required) and published in the online display or sent to the archiving system.

The gatherer reacts to state transitions, receiving orders from a controller. At the end of the run, for instance, the data is collected and it is sent to the archiving system.

IMPLEMENTATION: FIRST PROTOTYPE

The first prototype of the monitoring system has been implemented for the ATLAS combined test beam this year (summer 2004). The focus has been to provide a robust and user friendly monitoring system. The implementation was simplified with respect to the design due to the time constraints: the gatherer was configured only at start up, providing minimal interaction with the users once started (i.e., sending commands to the gatherer was not allowed), and only one protocol was implemented since there was no need to have more.

The communication protocol used is the standard one for the online applications at ATLAS, called Information System (IS)[2] and based on CORBA. In IS, all the information is stored in a central repository server. Each process sends its monitoring data to an IS server and the gatherer gets it from there. This method avoids slowing down the online processes by having too many requests coming from other processes.

To simplify the task of the software developers, an algorithm is executed at the Event Filter (the last trigger level) after all the reconstruction and monitoring packages. It gets the monitoring data, that in this case is only histograms, from the process memory and sends it to the gatherer. This procedure ensures that all the communications and data transfer is completely transparent for the users. It also ensures that new histograms are automatically received from the gatherer, without changing the databases.

Other processes from the Trigger DAQ[4] chain by default publish their information in one of the IS servers and therefore the users do not have to do anything special to get their information collected and summed up by the gatherer[2].

[2]There is only one restriction concerning the name of the published in-

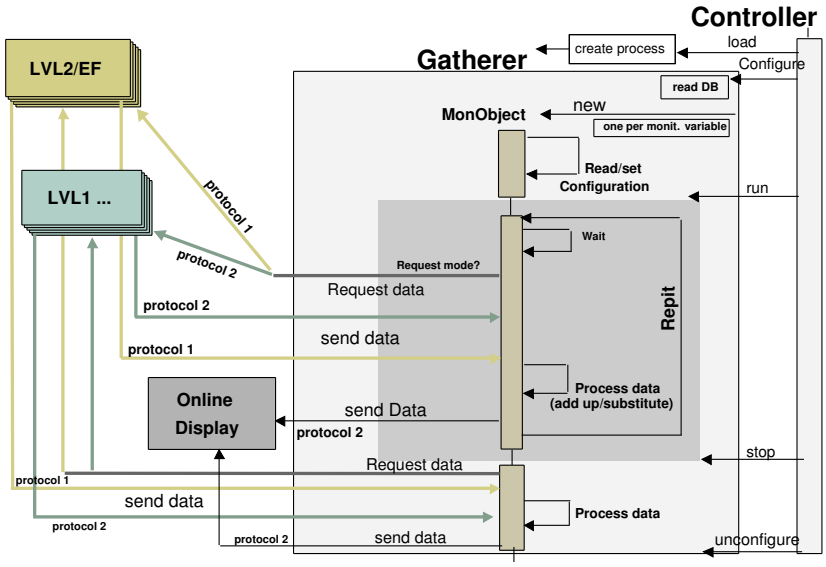

Figure 3: Scheme showing how the Gatherer works.

Performance

The gatherer has been running stably at the test beam since the beginning of August. Due to the fast evolution of the test beam software, new algorithms were added very frequently, so the average number of histograms and information transfered was increasing with time. The final size of data transfered was about 90 MB when the full Event Filter farm was running. In 2007, when ATLAS will start the commissioning, it is expected at least a factor 10-100 larger data size, since some of the detectors are planning to fill occupancy plots per detector channel[3] and the size of the Event Filter farm will also be larger.

The CPU consumption during the test beam was about 30 %, in a Xeon machine with a 3.2 GHz processor and 1 GB of memory, when the data is being transfered. The gatherer is idle about 30 % of the time, waiting some time before collecting and sending the data. The memory consumption was about 10 %, also during data transfer.

Since the IS server is not a real time system, when there are many processes sending large amounts of data at the same time, the data transfer may become slow. The flexible configuration of the gatherer, however, allows to improve the performance of the system by simply distributing the data within more servers or introducing more gatherers running in parallel. A tree of gatherer processes could be introduced if necessary.

SUMMARY AND CONCLUSIONS

The first prototype of a gathering system for the ATLAS monitoring has been designed and implemented for the ATLAS 2004 combined test beam, and it has been running stable for more than one month. The system has been designed to have maximum flexibility and user friendliness. It abstracts the communication layer, allowing transparency for the user and the possibility to use different protocols for the data transfer. It also makes the gatherer easily adaptable to any new experiment.

Having dynamic configuration and the possibility to use different communication protocols allows to tune the system in order to optimize the performance during the data taken.

formation: the name of the monitoring information should follow a standard, to allow the gatherer having dynamic configuration.

[3]There will be of the order of few hundreds million channels.

REFERENCES

[1] ATLAS TDAQ HLT group, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/HLT/AUTHORLISTS/chep2004.pdf

[2] http://atddoc.cern.ch/Atlas/DaqSoft/components/is/Welcome.html

[3] E. Gamma, R. Helm, R. Johnson, J.Vlissides, *Design Patterns*, Addison Wesley, 2003.

[4] The Atlas Collaboration, *Atlas High Level Triggers, Data Acquisition and Controls, Technical Design Report*, CERN/LHCC/2003-022, 2003.