The usage of ROOT for Online Monitoring in the ALICE DATE system

Wisla Carena[1], *Roberto Divià*[1], *Fons Rademakers*[2], *Klaus Schossmaier*[1], *Pierre Vande Vyvre*[1], *Alessandro Vascotto*[1], *Krzysztof Zelazowski*[1]

For the ALICE Collaboration

[1] European Laboratory for Particle Physics (CERN), Geneva, Switzerland
[2] Gesellschaft für Schwerionenvorschung (GSI), Darmstadt, Germany

Abstract

Data Acquisition systems for HEP applications need constant monitoring (online and offline) of their data streams to accomplish several tasks: quality checking, tuning, statistics, pre-analysis. Monitoring tasks can and should use the same tools as data analysis products (conventions, libraries, environments) to reduce training, installation, development and support efforts and - at the same time - to strengthen the liaison between the online and the offline worlds. The ALICE DATE Data Acquisition system available today for R&D and for test beams is fully integrated with the ROOT environment. A simple DAQ-oriented approach and a more complex OO-based model have been developed to allow a variety of programming paradigms and to validate the complete life cycle of monitoring tools, both for online and offline environments.

Keywords: Monitoring, DATE, ROOT, ALICE

1 Introduction

Developing the Data Acquisition (*DAQ*) system for an experiment in High Energy Physics implies the accomplishment of several ancillary tasks, such as online control of the quality of the data streams, fine tuning of hardware and software parameters, gathering of statistics, data pre-analysis. All these jobs are at the same time detached and correlated with the main DAQ system, they are not under direct control of the main DAQ state machine and do not necessarily obey to the same rules as the DAQ software. However, they need to cope with the data and control patterns coming from the online and they must follow several implicit and explicit rules. Moreover, these tasks often require sophisticated programming environments (tools, libraries, resources, programming techniques) that can make development, debugging and integration rather difficult. In this paper we illustrate how the development of these monitoring processes can be sensibly facilitated by using the right techniques and programming environments, provided that an appropriate interface with the Data Acquisition stream is available.

2 Online vs. Offline: two different worlds?

Traditionally, High Energy Physics experiments divide their data handling activities into two main categories: Online and Offline. The Online attribute is given to all components active during the transfer between the front-end electronics (detector(s), triggers, sensors) and the Permanent Data Storage (hard disk, tape, CD). The Offline activities cover the libraries, algorithms and simulation sessions used before and after the data collecting process. The boundary of this partitioning can vary and it is not unusual to “import” each other’s products whenever needed (e.g. for online filtering of raw event data or for offline decoding of experimental data).

Why this separation? The main reason lies in the difference between the online and offline development and production environments (platforms, tools, frameworks). However, the current trends in hardware and software products are reducing this gap and aim to make it more cultural then real. Online and Offline applications can now use similar - if not identical - hardware platforms, operating systems (in their standard form or with real-time extensions) and support libraries. The same tools and code can now be shared (logically and physically) throughout the whole collaboration. More performant resources (CPU power, system/auxiliary/permanent storage, interconnecting medias, accessories) are available at reasonable costs while the requirements for the collection of data from the hardware remain unchanged. The use of cooperating processes and multiple processors makes load distribution and balancing easy to achieve and effective in its implementations. It is therefore possible to have a considerable amount of processing power with the right tools and the needed libraries about everywhere in the experiment. All this should lead to a broader uniformity, a tighter collaboration and a better use of resources throughout the complete processing chain.

3 The ALICE Data Acquisition and Test Environment

The R&D groups involved in the ALICE [1] collaboration felt the need of a product capable of providing an environment suitable for rapid prototyping of hardware and software modules and to support the Data Acquisition system for test setups, both at CERN and at external locations. To meet this requirements, the ALICE DAQ team at CERN developed the ALICE Data Acquisition and Test Environment (*DATE*) [2]. DATE provides the tools, libraries and skeletons needed for the quick setup of a test acquisition system, for its integration in multi-detector environments and the development of the ancillary utilities needed for control and monitoring. Available on several Unix platforms (Solaris, AIX, OSF-1 and Linux), DATE uses a parallel processing approach and relies on Unix Interprocess Communication (IPC) features such as shared memory, semaphores and multi-process locking.

The main Data Acquisition process follows a well defined procedure under the strict control of the DATE global state machine. For what concerns monitoring, more flexibility must be granted and therefore a standard skeleton is not provided. What is available instead is a set of libraries that allow a liaison between the raw data stream(s) and the monitoring task(s). This liaison permits monitoring from the same CPU responsible for the generation of the online data stream, from other CPUs part of the same Data Acquisition system (e.g. the event builder or the run control host), from CPUs connected to but not part of the DAQ system and from non-DAQ streams, such as Permanent Data Storage medias (tapes, CDs, etc) or replay of online data stream sources. DATE provides libraries and tools for all the above styles of monitoring on all platforms where a DAQ system can be implemented. Furthermore, libraries with reduced capabilities are available for other Operating Systems, where only remote or file monitoring can be performed. This allows “foreign” machines to process raw data coming from the live DAQ stream or from Permanent Data Storage systems such as the Central Data Recording (*CDR*) facility available at CERN.

4 Features and limitations of the DATE monitoring scheme

Monitoring activities within a Data Acquisition system based on DATE must have little impact on the main experimental data stream. Several “fall-through” paths have been provided to avoid overhead whenever monitoring is not active (either disabled or without clients). Transfer of events between the online and the monitoring streams has to be performed as smoothly as possible and should not sensibly affect the global behaviour of the DAQ system. Other important features of the

DATE monitoring scheme allow to effectively select only the desired events as soon as possible in the transfer process (to offload CPU, system resources and communication medias). Several selection mechanisms are available, based on the type of event, on user-defined attributes and on different policies and aging algorithms.

The chosen design of the DATE monitoring system prevented the development of features that may have introduced a negative impact on the overall performances. Monitoring clients can receive only one stream at a time and do not combine their requests to offload the charge on the network. Finally, the format of the data exchanged via the DATE monitoring scheme follows closely the same conventions used for the readout and recording stages. The output is not trivial to decode and it is most likely incompatible with what higher level facilities - such as visualisation, features extraction and filtering algorithms - expect to receive.

5 The ALICE ROOT framework

A High Energy Physics collaboration such as ALICE must define a framework suitable for the development and the support of complex data production and data analysis environments. This framework must offer an adequate set of tools and libraries allowing the creation, interpretation, manipulation and visualisation of data objects coming from the Data Acquisition and from simulation sessions. Furthermore, this framework must be available on all the platforms - hardware and software - in use by the collaboration. At CERN, the choice of ALICE went mainly to HP-UX hosts and commodity PCs running Linux, although other types of platforms such as Solaris and AIX are also in use, especially for online tasks. Note that we are talking about different implementations of the Unix Operating System, so a common Unix approach could be envisaged. However, a Unix implementation may differ from another in several details and this can make code portability across different platforms quite tricky. High-level functions and libraries may not be available under the same form. Tools such as editors, compilers, data access routines, may not be consistently available.

Looking for a solution to the above problems, the ALICE collaboration decided to use the ROOT package[3] (see also `http://root.cern.ch`) and to propose it as a part of the common LHC framework. We believe that ROOT is an ideal environment to introduce physicists quickly to the world of Objects and C++. ROOT has the capability to handle efficiently large amounts of data objects (accessible via their individual attributes) and includes a remarkable set of libraries for producing histograms, plots, graphics, data visualisation and interaction with the Operating System. ROOT provides a built-in C++ interpreter that allows fast macro prototyping and offers an good environment to learn and use C++. The availability of ROOT on all the major platforms in use in ALICE gives an optimal common development and run-time environment, independent from the hardware and software architectures.

6 DATE/ROOT monitoring scheme: implementation

We decided to put DATE and ROOT together in order to achieve the following goals: reduced training for the Online environment, uniformity of coding conventions, use of standard libraries (system and ALICE-specific), independence from the Operating System, use of special ROOT capabilities for inspecting and debugging Objects.

Our first step was to convert the existing DATE monitoring libraries into sharable objects (to allow their dynamic loading within the ROOT framework). This gave a working environment suitable for simple monitoring programs getting and handling raw DATE events. However, this approach introduces some conflicts between DATE and ROOT, as they share - and need complete

control over - several system resources (semaphores, process control, sockets).

To avoid this interference, we have decided to use a more complex model, where a specially-tailored Object - called Monitoring Module - would provide an enhanced paradigm to access DATE monitoring streams. Based on a set of separate cooperating actors, this new model also extends the capabilities of the basic approach. At first we adopted the same mechanisms used by DATE (multiple processes using shared memory segments and system semaphores). Unfortunately the result did not suit very well the ROOT architecture and we therefore switched to a new method based on Unix sockets. This implementation proved to satisfy our requirements while providing a reasonable efficiency. Data copying is reduced to the minimum possible and events requested by multiple monitoring modules are transferred from DATE into the ROOT environment only once. Modules belonging to different processes (either different ROOT sessions or different processes using pre-compiled ROOT environment) do share actors and event data. DATE events are converted into ROOT objects using a copy-less approach that makes objectification rather efficient, so efficient that we have proved to be more effective to objectify at the destination rather then at the source. While it is still possible to use the traditional monitoring approach (where the process stops and waits for the next available event), we provide a more Object-Oriented paradigm where the module needs to be subclassed to create an extended ROOT environment: this allows the application to run the ROOT system event loop while handling the DATE monitoring stream(s). The Monitoring Module is now available within the standard DATE kit on Solaris and Linux, with porting to AIX in progress. Finally, the Monitoring Module was used to develop a utility called *EventBrowser*. This tool allows the dynamic inspection of any DATE Data Acquisition system starting from the upper level down to the last byte of the monitoring stream, to explore the structure the DAQ streams and to perform some simple statistics. The EventBrowser proved to be reliable and effective, and it is now part of the standard DATE distribution kit.

7 Conclusion

We have been impressed by the effectiveness of the ROOT environment, by its features (the interactive compiler, the online inspection of ROOT objects, the whole framework) and by the ease of integration between the DATE monitoring scheme and ROOT. Never before we had such a powerful framework available on all sorts of hosts (from the small Data Acquisition front-end cards up to the central development cluster). Developing, debugging and validating C++ code has been completed efficiently. We had to play a bit with the ROOT environment to find the right way of doing things and this gave us several hints on the proper way to integrate ROOT in online environments, where efficient interactions with external actors are required. We ourselves have been able to develop our monitoring application quickly and effectively. We think that ROOT is an excellent tool to integrate Online and Offline activities within the ALICE collaboration and this justifies our efforts to offer an efficient and reliable liaison between ROOT and DATE.

References

1 ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC, CERN/LHCC/95-71, 15 December 1995

2 ALICE DATE V3.5 User's guide, ALICE internal note INT-99-46, December 1999

3 R.Brun and F.Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, September 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86.