Use of ROOT in the DØ Online Event Monitoring System

J. Snow[1], *P. Canal*[2], *J. Kowalkowski*[2], *J. Yu*[2]

[1] Langston University, Langston, Oklahoma 73050, USA
[2] Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, Illinois 60510, USA

Abstract

The DØ experiment is one of the two High-Energy proton anti-proton collider experiments at Fermilab, USA. Since the detector serves multiple physics purposes, it consists of many different sub-detector systems together with supporting control systems. Therefore, online event monitoring plays a crucial role in ensuring detector performance and data quality by parasitically sampling events during the data taking. ROOT, a physics analysis package developed at CERN, is used in the DØ online monitoring as the main analysis tool, providing a graphical user interface that interacts remotely with an analysis executable and tools to monitor informative histograms as they get updated in shared memory throughout the data taking. In this paper, we present the basic structure of the DØ online monitoring system and the use of ROOT in the system.

Keywords: chep, DØ, D0, DZero, D-Zero, ROOT, online

1 Introduction

In DØ the online event monitoring system is comprised of multiple event monitor programs attached to the DAQ system, requesting events with desired trigger types. The programs are built under a fully asynchronous interactive framework [1] written in C++. The details of the DØ Data Acquisition (DAQ) system are described in Ref. [2]. The availability of physics analysis tools, shared memory, a browser, network sockets, and Graphical User Interface (GUI) classes, makes ROOT [3] an attractive choice for online monitoring applications. Therefore, ROOT is used as the main analysis tool for the monitoring programs in the DØ experiment. The results (mostly histograms) from the monitoring programs are stored in shared memory in ROOT object format. The main mode of accessing the results is to browse the objects in shared memory with a ROOT GUI via socket connections. In the next sections the DØ online event monitoring system is described in more detail.

2 DØ Online Event Monitoring System

The DØ online monitoring system consists of three major components:

- Data Acquisition System (DAQ)
- Monitoring Executables (Examine)
- Graphical User Interface (GUI)

These three components can be further subdivided into smaller components. These smaller components run on one of the three operating systems - Windows NT, Linux, and OSF1 - due to hardware specifications. Therefore the DØ online monitoring system requires portability of software across the operating systems and platforms. The DAQ consists of front-end electronics, two

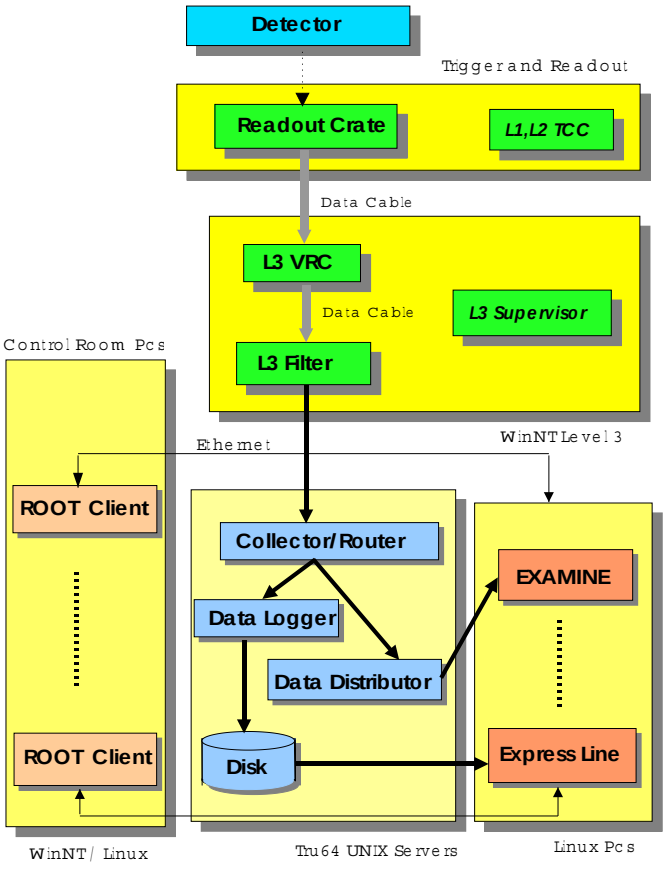

Figure 1: DØ Run II Online Event Monitoring Architecture

levels of hardware triggers, a software trigger, and data transfer systems. A detailed description of the DAQ system for the DØ experiment [4] can be found in Ref. [2]. Figure 1 shows the logical data flow of the DØ online event monitoring system architecture. The system is designed to be fully expandable depending on the bandwidth needs the system will encounter.

A collector/router (C/R) can handle the data being output from multiple level 3 (L3) software trigger nodes. The C/R distributes data to Data Loggers (DL's) that records the data into files based on the trigger condition a particular event satisfied. This path is tightly controlled to ensure proper normalization and weighting of each event across all the L3 nodes. In other words, if any of the DL's has a problem in writing out an event to a file, all other DL's must stop and the deliberate clogging of the data path must be propagated to all L3 nodes. The C/R also sends all events it receives to the Data Distributor (DD) [2] that assigns event buffers and passes the events based on the selection conditions transferred to it by the connected monitoring processes. The data flow in this path is uncontrolled, because it is desired to continue taking data even if a monitoring process is in a stalled state. It is in this path where the event monitoring occurs.

Therefore, the entire monitoring system involves, starting from L3, four separate software processes, excluding the GUI, running on three different operating systems and platforms. The executables of the DØ online monitoring run on a farm of personal computers (PC's) that run under the Linux operating system while the GUI may run under Windows NT or Linux.

2.1 The Executable - Examine

The executable portion of the DØ online monitoring system is called Examine[1]. Examine executables are built under the DØ fully asynchronous interactive program framework [1]. Examine executables unpack raw data, reconstruct each event to the level required by individual programs, define and fill necessary histograms, and provide event displays.

Multiple Examines for various purposes run on the Linux PC farm. The Examines can be, however, categorized into two large groups. The first is detector performance monitoring which is geared toward physical quantities that provide information for detector hardware diagnostics. The second is a global monitoring Examine. This Examine performs full reconstruction of the events to provide information concerning physics objects reconstructed based on specific algorithms. In other words, this Examine provides the user the information on how many electrons, muons, or jets have been produced during the run. Therefore, this Examine allows users to obtain an overall quality of the data being taken. The global Examine also provides an online real-time event display function for more instructive information in an event-by-event basis.

An Examine makes an event selection request via a Run Control Parameter (RCP) file [5] editable by the user. It transfers the selection conditions - any combination of the trigger names, data stream numbers or names - to the DD, where it makes a connection to the DD via a client-server package based on an ACE protocol [6]. This request causes the DD to assign an event buffer whose characters are controllable in an RCP file, and at the same time the Examine starts up three independent threads for event transfer communication between the DD and itself. When this procedure finishes, the Examine starts another buffer, whose queue depth is RCP controllable, to store events transferred from the DD to ensure a guaranteed event presence in the buffer for the processing, independent of the whole analysis process. The Examine also starts up a separate thread for histogram display, interacting with the GUI to allow uninterrupted access of histograms by the user. It also puts histograms in a shared memory for updated accessibility of the histograms while they get filled during the computation.

While ROOT provides a global framework not only for physics analysis tools (PAT) but also for I/O, data packing, Monte Carlo, and GUI, the DØ online monitoring system only uses the PAT, GUI, and socket communication portion of the ROOT framework. The diagnostic histograms are booked and filled in Examine in ROOT format and stored in memory as ROOT TFile objects [3]. Communications between the Examine and a GUI uses the socket classes of ROOT. Employing the TServerSocket class enables multiple GUI's to attach to a single Examine from potentially diverse remote locations.

2.2 The Graphical User Interface - GUI

The GUI allows the user to interact with an Examine process. The user may obtain information (*e.g.* histograms, status, *etc.*) from the Examine process and may control the Examine as well (*e.g.* start/stop, set parameters, *etc.*). Figure 2 shows a prototype GUI window built entirely from the GUI classes provided by ROOT on an IRIX platform. The top portion of the GUI acts as Examine process control, while the bottom portion of the GUI acts as histogram control.

When a GUI starts up, it does not have any associated processes. Thus the GUI first inquires for existing Examine processes to the process registry, a name server, to allow users to attach to a pre-existing process that the user is interested in. This allows for a maximally efficient use of computational power and more effective sharing of event statistics. The GUI also provides an ability of starting up a new Examine process on a least used node among the Linux server nodes. The Examine processes are registered to a process registry by the version numbers of a governing

[1] The name inherited from the monitoring programs of the previous run.

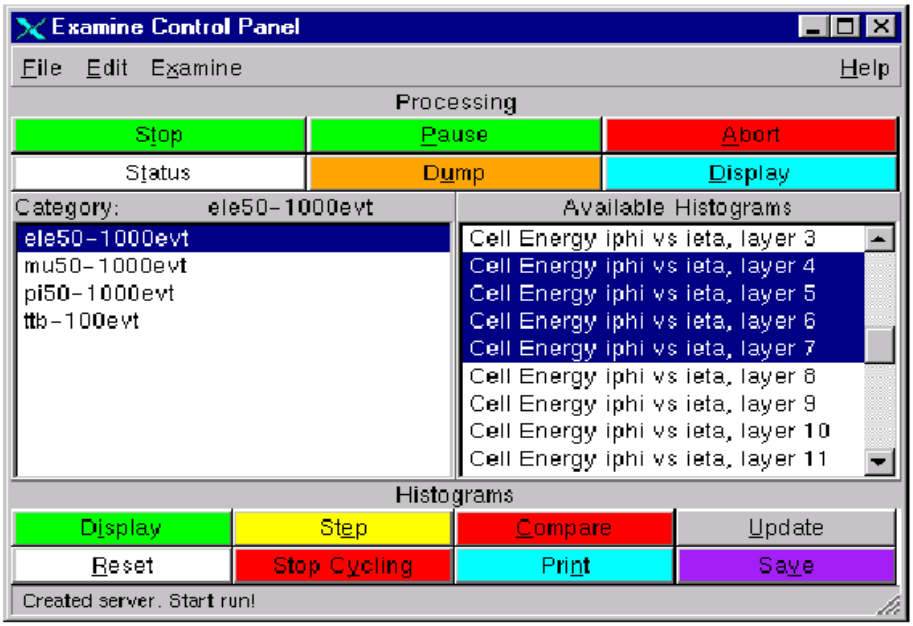

Figure 2: A Prototype of the DØ Examine GUI

RCP file. It is this scheme that allows users to parasitically attach to existing Examine processes to access the histograms from shared memory.

While many GUI's may attach to an Examine, only one, the creator is allowed to control the Examine. The creator GUI may Start/Stop, Pause/Resume, Abort, obtain Status information from, Dump an event from, and initiate an Event Display from the Examine process. For a non-creator GUI (observer) the control functions are inactive. On the other hand, both the creator and the observer GUI's have histogram control which allows various monitoring tools for more effective use of given information accessible in histograms. Users can access individual histograms one at a time, viewing a snap shot of the histogram in the shared memory. Users can select and cycle through one or more histograms continuously, updating the histograms every time histograms are displayed. The users may also enable a continuous updating of the selected set of histograms with a chosen update frequency. Users may initiate comparisons of the given histograms to a reference set, distinguished by the names of the histograms rather than traditional histogram ID numbers. In addition complex histogram operations between the current and the reference set are possible because these are generic functions of ROOT as a physics analysis tool. The control GUI may also Reset the Examine histograms, Save histograms to a file in ROOT format on a disk local to the GUI, and Print the displayed set of histograms.

The communication protocol used for the messaging between the GUI and the Examine executable is ROOT based socket communication. Presently the control and histogram portions of the GUI communicate synchronously with the Examine. ROOT socket classes allow complex objects such as histograms to be transferred as easily as string messages.

Significant events from the DAQ and Examine, whether expected such as End of Run condition, or unexpected such as error conditions must be communicated to the GUI user. These events may occur at any time, hence there is a need for asynchronous communications from the Examine to the GUI. This is accomplished using a TServerSocket which is active during the system idle loop. The idle loop is accessed by inheriting from the ROOT TApplication class.

3 Conclusions

The DØ online event monitoring system (DØEMS) utilizes a fully asynchronous interactive framework to reconstruct events, calculate relevant physical quantities, and fill histograms. The user interacts with the framework through a GUI which provides information and control functions. Features of the ROOT framework have been successfully integrated into the DØEMS. While the proton-antiproton collider data taking is about a year away, the DØ experiment is currently in the process of commissioning individual detector components. The DØEMS described in this paper is being used for verifying the detector component performances. The responses from the users will provide direction for improving the system. In the near future web based access of the event monitoring system information will be available to remote collaborators.

References

1 J. Yu and J. Kowalkowski, "DØ Run II Online Examine Framework Design and Requirements," DØ Note #3578, Unpublished (1999); J. Kowalkoski *et. al.*, "DØ Offline Reconstruction and Analysis Control Framework," in these proceedings, talk presented by V. White.

2 S. Fuess, "DØ Run II Online Computing," DØ Internal Note # 2935, Unpublished (1996); S. Fuess and J. Yu *et. al.*, "DØ Run II Data Distributor Requirements," DØ Internal Note # 3540, Unpublished (1998)

3 Rene Brun, Fons Rademaker, and M. Goto , `http://root.cern.ch/`

4 S. Abachi *et. al.*, DØ collaboration,"The DØ detector," Nucl. Instr. Meth. **A338**, 185 (1994); `http://www-d0.fnal.gov`

5 M. Paterno, "RunControl Parameters at D0," `http://cdspecialproj.fnal.gov/d0/rcp/D0LocalGuide.html`, (1999)

6 D.C. Schmidt *et. al.*, ACE Team, "The ADAPTIVE Communication Environment," `http://www.cs.wustl.edu/~schmidt/ACE.html`