Online Monitoring and Module Maintenance for CDF in the Upcoming Fermilab Tevatron Run II

B. Angelos[1], *T. Arisawa*[3], *F. Hartmann*[2], *N. Ho*[1], *K. Ikado*[3], *S. Jones*[1], *K. Maeshima*[1], *R. Pordes*[1], *H. Stadie*[2], *G. Veramendi*[4], *H. Wenzel*[2], *S. White*[1], *J. Yoh*[1]

[1] Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, Illinois 60510, USA
[2] Institut für Experimentelle Kernphysik, Universität Karlsruhe, Engesserstr. 7, 76128 Karlsruhe, Germany
[3] Waseda University, Tokyo 169, Japan
[4] Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA

Abstract

In this article we discuss two topics. Both are part of the CDF Run II-online system. First we present the design and current status of the CDF online monitoring project which is based on ROOT. The framework consists of three parts, the online event analysis programs to check the detector and data, the browser to display their results, and the server program which communicates with the display via socket connections. The second topic is an ORACLE based database application. It keeps track of the exact location, repair history, modification history and status of all DAQ electronic equipment during development, commissioning and physics running periods. We discuss the implementation and status of the database.

Keywords: daq, online, database

1 Introduction

The CDF experiment consists of many detector subsystems and will run in a high rate large bandwidth data transfer environment. In the experiment, it is crucial to monitor the performance of each subsystem and the integrity of the data, in real time with minimal interruption. Therefore multiple event monitor programs are attached to the DAQ system [1], requesting events with desired trigger types. We use the object oriented data analysis framework ROOT [2] for the monitoring programs. ROOT is written in C++ and the availability of physics analysis tools, shared memory, browser, socket connection, and GUI[1] classes, make it an attractive choice for online monitoring applications. The results from the monitor 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 via a server with a Display Browser using socket connections. In section 2 we describe the CDF online event monitoring system in more detail.

Section 3 describes the ORACLE based database application which keeps track of the exact location, repair history, modification history and status of all DAQ electronic equipment during development, commissioning and accelerator operations. Information about the modules is being input by the engineers and technicians during module construction, by the physicists debugging the modules in test systems, as well as automatically from testing programs during acceptance testing and commissioning. During the run the location and problem information will be updated by the shift crew, and the repair information by the operations staff. Keeping a full history of each module will allow analysis of existing and prediction of oncoming module problems and identify trends in the hardware performance that affect the quality of the data collected. The CDF Module Database provides a complete history of information about the hardware modules which allows later investigation of anomalies and problems during the many years of subsequent analysis of the data. We discuss the implementation and status of the database.

[1]Grapical User Interface.

2 CDF Online Event Monitoring System

A schematic view of the CDF online monitoring system is shown on the left in Fig. 1. The online event monitoring programs are called ‘consumers’, where a consumer is defined as a process which receives events from the Consumer-Server (CS) [3] in real time. The CS fetches events from the Level 3 software trigger [4]. A consumer can also be used for more than monitoring; it could perform other tasks, such as real time calibration. Here, we will focus on the CDF consumer-monitor framework.

2.1 The CDF Consumer-Monitor Framework

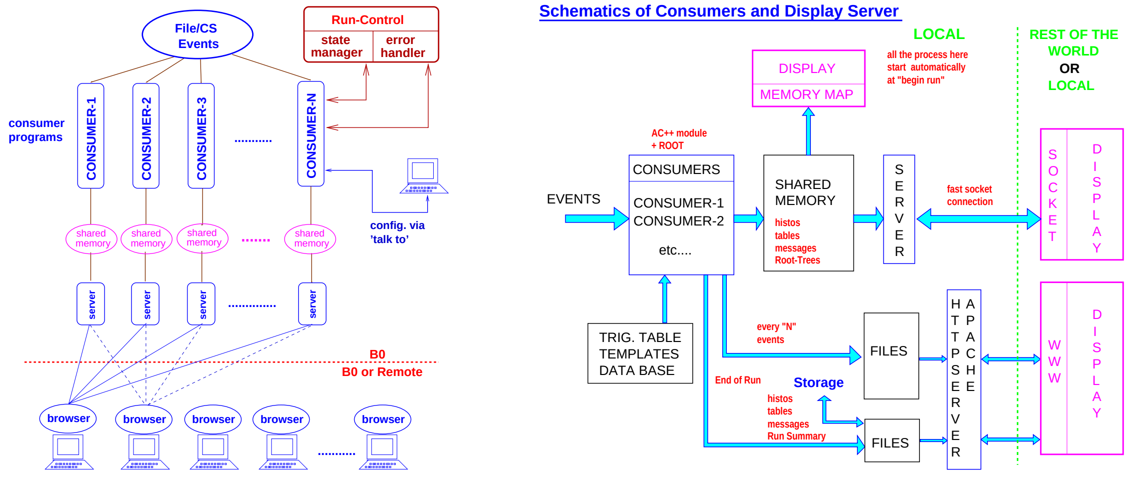

Figure 1: CDF Online Consumer-Monitor Framework Architecture.

A schematic view of the framework [5] with its elements is shown on the right in Fig. 1. For Run II we decided to separate the display part and Consumer-Monitor part, allowing us to monitor the experiment locally in the control room and remotely with no interruption. The task of the Consumer-Monitor process is to analyse and monitor the event data and to store the results into shared memory in the form of histograms, tables and warning messages. This information can then be viewed by the display browser via a server in real time. Results of the monitors are stored as data files periodically during a run and at the end of each run. These files can be viewed via a WWW server using the same display browser as described above. We will archive them systematically, so that we can readily access them, enabling us to do comparisons with the current data. The display browser provides a GUI to view the online monitored results conveniently, while also providing some basic utilities to do comparisons with previously stored results. The only parts of this framework that need to be run at the experiment are the Consumer-Monitors and servers. By separating the two tasks of monitoring and displaying, we remove the CPU and BUS load associated with displaying graphics from the machine which runs the Consumer-Monitors. Also, network traffic is minimised by serving only small objects (ie. histograms). During the data taking, multiple consumer processes run in parallel, receiving event data with the desired trigger types from the CS. Different consumers can run on independent CPU’s on different platforms[2]. Communication between a consumer and Run_Control[3] is handled using a commercial

[2]currently CDF supports IRIX, LINUX, SUNOS and OSF.
[3]The process which controls the CDF DAQ system.

package called Smart Sockets. The two types of communication that take place here are: 1) The State-Manager, which might be part of Run_Control or an independent process, watches the state of consumers. The consumers and Display servers are started automatically via this process. If for some reason, a consumer or server dies, the State-Manager will restart it. The Status of each Consumer-Monitor and server process is available on the WWW. 2) Severe errors detected by a Consumer-Monitor program which are in need of immediate attention are communicated automatically to the Error_Handler part of the Run_Control process. Besides this two processes the consumer framework has the following main components:

- **Consumers-Monitors:** are AC++ [6] modules, where AC++ is the CDF offline analysis framework, which allows to combine different modules, serving as input, output, event generator, reconstruction and analysis modules. AC++ handles the data flow between the modules. ROOT is used for event I/O and as an physics analysis tool. Things typically monitored by Consumer-Monitors are: detector occupancies (dead/hot channels), trigger rates and logic, luminosity, Level 3 reconstruction, physics objects, vertex positions, etc... Specifics of each of the Consumer-Monitors are determined and written in collaboration with the experts of each subsystem and are beyond the scope of this paper. The consumer framework provides a template consumer program, which includes the basic functions such as input/output, connection to data base (calibrations, trigger-table), and basic classes for the outputs. These base classes allow the server to interpret the stored objects, hence enable us to browse the monitored results efficiently. The template program also provides common utility methods which are useful for monitoring and some examples of how to use them. In the steady state running condition, the output format should remain unchanged, however, for debugging purposes we will also provide interactive operations of histogram handling controlled via a text file or a GUI. There are 3 ways to input events to Consumer-Monitors: 1) via consumer-server, 2) read disk files, and 3) generate random events at input stage. 1) is used during the data-taking, 2) and 3) are useful for developing and debugging programs.

- **Display Server:** is a program that allows the display browser programs to connect to it as a client and to access the information in the shared memory. Since it needs access to shared memory it has to run on the same machine as the Consumer-Monitors. Several Display Browsers can connect from anywhere in the world without having any effect on the consumer itself. The server will handle the requests, giving the process from the data-taking shift crew the highest priority.

- **Display Browser:** is a program that can run on the same or on a different machine with various ways to access the data:
  - access shared memory directly when running on the same machine. This option should be used just for debugging purposes.
  - access shared memory via socket connection to the Display Server. This is the default way to access the information. The browser should preferably run from a remote machine so that the CPU-load associated with displaying graphics is transferred to the remote machine. The graphic capability of a simple PC is more than adequate nowadays.
  - via the world wide web. There is a plug-in for the Apache web-server which makes the server ROOT aware and allows to access to ROOT files via a web browser.

  The current CDF online Consumer-Monitor GUI together with some histogram displays is shown in Fig. 2. On the left we see the Control GUI which allows the selection of different input modes and to browse the input to select the objects for display. In addition one can choose if the display is updated continously or per mouse click. On the right we see histograms from central calorimeter ADC data taken recently at the CDF experimental hall (YMON CEMD, Wedge Test) as well as some simulated data (Tracks). The data were analysed by Consumer-Monitor programs. The summary canvas shows results achieved by analysing and fitting the ADC histograms making use of ROOT as an Physics Analysis Tool. Shown are mean, RMS and fit results of the different histograms.

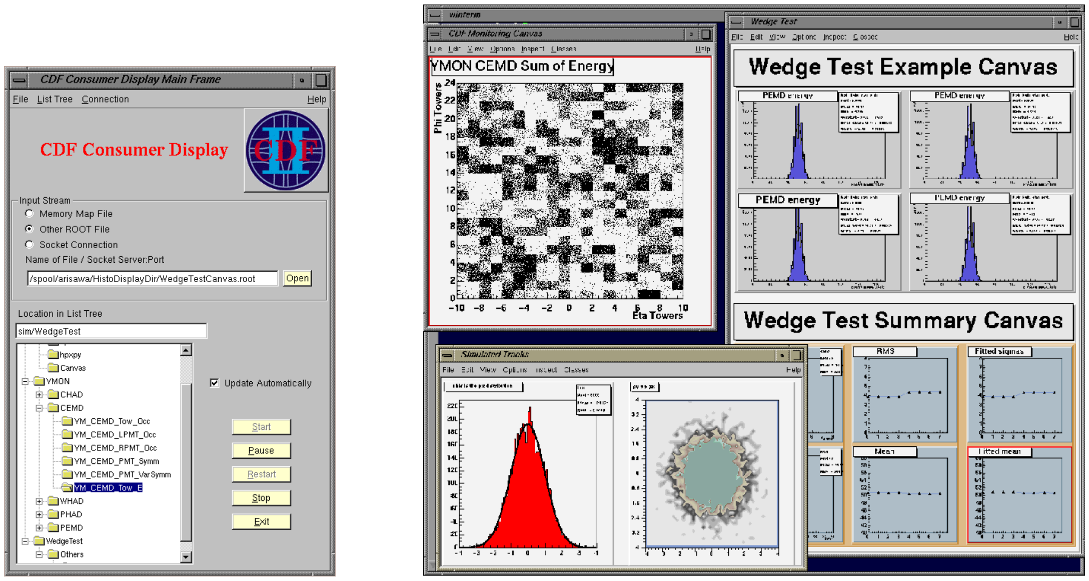

Figure 2: Example of CDF Online Consumer-Monitor GUI in development and some histogram display.

2.2 Use of ROOT in the CDF Online Monitoring System

ROOT is a very exciting new computing tool in high energy physics. It provides not only the physics analysis tools to replace PAW[4], but it also provides new features like classes for shared memory, socket connections, web server connections and GUI's. These features have been programmed using object-oriented design. Examples of how to use their functionality are provided. The object oriented approach makes is easy to design a data driven client server architecture. The CINT C++ interpreter and its debugging capabilities allow fast prototyping to some extent. One needs to be aware, that the interpreter has its limitations and that there are differences to ANSI C++. ROOT is still in the development stage. The source code is provided so that it is possible to find bugs and modify the code. The ROOT team has been very responsive to questions and requests. Fermilab actively supports ROOT for RUN II and provides tutorials and hands-on classes for new and advanced ROOT users [7].

2.3 Current status of the Consumer Framework

We have tested all the components of the CDF Consumer-Monitor framework and found them to work well. We are currently developing the full scale software programs to be used in the upcoming Run II. Since the CDF offline environment is still changing, a lot of time is devoted to maintain and adjust the code accordingly. Currently the first Consumer-Monitor programs to monitor detector occupancy and triggers [8] are being developed, using the provided templates

[4]Physics Analysis Workstation: physics analysis tool that we used during Run I.

and framework. We intend to use this programs during the ongoing commissioning of the CDF detector. Work is in progress to provide online documentation explaining the use of the framework and how to use the template to develop monitor programs.

3 Module Maintenance

The CDF Module Database [9] keeps track of the exact location, repair history, modification history and status of all CDF data acquisition modules during development, commissioning and accelerator operations. Many of these electronic modules are constructed and initially tested off site from Fermilab; they are then moved to Fermilab test stands for acceptance testing; moved to the CDF counting room for data taking; return to repair shops for fixing; perhaps go back to the original sites for upgrade; then are returned to the counting room and placed in a different location - crate, slot, rack - in the data acquisition system. The database supports all information about a module - serial number, ECO [5], prom versions, as well as specific location with in the CDF readout system. Full update and query access, with appropriate security, are provided over the web to allow the geographically distributed groups to update and view their information.

The CDF Module Database is a small extension to the main Computing Division (CD) equipment database, MISCOMP-EQUIPDB [10], which tracks information about all supported systems, their peripherals and software licences, as well as serial numbers, locations, ownership and repair history of most of the data acquisition modules used in the Fixed Target experiments as well as some CDF and D0 modules. The database supports the transfer of location and 'claim' on a module between groups such as CDF, CD repair groups, between experiments etc.. Access rules are implemented to ensure that only the appropriate groups or people are able to update the information.

The database management system is ORACLE; data input is through web based oracle forms; reporting is done through a combination of oracle reports, automated cron jobs, through the web using a a flexible PERL CGI script (MISWEB) or using commercial querying tools such as Crystal Reports. A custom JAVA interface has been developed by the CDF Silicon subproject which provides real time input of data from the module and detector test stands during acceptance testing. We anticipate that once the run has started, applications will make use of the various oracle API's to integrate information about modules into experiment applications and reports.

The success of such a database depends on its acceptance by the people "in the field". The efforts now are concentrating on tuning the application to deliver the most intuitive and easy to use update and reporting interfaces.

4 Acknowledgements

We would like to thank all the people who are contributing to this work. Special thanks to Rene Brun, Fons Rademakers, Philipe Canal, Suzanne Panacek, Peter Malzacher, Frank Chlebana, Kuni Kondo, Kevin MacFarland, Pasha Murat, Larry Nodulman, Jim Patrick, Tony Vaiciulis, Margaret Votava, the CDF online group, and the Fermilab Computing Division Online and Database Systems department. We also would like to thank the community of ROOT users who via the roottalk mailing list provide answers and solutions to many problems.

[5] Engineering Change Order.

References

1 M. Votava, *et al*, ”Data Acquisition Systems at Fermilab”, proceedings of 11th IEEE NPSS Real Time Conference, June 14-18, 1999, Santa Fe.

2 R. Brun, F. Rademakers, `http://root.cern.ch`.

3 M. Shimojima, *et al*, ”Consumer-Server/Logger system for the CDF experiment”, proceedings of 11th IEEE NPSS Real Time Conference, June 14-18, 1999, Santa Fe.

4 I. Kravchenko, *et al*, ”Event Builder and Level 3 Trigger at the CDF Experiment”, proceedings of CHEP 2000, February 6-12, Padova, Italy.

5 `http://kcdf1.fnal.gov/~wenzel/consumer_new`.

6 E. Sexton-Kennedy, *et al*, ”The Physical Design of the CDF Simulation and Reconstruction Software”, proceedings of CHEP 2000, February 6-12, Padova, Italy.

7 P. Canal, *et al*, ”ROOT at Run II”, proceedings of CHEP 2000, February 6-12, Padova, Italy.

8 S. Rolli, *et al*, ”L1/L2 hardware Trigger Simulation at CDF”, abstract 131 (Poster session) of CHEP 2000, February 6-12, Padova, Italy.

9 `http://miscomp.fnal.gov/cdfdb/`

10 V. White, ”MISCOMP, an Information System Tool-Kit”, proceedings of CHEP 1997, April 7-11, Berlin, Germany.