PROCEEDINGS

Online Monitoring in the CDF II experiment

Wolfgang Wagner[*3†], **Tetsuo Arisawa**[4], **Koji Ikado**[4], **Kaori Maeshima**[1], **Hartmut Stadie**[3], **Greg Veramendi**[2], **Hans Wenzel**[1]

[1] *Fermi National Accelerator Laboratory, Batavia, U.S.A.*
[2] *Lawrence Berkeley National Laboratory, Berkeley, U.S.A.*
[3] *Universität Karlsruhe, Institut für Experimentelle Kernphysik, Karlsruhe, Germany*
[4] *Waseda University, Tokyo, Japan*

ABSTRACT: We describe the online monitoring system of the CDF II experiment designed to check the data quality of all subsystems in real-time. A subset of the events accepted by the highest trigger level is made available to 10 analysis programs which check the data quality on an event-by-event basis and produce diagnostic histograms. The monitoring results are distributed to the user via a client-server scheme. The display clients feature an intuitive GUI which allows to browse and request the available results. The monitoring package is coded in C++ and makes use of the ROOT analysis framework which offers histogramming methods, networking classes and graphics handling. We report about the design and the implementation of the monitoring system and discuss our experiences during the first months of operation.

The Collider Detector at Fermilab (CDF) is a general purpose detector operating at the Tevatron $p\overline{p}$ collider located in Batavia, Illinois. During Run I, from 1992 to 1996, the Tevatron operated at $\sqrt{s} = 1.8$ TeV and CDF collected data with an integrated luminosity of $\mathcal{L} = 110$ pb^{-1}. Numerous high quality physics results were obtained, among them the discovery of the top quark, the discovery of the B_c meson and first indications on CP violation in the system of neutral mesons.

From 1996 to 2001 the Tevatron collider was upgraded to provide an instantaneous luminosity of about $2 \cdot 10^{32}$ cm^{-2}s^{-1}, i.e. 20 times more than in Run I, at $\sqrt{s} = 2.0$ TeV. The larger data samples to be expected in Run II, lasting from 2001 to 2007, will allow to repeat measurements of Run I with higher precision and perform new ones which were out of reach before. In order to prepare for the improved conditions CDF was considerably upgraded in all subsystems. Among these upgrades are new monitoring programs to control the data quality which are described in this article.

---

[*]Speaker.

[†]wagner@ekp.physik.uni-karlsruhe.de

1. Goals and system requirements

Our main goal is to monitor the data quality of all subsystems of CDF online, while the data are being taken. The monitoring is realized by consistency checks on an event-by-event basis, by filling diagnostic histograms or time-charts, which show the time development of certain quantities, and by performing periodic statistical tests of the bin contents of those histograms. Furthermore, we want to report discovered errors in an automatic way to run control and thereby allow to request an automatic action, e.g. to reset or reboot faulty front-end crates or even halt, recover and restart the run. At the end of each run each monitoring program should give a final assessment of run quality. The judgments should be stored in a data base to give an overview and allow access by higher level assessment programs. Additionally, we want to use the monitoring programs offline, for example to check reconstructed data, to analyze test beam data or for primary checks on Monte Carlo data sets.

To accomplish the goals mentioned above our monitoring system needs to fulfill certain technical requirements: (1) avoid interference with the data acquisition; (2) select events according to the trigger types; (3) parallelize concurrent tasks, like monitoring different subdetectors or different quantities; (4) guarantee each monitoring process an adequate input rate of minimum 1 Hz; (5) minimize the interference between the monitoring process and the display process, which delivers and visualizes the monitoring results to the user; the communication with the user should not slow down the monitoring process considerably; (6) implement structures and interfaces which are common to all monitoring processes; we named this common basis for all monitoring programs, which are in CDF jargon called consumers, the **Consumer Framework** [2]; (7) offer access to the monitoring results from remote institutions; that is important since not all subsystem experts are permanently located at Fermilab. (8) an easy-to use and versatile graphical user interface (GUI); (9) stable operation for long time periods; (10) a large part of the Consumer Framework should be kept independent of the CDF framework; this allows us to develop a universal tool which can be ported easily to other applications.

2. Design of the Consumer Framework

The highest trigger level (Level 3) of the CDF II experiment is a software trigger implemented on a large PC farm. The available bandwidth for data logging is 20 Mbyte/s, the approximate size of an event is 250 kByte. Therefore, the expected trigger output rate is typically 75 Hz. The data are transfered from Level 3 to a process called Consumer Server Logger (CSL) [3] which sends the data to the computer center where they are written to tape and forwards copies of a subset of these events to the online monitoring programs. The total bandwidth available to the consumers is limited to 10 MByte/s. The Consumer Framework is written in C++ and makes heavy use of the ROOT [4] package.

Monitoring programs: The CSL serves in total 10 monitoring programs: YMon (occupancies), XMon (trigger rates), LumMon (luminosity), TrigMon (trigger bits, trigger decisions), Stage0 (drift chamber calibrations), BeamMon (beam position), ObjectMon
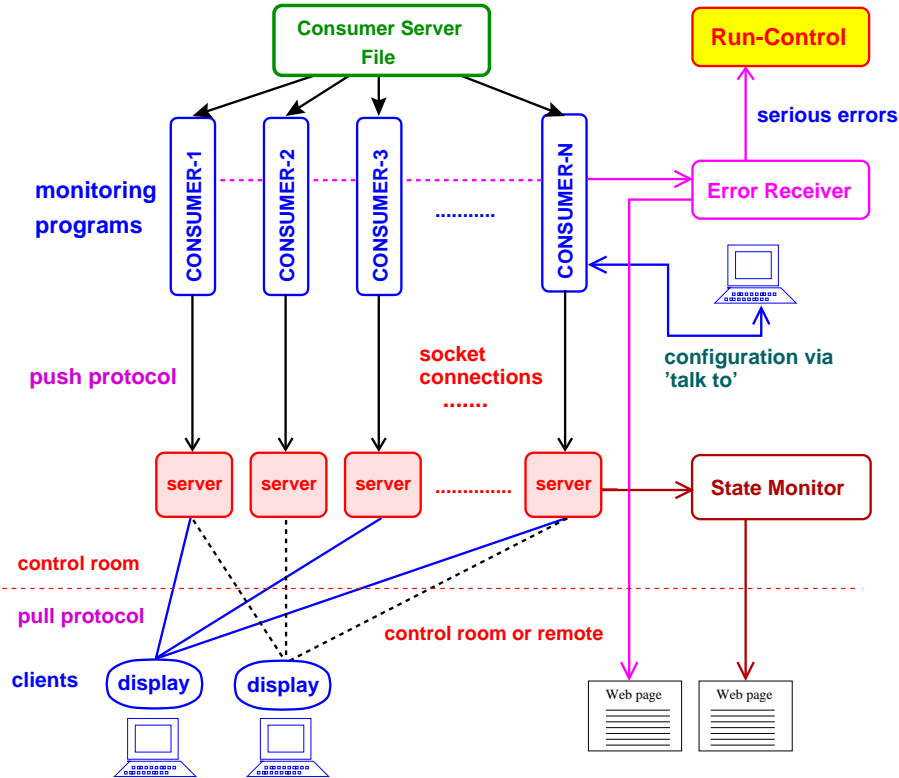
Figure 1: Design of the Consumer Framework.

(reconstructed objects like jets, electron and photon candidates, muon candidates and tracks), SiliMon (silicon hit maps, silicon tracks), SvxMon (chip-wise silicon information: raw ADC counts, pedestals, signal-to-noise) and L3RegionalMon (silicon Level 3 regional tracking). The consumers can be configured and parameters can be customized via a steering file (in CDF jargon called a talk-to) which is read in at start-up.

The monitoring programs are written by the subsystem experts. However, the Consumer Framework provides common features: the embedding in the CDF offline software (AC++), interfaces for initialization and event processing, the transport of results to the users, a common display program and error logging. This allows the subsystem experts to concentrate on their monitoring tasks, insures a large coherence between the individual consumers and eases additions and maintainability. The availability of such a common basis for all online monitors has to be viewed as a major improvement as compared to Run I of CDF.

The design and the components of the Consumer Framework are shown in Fig. 1. One important design choice was to implement a distributed system with many consumers running in parallel on different machines rather than one huge central process. The reason for this is that consumer jobs are CPU bound rather than limited by the available input rate.

Display Servers: Another major design decision was to separate the publishing of monitoring results from the consumer jobs, in order to increase stability and smoothness

of operation. Thus, we designed a client-server model to transport the monitoring results to the user. Each consumer has its own server program. The display servers receive all monitoring results from the consumers via a push protocol. The update occurs by default every 10 events. The update frequency can be customized via a steering file. To increase the performance we do not instantiate objects inside the display servers but rather keep them in a buffer type format. The display servers are running on the same machine as the affiliated consumers. The bandwidth of the internal socket communication is about 10 Mbyte/s. Since each consumer and its display server are running on the same machine, we can start the server process from the consumer in an automated way using fork/exec. When the consumer exists, the server is stopped automatically.

Display clients can connect to several servers at the same time and request specific monitoring results via a pull protocol. After connection to a server the display clients request first a list of all objects which are available by that server. This list is presented to the user in a list tree of the GUI. The user has access to the available objects via atomized requests: The user chooses from this list an object he wants to view. The request is sent to the server and the object is returned. Inter-process communication between consumer and display server on one hand and display server and display client on the other hand is realized by socket connections provided by the ROOT package (TSocket).

The display clients offer an auto-update mode to users, such that a selected object will be automatically updated with a specified update frequency, the default being 5 seconds. Another important feature of the display is the slide show option. The consumer writers can flag certain histograms or canvases as particularly important. If the user switches the slide show on, the display window will pop up and the flagged objects will consecutively be shown on this window. The objects will be automatically updated and the display will cycle over them.

The standard input method for the display is via sockets as discussed above. However, it is also possible to use the GUI to view ROOT files written by the monitoring programs.

Error receiver: Above all it is the purpose of the monitoring to find errors in the data or produce warnings about faulty distributions. Thus, it is important to get an overview about the error messages produced by the monitoring processes. For this purpose we collect error messages of all consumers by a central process: the error receiver. The last ten messages of each consumer are displayed on a web page.

Serious error messages will be forwarded to run control to take necessary actions. These messages have to be registered in advance to acquire a specific error key, which allows the error receiver to identify them. The registered error messages are forwarded to run control and can prompt an automated action, for example to halt, recover and restart a run.

State Monitor: This is a watchdog process which monitors the status of each consumer and displays it on a web page. The state monitor provides information on how many events each consumer has processed, on which machine the consumer and the affiliated server are running and the port number on which the server offers its service to the display clients.

3. Performance

The online monitoring programs are permanently running in the CDF control room since April 2001. The 10 processes are distributed over 6 Linux PCs (4 1.4 GHz and 2 2 × 800 MHz). The PCs serve 10 screens where the display clients are showing the results. To start the consumers and displays in standard operation the shift crew runs two scripts which start the necessary processes on the assigned machines.

To commission our monitoring system we have made several performance tests. Some findings are reported here: We checked how the communication of a consumer with its display server influences its monitoring performance by running the consumer with the highest number of histograms (YMon) with and without server connected. With YMon running alone on one machine we got an event rate of 5.8 Hz if the server was off and 4.7 Hz if the server was on. We consider this decrease in processing rate due to the export of monitoring results to be acceptable.

Another test dealt with the resources used up by the display client if it is running on the same machine. It turned out that the update time when being in auto-update mode is a crucial quantity. If set to a reasonable value of 5 seconds the consumer gets 80% of the CPU, the display uses 4%, the X server 6% and the display server 2%. If the update time is set to a far too low value, e.g. 0.3 seconds the share left for the consumer goes down to 36%, the X server uses 34% and the display 21%.

With the full system under operation we find that all consumers reach an event processing rate of approximately 2 Hz which is within specifications.

4. Conclusions

We have designed and implemented a common framework (Consumer Framework) for all 10 CDF online monitoring programs. The package includes common base classes, a display server to publish monitoring results and a versatile and user-friendly display GUI. The monitoring programs are in continuous operation since April 2001. Monitoring results are accessible from remote sites.

References

[1] F. Abe et al., *The CDF detector: an overview, Nucl. Instrum. Meth.* **A 271** (1988) 387–403.

[2] H. Wenzel et al., *Online Monitoring and Module Maintenance for CDF in the Upcoming Fermilab Tevatron Run II*, CHEP 2000, Padova, Italy, February 2000;
T. Arisawa et al., *CDF Run II Run Control and Online Monitor*, CHEP 2001, Beijing, China, September 2001.

[3] M. Shimojima et al., *Consumer-Server/Logger system for the CDF experiment*, 11th IEEE NPSS Real Time Conference, June 1999, Santa Fe, U.S.A.;
B. Kilminster et al., *Design and Specifications for the CDF Run II Consumer-Server/Logger*, CDF internal note CDF/DOC/ONLINE/PUBLIC/4794, 1998.

[4] R. Brun, F. Rademakers et al., *http://root.cern.ch*.