CDF Event Monitoring System and Operation

Kaori Maeshima

(For the CDF Collaboration)

***Abstract* -The foundation of the CDF Run II online event monitoring framework was implemented well before the start of the physics runs, allowing development of a coherent monitoring software across all the subsystems, consequently making maintenance and operation simple and efficient. Only one shift person is needed to monitor the entire CDF detector, including the trigger system. High data quality is assured in real time and well defined monitoring results are propagated coherently to offline data sets used for physics analysis. We describe the CDF Run II online event monitoring system and operation, including the remote monitoring shift operation started in November 2006.**

***Index Terms*—CDF, Online Monitoring, Data Quality Monitoring, Remote Operations, Run II**

I. Introduction

The CDF Run II detector [1] started taking data in mid-year 2001. Since that date the Tevatron has continued to improve by delivering higher luminosity; CDF has recorded over 2 fb^{-1} of data for a wide range of physics analysis to date (April, 2007). There are many people and components involved in order to take high quality data with high efficiency. The CDF online monitoring plays one of the important roles to achieve this goal. The online event monitoring processes have been running continuously since the beginning of 2001 and their results are routinely checked in real time by one of the three people on shift, called Consumer Operator (CO), for all the sub-systems. Monitored results are also checked automatically. The other two on shift are: ACE (main DAQ person) and SciCo (scientific coordinator, a shift leader).

The CDF Run II online event monitoring system was designed in 1998. The main design goal is to deliver a coherent integrated online monitoring system across all the sub-detectors (including trigger system) to ensure high quality data to be taken with high efficiency. It is designed such that monitored results can be easily viewed by the shift crew in the control room as well as the experts located anywhere in offsite to ensure the prompt feedback in case of problems. This feature also allowed an almost transparent transition to later remote CO operation. Since the design stage in 1998, we have been going through different phases of the monitoring operation as the CDF run II progresses:

1. Implementation phase: core monitoring framework and the first order individual monitoring programs
2. Detector commissioning phase: (for the initial stage commissioning as well as for the various upgrades throughout the Run II period): we use the same monitoring system to debug and commission the detector sub-systems by the experts.
3. Steady state operation: use online monitor around the clock by the shift crew; also used by the sub-system experts for maintenance and debugging.
4. Throughout the period of 2 and 3, improve and refine the monitoring programs and mode of operation while maintaining the 24/7 working version of online monitor system in the control room. It should be noted, here, that though we have made many operational improvements, the initial design and implementation of the online monitoring framework and the interfaces to individual monitoring programs remain unchanged.
5. Establish official remote online monitoring shift in 2006.

We describe here, the design and implementation process of the CDF run II online monitoring framework and monitors (section II), maintenance and performance of the monitoring system (section III), issues related to the monitoring shift operation (section IV), and finally, the remote monitoring shift operation officially started in 2006 (section V).

II. Design and Implementation of Consumer Framework and Consumers

A. Requirements and Placement of the Monitors

In order to achieve the goal of efficient and good quality data taking, followings are the essential technical requirements for the design:

1. Avoid interference with the data acquisition
2. A monitoring process to be able to receive selected events according to the trigger types or streams
3. Modular design such that it is possible to run different monitoring modules in different processes in parallel as well as being able to run as a combined process.
4. Guarantee each monitoring process an adequate input event rate.
5. Implement structures and interfaces which are common to all monitoring processes; we named this common basis for all monitoring programs,

The author, Kaori Maeshima is with the Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-3917; e-mail: maeshima@fnal.gov).

which are called Consumers in CDF, the Consumer Framework.

6. Minimize the dependency between the monitoring process and the display process. This is to: a) allow a single display process to be able to connect to display results from any monitor; b) any user activities while using a display process should not slow down or interfere the monitoring process; c) multiple users can run multiple display processes to view the same monitored results
7. In order to make implementation and development of each consumer monitor program easier, so that the whole system will mature as a coherent package for users: provide to the monitor code writers the base monitoring program with all the common features, a simple script for getting started, examples, documentation and assistance. The monitoring code should be easy to debug and validate on online and offline without changes in code (only the run-time parameters change). Also to provide 24/7 event server independent of the DAQ status.
8. Provide online monitor release.
9. Provide a common structure in the display GUI in the form of separate folders with different characteristics (slide show, warning, experts, etc.)
10. Offer remote access to view the monitoring results.
11. Stable operation for long time periods
12. A large part of the Consumer Framework should be kept independent of the CDF analysis framework – this allows us to develop a universal tool which can be ported easily to other applications.

The CDF online event flow (a simplified version) is shown in Fig. 1. The highest trigger level (Level 3) is a software trigger using a large PC farm. The events passed Level 3 trigger are sent to a process called Consumer Server Logger (CSL) [3]. Information about the event size, rates, and number of triggers is indicated in Fig. 1. The CSL has two main functionalities. Its first principle function is to log all the events which passed Level 3 trigger for the physics analysis. The other is to serve events to the online event consumers. The box where the online event monitor system resides is indicated at the right hand bottom. Consumers run on dedicated pc nodes which are different from the Level 3 clusters.

CSL assures that having the functionality of data serving does not cause significant dead time to data acquisition, while delivering sufficient rate to consumers. Any problems

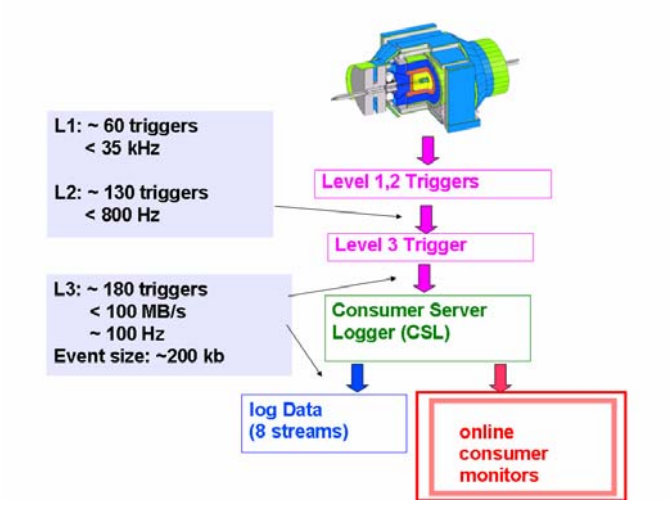

Fig. 1. Online event flow and online monitors

(crashes, memory leak, etc.,) associated with individual monitor programs do not interfere the data taking process. The CSL saves the begin-run record in each run; whenever a consumer connects to the CSL for the first time, the CLS passes the begin-run record the first thing to allow the consumer to initialize for the run configuration. At the time of the connection, a consumer can request to receive a choice of triggers from CSL. It is designed such that a consumer which is requesting a very rare trigger can receive the event even when there are other consumers requesting high rate triggers. There are certain classes of events which are needed for debugging purpose but those events may not pass the physics triggers in Level 3. For these cases, we have created new debug triggers which can be fed to appropriate monitor processes. The total event input rate to the consumers is ~ 20 Hz. This rate is sufficient to monitor almost all the system sufficiently, with the exception of SVTSPYMon which is a monitor for the displaced vertex trigger system (SVT) in the Level 2 trigger system. In this case the event source is not from CSL, but from the front-end crate level. Since Consumer Framework is designed essentially independent from the CDF event analysis framework (requirement 12), SVTSPYMon is able to use the large part of Consumer Framework. Consequently, users can use and view SVTSPYMon results just in the same way as all the other consumer monitor results. Above described is satisfying the requirement 1 through 4.

B. Consumer Framework Design

The core part of Consumer Framework is written in C++ and makes heavy use of the ROOT [2] package. The design and the components of the Consumer Framework are shown in Fig.2. Eleven monitoring programs (10 consumer processes including the event display program plus SVTSPYMon) are implemented by sub-system experts and running continuously based on the common framework. The major components of the framework are: 1) base classes

providing common functionality like creating canvases of histograms and storing log files, handling errors and alarms, 2) display server, which is used to distribute the monitoring results via a client-server mechanism, 3) display clients that feature an intuitive GUI which allows to browse and request the available results, 4) Consumer State Manager, 5) Error handling and forwarding. After the steady shift operation started, above core functionalities continued to be used, however, we have made some additional improvements which are described in sections IV and V.

B.1 Consumers

One of the important design choices was to implement a distributed system with numbers of different consumers running in parallel on different machines rather than one large combined monitor executable running on a central process or running multiple copies of the same combined executable in different nodes. This distributed design gives several advantages: 1) Consumers are developed by many sub-system experts. Less dependency between different types of monitors makes the process of development and debugging much easier. 2) One monitor's run-time problem (crashes, memory leak, etc.) does not affect the other monitor. These types of problems occurred quite frequently in the early run II running period when monitor programs were being worked on intensively, and we were very glad we designed the system in this way. 3) A different monitor can get a choice of different triggers. 4) Some consumer jobs are CPU bounded rather than limited by the available input rate. We can adjust and optimize the input rates to different monitors.

The CSL serves in total 9 monitoring programs and event display: YMon (occupancies for all the sub detectors except Silicon), XMon (trigger rates), LumMon (luminosity), TrigMon (trigger logic), Stage0 (drift chamber calibrations), BeamMon (beam position), ObjectMon (reconstructed objects like jets, electron and photon candidates, muon candidates and tracks), SVXMon (chip-wise silicon information: raw ADC counts, pedestals, signal-to-noise), DAQMon (DAQ), DAQErrorMon (examine events with DAQ errors, and L1 auto accept events). 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.

Consumers use the CDF event analysis framework called AC++. Each AC++ job consists of different modules. An input module receives the events from the CSL. The ConsumerFrameworkModule contains a list of all monitor objects that analyze the data. The monitor objects inherit from the BaseMonitor class and are written by the experts of the detector subsystem. The BaseMonitor supplies functions, e.g. to register Root objects, remove them, report errors, create a custom Root canvas for the histograms and save all

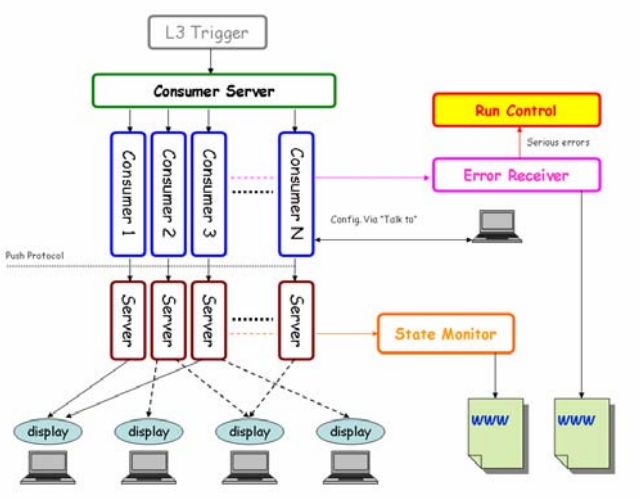

Fig. 2 Design of the Consumer Framework

Root objects into a file. The information about each registered object is stored in the ConsumerInfo object. This object and all registered Root objects are sent to the display server by the ConsumerFrameworkModule. The code for the sending of the objects is encapsulated in its own class. This allows extraction of all the classes of the client-server framework from the package and to use them outside the CDF software environment as a generic tool. At the end of each run the Root objects of each monitor are saved into a file and the results of the data checking are put into a database to allow the assessment of the run quality by combining all consumer results.

B.2 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 (system requirement no. 6). 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 raw buffer type format. The display servers are running on the same machine as the 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 and exec. When the consumer exits, the server is stopped automatically.

B.3 Display Clients

The display clients [4] connect to a server and request specific monitoring results via a pull protocol. Multi-point to

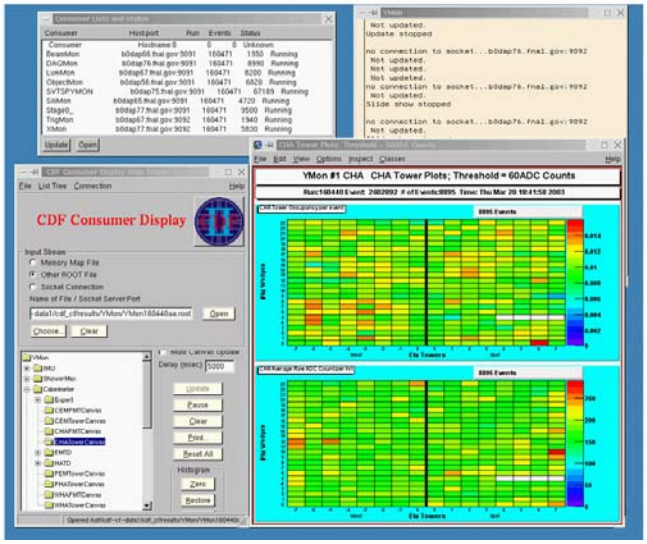

Fig. 3. a typical view of the computer screen when operating a display client (a YMon plot)

multi-point connections are available, i.e. several display clients can connect to one server and one display client can also connect to several display servers at the same time, as shown in Fig. 2. After connection to a server, the display clients first requests a list of all objects which are available by that server (the ConsumerInfo object). 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 selected objects 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. Aside from the fact that "slide show" feature is useful in the control room, it turns out that organizing the each monitor results to be sorted into "essential" (slide show folder), "expert", and "warning" folders, etc. causes an very important effect operationally. The organized monitor output across all the sub detector gives the coherence in viewing the results even when we use the alternative web based GUI.

B.4 Error handling and forwarding

Above all it is the purpose of the monitoring to find errors in the data and 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 consumer error receiver. The last ten messages of each consumer are displayed on a web page.

Serious error messages are forwarded to the main DAQ error handler process for the shift crew 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 the handler and can prompt an automated action, for example to halt, recover and resume data taking.

B.5 State Monitor

The State Monitor is a watchdog process which displays the status summary of all the consumer monitors 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. The status web page updates itself automatically in every 2 minutes, with an option of manual update at any time. Those monitors which have had a status change (for example, change in number of events processed, or change in run numbers) since the last update are indicated as "green", otherwise as "red" to show the active/inactive states of monitors on the consumer status web page. The display client reads the information about each monitor from this status page in order to connect to the appropriate port for the desired monitor process.

C. Implementation

The monitoring framework which deals with the common functionalities was implemented before any of the consumer monitor developers stared coding. In addition to the framework code, the framework team also provided various tools for the monitor code writers to make the job easier and also to ensure the code developed coherently across all the sub detector systems. Such tools include: documentation and instructions with examples, one line "getting-started" script which creates and builds an example monitoring program with all the hooks necessary to run in the online environment, coherent code management scheme which allows building an online monitoring release and easy ways to test the code online and offline. A consumer monitoring program can be tested on online or offline without a change in the binary. One only needs to modify the run time parameters in a trivial manner. In order to make online testing procedure of the monitors easier, we have setup a special CSL which reads raw data files and serves to consumers constantly in low rate (~1 Hz). The CDF DAQ system is setup so that it can run in different partitions. The main data taking DAQ typically runs in partition 0. This playback mode CSL (which is serving events to consumers constantly) always runs in partition 14.

A partition number is one of the parameter for a consumer to connect to an appropriate CSL process. Having the partition 14 allows us to test online monitors on the online environment at any time even when the data taking is not in progress. The existence of the partition 14 was especially useful in the early stage when the accelerator and CDF data taking performance was not as consistent, while at the same time a large amount of monitor code development was taking place.

III. MAINTENANCE AND PERFORMANCE OF THE MONITORING SYSTEM

Two cvs packages called ConsumerFramework and Consumer contain all the code directly related to the online event monitors. We build an online monitor release (cdfsoftb0 release) based on the offline release with patches necessary to run with the most updated online monitoring code. There are many other packages besides the Consumer and ConsumerFramework packages which are needed for inclusion from the newest version to build cdfsoftb0; for example, trigger emulation packages frequently change. Having an independent (independent from the Level 3 trigger release, for example) online monitor release is quite important, since the nature of the online monitoring task is to be versatile, while also needing to be organized (must be running all the time). Many unforeseen problems occur as we run the detector operation, and monitoring programs must be able to respond and adjust to the new finding promptly.

CPU, disk space, and memory usage of the nodes where monitors run must watched in an automated way, especially the disk space. We run a cron job to clean up, move and archive monitored result files and log files. The disk space where the result files are temporary stored is big enough to keep all the result files from the most recent ~3 months running.

IV. MONITORING SHIFT OPERATION

The original design and implementation of the ConsumerFramework and Consumers provided the fundamental functionalities for the online monitoring, and this framework works very well. However, as we gain more experiences using the monitoring system with the 24/7 shift operation, we identified some areas where improvement is required. None of the improvement is to alter the original design, but they are additions.

In case of the CDF shift operation, one person from the generic pool of CDF physicists, at a given time, is on the Consumer Operator (CO) shift. This person looks after all

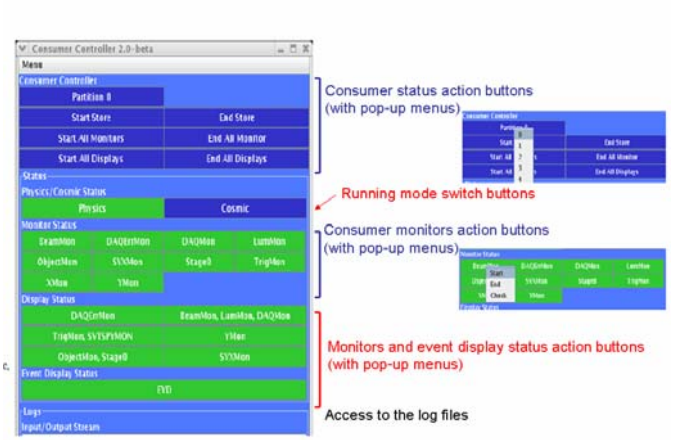

Fig. 4 ConCon (Consumer Control) GUI.

the detector and data taking components using this online monitoring tool (i.e., monitoring shifts are not done by each sub-system expert). Soon we reached the point that we needed the minimum but sufficient, official and clear instructions for CO: which plots to look at, how often, how one can decide the sub detector is good or bad, a clear set of instructions of what to do when a monitored result is found to be problematic, and provide with a method to propagate the good/bad decisions. The first non trivial part of this job is to converge to the clear set of official instruction, since it involves many people and systems and needed some time and experience for experts to learn what is good/bad and what criteria to designate.

The set of clear instructions for a CO materialized as a single interactive summary web page called “CO Check List”, from where the CO would always starts. From this page, the CO is provided the clear set of instructions, views of relevant histograms (with a reference plots side-by-side) via web links, and also buttons to click to store the checked results (good/bad, comments, etc..) which are stored in the run summary data base. Two things to note here are: 1) the detector conditions changes from time to time. Hot channels and dead channels come and go. Many of those problems are foreseen and dealt with in software automatically. But not all cases can be coded all the time. As for the detector operation and shift crew, it is important to be able to differentiate “known problems” from “new problems”. The first thing what CO person does before starting the actual detector check-list is to examine the web page called “Fix List” which is linked from the top of the “CO Check List” page. The “Fix List” web page keeps the up-to-date list of the “know problems”. 2) In the ConsumerFramework BaseMonitor class, we have added a functionality to save all the monitoring result root files in some time interval (set to 5 minutes) during a run. These root files are called “temp_root files”. They get overwritten in every 5 minutes. This is all done in the ConsumerFramework level, so that no Consumer code needs

to be modified in order to do this. These "temp_root files" are the ones linked from "CO Check List" so that a CO can check everything via web pages. In the early time when "CO Check List" web page was created, only the minimum essential plots could be viewed via a web page. But soon it was implemented such that we can see any monitored results using a web page in a similar way as the display client. This is especially useful for the experts who are paged and located offsite. Accessing information via a web page is typically much faster and easier than running a special application program locally to view the results.

V. REMOTE MONITORING SHIFT

Relatively recently in 2006, the CDF operation group started considering a possibility of remote CO shifts. Motivations are the following:

- Avoid oversea missions for CO shifts to save money to allow more people to be onsite for other activities.
- Reduce the number of people on shift during the night at FNAL.
- Acquire know-how for future applications of remote shifts (LHC, ILC…)
- Collaborators from Pisa and Tsukuba University were chosen as pilot institutions for the CDF Remote CO shift project.
- The project was developed in cooperation with CMS ROC group [5] at FNAL using the multipurpose remote operation room setup in the Wilson Hall 11th floor at FNAL.

The goal for the remote CO shift is: there should be no compromise in data quality nor efficiency — the remote CO person should be able to do all the things local CO person is supposed to be doing. Due to the fact that the procedure of the CO shift operation has been well established and basically all one needs to view and act on are web based (hence remotely accessible), the transition to making the remote shift was relatively painless and transparent. In addition to performing online monitoring during physics runs, a CO has another duty to check all the detector calibration results between the stores. As in the online monitoring case, the procedure of checking the calibrations has been well established and all tools (which CO uses) are web based.

There are a few things specifically for a remote shift which we needed to setup and test:

- Establish contact people for the remote operation (local and remote) and always have a back-up solution (local back-up CO) in case of network outage, etc. Establish and document any specific issues to do with remote CO (e.g. overlap shift information transfer procedure).

Fig. 3 Pictures of the CDF control room with online monitor displays (top) and Pisa CDF remote CO shift room (bottom)

- Setup and test reliable human communication system. We use Polycom with point-to-point connection which has been found to be very stable. International telephone call is always a backup solution.
- A few procedures which a local CO was doing must be revised due to the security issues. One of which was a procedure to start and stop all the monitoring processes. This was done by typing in a simple script from an online account before. A new consumer control GUI (ConCon) was created, and by having the ConCon GUI window pop up at the remote site, the remote CO can start/stop the processes without a problem. The ConCon GUI also improved the local CO operation. Fig. 4 shows the picture of the GUI.

All the above items were setup and tested after the summer of 2006. After several successful parallel CO shifts (simultaneous remote and local COs), the official remote shift operation was started in November 2006 from Pisa. Since then, typically in the frequency of one week per month, owl shifts (midnight to 8 am local time) have been taken from

Pisa. So far, after 6 months and 8 weeks of remote CO shifts, we have experienced no unexpected problems (there was one scheduled ~15 minutes network outage.) A picture of the CDF control room where online monitoring displays are (top), and a picture of the Pisa remote CO room (bottom) are shown in Fig. 5. We are currently testing the remote shift setup at Tsukuba University and expect to come officially online very soon.

The Screen Snapshot Service (SSS) [6] is a software package which puts images on a screen onto a web page, allowing remote viewing of the screen display. The SSS comprises three components: producer, server and client. The producer, implemented as a Java application, periodically captures screen images of a specified computer desktop and sends the snapshots to a server. After receiving the images from the producer, the server converts images to PNG format, and then serves them to clients. The server runs under Tomcat. The client periodically fetches snapshots from the server and displays snapshots in a web browser. The SSS was initially developed for the CMS remote operation in mind. After the initial development, multiple nodes are running SSS in the CDF control room making some of the information which was not previously available viewable from remote sites, such as the main run control state, etc. The SSS is strictly a viewer and does not allow any controls from remote location. Remote operation people at Pisa are now considering increasing the number of display screens now that SSS can serve useful screen images.

VI. Conclusion

Since the integrated online monitoring scheme was proposed in 1998, it went through various phases but the monitoring programs have been always running since early 2001 and they are essential and integral parts of the CDF Run II data taking operation. Over the course of years, we have developed significant amount of web based tools for the shift and detector operation, which enabled the recent transition to remote CO shift to be simple. The remote monitoring shifts were found to be just as effective as the local shift from the recent experience. It is important to have: a good coherent foundation for the monitoring framework, coherent development of the monitoring programs and operation, collaborative effort to converge on clear ways of determining the essential problems and instructions for shift crew, and to make the information easily accessible remotely for experts and remote monitoring shifts.

Acknowledgment

The author would like to thank T. Arisawa, W. Badgett, K. Biery, D. Fabiani, A.D. Hahn, D. Hirschbuehl, K. Ikado, T. Kubo Y. Kusakabe, J. Naganoma, K. Nakamura, C. Plager, E. Schmidt, F. Scuri H. Stadie, R. Tsuchiya, G. Veramendi, T. Vaiciulis, W. Wagner, H. Wenzel, M. Worcester, K. Yorita and the support from many CDF colleagues, especially from the CDF operation group, and the CMS ROC group at FNAL.

References

[1] D. Acosta, *et.al.,* Phys. Rev. **D71**, 032001 (2005)
[2] R. Brun, *et.a*l, http://root.cern.ch
[3] M. Shimojima *et al.*, *Consumer-Server/Loger system for the CDF experiment, 11*th IEEE NPSS Real Time Conference, June 1999, Santa Fe, U.S.A.
[4] Original design of display client comes from LHCb experiment
[5] http://uscms.org/roc
[6] http://home.fnal.gov/~beiry/ScreenSnapshotService