GlueX-doc-1721-v0

Plan for Monitoring Histogram System in Hall-D

David Lawrence

May 19, 2011

Abstract

This document supplies the initial plan for a system to allow monitoring of data quality by shift takers in Hall-D. This will be done via a select group of histograms that shift takers may monitor in real time. The histograms will also be archived for later inspection. This document applies to 12GeV project line 1532035b "Plan Monitoring Histograms".

Contents

1 Requirements

A requirement of data taking in Hall-D will be the monitoring of the data quality as it is acquired from the DAQ system. This will help ensure the integrity and quality of the data on tape by identifying problems quickly and limiting the loss of resources spent on data that will be discarded. The primary means of monitoring the data will be through histograms that are continuously filled as the data is acquired.

The data rates in Hall-D[1] will be large enough that a small farm of monitoring computers will be used in order to inspect a reasonable fraction of the data as it is acquired. Each node on the monitoring farm will process a portion of the events. Histograms will be made for each detector system as well as some higher level analysis histograms that combine information from multiple detectors. Multiple farm nodes will produce similar histograms (using independent event samples) that will be combined into a single histogram for inspection by shift takers and/or archiving for inspection offline.

For simple histograms such as hit occupancy, a large fraction ($\geq 50\%$) of the data should be handled by the system. For more complicated values requiring full event reconstruction, a much smaller fraction is needed ($\sim 1\%$).

Because multiple computer nodes will be used and the histograms combined, the system will require transfer of histogram contents over the network. It will also need to couple into an ET system that can access events from the DAQ data stream. A user interface for shift workers in the counting house will be required as well as a mechanism for saving the full histograms(not just image files) to disk for later analysis. Finally, the bandwidth and CPU resources required to implement the system will need to be modest enough that the monitoring system does not disrupt the other operations in the counting house associated with running the experiment.

In order to allow outside users the ability to monitor the online operations, a separate program will be needed that can act as the interaction point. This can then limit the network activity going into the counting house, lowering the risk of disruption of the online systems due to heavy network traffic. This program can then also server as an aggregator for multiple histogram sources which should also help conserve bandwidth.

Currently, the exact hardware has not yet been purchased that will form the network in Hall-D so the available bandwidth is not clear. The lower limit, however, will not be below 1GB ethernet speeds (100MB/s). Designing the system such that it does not require more than 10MB/s into or out of a single node ensures no more than 10% of the available bandwidth will be used for monitoring histograms.

A summary of the requirements for the monitoring histogram system is given in table 1.

[1] Hall-D data rates will be about 3GB/s into the L3 farm for high luminosity running and about 300MB/s for low luminosity running.

Table 1: Requirements for monitoring histogram system

- Multiple histogram producers per node
- Multiple nodes (up to 50) supported (network coupled)
- Automated summing of histograms from multiple sources
- GUI with histogram display and system navigation
- Support for multiple, simultaneous GUI instances
- Ability to archive online histograms for later analysis
- Gateway/repeater/aggregator program
- Coupling to ET system
- Discovery ability for histogram producers
- Ability to accommodate nodes dying and emerging at any time
- Archiver program for writing out monitoring histograms to file(s)
- $\geq 50\%$ of DAQ events processed for occupancy histos
- $>\approx 1\%$ of DAQ events processed with full tracking and calorimetry
- $\leq$10MB/s single node bandwidth in counting house

2 ROOT and RootSpy

The most commonly used histogramming package in use in HENP today is ROOT[2]. Given the nearly ubiquitous use of ROOT and the large amount of cumulative experience with it within the JLab community, it makes a natural choice to base the online monitoring framework on. It is with this motivation that the RootSpy project[3] was started. RootSpy was designed to make it easy for existing ROOT-based histogram producer programs to publish histograms over the network as they are being filled. Proof of principle tests of the primary features and functionality of the RootSpy framework have been successfully performed and are described later in this document.

RootSpy works, in part, by taking advantage of a design choice built into ROOT. Specifically, that ROOT makes pointers to histogram and tree objects available through global memory. A thread running within a ROOT-enabled program resides in the same memory space and therefore has access to the list of histograms/trees that are currently defined as well as pointers to those objects which can be used to access their contents. A separate thread is used to listen and satisfy network requests without disturbing the main portion of the histogram producer program. Furthermore, the system allows the author of the histogram producer to remain ignorant of the presence of RootSpy placing less onus on them to write software that couples into the online systems.

RootSpy currently includes a JANA[4] compatible plugin that allows an existing JANA-based program to publish all of its histograms by simply attaching the plugin at program startup. The plugin uses cMsg and facilities within ROOT to serialize

[2]ROOT website: http://root.cern.ch
[3]RootSpy website: http://www.jlab.org/RootSpy
[4]JANA website: http://www.jlab.org/JANA

and deserialize objects. This means that the compiled program and any histogram producing JANA plugins need not know of the existence of RootSpy when they are compiled for their histograms to be accessed via the RootSpy framework. This is the nature of Root*Spy*.

3 Status of RootSpy

The RootSpy project was started in August 2009 as part of the JANA framework. It was moved to a separate project shortly thereafter. Features have been gradually added utilizing student labor. A functioning GUI exists (though not necessarily in its final form). A screenshot of it can be seen in figure 1 A JANA-compatible plugin has been written allowing any existing JANA program to include RootSpy.

At this point, work has begun on adding a feature to interact with trees, but that will require significantly more effort to complete. Other features such as configuration files or automatic refit-on-update have not yet been implemented. The project as it stands is usable in an offline environment (though it is not used regularly at the moment). A list of features that do not yet exist is given in table 2.

One of the primary requirements of RootSpy is the ability to sum histograms produced by multiple producers and display the summed histogram to the user. As a diagnostic feature, the user may also need to view a histogram from a single server. This functionality exists in the existing RootSPy prototype in multiple forms. Figure 2 shows dialogs displaying *View By Server* mode and *View By Object* mode. In *View By Server* mode, histograms are cycled through for display one server at a time, without summing. For normal operation, one will use *View By Object* mode which will sum similar histograms from all producers prior to display. Figure 4 shows an alternate of viewing individual producer histograms. This is a separate window that can be brought up to display all histograms at once. The user can choose to display them overlayed, or in a grid.

3.1 Proof of principle tests

A full proof of principle test of the prototype RootSpy system was performed on April 16th, 2010. This test employed 14 computer nodes: 8 from the JLab DAQ group farm, 4 interactive scientific computing farm nodes, and 2 additional machines in the office serving as the control center for the test. For the test, histograms were produced using a generated data set. A RootSpy GUI collected and summed the histograms from all 14 sources and displayed them. The system functioned as expected and no issues were encountered that might suggest the framework to be unsuitable for the online monitoring system. Figure 5 shows a screen shot of the cMsg monitor program (not part of RootSpy) taken during the test.

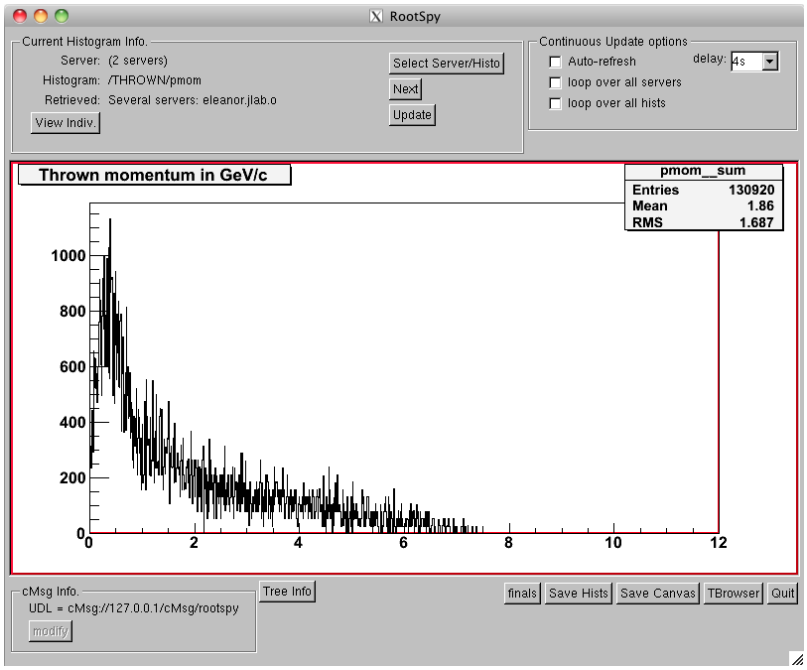

Figure 1: RootSpy Graphical User Interface (GUI) for the prototype version of RootSpy.

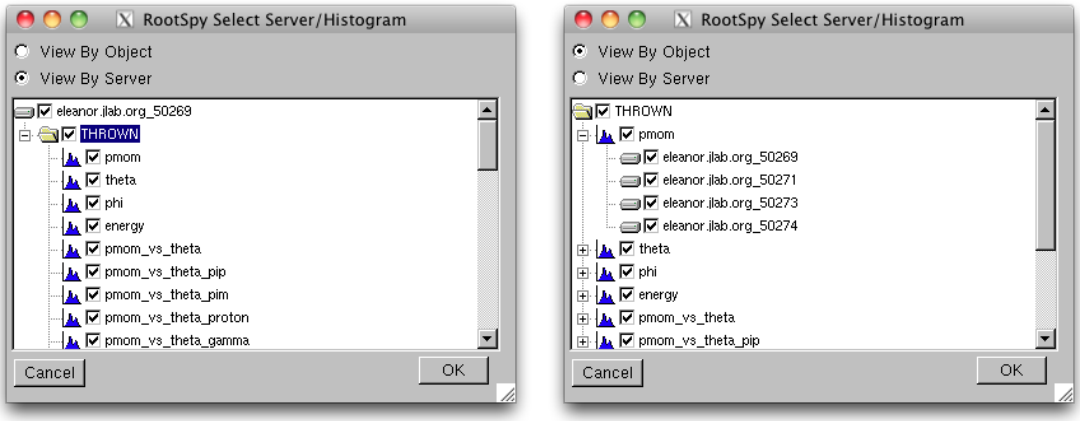

Figure 2: Dialog for selecting which histograms to display in both the *View By Server* mode (left) and the *View By Object* mode (right). See text for details.

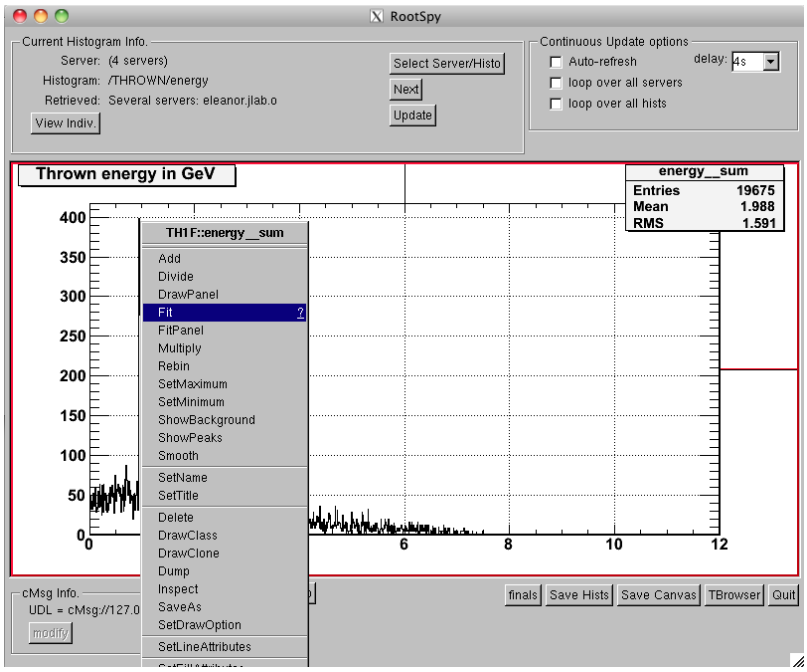

Figure 3: Right-clicking on the histogram displayed in the RootSpy GUI brings up the same menu options when running a ROOT interactive session. This is supplied automatically by ROOT requiring no additional programming on our part and presenting users with a familiar interface.

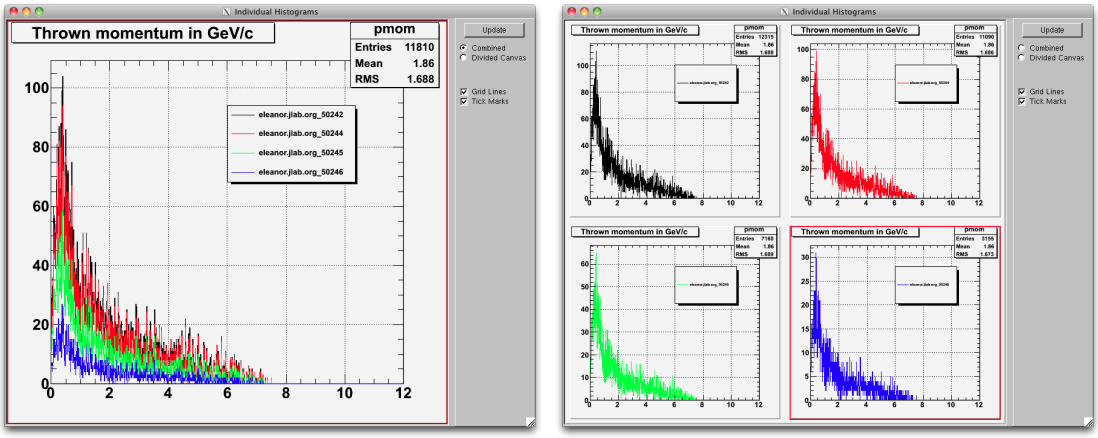

Figure 4: The window for viewing histograms from individual producers. Two modes of display are available: Overlay (or Combined) is shown on the left while Divided canvas is shown on the right.

4 RootSpy Development Plan

Development of RootSpy will proceed as part of the 12GeV Project[5]. Work that is required prior to deployment for Hall-D commissioning is:

4.1 Testing

Continued testing during development will be done to identify problems early and to gain experience with the system. This will include future beam and DAQ tests. In addition, another large scale test will be performed using multiple nodes[6] (∼20). This should be done after several of the features in table 2 have been implemented so that they may be tested. The time frame would likely be in the winter of 2012-2013.

[5]Development will continue under activity ID 1532080b: *Write Monitoring Histograms*.

[6]The test nodes should include multiple cores as well so the multi-threaded aspects of the monitoring can be tested in conjunction with the other aspects of the system

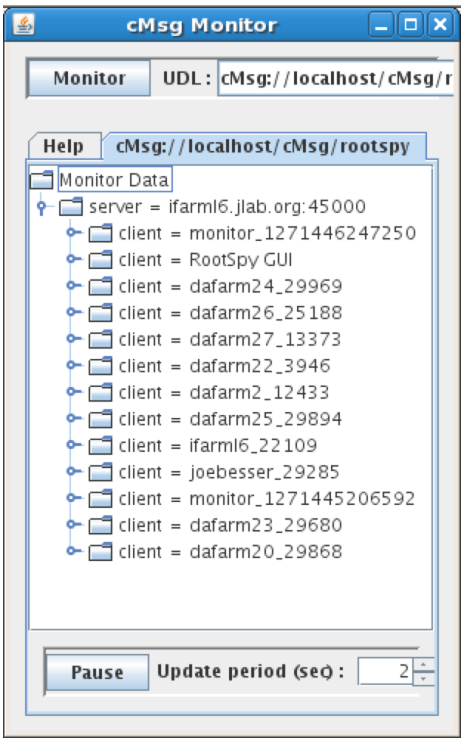

Figure 5: cMsg monitoring program (supplied with the cMsg) indicating multiple histogram producers and the RootSpy GUI attached to the server during the test in which 14 producers (10 shown) supplied histograms.

Table 2: Features to be added to RootSpy

Configuration Files: Implement configuration file feature to save and restore histogram display configuration. In the counting house environment, this will allow shift takers to quickly bring up windows meant to continuously monitor specific detector systems, or the detector as a whole.

"Final" Histograms: The system will need a way to request producers to automatically send the final histogram from a run when it detects the last event from that run. This will allow a RootSpy archiver to record cumulative histograms that contain all events processed for a specific run before opening a new file for the next run.

Archiver: An archiver program that collects, sums and eventually stores the final summed histograms for a run will be written.

Local reset: A client viewing a histogram may want to reset the histogram being displayed without disturbing the histogram on the server. This will be done by recording the histogram when the local "reset" is performed and subtracting it from subsequent histograms read from that producer. Producer generated timestamps for each histogram delivered will be implemented to help with the bookkeeping. An "unreset" feature will also be present in case the users wishes to return to viewing the full statistics.

Histogram overlay: It may be desirable to view histograms from an archived file overlayed on the current histogram. The user could specify a file that could be searched (automatically or by the user) for a histogram to overlay. An option to re-scale the file histogram should be present to allow histograms with different statistics to be compared visually. (This could be done by plotting versus the right hand axis and having radio buttons to allow a choice from having the axes be independent or forcing one to be the same as the other.)

Tree handling: Remote processes should be able to access tree information through RootSpy. Sending entire trees is usually not practical so the remote process will need to be able instruct the histogram producer to create a histogram and project a tree onto it using specific cuts. The resulting histogram is then returned in the same way other histograms are. This feature will require some coordination between the RootSpy thread and all other threads in the producer to prohibit simultaneous access. A flag indicating the producer program will support this feature of RootSpy will be used to allow/prevent RootSpy from structurally modifying the ROOT global memory.

Alternate Histogram Formats: A field indicating the format type of the payload containing the serialized histogram information will be added. This will allow non-ROOT programs to supply histograms to the system. A simple XML-based format will also be implemented to support this feature.

Histogram Categories: Categories for histograms will be introduced such that producers may indicate a specific category certain histograms belong to. Categories will allow clients to limit the list of histograms they deal with. They will also be used to publish/subscribe to specific subjects and types using the cMsg system in cases where the bandwidth may be reduced by doing so.

Web Interface: A web interface will be added that allows offsite access with limited functionality to histograms via the web.

5 Interface with DAQ system

The monitoring system will need to interface with the DAQ system in two ways. First, it must access events from the DAQ system in real time (see below). Second, it must be able to identify state changes so that archive files may be closed and histograms reset when a run is ended. The latter is complicated by the latency in event processing through the ET system with respect to the actual state transition of the DAQ system. Signaling each monitor program when a run ends will typically happen before the last events from the run have been processed. This means the monitoring programs will need to remember end state transitions, but not act upon them until it sees the ET buffer has emptied. Run number changes within the events will provide an alternate, implicit indication that all events from the current run have been processed. For contrast, it is noted that in previous experiments, special control events in the data stream were used for this purpose. In those cases, however, a single program would see every event. The parallel system of monitoring programs in Hall-D precludes that solution due to each process receiving an independent set of events such that only one process will see the control event.

Figure 6 shows a diagram of the monitoring system DAQ interface. This will require a AFECS-enabled farm manager program. This will be a full CODA component that will present the monitoring farm to the DAQ as a single entity. The monitoring farm may therefore be included as a required component in certain CODA configurations. This allows the farm to have a flexible number of nodes that may come online and offline during a run with the farm manager monitoring that a minimal number are present to maintain running status. The farm manager will communicate with the monitoring programs using the cMsg system, publishing state transitions as they are generated. This allows the flexibility of hosting the monitoring system communication on a different cMsg server than that used by AFECS.

Events will be read into the monitoring programs via an ET system. This functionality will be handled through JANA which have built-in ability to read from ET systems or files.

6 Offsite Access

Collaborators offsite may wish to monitor the status of the ongoing experiment. Direct access to the histogram producer nodes (via ssh tunnel or otherwise) could disrupt operation so an alternate mechanism for external access will be developed. There will be a web-based component to provide simple, universal access. An aggregator/cache system will also be developed for those case where access to the ROOT objects themselves is desired. Both systems will be implemented for use outside the counting house (including those in offices at CEBAF Center). They will be implemented with maximal bandwidth restrictions on traffic to/from the counting house to limit potential disruption to data taking operations.

Implementation of the offsite access features will be implemented later as it is non-critical for experiment operations to begin.

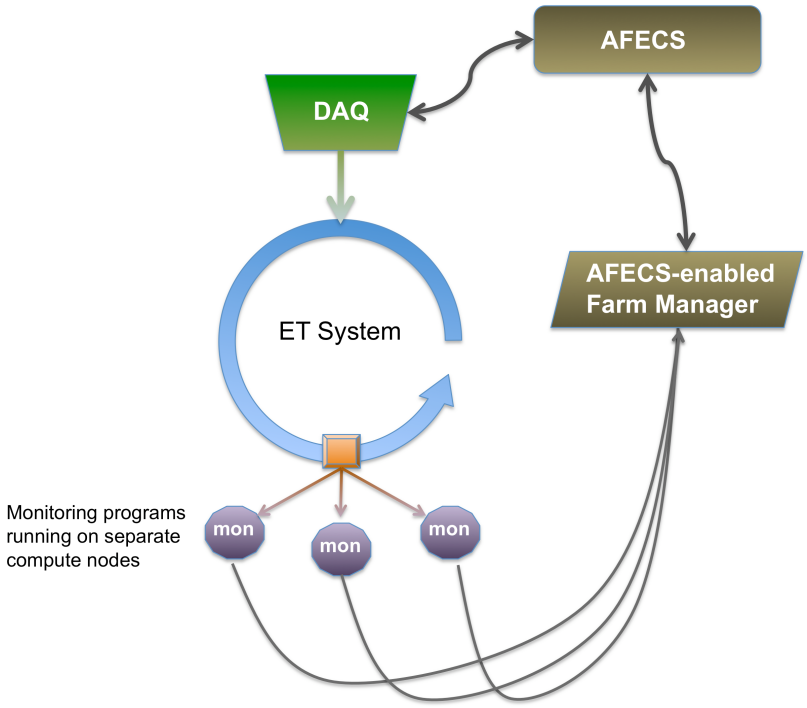

Figure 6: Diagram illustrating how the monitoring programs interface with the DAQ system. Arrows indicate network communications. Connections to RootSpy from the monitoring programs is not shown.

7 Summary

A working prototype of a histogram monitoring system has been written and tested for proof-of-principle. The system will need several additional features (table 2) to be fully functional for data taking. Work is scheduled to begin on implementing these features starting with the writing activity[7] in June 2012. However, work can begin as manpower becomes available. A large scale test will be planned when sufficient progress has been made, nominally in winter 2012-2013.

[7] 12GeV Project Activity 1532080b "Write Monitor Histograms"