GlueX-doc-2287-v1

Hall-D Online Data Challenge 2013 Report

David Lawrence[1], Elliott Wolin[1], and Sean Dobbs[2]

[1]Jefferson Laboratory, Newport News, VA
[2]Northwestern University, Evanston, IL

September 14, 2013

1 Introduction

An Online Data Challenge was performed in the Hall-D counting house the week of April 26, 2013 (see fig. 1). The scope of the challenge was to test the section of the online data flow path that lies between the final stage Event Builder and the tape silo. The primary goals were to test the online monitoring and level-3 (L3) trigger systems. A secondary goal was to test the data transfer mechanism and rate from the counting house to the tape silo in the Computer Center. Success was achieved for all tests with rates limited by the hardware currently installed in the counting house. No major software or hardware design issues were encountered, although several minor ones were identified. This document summarizes the tests performed and the milestones achieved through preparing for and executing the Data Challenge.

Figure 1: Exciting high-action shots of the Data Challenge underway.

2 Online Environment and System Accounts

Accounts were created on all counting house computers and the required software installed. The configuration of using *hdsys* and *hdops* accounts for administration and operations respectively as described in reference [1] was implemented. Software required for the Data Challenge included:

- EVIO Package for reading and writing JLab custom binary data format
- ET Event Transfer system for efficiently moving event-based data buffers between processes (locally or remotely)
- cMsg Message-based publish/subscribe interprocess communications package
- JANA Multi-threaded event processing framework for offline event reconstruction and L3 framework
- *sim-recon* Hall-D Simulation and Reconstruction software
- Xerces-C, Cernlib, Root, curl Misc. packages required for building *sim-recon*

The computer consoles that provide the human interface to the counting house operations were installed two weeks prior to the start of the online data challenge. These consisted of five identical systems, each with a set of three monitors. Two of the monitors were standard high-definition computer monitors while the third was a larger television monitor connected via HDMI. The systems were configured such that all three monitors formed a single desktop space where windows could be easily dragged between any of them. The console computers themselves were dual core 3.2GHz Intel machines. Table 1 lists all of the computers used for the Data Challenge.

Software was compiled using the GCC 4.8.0 compiler installed on the JLab CUE system and managed by the Computer Center.

3 Raw Data Format and Translation Table

Data files used in the Data Challenge were generated using the *bggen* program, a PYTHIA-based[2] program for generating hadronic events from the single pion threshold up to photon energy end point (12GeV). The values were randomly smeared to give resolutions consistent with expected detector resolutions. The values were then converted into digitized units and written into an EVIO formatted data file using the *mc2coda* library provided by the JLab DAQ group. The translation table that will be used to convert the crate, slot, channel indexing to detector indexing was used to produce the files. We note that this is the actual translation table that will be used during production operations.

Table 1: Computers used for the online data challenge

node	type	cores	RAM	function
gluon01a gluon02 gluon03 gluon04 gluon05	3.4GHz Intel i5	2+2ht	16GB	console (human interface)
gluon20 gluon21 gluon22 gluon23	1.9GHz AMD Opteron	8	8GB	monitoring
gluon40 gluon41 gluon42 gluon43	1.9GHz AMD Opteron	8	8GB	L3 trigger & monitoring
gluon44 gluon45	2.53GHz Intel Xeon	8+8ht	48GB	L3 trigger, event source, & monitoring server
gluon100 gluon101 gluon102 gluon103	1.9GHz AMD Opteron	8	8GB	L3 trigger
halldraid1	2.0GHz Intel Xeon	4+4ht	12GB	RAID disk

4 The Data Flow Architecture

Initial GlueX data taking includes multiple phases where the rate and volume of data will be gradually ramped up[3]. The L3 trigger will not be required through phase III running and only becomes a requirement for phase IV. The project plans, however, include building in L3 infrastructure early on so as to make the implementation of L3 a smoother transition. Therefore, the design of the data flow for the Hall-D DAQ system includes processes that can easily add a L3 trigger filter. Figure 2 shows the design implemented for the Data Challenge and what is anticipated for the final system. The system uses two ET systems for transport of the data from the Event Builder (EB) to the Event Recorder as shown in the upper half of the diagram. Two additional ET systems are used for monitoring the pre-L3 and post-L3 event streams. Both of the ET systems that are dedicated to monitoring read events from the remote ET systems via *non-blocking* stations. This means that every event need not pass through the monitoring system. All other connections are set to *blocking* mode which guarantees all events coming from the EB pass through a L3 process and into the ER. It also means that once an event gets into one of the monitoring ET systems, it will be processed. The design choice to make the monitoring ET systems run in *blocking* mode was to limit the rate that events are pulled from the EB and ER systems for monitoring. This configuration limits those ET to ET transfer rates to what the monitoring farms can handle. Specifying the L3 algorithm to use is done by specifying which plugin the L3 process uses.

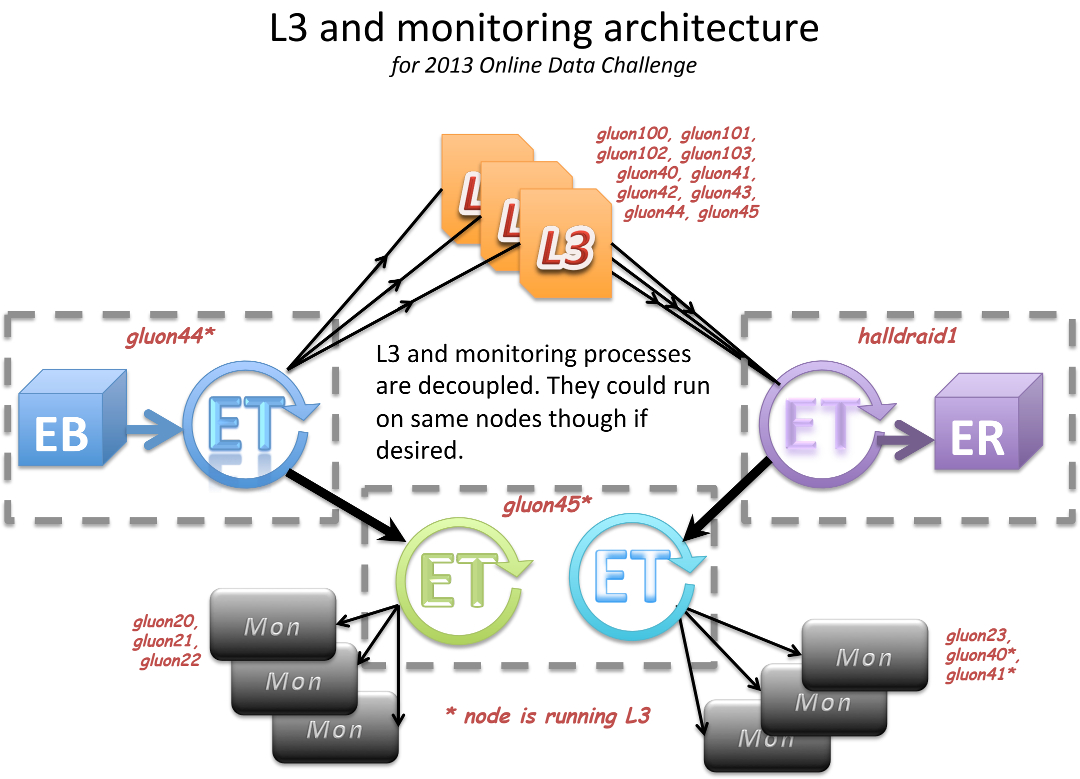
During the Data Challenge, the *hdl3* program was used for L3. This program was designed to write EVIO formatted output to either a file or an ET system.

4.1 Implementation of the L3 Trigger System

The design of the L3 system combined the ET system and JANA[4, 5]. As shown in figure 2, multiple L3 processes are used to transfer events from the EB to the ER. Each of these is a JANA-based program running on a separate node. The JANA program itself is multi-threaded and able to fully utilize the available CPU on the node in a single instance of the process. The L3 trigger software mechanism works by creating a DL3Trigger object for each event. The object contains a flag indicating whether the event should be kept, discarded, or no decision could be made. The “no decision” option could be used to chain different L3 algorithms together if desired.

4.2 Data Transfer to JLab Tape Silo

Events accepted by the L3 system were sent to an ET system on halldraid1. There a process (et2evio) wrote the events to an output file with maximum length 10 GBytes (this size is currently preferred by the GlueX Offline group). If the size limit was reached the file was automatically closed and another open with a new file name that incorporated an incremented sequence number.


L3 and monitoring architecture

for 2013 Online Data Challenge

L3 and monitoring processes are decoupled. They could run on same nodes though if desired.

gluon100, gluon101, gluon102, gluon103, gluon40, gluon41, gluon42, gluon43, gluon44, gluon45
L3
gluon44*
EB
ET
halldraid1
ET
ER
gluon45*
ET
ET
gluon20, gluon21, gluon22
Mon
Mon
Mon
gluon23, gluon40*, gluon41*
Mon
Mon
Mon
* node is running L3

Figure 2:

Note that we only wrote L3 accepted events to disk during part of the data challenge to save disk space, and we did not save all the files (recall they contain multiple copies of events from the monte-carlo input data file). In total we kept 1.4 TBytes of data from one run on disk.

While writing the 1.4 TBytes we simultaneously copied 100 GBytes from halldraid1 to the JLab tape silo. We initiated the copy manually, using jput from the hdops account, simulating the action of the CRON job that would be running in a production environment. The elapsed time for the copy was about half an hour, giving an average transfer rate of about 50 MBytes/sec (recall in production we expect 300 MBytes/sec). We are currently working with the JLab computer center to determine actual disk staging times and tape writing times during that period.

Note that testing file transfer in this way does not simulate conditions expected during production running very well. We used an old raid system (new one expected Fall 2013), we used older tape drives (new drives on order have higher copy speeds), and we did not run in the high-priority mode typically set up for running experiments. In other words our disks were slow, the tapes were slow, and we only got low-priority access to the silo system via the use of jput.

We are current planning a more realistic tape transfer test in early Fall 2013 using a high-priority access mechanism and dedicated tape drives. We expect our new raid system should be installed by then, and although we still will be using old tape drives, scaling up to the new drive speeds should be straightforward.

5 The RootSpy System

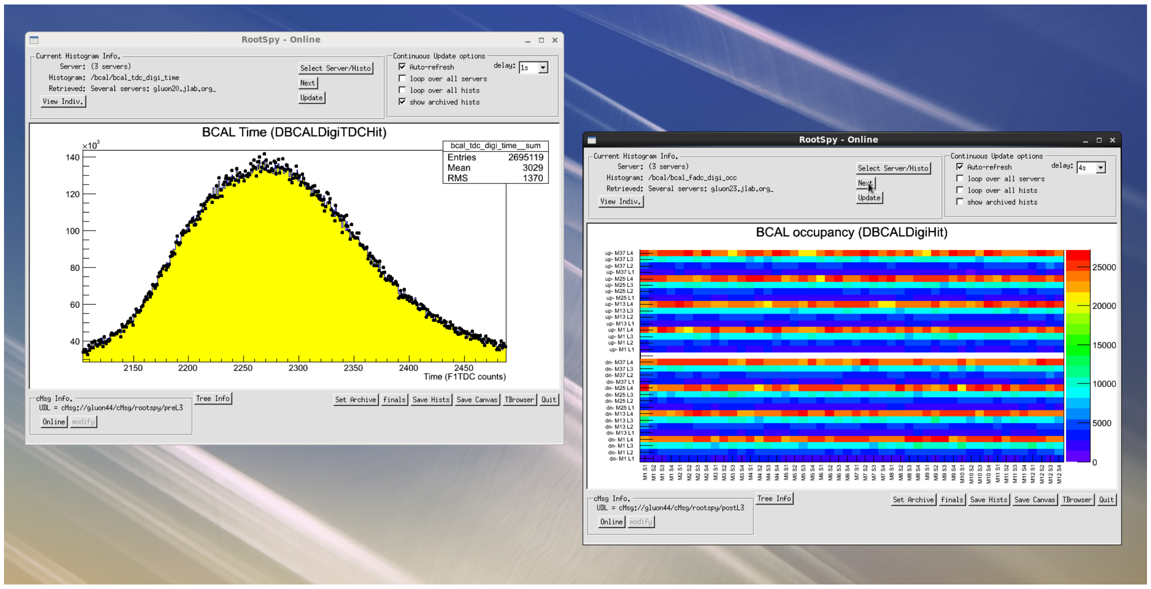
The RootSpy system[6] uses ROOT object serialization and the cMsg message passing architecture to provide a distributed architecture for accessing and collecting histogrammed data. Generally, a JANA-based program is used to generate the histograms, and the RootSpy client is loaded as a plugin, which then allows a master RootSpy program to view or otherwise access these histograms.

For the Data Challenge, the monitoring histograms were generated by a series of plugins corresponding to the various detector subsystems. Only histograms of low-level detector information were generated, such as channel occupancies and timing information. The filling of these histograms provided an important test of the analysis framework's ability to correctly parse and interpret the raw data format.

We divided the monitoring farm evenly into two parts, as illustrated in Fig. 2, in order to separately monitor the events before and after L3 processing. As with the L3 system, the JANA program is multi-threaded, so one program per node was run which utilized all available CPU resources. The pre-L3 and post-L3 processes were run in separate cMsg namespaces so that they could be viewed separately. We verified that the event rate processed by the monitoring farm was the same as that being passes through the L3 system at low event rates. When the event rate increased beyond the monitoring farm's ability to process events, we verified that the rate through the L3 system was not affected.

The histograms from the monitoring processes were viewed using the main RootSpy program with a ROOT-based graphical interface (GUI) on the operator consoles. The RootSpy GUI has the ability to view histograms generated by individual client proceses or the summed histogram of multiple clients. Two different instances of the RootSpy GUI were used to seperately view the pre-L3 and post-L3 monitoring histograms. We also verified that we could monitor the histograms generated by the L3 system.

Figure 3 shows a screen capture of the RootSpy GUIs used for pre-L3 and post-L3 monitoring. The pre-L3 is on the left where the yellow filled area is a histogram from an archive file, normalized to the "live" histogram drawn with the black points. This will provide an easy visual to compare to previous runs to identify discrepancies. The post-L3 window is on the right in the figure and shows the occupancy of the BCAL for fADC hits. The upstream end hits are shown in the top half and the downstream hits in the bottom half. Module, layer, and sector numbers are as labeled.


RootSpy - Online

Current Histogram Info.
Server: (3 servers)
Histogram: /bcal/bcal_tdc_digi_time
Retrieved: Several servers: gluon20.jlab.org_

Continuous Update options
☑ Auto-refresh   delay: 1s
☐ loop over all servers
☐ loop over all hists
☑ show archived hists

bcal_tdc_digi_time__sum
Entries 2695119
Mean 3029
RMS 1370

cMsg Info.
UDL = cMsg://gluon44/cMsg/rootspy/preL3

RootSpy - Online

Current Histogram Info.
Server: (3 servers)
Histogram: /bcal/bcal_fadc_digi_occ
Retrieved: Several servers: gluon23.jlab.org_

Continuous Update options
☑ Auto-refresh   delay: 4s
☐ loop over all servers
☐ loop over all hists
☐ show archived hists

cMsg Info.
UDL = cMsg://gluon44/cMsg/rootspy/postL3

Figure 3: RootSpy GUI screen capture taken during the Data Challenge. The window on the left shows the pre-L3 monitoring while the window on the right shows the post-L3 monitoring. See text for more details.

5.1 RootSpy Archiver

Besides the monitoring of histograms during data taking, it is desirable to store the histograms that are produced for later viewing and analysis. To accomplish this, we have written a program that, when the end of a run is signaled, will collect all of the histograms generated by the monitoring processes and writes them all out to an ROOT file which is then archived in a common location on the file system. We successfully tested an initial version of this program during the Data Challenge. Some work was also done to integrate this program with the run control system that will be used during actual operations. As this system is still in the early phase of its design, we expect that additional development and testing will be done as the system matures.

6 Summary

Several important milestones were achieved as a result of the Data Challenge. These were:

- L3 trigger framework tested

  - "pass-through" algorithm used with multiple nodes to achieve data rates at network limits

  - Boosted Decision Tree based algorithm from MIT group tested (with some issues)

- Monitoring framework tested

- RootSpy tested on the user consoles that will be used for monitoring Hall-D experiments online

- Archiving of monitoring histograms tested

- Link connecting Hall-D counting house and Computer Center tape silo tested:

  - *jput* command used

  - security certificate authentication

References

[1] Elliott Wolin and David Lawrence. Hall d daq software code management. Technical Report GlueX-doc-1892-v1, Jefferson Lab, 2011.

[2] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 0605:026, 2006.

[3] GlueX Collaboration. Hall-d offline software status. Technical Report GlueX-doc-2017-v2, 2012.

[4] D. Lawrence. *Multi-threaded event processing with JANA*, number 062 in PoS, http://pos.sissa.it/archive/conferences/070/062/ACAT08_062.pdf, Nov 2008. SISSA.

[5] David Lawrence. The jana calibrations and conditions database api. *Journal of Physics: Conference Series*, 219 part 4:(6pp), 2009 doi: 10.1088/1742-6596/219/4/042011.

[6] David Lawrence. Plan for monitoring histogram system in hall-d. Technical Report GlueX-doc-1721-v1, Jefferson Lab, 2011.