CONTROL OF ACQUISITION AND CLUSTER-BASED ONLINE PROCESSING OF GRETINA DATA*

C. Lionberger[#], M. Cromaz, LBNL, Berkeley, CA 94520, U.S.A.

Abstract

The GRETINA gamma ray tracking detector will acquire data from 112 digitizer modules in 28 VME crates. The data will be distributed to a cluster of on the order of 100 computer servers for the computation-intensive initial processing steps which will be run concurrently with data acquisition. A slow-controls system based on EPICS controls all aspects of data acquisition and this online processing. On the cluster, EPICS controls not only when processing is occurring but which processing programs are running on which nodes and where their inputs and outputs are directed. The EPICS State Notation Language is used extensively both in the VME and cluster Environments.

GRETINA

GRETINA (Gamma Ray Energy Tracking In-beam Nuclear Array) [1] is an array of 28 36-segment germanium crystals. The crystals are grouped into 4-crystal modules designed to be closely packed to cover one-fourth of a sphere, with an inner radius of 15 cm and a germanium thickness of 9 cm. The intent is to distinguish the locations and energies of the multiple interactions a gamma ray makes within the germanium shell as it decays, with a spatial resolution of about 2.0 mm.

Digitization of the segment signals is performed by custom VME-based signal digitizer modules. Each VME module hosts 10 digitizer channels. The digitizers have 16-bit ADCs which sample at a 100 MHz rate and can store traces of up to 498 readings per channel per event. In addition to the trace information the digitizers calculate channel energies and associate timestamps derived from a global clock with each segment event. Readout of signals within the entire array at a selected time is enabled by a trigger system composed of custom VME modules connected to the digitizers by high-speed links. The four digitizer modules necessary to instrument a crystal are serviced by a single VME processor board. This board reads out the signals from FIFO memories on the digitizers, sorts them into a time-ordered data stream and provides the data to a computing cluster for *online processing.*

The first stage of online processing is *signal decomposition*, a compute-intensive process of determining the number, location and energies of interactions within the crystal during the time sampled by the digitizers. Signal decomposition is trivially parallelizable as the signals from one event in one crystal, termed a *crystal event*, may be decomposed independently of signals from other crystals or other times. The resulting position and energy information is combined with time-corresponding information from the other crystals in the array by the global event build process, so that the data appears for the first time as a single stream. This stream is fed to the tracking process which uses physics constraints to order the interactions which occurred during the same trigger and reject illegal combinations.

COMPUTING ARCHITECTURE

When looked at from the point of view of a single stream of data acquisition and processing the system data flow can be represented as a pipeline. Each stage of the pipeline is composed of an architectural unit or pattern termed a *pipeline component*. (Figure 1)
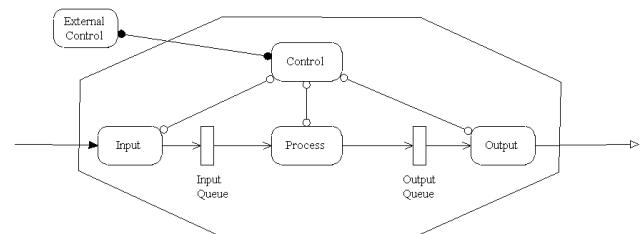

Figure 1: Pipeline component internal elements

The *control* element mediates all external controls, the *input* element obtains input data from the previous pipeline stage or from the digitizers at the beginning, the *processing* element performs the data manipulations for which the pipeline component is intended, and the *output* element sends data to the next stage of the pipeline, or to disk storage at the end. Input and output elements are interfaced to the processing element using queues to isolate processing from the vagaries of I/O and attention is given in the interfacing of the control element to the other elements to responsiveness of the controls combined with graceful termination or starting of processing. Except for the input and output queues each element is a separate thread of execution.

The entire pipeline is depicted in figure 2. Data originates in the digitizers and is sent to the signal decomposition nodes. From there it is gathered by the global event builder and sent to the tracking nodes. The outputs of the tracking nodes are again sorted into a time ordered stream and stored to disk by the data storage node. In addition there is a logging node. Although not in the pipeline it has the same architecture as a pipeline node. It can accept log entries from any node at any time

[#]calionberger@lbl.gov

and stores them in a first-in-first-out order in a log file for analysis or debugging purposes.
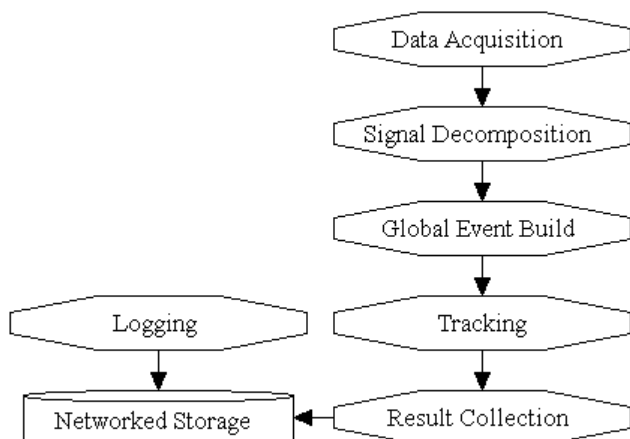

Figure 2: Data acquisition and processing

A high degree of parallelism is employed both in the data acquisition and online processing stages. As mentioned above data acquisition is performed in parallel from the 28 crystals by 28 VME-based systems. Each VME system in turn provides data to several cluster computers for signal decomposition. Signal decomposition relies on large static crystal-dependent data structures, so decompositions of data from only one crystal are performed in any one computer. As well as rendering the data more comprehensible, signal decomposition reduces its size by a factor of about 30. Because the data does not come together into a single stream until this compression has been performed it is possible to use gigabit Ethernet to connect all computers even though each VME computer is required to be able to provide 10 Mbytes per second of data.

Currently quad-core computers have become common, and it is expected that by the time the final system is implemented in 2009 most of the computing cluster will consist of 8-core processors. The pipeline component pattern is adapted to multiple-core CPUs by incorporating additional threads running whatever elements are most critical. In the case of signal decomposition this is the central computing element; in the case of global event build it is the input element.

CONTROLS

EPICS (Experimental Physics and Industrial Control System) [2] is used for the hardware interfacing of slow controls for the digitizer and trigger modules, for controlling data acquisition and processing, and for configuring the computing cluster.

EPICS device supports are used for interfacing EPICS PVs (Process Variables) to the digitizer modules and the trigger modules. Each digitizer board has about 40 PVs which control and monitor board-wide items and 16 PVs for each of the 10 digitizer channels. The trigger module supports 175 I/O points; 5 trigger modules are used in the

system. Because a given parameter is usually set to the same value on all boards and channels in a run, provision is made for single-point control using a set of global controls distributed by data fanout PVs to the hardware PVs. EPICS autosaverestore is used for the individual parameters with PINI=YES set, but is used only to set the value of but not process the global PVs, so that if certain channels are set up uniquely this will survive rebooting. The EPICS PVs which monitor and control the hardware can be accessed by a set of edm screens.

The highest level of control is *run control*. Run control has two major modes, *run* and *setup*. During setup mode system parameters can be changed but data cannot be acquired; during run mode data can be acquired but parameters cannot be changed. Parameter change control is implemented using channel access security and can be compromised at that level for testing or tuning purposes. The user interface to run control is a python GUI using wxPython for the graphics. The script uses the fermilab python channel access interface with some enhancements allowing monitoring that were added at LBNL. Run control requires the setting of certain parameters such as data storage location before a run is allowed, stores run information, and sets a run start flag. All pipeline components are expected to respond to the run flag by going into run mode; if any do not it is detected by a sequencer program that monitors the state of each component. Similarly, retraction of the start flag to stop the run is expected to cause the pipeline components to respond but there is no hard interlock. The summarizing program can be told to ignore any component by setting it offline.

Pipeline components are implemented using SNL (EPICS State Notation Language). SNL is suitable for this role because it implements each of the state machines in a state program as a separate thread, it has a very straightforward yet event-driven interface to EPICS PVs, and it interfaces C code directly. Each element (except the queues) is a separate state machine; all state machines in a pipeline component are generally in the same SNL program.

A major problem in parallel systems is that of load-leveling: making sure that computing resources are available proportional to need. The loading of the signal decomposition nodes connected to a single crystal will be evenly spread between those nodes because they obtain data from the data acquisition computer on demand; architecturally the data acquisition computer is a data server and the decomposition nodes are clients. Not all crystals will have the same data rate, however, and so different numbers of signal decomposition nodes may need to be assigned to each. This is just one example of the flexibility that is required in assigning jobs to computers in the cluster.

The computing cluster runs Warewulf [3], a cluster software suite used at LBNL. Thus all the nodes except the master node boot the same linux software image. The pdsh parallel shell command is used to start a program launcher SNL program on all the nodes. The launcher

program examines a set of EPICS PVs which specify what program should run on each cluster node and start that program. Other parameters such as which crystal each decomposition node is to obtain data from are also indicated by EPICS PVs. The launchers monitor the program name and number specifications and if any is changed by the operator the launcher will kill the
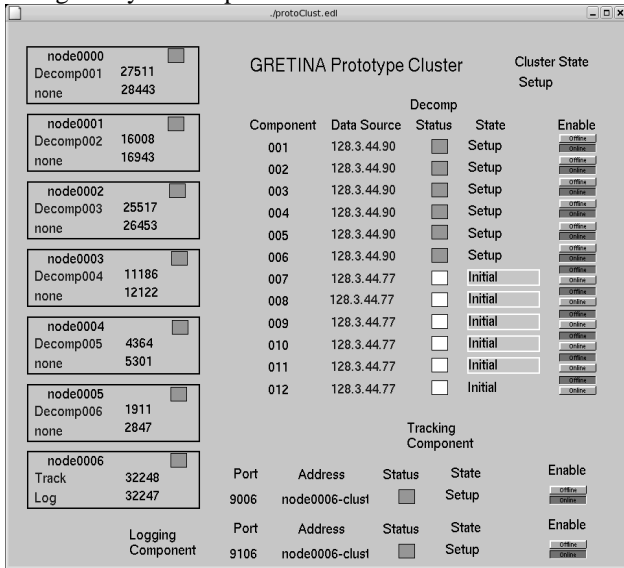

Figure 3: edm screen controlling program assignment to cluster nodes

currently running program and start the new one. If the new one fails to start, eg, if it doesn't exist, the launcher will retry every 30 seconds. Also if the program crashes the launcher will be signaled and will restart it. Thus the mix and assignment of programs running on the cluster can easily be changed without a cluster restart. Currently these assignments are all controlled from a GUI (figure 3).

CURRENT STATUS

The GRETINA software package development is scheduled to have 5 major releases, the final one to be shipped with the instrument in early 2011. The second release has just been completed and tested on a cluster containing 8 servers from a prototype acquisition system serving data from 3 crystals using 1 VME cput and 15 8-channel prototype digitizers. At this time the digitizer control, program launchers, data acquisition code, signal decomposition nodes and logging nodes are considered to be nearly in their final forms, but global event build, tracking and data storage functions are combined in one program. In addition considerable development is still required in the trigger system and a relational database system to store crystal characteristics and an EPICS interface to it are still in the design stages. An online system to sample the data stream and do some higher-level analysis in near real time is also planned.

CONCLUSIONS

The EPICS system is suitable for implementation of parallel system control; this has been well accepted at the level of hardware control. In GRETINA it is being used for control of parallel software system running on a computer cluster as well.

REFERENCES

[1] Project home page at http://grfs1.lbl.gov/

[2] http://www.aps.anl.gov/epics/

[3]   http://warewulf.lbl.gov