Jefferson Lab
Thomas Jefferson National Accelerator Facility

Streaming Mode Data Acquisition at Jefferson Lab

Graham Heyes

Thomas Jefferson National Accelerator Facility

Experimental Nuclear Physics Division

Data Acquisition Support Group

Information Technology Division

Scientific Computing Department

Abstract

Over the last decade advances in electronics, computing, and software have changed the assumptions upon which data acquisition system designs for nuclear physics experiments are based. This is true at Jefferson Lab (JLab) as well as at other laboratories. Looking forward to future experiments that are in various stages of planning we must reevaluate the near- and long-term course that the development of readout systems for JLab experiments will take. One technique that is growing in favor in the data acquisition community is streaming mode readout, whereby detectors are continuously read out in parallel streams of data. The goal of this document is to introduce streaming mode readout, discuss the advantages and disadvantages in the context of JLab experiments, and make comparisons to the pipelined trigger mode currently used by JLab experiments.

Document written as contribution to JLab agenda, Science & Technology Outlook 1. Strategic initiative 3.

Introduction

For many years the design of data acquisition (DAQ) systems for nuclear and high energy physics experiments was strongly influenced by assumptions about the types of data generated by detectors and the associated data rates. Some of these assumptions can be summarized as follows:

- *The data rate from a detector is impossible to capture with an affordable data acquisition system without a trigger to reduce event rates.*
- *Even if the untriggered data rate could be captured it would be impossible to store.*
- *Even if it could be stored the full dataset would represent a data volume that would require impractically large computing resources to process.*

Based on these assumptions a trigger-based readout system, CODA, was designed and implemented as the foundation for DAQ systems at JLab over the last 25 years. Lessons learned from the operation and evolution of CODA, and projected requirements for future experiments, have led to a search for an alternative DAQ design.

The ubiquitous presence of computing in 21st century life has driven down cost and driven up performance to a point where one can argue that, at least for the nuclear physics experiments envisaged over the next couple of decades, the assumptions that held for most of the last 25 years are no longer valid. In fact, extrapolating forward the evolution of both experiment and technology it is likely that these assumptions may never be valid again. In addition, trigger-based readout systems have some well-known intrinsic limitations for certain experiments. They can carry bias to low-energy particles, do not deal well with event-pileup, and are not an ideal match for complex, general-purpose detectors. The compromises put in place to limit the data rate inevitably lose information. For example, detector channels are often summarized by single numbers rather than full wave forms.

To take advantage of the updated assumptions, and to make minimal or trigger less acquisition possible, alternative modes of readout are an option and one, streaming mode, is an ideal fit to nuclear physics DAQ. This document defines streaming mode where:

- *Data is digitized at a fixed rate with thresholds and zero suppression applied locally.*
- *Data is read out in continuous parallel streams that are encoded with information about when and where the data was taken.*
- *Event building, filtering, monitoring, and other processing is deferred until the data is at rest in tiered storage.*

The discussion that follows shows how a practical streaming system could be implemented that does not suffer from the limitations that the current generation of CODA based systems have. A streaming system design has much in common with a CODA based system at the hardware level and should not be any more expensive to implement and operate. The global clock, front-end digitizing, data movement, and data processing systems are essentially the same but operated in a different mode. Also, the lack of strong requirement for a global trigger using custom hardware and firmware is a significant simplification of the overall system that leads to savings in implementation and operation. Use of a streaming DAQ also opens opportunities to streamline workflows and take advantage of other emerging technologies.

For all of these reasons streaming mode should be the adopted mode for a universal DAQ to replace CODA.

DAQ Readout modes

Triggered pipeline DAQ is the mode currently used by CODA at JLab. In this mode analog signals are digitized, and the result held in a memory. A trigger system reads some or all of this data, decides whether the data merits collection or should be discarded, and initiates full readout via the main DAQ data path. An alternative mode of operation is to allow the digitizer to freely run and clock data into the memory continuously. The DAQ system must then continuously read the memory on the electronics so that it never fills. Consider a single channel of a 250 MHz flash ADC. If each sample was encoded as 4 bytes a single channel would generate over 1 Gbyte/s of data. Summed over a large detector the data rate would be beyond the capability of any DAQ system that could be affordably implemented. For this reason, free-running readout is not a practical option. However, there are several techniques that can be used to reduce this rate to something that is manageable without resorting to a traditional trigger system. These make use of occupancy and compressed formatting. In most cases detectors with high channel count also have low occupancy. If this were not the case track finding and reconstruction would be impossible. Consider the single ADC channel discussed earlier, there are periods of time when the 1 Gbyte/s sequence of ADC samples encodes no useful physics. This is illustrated in Figure 1.

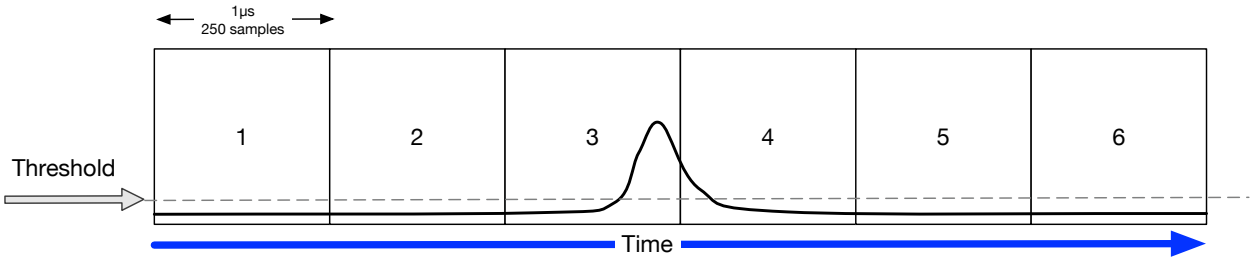

FIGURE 1 – DATA FROM A SINGLE ADC CHANNEL.

Dividing the time axis into fixed length 250 sample, or 1 μs, "frames", only frames 3 and 4 have a signal that is above threshold, the other frames have no useful data. Furthermore, the first 150 samples of frame 3 and the last 200 of frame 4 also contain no data above threshold. So, we could encode this sequence in a compressed form as a sequence of variable length blocks of data each beginning with a length and offset. The first two blocks have zero data words and are encoded as length zero. The third block has length 100, offset 150, and contains 100 words of data. The next block has length 50, offset 0, and contains 50 words of data. The final two blocks are also zero length. This encodes the data in a total of 156 words compared with 1500 if all of the below threshold data was included. One thing to note is that, even if there is no useful data at all there is, at some level, still a sequence, or stream, of data frames that state that "there was no useful data in this time period". This method of taking a data source and encoding it as a continuous time indexed sequence of data blocks is *streaming readout*. In general, for a low occupancy detector, streaming generates a lower data rate than free-running but not as low as triggered-pipeline which uses a trigger to discard above threshold data. Here is a summary defining the three modes:

1. **Free-running** - Data is digitized at a fixed rate and read at the same rate.
2. **Streaming -** Data is digitized at a fixed rate, thresholds and zero suppression are applied locally. Data is read out as a continuous stream that contains timing information that allows the original free running data to be reconstructed.
3. **Pipeline** - Data is clocked from the digitizer at a fixed rate and read out on a trigger that can depend on global data from all detectors. Each block of data has information about which trigger the data corresponds to, but blocks are read out with random timing due to the trigger.

As can be seen all three modes are similar but differ in the criteria used to manage the rate.

Practical considerations for DAQ using streaming mode

1) Readout and control

At the front-end, close to the digitizer, all three readout modes discussed in the previous section are identical. Several electronic modules in use at JLab, that were designed to operate in triggered mode, can be easily adapted to either of the alternative modes. Streaming mode uses a data format and transfer protocol that creates data blocks that are varying in length, depending on zero suppression, yet preserve the timing of when the data was generated. This well-defined flow of time ordered data blocks defines a data stream. Unlike a system with a global hardware trigger, where there is association of data with events occurring with random timing, streaming readout is driven entirely by time and the data flow through the system is deterministic. To be practical a streaming system requires a time and clock distribution system so that data frames are correctly timestamped. In practice this hardware is not too different from the global clock distribution already needed for triggered readout.

With a detector of many thousands, or millions, of channels it is not practical to have data transport hardware, for example optical fiber and transceivers, dedicated to every stream of data. Instead, though still logically independent, streams will be aggregated to share hardware. The streams of data are a sequence of data blocks labeled with time and detector. Hardware or software can combine blocks from different data sources but the same time range. This *stream aggregation* differs from event building because the process is completely blind to the data contained within the blocks and aggregation is based on time rather than trigger number. Being blind to the data payload allows use of general-purpose data handling techniques to manage the data streams. To read out a detector, channels would be aggregated at the board level to give a single stream off board. Streams from individual boards could then be aggregated by software or hardware into larger subsystems or streamed in parallel without further aggregation. Practical examples of stream aggregators are the VTP at JLab and Felix PCI card used at BNL. When subsystem level aggregation takes place, one could imagine implementing a further level of processing, for example, finding clusters of hits. This is an example of a concept that streaming readout introduces, *stream translation*, where hardware or software components accept a stream, or streams, of one data type and output a stream of a different data type.

A streaming system enables data driven control of the DAQ via the timing system. Commands would be distributed by the time distribution system and status embedded in the data via control or status codes in the data frame header. This would be simpler and more responsive than the existing CODA run control system that requires initiating and monitoring state changes by potentially hundreds of individual software components and forcing all system components to implement the same state machine.

As described above, streaming mode readout addresses several of the drawbacks observed in the operation of the CODA based triggered-pipeline readout systems at JLab.

1. There is no complex hardware-based trigger system. Most subsystems are self-triggered, and filtering based on multiple detectors deferred.
2. There is no independent trigger path for data, all data is streamed off the detector in parallel using the same hardware.
3. There is a well-defined and very deterministic data transport mechanism. Data is packed into frames representing a time interval and clocked out of the hardware at fixed block rate.
4. Apart from aggregation the streams remain independent without feedback loops that can cause instability.

5. Run control becomes data driven with control and monitoring via the data streams themselves.

With existing technology, it is possible to use Streaming Mode and read out a detector with an affordable data acquisition system without a global hardware trigger to reduce event rates.

2) Data storage and event building

Since streaming mode has the potential to generate higher data rates off the detector than the triggered pipelined mode currently used at JLab, there has been concern about storing these large datasets. In CODA based systems events are built in real time and the DAQ chain ends with the Event Recorder writing sequences of events to files on disk. These files are permanently archived as quickly as possible after the data is taken. The rationale is that data files represent beam time, beam time is costly therefore the raw data files are of high intrinsic value. This is only partially correct. In reality, for most experiments, the fraction of data representing useful physics is low. For a streaming system there is no global trigger before acquisition, so an even lower fraction of the acquired data is useful. A practical streaming system would acquire data to temporary storage and use software to generate a virtual trigger to reduce the rate. Effectively, compared with pipeline readout, the trigger has been moved from the start of readout to the end, it is calculated after the data is acquired and is in temporary storage. An advantage of this is that the data acquired is no longer biased by a trigger. The experiment then has the flexibility to process the unbiased data, i.e. running with no global trigger at all, or use a trigger to reduce the rate but interspace filtered data with an unbiased sample. This would be useful operationally by reducing the need for special unbiased runs for calibration etc. This deferment of the global trigger is illustrated in Figure 2.

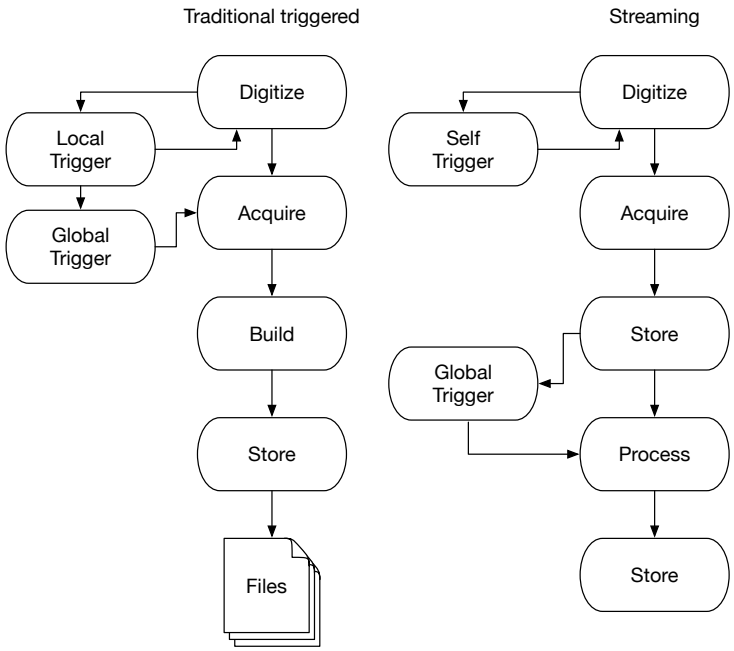

FIGURE 2 - TRIGGERED VS STREAMING

On the left of Figure 2 is a simplified block diagram of a traditional, CODA like, DAQ. The global trigger system works in real-time to select which data is acquired. In the streaming, shown on the right, the global trigger is deferred and takes the form of software, acting after the data is already acquired and in intermediate storage. The trigger can then be used to determine which data is selected for further processing and storage stages.

In a streaming system the use of a virtual trigger as a filter may not be required if the rate is low enough for permanent storage of the unfiltered data. A software trigger is always required when data processing requires the identification of events in the dataset. In this case the trigger identifies time regions containing potentially useful physics. An advantage of this model is that the use of the trigger to filter is entirely under the control of the experimenter and does not require hardware modification. Several software triggers could operate in parallel.

A further advantage of the intermediate data store is that data can be processed in near real time, for example, full or partial reconstruction, and detector calibration, this will be discussed in more detail later. Processed data is generally more compact than raw and would likely be output at a rate lower than the incoming raw streams. A streaming system would use several tiers of storage. Each tier, except the last, would temporarily store data with data processing between tiers reducing the data volume permanently stored in the final tier. The diagram in Figure 3 shows one way that tiered storage could be used to implement a streaming system.

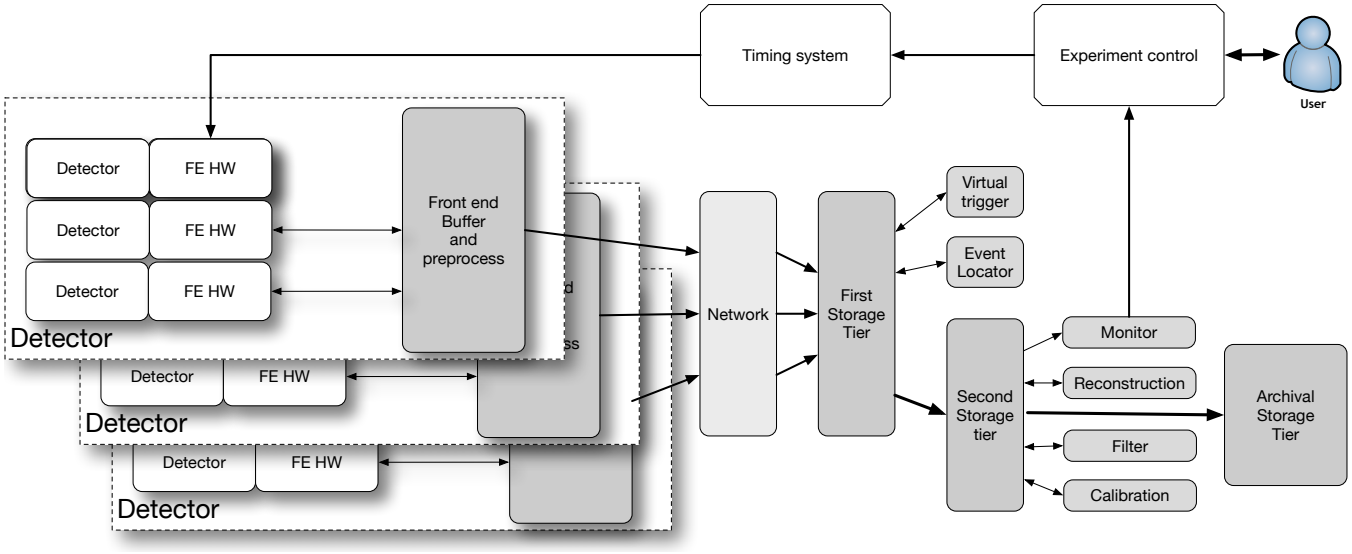

FIGURE 3 - TIERED STORAGE MODEL

On the left of the diagram detector hardware is read out by digitizers clocked at a fixed rate into front-end buffer electronics. Firmware or software at this level applies threshold cuts on the data to reduce the rate. The output of this electronics is sent in a common stream format in parallel from each detector. All of the streams are connected via network links to one or more high-speed random-access storage devices capable of accepting the incoming rate. The format of this first storage tier would most likely not be file based but would allow rapid random retrieval of data based on detector ID and timestamp. Since this tier must be fast cost will constrain it to be relatively low capacity. However, data will not need to sit in the tier for more than a few minutes. This constrains the technology to one that can survive a large number of read-write cycles, for example DRAM memory.

The virtual trigger is generated by software using data from the online storage. The trigger indicates time ranges where interesting events are located. Event locator software uses the trigger output, identifies data belonging to the event, and copies it to the next tier of storage. If the virtual trigger acts as a filter the incoming rate to the second tier will be lower than the rate into the first. This allows the second tier to be implemented using slower, and so cheaper, technology, and be fairly large. As an implementation example, solid state or RAID disk would

be a good fit. In a CODA based system events are built in real time by an EB that sits between the front-end readout and the Event Recorder. The event builder is required because CODA indexes data by trigger. Further processing requires gathering of data from various detectors that correspond to individual triggers. This is difficult to implement in real-time and an inevitable bottleneck since all data must pass through the event builder. The process used in a streaming system, outlined above, is much simpler than the CODA EB since it only identifies where the data belonging to an event is stored. The software implementation is not on the main data path and is less of a bottleneck, indexing is faster than collation. Event building in the traditional sense need not take place. Once the data is copied to nearline storage it is available for reconstruction, calibration and maybe further filtering. Software can then access the data according to the criteria defined by the global trigger and/or time, whichever makes the most sense for a particular experiment. Whether all, some, or no unprocessed data is archived to long term storage is an option under the control of the experiment.

As mentioned earlier, a streaming system could be controlled using its timing system. In Figure 3 the user interacts with the system via Run Control which controls the timing system. A monitoring process watches for tagged data and provides feedback to Run Control and the user. Since the monitor is using tags in the data, Run Control does not have to communicate directly with various parts of the system that may be too busy to respond.

3) Calibration and analysis

In a CODA based DAQ system data is either in motion through the online system, or at rest in files. The approach of tiered storage, outlined in the last section, provides the flexibility to temporarily record, or delay, the flow of the data streams while some time-consuming task, like calibration or filtering, takes place. This also allows the introduction of different formats and storage patterns that may be more appropriate for the type of data available at each stage. Most importantly it blurs the line between what may be considered to be online and offline. This allows data processing tasks to migrate up or down stream to more logical locations rather than being restrained by an online system. An example of this is the firmware that CLAS12 use to very efficiently identify track candidates. Currently this is part of the global trigger and constrained to algorithms that fit that hardware. In a streaming system it could be relocated to a more powerful FPGA accelerator card in a server.

Considering practical implementation using the example in Figure 3, the cost of a high performance first tier of storage that can handle the data rate at that stage would limit its size, and therefore the average amount of time that any block of data can sit in that storage tier. Even so, 1TB of memory can buffer a 10 Gbyte/s data flow for 100 seconds. This is ample time for many types of data processing. In practical terms the required 1TB of memory, 10 Gbyte/s data throughput, and the associated processing power to perform the virtual trigger and event location, is already exceeded by the DAQ nodes used by GLUEX and CLAS12 (because of the higher requirements of real time event building in CODA). This performance can only be expected to increase in the future generations of hardware. Since a streaming system already operates on parallel streams scaling to higher than 10 Gbyte/s using several nodes in parallel is natural. The reduction in data rate after this first processing reduces the performance requirements for the second storage tier and allow it to be large enough that data can sit in this stage for weeks or months as the primary data source for calibration, monitoring, and analysis.

In a traditional system unprocessed data off the detector, raw data, is written in files. Since monitoring of detector performance during data taking requires prompt access to data, monitoring has instead been traditionally performed by software reading data that is still in motion through the online system. Conversely, tasks such as calibration and reconstruction have been performed offline using the raw data files. The three different data processing tasks have been treated relatively independently and intermediate results are seldom stored or passed between them. The blurring of the line between what is offline or online means that any data processing tasks can

use the same data types and share data between each other. Monitoring is no longer purely online; calibration and analysis are no longer purely offline. Earlier it was mentioned in passing that one concept that streaming introduces is stream translation, where a software process or hardware component accepts streams of one data type and translates the data into another data type. This is an interesting concept in the context of monitoring, calibration, and reconstruction/analysis. For any detector there are sub-detectors for which tasks like monitoring and calibration are computationally simple and others that are more difficult. The stream translation technique would allow the uncalibrated raw data from a sub-detector to be calibrated as it is taken and converted into a stream of calibrated, even fully reconstructed, data. There are many advantages to this, one that stands out is that easily processed data is processed once, very soon after it is taken. Currently every time that reconstruction or calibration code is changed, or any part of the calibration is updated, the entire dataset is reprocessed rather than only those parts affected by the changes. In a streaming system everything is fluid, the researcher can decide whether to go back to raw streams or to preprocessed streams of intermediary results on a case by case basis. Furthermore, several groups sharing the same dataset can, if they wish, share intermediary results. This is useful in the context of data management and long-term data access; pre-processed data is much more accessible for future data-mining activities.

The architecture of the software that would be used to implement a streaming readout is beyond the scope of this paper. It is useful to note that the concepts of aggregation, translation, and processing of parallel streams of data, map very well onto a micro-services architecture. In such a system well defined data types, encapsulated algorithms, and data-driven system control allow large computational tasks to be broken down into sub tasks between which data flows. An example of such a data processing system is the CLARA reconstruction framework used by CLAS12. Here the large task of event reconstruction is broken down into smaller tasks, like tracking, that communicate between each other. Following this model allows flexibility in the implementation of data processing algorithms. Machine learning and other advanced techniques could be encapsulated in software services.

A streaming system has the potential to generate a large, shareable, dataset in an unrestricted random-access format that would be ideal for novel data processing techniques that are beyond the scope of this paper. One example would be to use machine learning techniques to compare experiment directly to theory without the laborious event-by-event reconstruction and statistical analysis that is currently standard practice. Instead of the previously accepted view that streaming readout would generate a data set that would be difficult to process it has the potential to open new approaches that will speed the path from experiment to physics results.

4) Streaming system cost comparison

At first glance streaming mode seems like a *brute force* method, solving the challenges of DAQ by using high performance and therefore costly hardware to store everything and do the processing later. Figure 4 is a comparison of streaming and non-streaming (GLUEX) DAQ systems. The central column of left and right parentheses joined by lines indicates which components of each system have equivalent function. Based on this comparison it is reasonable to assume that the cost of the readout of the GLUEX detector using streaming mode, and current technology, would be similar to, or less, than the cost of the DAQ system currently deployed. Less, because, if it had been designed to operate in streaming mode from the start the front-end electronics would not have the expensive interface to the global trigger. In addition, there would be the cost saving from not having to implement the global trigger itself which uses custom electronics and firmware. The cost of implementing the global trigger functionality in software is negligible since similar software would have to exist to model the behavior of a hardware trigger

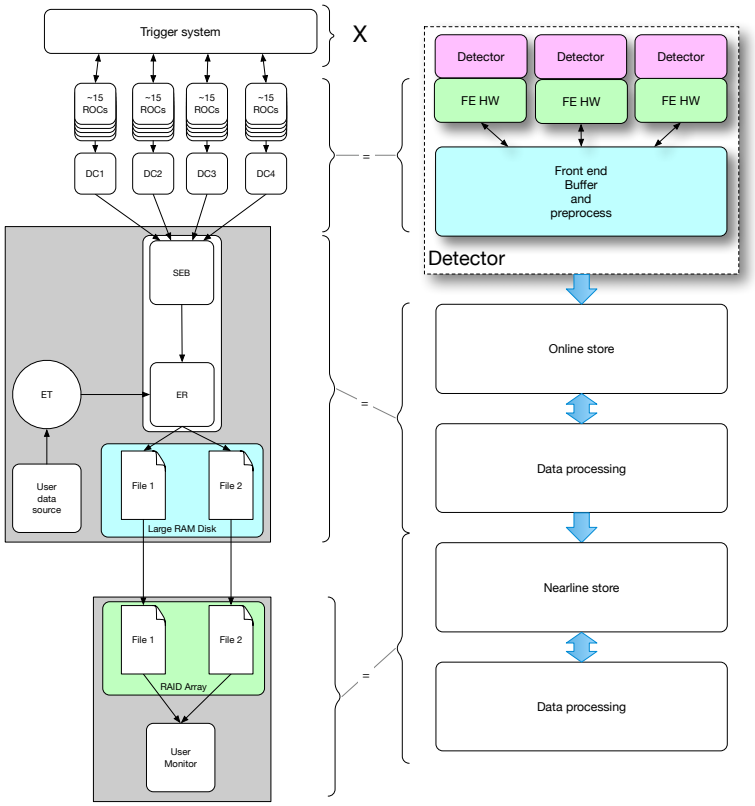

FIGURE 4- STREAMING VS NON-STREAMING

The path to a working system

The discussions in the previous pages leads to the conclusion that streaming mode should be the adopted mode for a universal DAQ to replace CODA. It is worth noting that JLab is not alone in the investigation and adoption of streaming mode readout as an alternative to traditional trigger-based systems and much can be learned from those groups. Unfortunately, streaming systems that exist now, or are planned in the next few years, are very experiment specific. The diagram, Figure 3 - Tiered storage model, shows a concept for a streaming DAQ system based upon tiered storage. It is not the only model that could be followed but is an appropriate one for JLab because of the flexibility that it enables, which is important because of JLab's broad range of experiments and detectors. The tiered storage model provides well defined places where the data is at rest for a period of time and allows discontinuities between data formats and workflows in different parts of the system.

Several critical components are required to implement a practical streaming system :

- *A stream-oriented timing system.*
- *A standard stream data format.*
- *Front end electronics that outputs time-based streams of data in the standard format.*
- *Efficient and robust streaming data transport. (probably using network protocol over fiber.)*
- *A stream oriented random-access data storage tier and associated data access tools.*
- *A framework for data management tasks such as event location and virtual trigger.*
- *A framework for nearline tasks such as calibration, monitoring, reconstruction, etc.*

Several groups at JLab are involved in projects to develop these components and are collaborating to produce a self-consistent suite of streaming capable tools. In particular, some hardware already in use with CODA to

instrument both GLUEX and CLAS12 is able to operate in streaming mode after firmware modification. To aid this work JLab has set up a small development lab, INDRA, that can be shared by various projects to gain experience with operation in streaming mode. The first project in the INDRA lab has given JLab valuable experience with the readout of GEM detectors in streaming mode. This work is a collaboration between several groups in ENP division that is of immediate use for planned experiments in halls A/C as well as being valuable R&D for EIC detector readout. We have also already modified existing firmware to allow detectors used in halls B and D to operate in streaming mode by reading out the JLab designed fADC 250 flash ADC using the VTP module that was originally designed as part of the CLAS12 and GLUEX global triggers. In collaboration with INFN, who have experience with streaming readout, we had our first data from a streaming system, in beam, in hall-B, in February 2020. Since much of the software that we would require for a general purpose JLab streaming system does not exist, we used the TriDAS software, used by the KM3NeT neutrino detector. We sucessfully integrated TriDAS with CODA (for configuration and control) and our existing streaming readout pilot projects. We are looking forward to repeating these tests using detectors in hall-D. We are actively working on two approaches to streaming reconstruction and calibration, one using the JANA2 framework and the other using CLARA.

Summary

The need of an alternative, data driven, DAQ design at JLab is driven by the lessons learned from CODA and looking forward to future experiments. This paper began with the statement that traditional event-by-even trigger based DAQ system designs are based upon assumptions that are no longer true. This was followed by a discussion of the consequences that those assumptions have for the design of a system to replace the existing implementation of CODA, leading to the conclusion that streaming mode was an ideal fit as a basis for this design.

This document defines a streaming mode of detector readout as a mode where:

- *Data is digitized at a fixed rate with thresholds and zero suppression applied locally.*
- *The data is transported from the detector as a stream, a sequence of time ordered data packets in a format that is efficient for data transport yet allows the original data sequence from the detector to be reconstructed in a lossless way.*

The discussion that followed showed how a streaming system could be implemented and how it does not suffer from the same challenges that the current generation of DAQ systems have. During the discussion concepts were introduced, two of which are of particular note:

- *Stream aggregation – where several otherwise independent streams are formatted to allow them to share the same hardware transport layer.*
- *Stream translation – where an incoming stream, or streams, of data are translated from one data type to another.*

A simple conceptual design for a practical streaming system was described based on a tiered storage model:

- *Tiers of storage, maybe using different hardware at each stage, allow the introduction of different formats and storage patterns that may be more appropriate for the type of data available at each stage.*
- *Event building, filtering, monitoring, and other processing are deferred until the data is at rest in tiered storage. This removes many of the challenges related to real-time event building and monitoring that the existing CODA systems experience.*
- *The system can be configured to provide a large random-access dataset that is available for use by several groups simultaneously and would enable novel data processing techniques.*

During the discussion of this conceptual design comparisons were made between the hardware requirements of a streaming system and the existing CODA based DAQs already in use at JLab. The streaming system is simpler to implement and operate. In many cases the hardware requirements are less stringent than those required for the CODA system. This is mostly due to the lack of real time event building and the implementation of any global trigger in software, which is cheaper and more flexible than the hardware equivalent.

If this were purely a theoretical discussion it would be compelling enough to strongly recommend streaming mode. In practice, several groups outside JLab are well on the way towards practical streaming systems, or already have them operating. At JLab we have been working on GEM readout for use in future experiments for halls A and C, notably TDIS and SoLID. We also have existing installed hardware that can be adapted, for example, almost 60% of CLAS12 readout could be converted to operate in streaming mode. A rudimentary streaming implementation using this hardware is being used, with beam, in halls B and D to gain experience. This provides a possible path to future high luminosity operation of CLAS. This work with streaming calorimeter readout is also partly driven by foreseen experiments in hall-C as well as BDX follow-up efforts. On the software side, the lab's two main ENP offline frameworks, JANA and CLARA, are both suitable as a basis for development of a general-purpose streaming DAQ. The work required to merge the best features of both into a consistent software suite is starting. For example, the successor to JANA, JANA2, is being used in our streaming tests and the design of JANA2 supports streaming as a readout mode. Meanwhile, CLARA is being modified to support data in stream format and to enable use of JANA2 at the node level while CLARA takes care of building and operating a distributed system. There is also a streaming LDRD project towards real time calibration. The immediate plan is to continue these various existing projects in parallel while merging them into a coordinated effort leading to the successor to CODA, which is the goal at JLab.