DUNE FD TPC Stage-1 processing use cases for offline software framework

Brett Viren

May 24, 2024

1 Context

DUNE expects the far detector (FD) site to be complete with total of four liquid-argon time-projection chamber (LArTPC) detector modules. They shall employ a variety of technologies to read out information about activity in the form of ionization electron charge from the time-projection chamber (TPC) and scintillation light from the photon-detection system (PDS) that is caused by energetic, charged particles traversing their argon-filled volumes.

A high-level and generalized model of the initial stages of processing of these data from each module is shown in figure 1. The data-acquisition system (DAQ) will read out raw data from real TPC and PDS detector electronics and simulation will generate data that mimics it. Each of the TPC and PDS data will then be input to independent processing chains. These branches will be merged downstream so that the PDS information can be used to determine the absolute time of various groupings (clusters) of elements of reconstructed TPC activity. Subsequent processing will derive high level physics quantities such as particle identification and momentum determination and ultimately neutrino flavor, energy and interaction type.

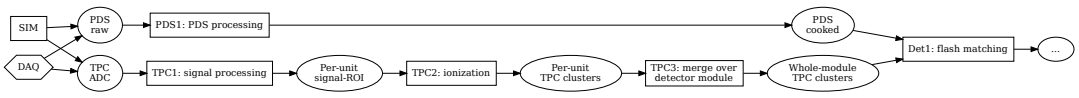


Figure 1: General, conceptual view of the initial few stages of data processing from DUNE far detector modules.

This graph represents a high-level view. In fact, each stage box can represent substantial complexity. Figure 2 shows structure internal to the "TPC1" stage of figure [[fig:stage-all].



Figure 2: Internal structure of processing stage TPC1 consisting of noise filter and signal processing components provided by the Wire-Cell toolkit.

The noise filter (NF) and signal processing (SP) stages are implemented by components of the Wire-Cell toolkit running (currently) from an *art* module that consume and produce data products

expressed as LArSoft types. The output are "signals" represented by sparse `recob::Wire` giving waveform fragments with samples in units of number of ionization electrons per channel per sample period.[1].

Additional complexity that is not expressed in figure 1 exists in the TPC2 stage. This stage represents a major branching point with both branches consuming the same set of signals as depicted in figure 3.

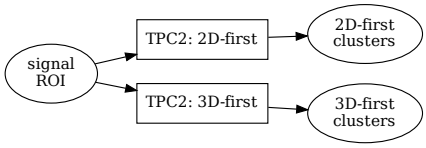


Figure 3: Internal structure of processing stage TPC2 consisting of a major branch point with each branch initating a unique reconstruction strategy.

The "2D-first" branch begins with fitting a multi-Gaussian model to the signal waveforms followed by reconstruction operating on 2D views of time vs channel for example by Pandora algorithms.[2] The "3D-first" branch begins by application of the Wire-Cell algorithm that inverts the tomographic Radon transform performed by the detector to produce a sparse 3D structure that localizes ionization charge.

Each of these two chains then continue to form clusters, stitch these clusters across detector module units (APAs/CRPs) and eventually match PDS and TPC information to identify cluster times. The exact order of stitching vs matching is a detail of the algorithms and in principle may differ for the 2D-first and 3D-first strategy.

Further processing starting with stage "Det1" is kept out of scope for this document.

2 Wire-Cell toolkit

Wire-Cell provides several major features to DUNE and other LArTPC detectors as illustrated in figure 4. Internally, Wire-Cell toolkit execute many individual components in a data-flow graph. Each component can be understood as being similar to an *art* module but with finer-grain scope. The data-flow graph can be understood as being similar to the "module paths" implemented by *art* except that execution is data-driven.

The Wire-Cell data flow graph can run in single or multi-threaded mode and it can be run as a stand-alone program or, when performing input/output with `art::Event` it may execute in the context of an *art* module. Data that passes along edges in a Wire-Cell data-flow graph are ephemeral and are only exposed to the `art::Event` by certain data-flow graph nodes that explicitly convert between LArSoft data product types and internal Wire-Cell data interface types. Essentially any Wire-Cell data-flow graph can run as a monolith or be partitioned along its "transverse" and/or "longitudinal" axis so as to run from a number of *art* modules or a number of distinct executable.

[1]Note, it is common but utterly wrong for experiments to arbitrarily normalize signal samples so that the result is numerically similar to collection ADC waveform samples. This is not a physical interpretation, adds useless confusion and the practice should stop.

[2]Current processing combines TPC1 with this initial part of the 2D-first TPC2 stage. The Gaussian fitting should be moved out ot TPC1 as it is only of interest to 2D-first and it inflate the data by almost a factor of two compared to the input signals, which must continue on to 3D-first branch.

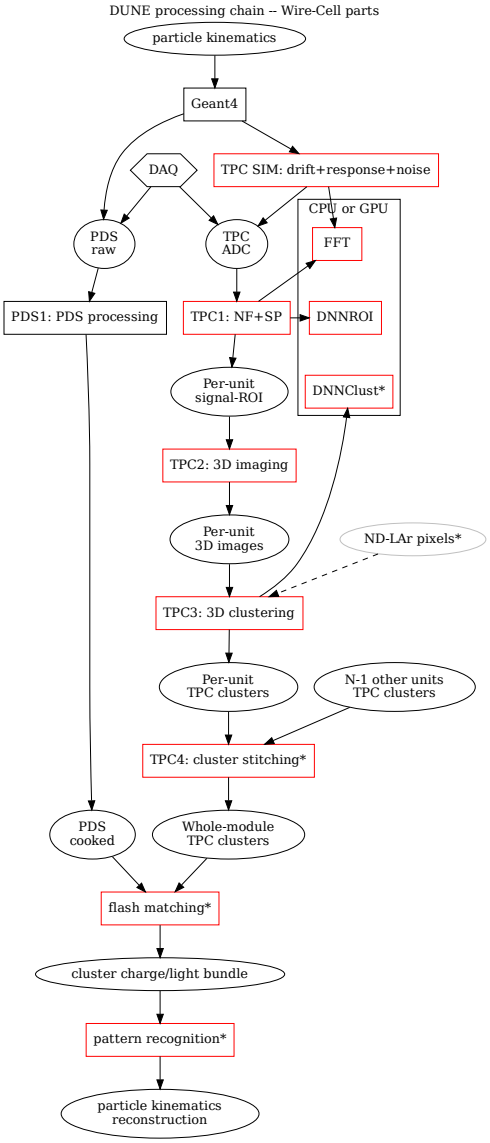


Figure 4: General, conceptual view of the initial few stages of data processing from DUNE far detector modules.

3 Simulation use cases

Wire-Cell simulation consumes energy depositions ("depos") and outputs raw ADC waveforms. Typically, depos are the product of a Geant4 simulation *eg* running in the LArG4 *art* module provided by LArSoft. They may also be provided from file which can enable Wire-Cell to run in stand-alone mode which may have certain memory benefits.

3.1 Truth-triggered simulation.

The most common method of simulation is to provide Wire-Cell simulation independent sets of depos each of which are grouped around a fixed time value, such as 0. A framework component must be able to input a set of depos, pass it into a Wire-Cell data-flow graph with simulation components, receive the output corresponding raw ADC waveforms and hand them off to the framework for whatever dispatch is required.

3.2 Intermediate results

From time to time, users want to apply only the Wire-Cell drift simulation to depos and get back the results also as depos. For performance reasons, this typically must be done in batches of multiple depos. Often this intermediate result is computed as part of the full detector simulation. In this usage, the framework component that interfaces to Wire-Cell must be able to provide two types of data product back to the framework for further dispatch. This is one of a class of usages.

3.3 Combined simulation and signal processing

Relative to the full ADC waveforms, the data representing the input depo sets are relatively small in size. Likewise, the signals output by stage TPC1 are small. As an optimization, the SIM and TPC1 stages are best combined into a single Wire-Cell data flow graph in order to avoid realizing the relatively large volume of intermediate ADC waveforms. This pattern does not explicitly place unique requirements on the framework but except to allow for a monolithic SIM+TPC1 and an (obviously) separate DAQ and TPC1 stages.

3.4 Full far detector module simulation

DUNE FD modules pose a scaling problem to the detector simulation due to their size and large number of sub-detector elements (APA/CRP). The simulation itself operates on a per detector element basis while the depos are typically produced holistically across the detector module. The Wire-Cell data-flow graph for such a job would consists of a 150-way fan-out (using the number of APAs in DUNE FD horizontal drift module) with each finger of the fan processing a single APA. Wire-Cell provides support for an optimization mechanism called "sparse simulation" where only detector elements with "interesting" depos are simulated. However, to achieve such a graph requires large memory and multi-threading.

The framework can allow for optimization if it allows this necessary fan-out to be moved out of Wire-Cell and into file management layers. This would allow small-memory, single detector element jobs each executing on single-cores. For this to work the framework must, at least, allow the same input file holding depos to be given to 150 jobs, each targeting one APA. This would allow Wire-Cell to filter out depos from all but one APA. This filtering may also be considered a job for the framework (perhaps with a user component) or be pushed upstream in depo file preparation.

3.5 Multi source mixing

A common use case is prepare multiple sources of depos, for example from different physical processes, and to interleave them in time prior to simulation. This "mixing" can be done naturally and inherently via the Wire-Cell data-flow graph. For it to work, the framework must allow for the component interfacing with Wire-Cell to load zero or more depo set from its data store. This loading multiplicity will be dynamically varying as consumption rates will not in general match production rates. The variability can be absorbed by Wire-Cell at the cost of more than needed input buffer memory usage.

3.6 Streaming simulation

The memory usage required for simulation as well as typical detector readout window durations have so far limited simulations to on order 5ms or 10,000 ticks or less. DUNE must perform simulations of longer contiguous duration if it is to produce simulated data that is realistic to what is acquired by a supernova neutrino burst (SNB) candidate trigger decision. While the full 100s would be challenging, even 100ms requires a different, streaming simulation strategy.

Wire-Cell supports streamed simulation but for this to work the framework must be able to provide streams (of sets) of depos in time-order and must accept chunks of raw ADC waveforms. The relative time spans of consumed depos and produced waveforms will not in general coincide. Thus, framework must allow zero or more depos sets to be consumed and zero or more ADC waveform chunks to be produced each time it calls the framework component that interfaces with Wire-Cell.

4 TPC1 - first stage TPC processing

The TPC1 stage consumes all raw TPC ADC waveforms and produces all signal waveforms. It is a critical stage both in data volume, computational cost and downstream clients.

4.1 Per detector element processing

The memory requirements of TPC1 jobs and the typical batch allocation limitations imply that each job that makes up TPC1 should ingest only the portion of data from a given trigger record that was read from a single detector element (APA/CRP). This is an exact match to the content of a raw DAQ data file, which provides a per detector element stream. In the case that raw DAQ data files span multiple detector elements, the framework should be able to serve data to TPC1 from only a specific detector element.

4.2 Streaming signal processing

Like streaming simulation, the TPC1 stage must be able to consume a stream of raw ADC waveforms that are of larger duration than can be loaded entirely into memory and it must produce a stream of signals. This output must not include any artificial boundary effects due to arbitrary data chunking that is inevitably required.

A particular boundary condition is due to the mathematical nature of deconvolution (or convolution in general) whereby a sequence of size N is deconvolved with a kernel of size K must be performed using intermediate sequences at least as large as $N+K-1$. Any smaller and the periodic nature of the convolution will cause information to "wrap around" and thus produce artifacts near the edges of chunk size N. To avoid the artifacts methods such as *overlap-add* or *overlap-save* are

employed. They work by retaining the $K-1$ "extra" result from one chunk and applying it to the subsequent chunk.

To accommodate this, the framework must allow for different duration of input raw ADC waveform chunks and output signal waveform chunks to be produced/consumed each time the framework component that interfaces to Wire-Cell is called. Note, this similar to the use case for streaming simulation. Note, the framework need not deliver chunks exactly matching the operational chunk size N nor the kernel size K. Wire-Cell can perform buffering.

5 TPC2 splitting

As described above, data output by TPC1 represent a split-point as it is input to two branches of TPC2. The framework must be able to accommodate this splitting. Though, most likely this will be a user duty and accomplished by appropriate file handling and job configuration.

6 DET1 merging

6.1 Localized data

The framework must be able to input and synchronize different types of data. Here, this means the "cooked" PDS data must be synchronized with whole-module TPC cluster objects. Details of this synchronization may differ from the 2D-first and 3D-first branches. The data model must provide some "key" (likely time) in both the PDS and TPC objects that is in someway exposed to the framework to allow for synchronization. The synchronization must be performed in some time-windowed manner as both types of object will span some time duration. These spans are originally defined by the DAQ but may change through the processing of each PDS and TPC leg.

6.2 Extended data

This merging of PDS and TPC data must be performed for extended SBN readouts. This requires a streaming merge algorithm followed by chunked processing. This processing likely places the same kind of needs on the framework as streamed simulation and TPC1. Namely input and output chunk duration will not coincide.