DUNE Far Detector Framework Drivers

Brett Viren

June 6, 2023

Overview

Themes

- A view of DUNE far detector (FD) design and its data from offline framework eyes.
- The initial stages of offline processing.
- File-based branching and merging patterns.
- Software build/run/devel ecosystem.
- Effective exploitation of varied mixes of CPU/GPU resources.

DUNE far detector: four detector modules, two subsystems

Time projection chamber (TPC) sensitive to ionization electrons

TPC is source of the vast majority of the FD data.

- **DAQ per-module input data rate:** ≈ 1.5 TByte/s ≈ 50 EB/year.
- **Offline total “allowed” input data rate:** 30 PByte/year

As much as a 10^{-4} reduction in data volume is required.

Photon detection system (PDS) sensitive to scintillation light

Minor portion of data but **essential** for vertex t_0/x_0, trigger, reco, etc.

Functionally-identical **detector units (DU)** compose each detector module

DU examples: Horizontal Drift has 150 APAs, Vertical Drift has 160 CRPs.

DUNE DAQ Trigger Record (TR)

DUNE DAQ trigger and readout

- The DAQ **self-triggers** by considering **all** input data.
- Writes subset of data to file in form of **trigger records** (TR - nee "event").

Expected types **trigger records**

- **Localized**: $\mathcal{O}(5 \text{ GByte})$, $T \approx 3$ ms, $\lambda \approx 0.1$ Hz, may include a subset of DU.
- **Extended**: $\mathcal{O}(150 \text{ TByte})$, $T = 100$s, $\lambda \approx 1/$month, **supernovae** candidate, all DUs.

DAQ packed files *vs* Offline working memory sizes

- Need 2 working copies and 14bit $\rightarrow$ 32 bit $\Rightarrow$ $4.5\times$ inflation.

DAQ file model

Driver: target file sizes in range of 2-10 GB for efficient tape utilization.

Localized TRs: **monolith**

- Early running, whole-module readout: $\mathcal{O}(1)$ TR/file.
- Later, safely **decimate** TR, nullifying some DUs: $\mathcal{O}(10)$ TR/file.

Extended TRs: **sliced monolith**

- Time slice single 100 second TR into *eg* 2×10^4 files $\Rightarrow$ 5 ms, $\approx$ 7 GB per file.

Offline processing issues with monolithic file model

Processing of dense ADC arrays faces strong **memory pressures**.

- **ProtoDUNE-SP**: 6 APA $\times$ 3 ms difficulty hitting 2-4 GB/job.
  - DAQ FD **decimation** leaves $\approx$ 15 DU/TR average.
- One extended TR time slice is full $\approx$ 150 DU/TR, not tenable.

Something new is needed.

DAQ file loading strategies

Initial processing stages' algorithms require data from only a single DU!

Serial: iterate over the TR to load **one DU at a time**.

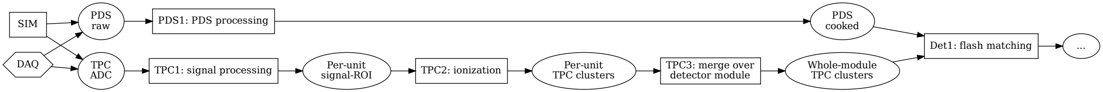
- Requires **lazy loading**, **eager saving** and **purging of transient** data.
- Requires framework **hierarchy extension**: run / subrun / event / **unit**
- If achieved, should allow for single-core (2GB) jobs.

Parallel: load **all DU in TR** and allocate large memory.

- Allocate **multiple CPU** cores to provide **sufficient memory**.
- Utilize **multi-thread** processing so as to not waste allocated CPU cores.
- DAQ decimation will lead to **variability in jobs size** over job lifetimes, must allocate for worse case.
- **SBND has demonstrated** this approach with art+larsoft+Wire-Cell jobs.

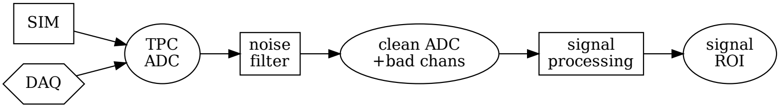
Initial DUNE processing stages

High-level conceptual processing chain:



- Each box is one batch processing campaign stage.
- Bubbles are file sets.
  - SIM may be combined into TPC1 (and/or PDS1) to if intermediate need not hit files.

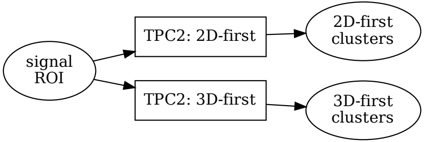
TPC1 processing stage one



RAM limits processing to $\mathcal{O}(5\ \mathrm{ms} \cdot \mathrm{DU})$

- **localized TRs** naturally satisfied, assuming serial/parallel loading strategy.
- **extended, sliced TRs** additionally require **duplicate** $\approx 100\ \mu s$ at slice boundaries.
  - Needed to avoid introducing FFT related artifacts.
  - Framework must allow a 2-slice buffer inside signal processing to be maintained.

TPC2 processing stage two

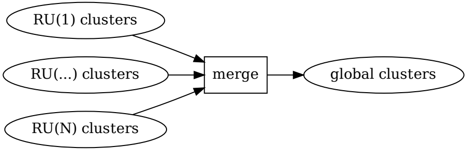


Must **branch** at the signal-ROI data tier to feed different **analysis strategies**.

- **2D-first** takes signal-ROI as $3 \times$ 2D views (Pandora, etc)
- **3D-first** performs computed tomographic imaging (Wire Cell)

No big framework issues, but downstream merge required.

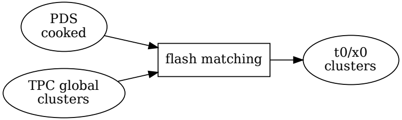
TPC3 processing stage three



The TPC3 stage merges clusters from a TR across all DU.

- Framework must transition from per-DU loading to **whole-TPC** processing.
- Extended TR: may need to merge both over DU and slices.
  - Or not: SNB interactions are small. Analysis groups need to decide.

DET1 processing - first whole detector module stage



Combine PDS + TPC for **flash matching**.

- Framework must allow synchronized reading of files with different types of data.
- Records may not share common I/O keys, eg due to previous **slicing**.

Software ecosystem issues

Problems related to binary/UPS-based ecosystem.

- Good support for some OSes, but not all OS used by DUNE collaborators.
- "Challenging" to build the stack on unsupported GNU/Linux OS.

Problems reported with hard-locked versions in software dependencies.

Can not "swap out GENIE versions easily without cutting a new LArSoft release".

Migration to Spack

- I expect the move to Spack will solve most of these kind of problems.

Personally, I **greatly appreciate** the effort put in to this important change.

DUNE production processing wants to use GPUs

Many spots in DUNE code can be accelerated with GPU. A sampling:

- Wire-Cell TPC sim ($18\times$) and signal processing (currently $2\times$)
- `GaussHitFinder` ($10\times$) and `EmTrackMichelId` ($14\times$)
- Deep neural network (DNN) **inference** (typical $\approx 20\times$)

Ignoring special **DNN training**, the processing is still **CPU dominated**.

Full jobs need 5-100 CPU / GPU

- In general, must share GPU with multiple threads/processes/boxes.
- Or accept idle GPU, verboten in some computer facilities.
- Sharing requires some form of **GPU task queuing and scheduling**
  - Else GPU idles and/or RAM overloads, jobs crash.

GPU queue schedulers in DUNE

- "GPU as a service" with nVidia Triton Server
- Wire-Cell/ZeroMQ distributed task offload
- Wire-Cell GPU algs with inter-thread semaphore

Issues shared by current queue schedulers

- All s/w in a job must share a common interface.
  - Circumventing the will lead to GPU overload / out of memory errors.
  - Same time, must reject code lock-in, keep CPU/GPU portability.
  - Need an **insulating mini-framework**.
- Normal usage pattern leads to idle CPU cores
  - Thread/core launches GPU task, waits for response.
  - Queue depth fluctuates high, many CPU cores go idle. A very dynamic problem.
  - Some mitigation possible (CMSSW “external” and TBB `tbb::flow::async_node`)
  - Requires framework, toolkit, user code adherence.
    - Need **independent mini-framework** used in different code contexts/packages.

Cross-project task group?

Summary

DUNE far detector drives various framework issues including at least:

- file loading at different scales and managing monolithic TRs.
- file-level branch/merge patterns,
- software ecosystem portability/flexibility and
- GPU support given variety of software and hardware deployments.

$\mathcal{FIN}$

backups

Monolithic *vs* striped DAQ files

Current file model: localized **monolithic** and extended, sliced monolithic.

However, it is possible for DAQ to **stripe** a TR across per-DU files.

- Striped files will simplify requirements on the framework.
- Allow simple, low-memory, single-core jobs.
- Forget hierarchy extension, special DU-iteration, lazy loading, eager write, data purging.
- The extended, TR **slices** are longer (eg 1 s) but then further segmented into **chunks** on reading.
- But, some additional file bookkeeping and data aggregation issues to solve.

We should study if a **striped model** is preferred over monolithic.