DUNE Perspective on Future LArSoft

Tom Junk
LArSoft Workshop
25 June 2019
ArgonCube: Pixel-based LArTPC, unmagnetized (150 Tons)
MPD: "Multi-Purpose Detector": High-Pressure Gas TPC, solenoid, ECAL, muon stack
3DST-S Plastic scintillator, gas TPC, magnet, and ECAL
DUNE ND Prism Hall

MPD and ArgonCube plan to move up to 35m off axis.

3DST-S stays on axis in an alcove
An Event in ArgonCube

J. Sinclair

K. Terao is the ArgonCube software coordinator

1 MW 3 horn optimised spill, FHC, including rock. 4x5 geometry. Colouring by nu.
Simulation Chain

Event Generator → Tracking (Geant4) → Drift/Photon Propagation → TPC Readout

TPC Readout

Digitized waveform for electronics response
- “Vectorized code” written by Dan Dwyer (LBNL), ready(?) to be integrated with drift/E-field simulation.
- Seek for ways to integrate vectorization scheme made available(?) in LArSoft

Photon Propagation

Individual photon ray tracing is time consuming. Study a solution using a photon library for ArgonCUBE 2x2 prototype.
- LArSoft: a “photon library” (look-up table) for photon collection efficiency and timing estimates at different detector locations.
- Need to run a full simulation to build the library
Simulation Chain

Event Generator → Tracking (Geant4) → Drift/Photon Propagation → Detector Geometry → TPC Readout → Optical Readout

Pixel Geometry in LArSoft

LArSoft assumes “wire” in the core of Geometry design & APIs to query geometry information (PlaneGeo/WireGeo). Needs to be changed.

- Designed a generic “charge-sensitive element” to replace the current implementation in non-distruptive manner (by Gianluca @ SLAC)
- Now in testing stage: goal is to run largeant for wire & pixel geometry
Simulation
1. Debug the pixel geometry implementation
   ○ **Goal**: run large ant for wire & pixel geometry

2. Generate photon library
   ○ **Goal**: photon library within TPC active volume

3. Implement E-field response into LArSoft
   ○ **Goal**: run drift simulation for wire & pixel geometry

4. Implement pixel readout response into LArSoft
   ○ **Goal**: run the whole readout chain for pixel (no wire)

5. Keep working till we are happy
MPD ECAL weighs 300 tons + 100 tons for the magnet.
1 Ton of GAr

Re-Use ALICE Readout Chambers add ECAL and Muon Detector
MPD: Reconstruction: GArSoft

Implemented

- Event Generation
- Detector Geometry
- Particle Interactions & Energy Deposits
- Drift and Diffusion
- Digitization
- Hit finding and clustering
- Pattern recognition
- Track fitting
- ECAL Digitization
- ECAL Reconstruction
- Ionization-Based Particle ID
  - Initial version exists – needs work

To do (to some degree optimization)

- TPC Field Response and Electronics Response
- Optimize pattern recognition in difficult cases
- Optimize track fit
- Very short tracks in crowded environments will require innovative algorithms
  - Deep learning methods being studied now
- Vertexing
  - Preliminary vertex-finding algorithm written and tested
- ECAL
  - Cluster-Track matching
  - Full energy reconstruction (only visible energy for now)
  - …..

DUNE-Doc 13933
A Simulated and Reconstructed $\nu_e$ Charged Current Event in the HPGTPC

$\nu_e + \text{Ar} \rightarrow e^- + \pi^+ + p + n$

Neutron with $p = 0.23$ GeV/c at the P.V. not shown
The dunetpc Dependency Tree (v08_18_00)
The GArSoft Dependency Tree (depends on art, nutools)
Near Detector Integration Thoughts

• Running GArSoft and LArSoft modules in the same job "should" be possible
  – Both are based on the art framework
  – art loads modules dynamically based on FHiCL configuration
  – Data products for GArSoft have names in the gar namespace. e.g. gar::raw::RawDigit, so as not to collide or be confused with raw::RawDigit in LArSoft

• But there is some work to do to keep it all together
  – Dependency trees have to match. Must use same version of art for example. "A tree with two trunks"
  – GArSoft is updated to art V3. LArSoft has followed a few point releases since then but they involve few breaking changes.
Integration: Easy Issues First

Running detector-specific simulation and reconstruction are all independent pieces – modules work on independent data.

- channel response
- data output from sim job and readin in reco job
- noise filtering
- deconvolution
- TPC clusters and hit-finding
- tracking
- shower reco
- calorimetry

• Some modules and services may duplicate names with those in LArSoft. Can fix those easily.
Integration: Harder Issues

• Unified GEANT4 simulation
  - Current modules: LArG4 and GArG4. Consume MCTruth data products, make sim::SimChannel and energy deposits
  - particles produced in LAr -> GAr -- one can imagine running LArG4 first and then piping particles that come out of the LAr as MCTruth for GArG4, which gets run second.
  - Particles produced in GAr traveling back into LAr. Our CDR-Lite Executive summary mentions that backwards-going cosmic rays are an important calibration source for the LAr
    • Either need to iterate this, or run a unified GEANT4 step

• Unifying the GEANT4 step means having a single geometry description GDML file (or files), and calling GEANT4 once to follow particles back and forth.

• Hans Wenzel's new Energy Deposits in LArG4 look a lot like Brian Rebel's solution in GArG4.

• Data products have different names but that's okay
Integration: Event Display

• The three-detector ND Complex will have particles exiting one detector and possibly going into the other two.

• Visualizing the events will be useful in developing (traditional) reconstruction and track-matching algorithms

• Currently we are working independently

• How does MINERvA/MINOS deal with this?
Integration with 3DST-S

- Less understood on the MPD software side how the 3DST-S would fit in.
- GEANT4 step needs to be unified with ArgonCube and MPD for reasons explained before
- Off-axis positions are interesting – five combined geometry descriptions may be necessary, since 3DST-S does not move off axis.
- 3DST-S has gas TPC components. May want to re-use GArSoft algorithms, as they are intended to be homogeneous and isotropic. GArSoft assumes pixel readout however.
The 3DST Spectrometer (3DST-S)

- B-field = 0.6 T
- 0.5 m depth for both TPC and ECAL
- TPC:
  - space-point resolution <0.5 mm
  - 5% p resolution @3 GeV/c

2018 *JINST* **13** P02006

- Muon detection efficiency >90% at $4\pi$
- Muon p resolution by range ~2-3%
- Detect protons above ~300 MeV/c
- Very good neutron detection capability

T2K Near Detector will be upgraded with 2 tons 3DST-like detector and TPC
Definition of a Near Detector "Event"

- *art* handles events as the smallest bit of independent data.
- We associate these with triggered detector readouts.
- The entire complex will want to share a single trigger:
  - beam spill signal from LBNF
  - Random triggers for background constraints
- We will also want to partition the DAQ for commissioning and tests.
Supported Detectors in dunetpc

- 35-ton: Support is thin. Pandora stopped supporting 35-ton about a year ago. lbne_raw_data (DAQ interface) should be removed at some point. Data preservation?
- DUNE FD SP 10 kt
- DUNE FD SP 1x2x6 Workspace
- DUNE FD DP 10 kt
- ProtoDUNE-SP 6x6x6 meters cubed (+DAQ)
- ProtoDUNE-DP 6x6x6 meters cubed
- WA105 3x1x1 dual-phase prototype (+DAQ)
- ICEBERG (+DAQ)
- Coming: (?) ArgonCube ND. 2x2 ArgonCube Prototype in the NuMI hall near MINOS ND. (+DAQ)
# Timing of a 7.5 ms ProtoDUNE-SP Reco Job

/pnfs/dune/scratch/dunepro/beam_prep/logs/protodune-sp_sce_sample_keepup_7.5ms_v07_08_00_05_snapshot_id_192222_slice_0_stage_500_17066634_0.out

<table>
<thead>
<tr>
<th>TimeTracker printout (sec)</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>Median</th>
<th>RMS</th>
<th>nEvts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full event</td>
<td>1402.6</td>
<td>2155.65</td>
<td>3403.72</td>
<td>2100.68</td>
<td>435.928</td>
<td>49</td>
</tr>
<tr>
<td>source:RootInput(read)</td>
<td>0.000543632</td>
<td>0.00688822</td>
<td>0.111605</td>
<td>0.00189346</td>
<td>0.0191799</td>
<td>49</td>
</tr>
<tr>
<td>decode:timingrawdecoder:TimingRawDecoder</td>
<td>0.00046275</td>
<td>0.0137007</td>
<td>0.178662</td>
<td>0.0115812</td>
<td>0.0252344</td>
<td>49</td>
</tr>
<tr>
<td>decode:ssprawdecoder:SSPRawDecoder</td>
<td>0.233429</td>
<td>0.324645</td>
<td>0.806665</td>
<td>0.314574</td>
<td>0.0825951</td>
<td>49</td>
</tr>
<tr>
<td>decode:tpcrawdecoder:PDSPTPCRawDecoder</td>
<td>24.8511</td>
<td>29.8284</td>
<td>33.0416</td>
<td>30.0935</td>
<td>2.21615</td>
<td>49</td>
</tr>
<tr>
<td>decode:ctbrawdecoder:PDSPCTBRawDecoder</td>
<td>0.000214046</td>
<td>0.00367091</td>
<td>0.0828906</td>
<td>0.000400356</td>
<td>0.0124469</td>
<td>49</td>
</tr>
<tr>
<td>decode:beamevent:BeamEvent</td>
<td>0.000726402</td>
<td>0.00678737</td>
<td>0.257794</td>
<td>0.00117761</td>
<td>0.036297</td>
<td>49</td>
</tr>
<tr>
<td>decode:caldata:DataPrepModule</td>
<td>75.3518</td>
<td>104.001</td>
<td>135.345</td>
<td>103.056</td>
<td>12.6566</td>
<td>49</td>
</tr>
<tr>
<td>decode:gaushit:GausHitFinder</td>
<td>34.2085</td>
<td>52.2964</td>
<td>70.2773</td>
<td>52.8011</td>
<td>7.57757</td>
<td>49</td>
</tr>
<tr>
<td>decode:hitpdune:DissambigFromSpacePoints</td>
<td>50.9646</td>
<td>101.859</td>
<td>156.355</td>
<td>99.9169</td>
<td>25.2145</td>
<td>49</td>
</tr>
<tr>
<td>decode:reco3d:SpacePointSolver</td>
<td>24.8511</td>
<td>29.8284</td>
<td>33.0416</td>
<td>30.0935</td>
<td>2.21615</td>
<td>49</td>
</tr>
<tr>
<td>decode:hitpdune:DissambigFromSpacePoints</td>
<td>50.9646</td>
<td>101.859</td>
<td>156.355</td>
<td>99.9169</td>
<td>25.2145</td>
<td>49</td>
</tr>
<tr>
<td>decode:pandora:StandardPandora</td>
<td>744.267</td>
<td>1274.57</td>
<td>2191.16</td>
<td>1228.64</td>
<td>306.203</td>
<td>49</td>
</tr>
<tr>
<td>decode:pandoraTrack:LArPandoraTrackCreation</td>
<td>57.3564</td>
<td>115.366</td>
<td>171.417</td>
<td>112.522</td>
<td>25.1478</td>
<td>49</td>
</tr>
<tr>
<td>decode:pandoraShower:LArPandoraShowerCreation</td>
<td>45.2669</td>
<td>92.431</td>
<td>152.72</td>
<td>91.4082</td>
<td>23.764</td>
<td>49</td>
</tr>
<tr>
<td>decode:pandoraTrack:LArPandoraTrackCreation</td>
<td>57.3564</td>
<td>115.366</td>
<td>171.417</td>
<td>112.522</td>
<td>25.1478</td>
<td>49</td>
</tr>
<tr>
<td>decode:pandoraShower:LArPandoraShowerCreation</td>
<td>45.2669</td>
<td>92.431</td>
<td>152.72</td>
<td>91.4082</td>
<td>23.764</td>
<td>49</td>
</tr>
<tr>
<td>decode:pmtrackcalo:Calorimetry</td>
<td>13.9568</td>
<td>29.1075</td>
<td>44.7453</td>
<td>28.942</td>
<td>7.08136</td>
<td>49</td>
</tr>
<tr>
<td>decode:pmtrackpid:Chi2ParticleID</td>
<td>0.0072007</td>
<td>0.0137094</td>
<td>0.0312194</td>
<td>0.0134894</td>
<td>0.0043193</td>
<td>49</td>
</tr>
<tr>
<td>decode:opflashInternal:OpFlashFinder</td>
<td>0.00782905</td>
<td>0.0135512</td>
<td>0.0184147</td>
<td>0.0138113</td>
<td>0.00228544</td>
<td>49</td>
</tr>
<tr>
<td>decode:opflashExternal:OpFlashFinder</td>
<td>0.00328762</td>
<td>0.00421628</td>
<td>0.00570462</td>
<td>0.00440462</td>
<td>0.000656042</td>
<td>49</td>
</tr>
<tr>
<td>decode:TriggerResults:TriggerResultInserter</td>
<td>2.3567e-05</td>
<td>3.7177e-05</td>
<td>7.0934e-05</td>
<td>3.7005e-05</td>
<td>1.15572e-05</td>
<td>49</td>
</tr>
<tr>
<td>end_path:out1:RootOutput(write)</td>
<td>21.5251</td>
<td>26.9227</td>
<td>45.9711</td>
<td>26.1481</td>
<td>4.15128</td>
<td>49</td>
</tr>
</tbody>
</table>
Scaling Resources to the Far Detector

- Pandora needs the most CPU in the 7.5 ms readout window ProtoDUNE-SP event
- But its CPU scales nonlinearly with the activity in the event
- Far Detector data will mostly be empty
- Need to run Data Prep on a 25—bigger detector
- Data unpacking and preparation scale linearly with data size (= detector size x nticks)  FFTs scale a bit faster with nticks, but nchannels is the big scale factor here.

- Supernova burst: 30 seconds or more of non-zero-suppressed waveform readout
- All four detector modules processed separately (10 kt each)
ProtoDUNE-SP Raw Decoder tool that unpacks one APA at a time now implemented. Calls removeCachedProduct for the input artdaq::Fragments
Need to work on MC – break it into smaller pieces
Data Prep and Wire Cell may have to be serialized so that we are sure that everyone who needs raw digits for APA n is done and the raw digit remover can run. ART does not currently support removing produced cached products however, only ones input from files. -- may be an easy upgrade to allow produced cached products.

Threading processing of pieces of events is better than requiring multiple events to be in memory at a time. Serial processing of APA's may be as efficient as parallel, perhaps more so.

Intra-event threading requires shared modules (replicated modules in art3 are made per schedule)

We care more about throughput than latency, though the Event Display has latency issues.
New DAQ Format Ideas

- DUNE DAQ Consortium is exploring ideas to not use art-formatted rootfiles as output from artDAQ.
- It is difficult to reshape art events after they are written. Possible, but difficult (LArIAT and 35t do this)
- It is difficult in art to process less than an event.
- It is impossible in ROOT to read in less than one entry on a branch (or leaf). (solution: just make more branches).
- Another solution: a file per APA – 150 Files per event.
  - The filesystem becomes part of the event builder.
- We'd like to keep them together. tar or something like it can keep the files on the same tape.
External Source Code (e.g. GPL3)

• Question on the DUNE Slack #larsoft-beginners channel:

What is the collaboration/LArSOFT/Art's position on copying from open source libraries? Specifically, I want to use a function released under LGPL 3.0 that I have modified, but left all the original inline documentation and author information, as well as my name and modification date

• Jeremy Hewes's request for central management of HighFive, a header-only convenience interface to HDF5. Lynn says DUNE would have to maintain it.
External Source Code Concerns

• What-if:
  – Original developers abandon project. We're left maintaining it
  – Developers take project in a new and interesting direction, leaving us behind or incompatible

• Concern if we modify external source
  – what if someone wants original, unmodified behavior
  – new names

• If code breaks, we can decide to maintain it or disable it. Simple updates are okay. Breakage is harder

• Tutorials?

• There's a lot of open-source code out there. Do we have to maintain every piece a DUNE collaborator wants to use?
GPU-Enabled Code

• We would like to make better use of GPU's of course
  – machine learning training (already did. Robert and Dorota's track/shower discriminator was trained using GPU's)
  – machine learning discriminant calculation (¿ less CPU intensive than training presumably)
  – Data prep
  – Event Display

• Development platform and examples would be welcome.
  – gpvms?
  – desktops and laptops?
  – Wilson Cluster?
Operating System Support

- We support
  - SL6
  - SL7
  - macOS 10.13 "High Sierra"
  - macOS 10.14 "Mojave"

- Many people currently use SL7. We just got a new build node, dunebuild02.fnal.gov, which runs SLF7.

- Probably not too disruptive to move away from SL6, though we do have a TDR to finish writing.

- I am also okay with containers replacing flavor support. Supporting one flavor and providing containers sounds good to me.
dunetpc Is Getting Too Big

- Slow git clone. Several minutes, even at FNAL. Lots of old history (GDML files came and went).
- build takes ~1 hour of CPU and around 3 GB of storage. On a build node, it's not so bad.
- Split into repositories and UPS products?
- Need a different build and release strategy (a la LArSoft and MicroBooNE)
- dunepdsprce source tarball? Not currently. We download from github
- We had been waiting to see how Spack(Dev) affects this model.
- At the LArSoft coordination meeting June 4, we found that Spack(Dev) will target replacing mrb and ups – we still need to build a repository at a time.
- Will need some refactoring of code to break this up.

- Pull requests a la LArSoft with GitHub as the service provider
Extras