SPNG Full TDM Chain Update 2

Brett Viren

December 19, 2025

Topics

- TDM'ified DNNROI working
- Flexible configuration.
- Job comprehension (but still just PDHD)
- Todo list

TDM DNNROI in SPNG

Mystery from last time solved.

- OSP gives MP2/MP3 “boolean” values 0 and 4000.
- `DNNROIFinding` scales MP2/MP3 and “dense” images down by 4000.
- SPNG (now) keeps MP2/MP3 truly boolean 0,1 and only scales down “dense”.

Now get reasonable output from full SPNG

- Still need to remove “negative signal” due to noise ROIs like OSP does.
- Some tuning may be needed for SPNG’s “rebaseline” algorithm.

Flexible configuration

All lives under `spng/cfg/spng/` following a hierarchical design.

- `detector.jsonnet` abstract API for defining a detector
- `detconfs/pdhd.jsonnet` a concrete PDHD detector
- `{drift,detsim,deposplat,decon,crossviews,roi,drift,dnnroi}.js` define functionality subgraphs
- `frame.jsonnet` related to interfacing with `IFrame`
- `tpc.jsonnet` defines per-TPC aggregations
- `det.jsonnet` defines whole detector aggregations
- `{control,fans,io,torchio,util}.jsonnet` infrastructure
- `test-*.jsonnet` many tests (18 so far). some serving as user CLI

User CLI

The most flexible main Jsonnet so far: `test-det.jsonnet`, with TLAs for:

- `input` file name
- `output` file name pattern, eg `"test-det-%(tier)s-tpc%(tpcid)d.npz"`
- `input_type` when not default `depo` to use, eg `frame_array`
- `job` name the core (non I/O) subgraph, eg `depos_to_adc`
- `detname` just default `pdhd` but eventually more
- `engine` default to `Pgrapher` or give `TbbFlow`
- `device` default `cpu` or give `gpu` or `gpuN` for specific GPU number N
- `verbosity` for common `Logger` class for data-aware logging.

Ways to use the config

- `wcsonnet` for testing syntax.
- `wcpy pgraph dotify` for generating graph viz in PDF
- `wire-cell` of course for running actual job
- art/LArSoft/FHiCL is t.b.d. (I'll need help here)

Examples

Command prefix:

```
wire-cell spng/cfg/spng/test-det.jsonnet -l stderr -L debug
```

Options to input depos or frames:

```
-A input=depos.npz
-A input=frames.npz -A input_type=frame_array
```

Options to pick a `job`:

```
-A job=JOBNAME
```

Options to select TPC and device

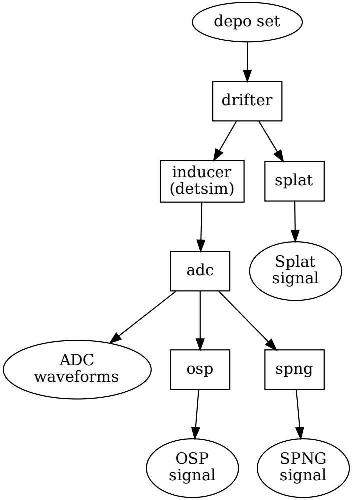
```
--tla-code 'tpcids=[0]'  -A device=gpu
```

A job is a subgraph of a possible whole graph

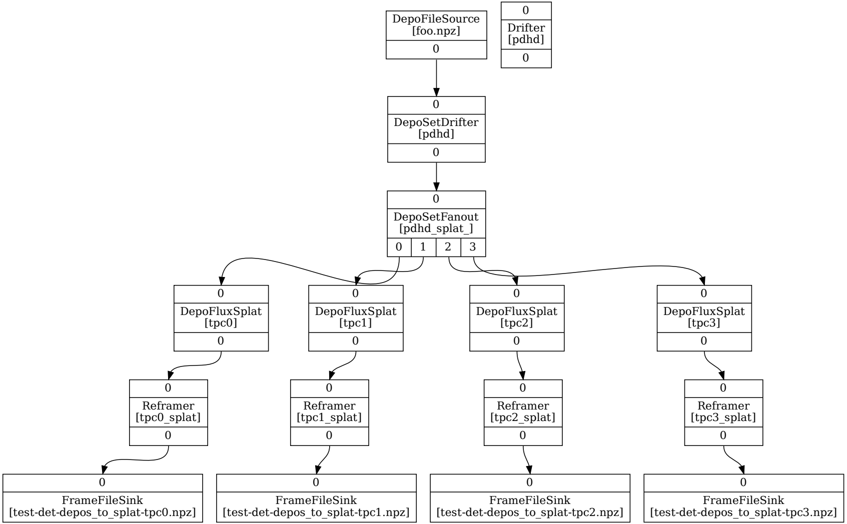
- **units**: `drifter`, `splatter`, `inducer` (detsim), `osp`, `spng`
- **full**: `depos_to_splat`, `depos_to_adc`, `depos_to_osp`, `depos_to_spng`
- **union**: `kitchen_sink`

Current constraints:

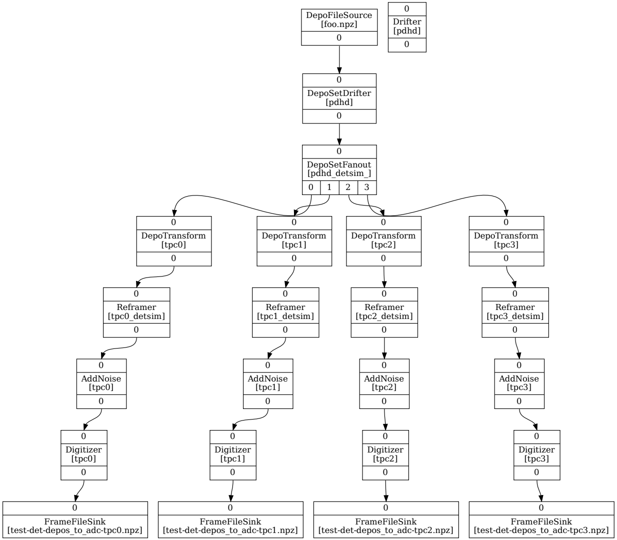
- **Depo set input** implies whole-PDHD job
- **ADC waveform input** implies per-APA job
- **ADC/signal output** to per-APA file.



PDHD true signals: *depos* → *splat* (7 CPU-seconds)



PDHD ADC waveforms: *depos* → *adc* (30 CPU-seconds)



DepoFileSource [foo.npz]

Drifter [pdhd]

DepoSetDrifter [pdhd]

DepoSetFanout [pdhd_detsim_]

0 1 2 3

DepoTransform [tpc0]

DepoTransform [tpc1]

DepoTransform [tpc2]

DepoTransform [tpc3]

Reframer [tpc0_detsim]

Reframer [tpc1_detsim]

Reframer [tpc2_detsim]

Reframer [tpc3_detsim]

AddNoise [tpc0]

AddNoise [tpc1]

AddNoise [tpc2]

AddNoise [tpc3]

Digitizer [tpc0]

Digitizer [tpc1]

Digitizer [tpc2]

Digitizer [tpc3]

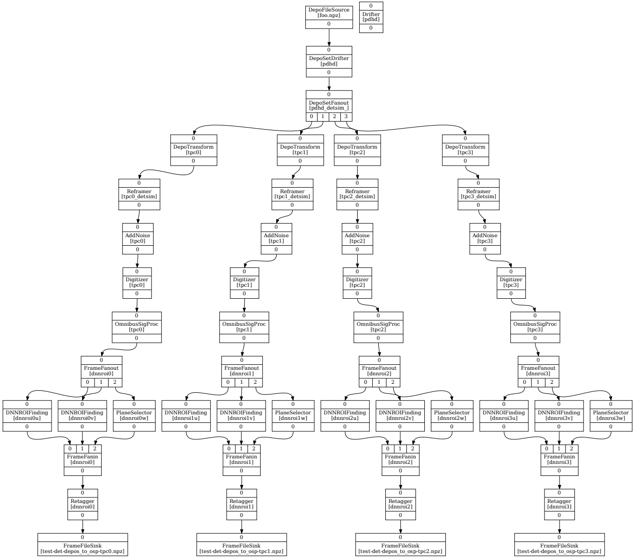
FrameFileSink [test-det-depos_to_adc-tpc0.npz]

FrameFileSink [test-det-depos_to_adc-tpc1.npz]

FrameFileSink [test-det-depos_to_adc-tpc2.npz]

FrameFileSink [test-det-depos_to_adc-tpc3.npz]

PDHD OSP signals: *depos* → *adc* → *osp* (70 CPU-seconds)



DepoFileSource [foo.npz]

Drifter [pdhd]

DepoSetDrifter [pdhd]

DepoSetFanout [pdhd_detsim_]

DepoTransform [tpc0]

DepoTransform [tpc1]

DepoTransform [tpc2]

DepoTransform [tpc3]

Reframer [tpc0_detsim]

Reframer [tpc1_detsim]

Reframer [tpc2_detsim]

Reframer [tpc3_detsim]

AddNoise [tpc0]

AddNoise [tpc1]

AddNoise [tpc2]

AddNoise [tpc3]

Digitizer [tpc0]

Digitizer [tpc1]

Digitizer [tpc2]

Digitizer [tpc3]

OmnibusSigProc [tpc0]

OmnibusSigProc [tpc1]

OmnibusSigProc [tpc2]

OmnibusSigProc [tpc3]

FrameFanout [dnnroi0]

FrameFanout [dnnroi1]

FrameFanout [dnnroi2]

FrameFanout [dnnroi3]

DNNROIFinding [dnnroi0u]

DNNROIFinding [dnnroi0v]

PlaneSelector [dnnroi0w]

DNNROIFinding [dnnroi1u]

DNNROIFinding [dnnroi1v]

PlaneSelector [dnnroi1w]

DNNROIFinding [dnnroi2u]

DNNROIFinding [dnnroi2v]

PlaneSelector [dnnroi2w]

DNNROIFinding [dnnroi3u]

DNNROIFinding [dnnroi3v]

PlaneSelector [dnnroi3w]

FrameFanin [dnnroi0]

FrameFanin [dnnroi1]

FrameFanin [dnnroi2]

FrameFanin [dnnroi3]

Retagger [dnnroi0]

Retagger [dnnroi1]

Retagger [dnnroi2]

Retagger [dnnroi3]

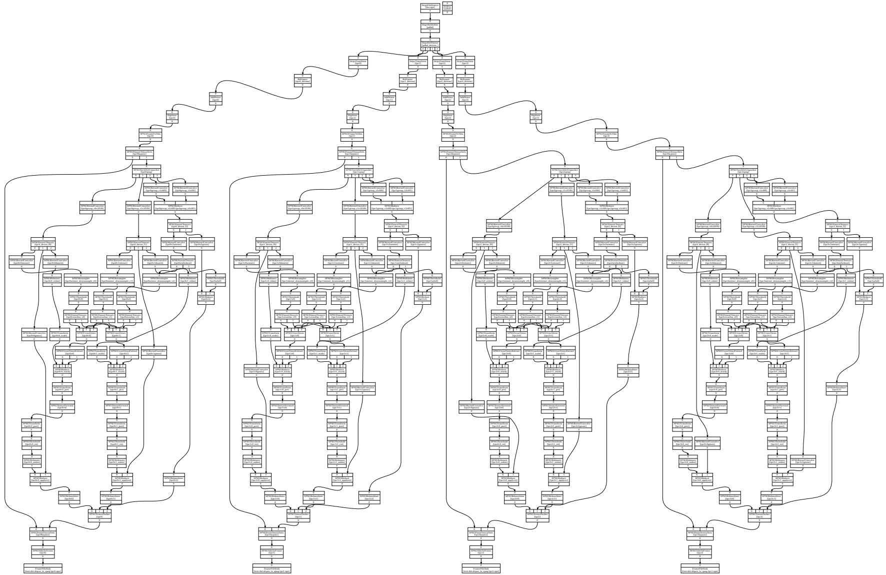
FrameFileSink [test-det-depos_to_osp-tpc0.npz]

FrameFileSink [test-det-depos_to_osp-tpc1.npz]

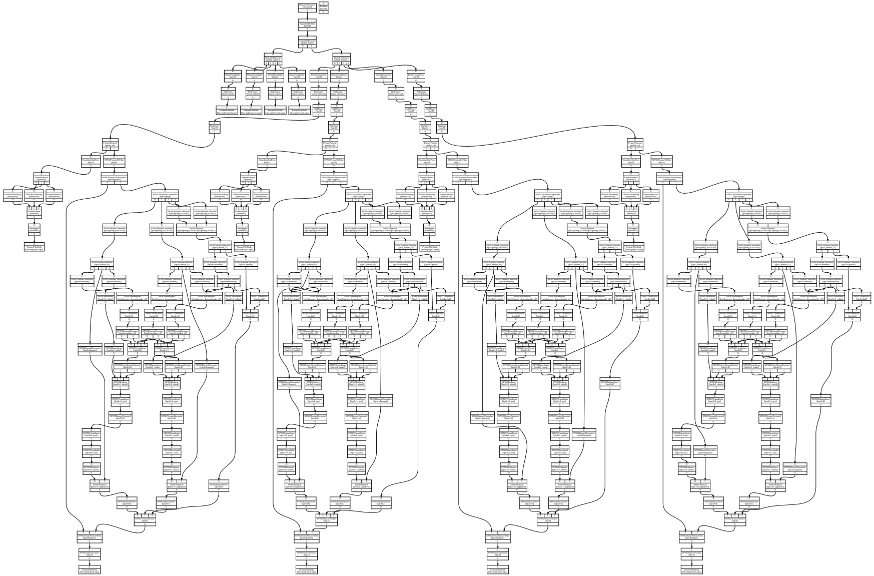
FrameFileSink [test-det-depos_to_osp-tpc2.npz]

FrameFileSink [test-det-depos_to_osp-tpc3.npz]

PDHD SPNG signals: *depos* → *adc* → *spng* (51 CPU-s, 13s with GPU)*



PDHD Kitchen sink: $depos \rightarrow \{splat, sim \rightarrow \{osp, spng\}\}$



Along the way: WCT config speed improvement

Found that full SPNG Jsonnet config takes **6 seconds** just to compile!

- Actually worse as Jsonnet is MT'ed and that is **wall time**.
- Aside: this with WCT's `wcsonnet`, oddly bare (Go) `jsonnet`, is even slower.

Discovered an $\mathcal{O}(N^2)$ scaling bug in core graph construction Jsonnet code.

- `unique_list()` was the culprit.
- Replace with better $\mathcal{O}(N \log N)$ algorithm.
- Also learned some new Jsonnet tricks
  - Liberal use of `local` to causes parser to cache values.
- This gives about $10\times$ Jsonnet compilation speed up.

This should have been a problem for mainline WCT, maybe people just accept the slowness?

- Though, SPNG graphs are **big** so maybe not surprising we see scaling bugs.
- Still, mainline will benefit once we **merge the spng** branch.

Merging - aside

Speaking of merging, at some point “soonish” I want to take a pause and:

1. Merge WCT **master** back into **spng** branch.
2. Ask Haiwang to merge **spng** into **master**.

We should think how best to schedule this w.r.t. doing some “real” processing.

- Ideally, we merge and make a real, full WCT release prior to batch tests.

Next up (continued from last week)

- ☒ Validate post-CrossView outputs.
- ⊟ Configuration:
  - ⊟ Output `IFrame` from SPNG graph
    - ☒ SPNG graph wrapped by `IFrame`
    - □ **(new)** Some bugs to work out for `TdmToFrame`
  - ☒ Make sim and SPNG test files to be “modular”.
  - ☒ A “depo flux splat” subgraph (“true signals”)
  - ☒ An OSP subgraph
  - ☒ Individual and combined main config files.
- □ More rigorous profiling, performance eval.
- □ Revisit run-time story (eg containers, CVMFS).
- □ Incorporate depo files from Jake and “SPDIR” depos.
- □ Continue DNNROI++ ideas (MPn → CrossViews and Jake’s Multi-UNet)
- □ **(new)** Speed up CrossViews+MP2/MP3 with new ideas from Jake + me.
- □ … What else?