Wire-Cell Toolkit
Sim and Graph Updates
Config and NF Integration

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

bnl-ub – 2018-03-23

Outline

New Stuff: Sim and Graph

Configuration

NF Integration Issue

New Stuff: Sim and Graph

Configuration

NF Integration Issue

New WCT Package: `pgraph`

The "P" could mean "pipe" or "process".

- Provides a **single-threaded** implementation of WCT's directed graph execution interface.
- Uses an invented "ASAP" graph execution model.
  - → Always execute the "deepest" graph nodes possible.
    - Minimizes amount of "in flight" data (reduce memory usage).
    - Note: **exactly wrong strategy** if multi-threaded:
      - → see I. Shapoval, thesis on GaudiHive
- Finally, we can configure an **arbitrary graph of components**
  - A new, generic "app" component: `Pgrapher` executes the graph.
  - The `Fourdee` app is still left in the `gen` package.
  - Hard-coded `Fourdee` C++ structure now can be expressed in configuration with help of a few new components.

New and Newish Components

In `gen`:

DepoMerger merge two streams of deposition into one time-ordered stream (was hard-code C++ in `Fourdee`).

FrameSummer add frames from two streams together (also hard-coded in `Fourdee` C++).

BlipSource a point-like deposition source of low, fixed energy or ^{39}Ar spectrum, uniform vertices in a box.

TrackDespos generate ideal, straight-line tracks in various ways (not new).

MultiDuctor a rule-based switchyard managing other Ductors (newish).

In `sio`:

NumpyDepoSaver save depos to Numpy array files.

NumpyFrameSaver save frames to Numpy array files.

In `pgraph`:

Pgrapher WCT app executing the graph

New Stuff: Sim and Graph

Configuration

NF Integration Issue

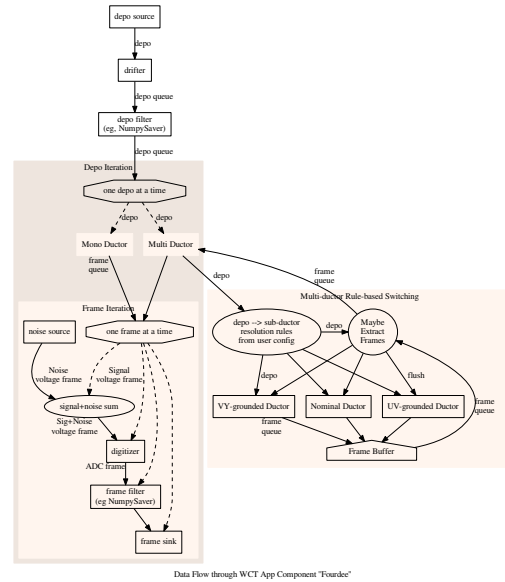
Pgrapher Configuration Jsonnet file

```
local base_params = {
  lar: {...}, detector: {...}, daq: {...}, adc: {...}, elec{...},
  sim: {noise: true, ...},
};
local uboone_params = base_params {...}; // also one for DUNE
local params = uboone_params;

local cosmics = {type: "TrackDepos", name: "cosmics", data: {...}};
local beam = {type: "TrackDepos", name: "beam", data: {...}};

local joincb = { type: "DepoMerger", name: "CosmicBeamJoiner" };
// ... more nodes defined

local app = { type:"Pgrapher", data: { edges: [
  {tail: {node: wc.tn(cosmics)}, head: {node: wc.tn(joincb), port:0}},
  {tail: {node: wc.tn(beam)}, head: {node: wc.tn(joincb), port:1}},
  // ...
]}};
[cosmics, beam, joincb, app, ...]  // resulting configuration sequence
```

Noise+Signal and Just Signal

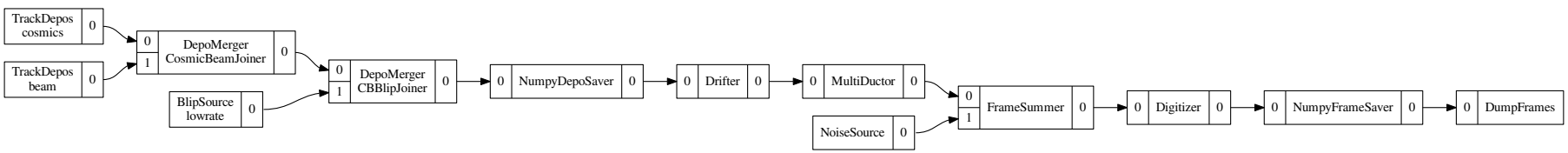
Can switch noise on/off by drawing a different graph.

- Currently do it by editing the Jsonnet file.
- Switch could use Jsonnet's "external variable" feature.



hard-coded Fourdee structure

signal + noise



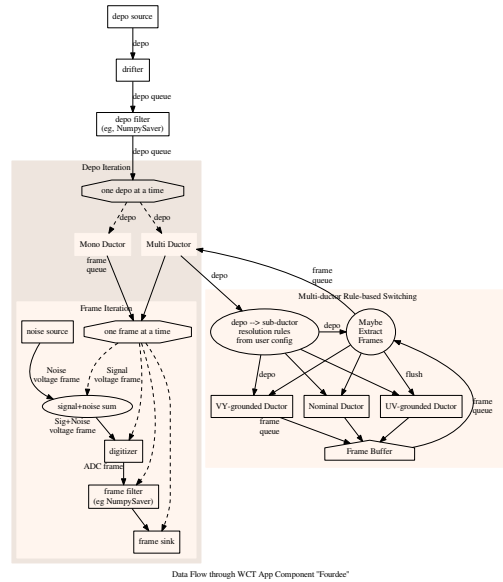
(generated directly from config)

No longer need to develop new C++ "app" components just to change graph structure!

Noise+Signal and Just Signal

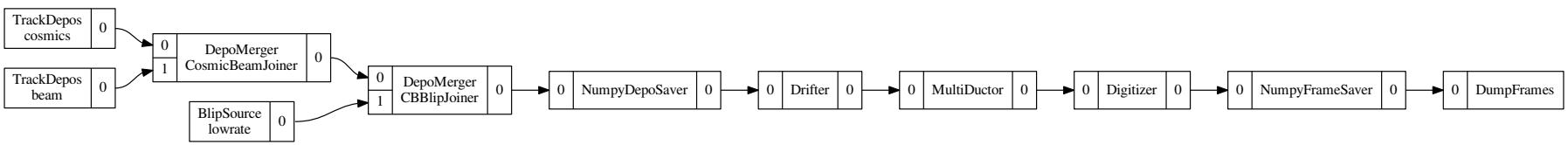
Can switch noise on/off by drawing a different graph.

- Currently do it by editing the Jsonnet file.
- Switch could use Jsonnet's "external variable" feature.



hard-coded Fourdee structure

just signal



(generated directly from config)

No longer need to develop new C++ "app" components just to change graph structure!

Jsonnet External Variables

- Jsonnet has function to "inject" an external variable.
- Can be used to implement "user visible" switches.
  - Currently we use it to switch "configuration epochs" (before/after hardware fix) when compiling the *.json.bz2* config files.

```
local hwnf_epoch = std.extVar("hwnf_epoch");
if hwnf_epoch == "before" then ...
```

- Inject from `wire-cell`, `jsonnet` command line programs:

```
$ jsonnet -V hwnf_epoch=before ...
$ wire-cell -V hwnf_epoch=before ...
```

- Inject **from** *art* FHiCL but no way to inject **to** FHiCL.

New Stuff: Sim and Graph

Configuration

NF Integration Issue

Configuration Epochs

WCT "channel noise DB" needs different configs (for "RMS cut") in pre- and post-hardware fix "epochs".

- Currently provide 2 pairs of WCT JSON and *art* FHiCL config files.
- M. Kirby and Herb say two configs will lead to mistakes, want us to fix.
- No DB support, Mike says use hard-coded C++ switch on run number.
- But, WCT says, "configure exactly once before data is processed!"

Suggested Fix

1. Develop new *facacde* `ChannelNoiseDB` inside `larwirecell`
   - It manages N "real" `ChannelNoiseDB` instances.
2. Configuration associates a **run range** to each instance.
   - Right now $N = 2$: before/after.
   - Each job creates and configures both `ChannelNoiseDB` instances.
3. The *facade* `ChannelNoiseDB` IsA `IArtEventVisitor`:
   - Can check run number on each event and then switch implementation when needed.
   - Must check no WCT code caches some value from the CHNDB.