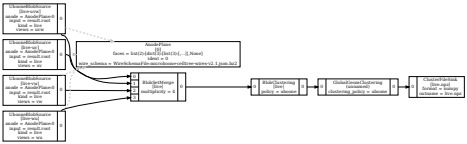
Uboone cluster+flash loading

Brett Viren

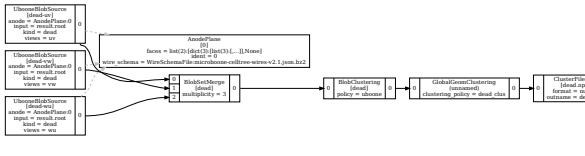
February 10, 2025

Reminder: loading uboone blobs

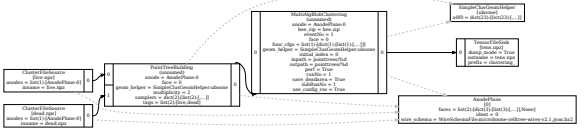
- 2 loads of uboone blobs
  - “live” blobs (uvw/uv/vw/wu)
  - “dead” blobs
  - effectively skips **WCT imaging**.
  - results in `ICluster` files.
- Load `ICluster` files run **WCT clustering** to get PC-tree as `ITensors`.



live



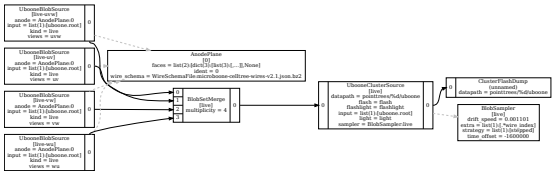
dead



clustering

New: loading uboone clusters + flashes + matches

- Load uboone “live” blobs using same `UbooneBlobSources`.
- Insert a new `UbooneClusterSource`
  - Ingest `IBlobs`, sample them.
  - Load Uboone TTrees (again).
  - Build “light ”flash“, ”flashlight" arrays.
  - Build cluster-flash match
- Effectively skips **WCT clustering**.
  - makes functionally identical PC-tree
  - but, adds the optical info.
- Graph ends with `ClusterFlashDump`
  - logs info for validation.



Cluster-flash representation

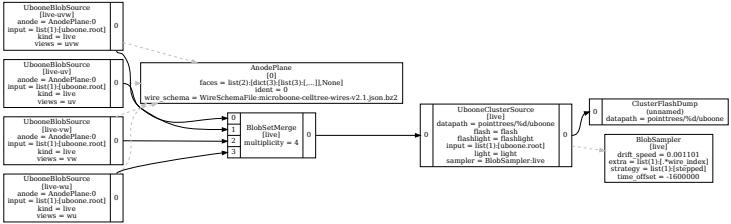
Cluster nodes get a local PC `Dataset` named `cluster_scalar` with arrays:

- `flash` the **index** into the `flash` `Dataset` or `-1` if no flash.
- `cluster_id` the application-determined identifier for the cluster.

Consequences:

- User can mutate the PC-tree without breaking `cluster_scalar`.
- User must the rewrite `flash` index if the optical datasets are mutated.
- Cluster-to-flash is strictly 1-to-1 relationship.
  - An implicit many-to-1 flash-to-cluster relationship is allowed.

Test jobs



Stand-alone WCT test

```
wire-cell -l stderr -L debug -A infiles=nuselEval_5384_137_6852.root \
  root/test/uboone-clusters.jsonnet
```

Script to download test data file, run `wire-cell`, make graph, etc

```
./root/test/uboone-clusters.smake dump dotify
```

Validation needed

So far, just overall counts of some things are checked.

Flash representation

In `ClusterFlashDump`, dump `flash`, `light` and `flashlight` arrays. Compare against ROOT file contents.

Cluster-flash matching

In `ClusterFlashDump`, dump `cluster_id`, lookup `flash.ident` in `flash` array given `flash` index. Compare against ROOT file contents.

The optical tensor data model

TDM doc updated in `apply-pointcloud` branch

- We may extend each dataset to novel arrays and arrays can be optional.

`light: time, value, error, ident`

The `light` array is like "ophits". The `ident` may be a PMT/SiPM channel number.

`flash: time, tmin, tmax, value, error, ident, type`

Note, all per-flash scalar, no "ophits" here directly.

`flashlight: flash, light`

A PC with two arrays giving `(flash index, light index)` pairs.

- May represent an arbitrary bipartite graph.

Recommendations for the SBND Q-L development

Let's synchronize SBND Q-L development:

- Use `light`, `flash` and `flashlight` tensor data model for optical info.
  - Update existing LArSoft `Op*` to the optical TDM converter.
  - The new `root/UbooneTTrees` helper class may provide an example.
- Use the `cluster_scalar.flash` PC array to represent Q-L matches.
- Set `cluster_scalar.cluster_id`.
  - Freedom to choose `cluster_id` numbering convention.
  - WCT cluster IDs **emerge** from "connected components" graph function.
  - A "stable" ID method is technically impossible, but something practical might be invented.

Vague pointers for future development

Flash making

Take a PC-tree, ignore and overwrite `flash` and `flashlight` local PCs in the "grouping" node.

Cluster-flash making

Take a PC-tree, ignore and overwrite the `flash` **index** scalar array in `cluster_scalar` PC on each cluster node.

Refine clusters

Mutate the PC-tree as you wish. Any new cluster needs a `cluster_scalar` PC with `flash` and `cluster_id`.