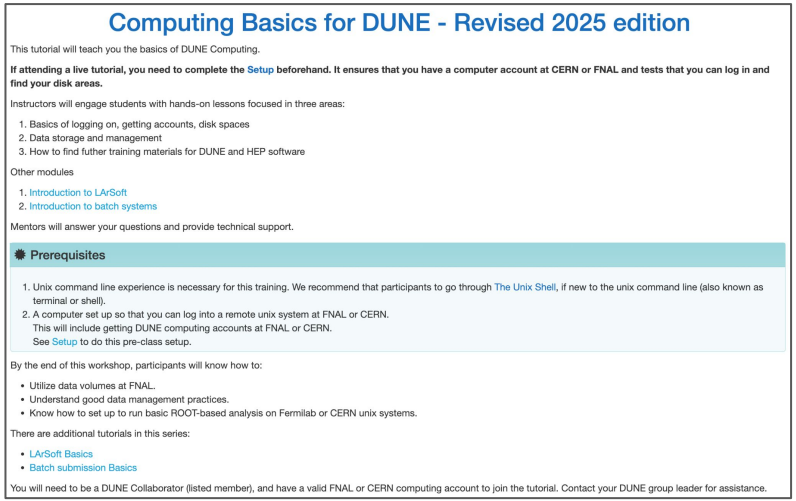
Status report on

DNNROI sigproc

Hokyeong Nam
Chung-Ang University

Pre-preparation before BNL visit



Computing Basics for DUNE - Revised 2025 edition

This tutorial will teach you the basics of DUNE Computing.

If attending a live tutorial, you need to complete the Setup beforehand. It ensures that you have a computer account at CERN or FNAL and tests that you can log in and find your disk areas.

Instructors will engage students with hands-on lessons focused in three areas:

1. Basics of logging on, getting accounts, disk spaces
2. Data storage and management
3. How to find futher training materials for DUNE and HEP software

Other modules

1. Introduction to LArSoft
2. Introduction to batch systems

Mentors will answer your questions and provide technical support.

Prerequisites

1. Unix command line experience is necessary for this training. We recommend that participants to go through The Unix Shell, if new to the unix command line (also known as terminal or shell).
2. A computer set up so that you can log into a remote unix system at FNAL or CERN.
   This will include getting DUNE computing accounts at FNAL or CERN.
   See Setup to do this pre-class setup.

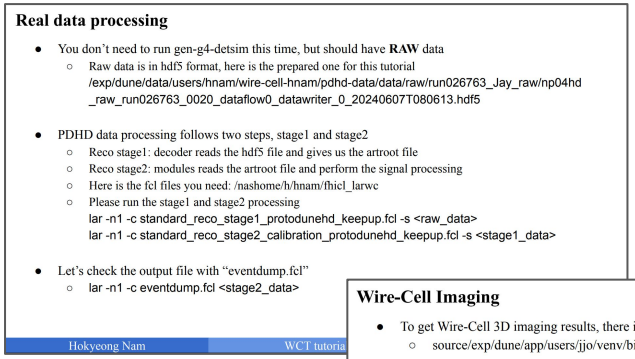
By the end of this workshop, participants will know how to:

- Utilize data volumes at FNAL.
- Understand good data management practices.
- Know how to set up to run basic ROOT-based analysis on Fermilab or CERN unix systems.

There are additional tutorials in this series:

- LArSoft Basics
- Batch submission Basics

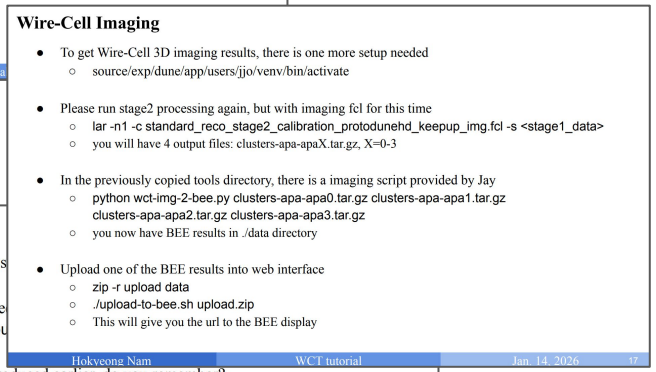
You will need to be a DUNE Collaborator (listed member), and have a valid FNAL or CERN computing account to join the tutorial. Contact your DUNE group leader for assistance.



Real data processing

- You don't need to run gen-g4-detsim this time, but should have **RAW** data
  - Raw data is in hdf5 format, here is the prepared one for this tutorial
    /exp/dune/data/users/hnam/wire-cell-hnam/pdhd-data/data/raw/run026763_Jay_raw/np04hd_raw_run026763_0020_dataflow0_datawriter_0_20240607T080613.hdf5
- PDHD data processing follows two steps, stage1 and stage2
  - Reco stage1: decoder reads the hdf5 file and gives us the artroot file
  - Reco stage2: modules reads the artroot file and perform the signal processing
  - Here is the fcl files you need: /nashome/h/hnam/fhicl_larwc
  - Please run the stage1 and stage2 processing
    lar -n1 -c standard_reco_stage1_protodunehd_keepup.fcl -s <raw_data>
    lar -n1 -c standard_reco_stage2_calibration_protodunehd_keepup.fcl -s <stage1_data>
- Let's check the output file with "eventdump.fcl"
  - lar -n1 -c eventdump.fcl <stage2_data>

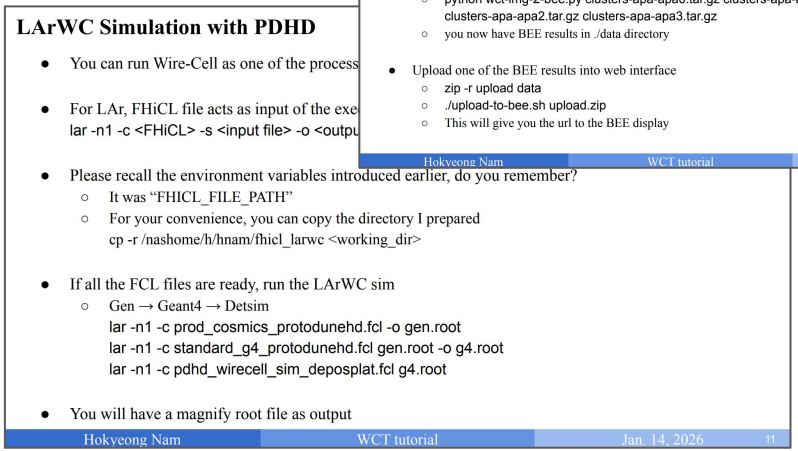
Hokyeong Nam WCT tutoria



Wire-Cell Imaging

- To get Wire-Cell 3D imaging results, there is one more setup needed
  - source/exp/dune/app/users/jjo/venv/bin/activate
- Please run stage2 processing again, but with imaging fcl for this time
  - lar -n1 -c standard_reco_stage2_calibration_protodunehd_keepup_img.fcl -s <stage1_data>
  - you will have 4 output files: clusters-apa-apaX.tar.gz, X=0-3
- In the previously copied tools directory, there is a imaging script provided by Jay
  - python wct-img-2-bee.py clusters-apa-apa0.tar.gz clusters-apa-apa1.tar.gz clusters-apa-apa2.tar.gz clusters-apa-apa3.tar.gz
  - you now have BEE results in ./data directory
- Upload one of the BEE results into web interface
  - zip -r upload data
  - ./upload-to-bee.sh upload.zip
  - This will give you the url to the BEE display

Hokyeong Nam WCT tutorial Jan. 14, 2026 17



LArWC Simulation with PDHD

- You can run Wire-Cell as one of the process
- For LAr, FHiCL file acts as input of the exe
  lar -n1 -c <FHiCL> -s <input file> -o <outpu
- Please recall the environment variables introduced earlier, do you remember?
  - It was "FHICL_FILE_PATH"
  - For your convenience, you can copy the directory I prepared
    cp -r /nashome/h/hnam/fhicl_larwc <working_dir>
- If all the FCL files are ready, run the LArWC sim
  - Gen → Geant4 → Detsim
    lar -n1 -c prod_cosmics_protodunehd.fcl -o gen.root
    lar -n1 -c standard_g4_protodunehd.fcl gen.root -o g4.root
    lar -n1 -c pdhd_wirecell_sim_deposplat.fcl g4.root
- You will have a magnify root file as output

Hokyeong Nam WCT tutorial Jan. 14, 2026 11

students will visit BNL from Feb. 2 to Feb. 24

Hokyeong Nam, Juseong Park, Sunhong Kim, Yujin Park

TagSelector for spliced pipeline

sigproc/inc/WireCellSigProc/TagSelector.h

```
+ /**
+    Make a new output frame with a set of traces selected from the
+    input based on trace tags.
+  */
+
+ #ifndef WIRECELLSIGPROC_TAGSELECTOR
+ #define WIRECELLSIGPROC_TAGSELECTOR
+
+ #include "WireCellIface/IFrameFilter.h"
+ #include "WireCellIface/IConfigurable.h"
+ #include "WireCellAux/Logger.h"
+
+ #include <string>
+ #include <vector>
+
+ namespace WireCell {
+     namespace SigProc {
+
+         class TagSelector : public Aux::Logger,
+                             public IFrameFilter,
+                             public IConfigurable
+         {
+           public:
+             TagSelector();
+             virtual ~TagSelector();
+
+             /// IFrameFilter interface.
+             virtual bool operator()(const input_pointer& in, output_pointer& out);
+
+             /// IConfigurable interface.
+             virtual void configure(const WireCell::Configuration& config);
+             virtual WireCell::Configuration default_configuration() const;
+
+           private:
+             std::vector<std::string> m_tags;
+             int m_count{0};
+         };
+     }  // namespace SigProc
+ }  // namespace WireCell
+
+ #endif
```

sigproc/src/TagSelector.cxx

```
+ #include "WireCellSigProc/TagSelector.h"
+ #include "WireCellAux/SimpleFrame.h"
+ #include "WireCellAux/FrameTools.h"
+
+ #include "WireCellUtil/NamedFactory.h"
+
+ #include <sstream>
+ #include <unordered_map>
+
+ WIRECELL_FACTORY(TagSelector, WireCell::SigProc::TagSelector, WireCell::IFrameFilter, WireCell::IConfigurable)
+
+ using namespace WireCell;
+ using namespace WireCell::SigProc;
+
+ TagSelector::TagSelector()
+     : Aux::Logger("TagSelector", "glue")
+ {
+ }
+
+ TagSelector::~TagSelector() {}
+
+ WireCell::Configuration TagSelector::default_configuration() const
+ {
+     Configuration cfg;
+
+     /// Only traces with these tags will be in the output.
+     cfg["tags"] = Json::arrayValue;
+
+     return cfg;
+ }
+
+ void TagSelector::configure(const WireCell::Configuration& cfg)
+ {
+     auto jtags = cfg["tags"];
+     int ntags = jtags.size();
+     m_tags.clear();
+     m_tags.resize(ntags);
+     for (int ind = 0; ind < ntags; ++ind) {
+         m_tags[ind] = jtags[ind].asString();
+     }
+ }
+
+ bool TagSelector::operator()(const input_pointer& in, output_pointer& out)
+ {
+     out = nullptr;
+     if (!in) {
+         log->debug("see EOS at call={}", m_count);
+         ++m_count;
+         return true;  // eos
+     }
+
+     if (m_tags.empty()) {
+         log->warn("TagSelector configured with no tags, passing through input frame");
+         out = in;
+         return true;
+     }
```

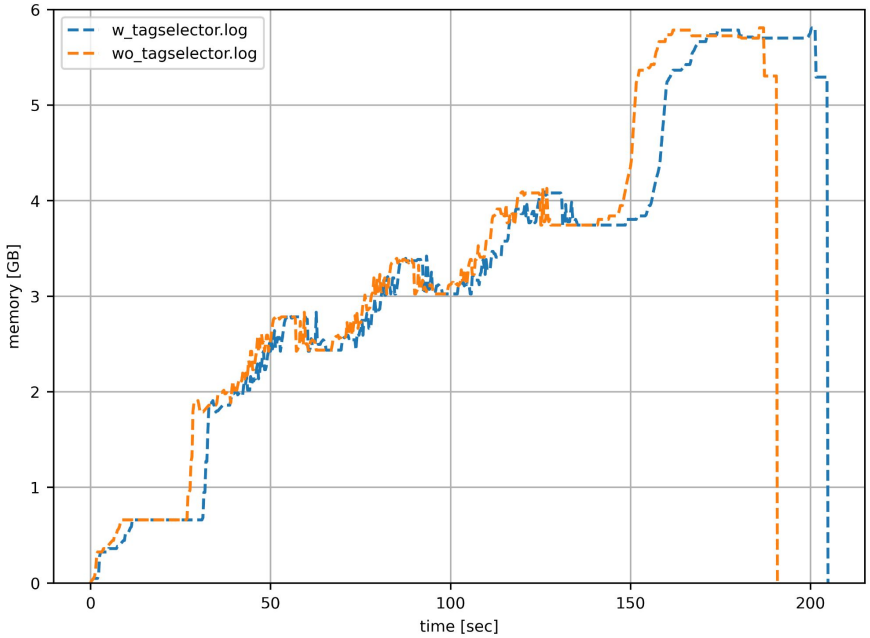
```
local sp_tag_selector = g.pnode({
  type: 'TagSelector',
  name: 'sp_tag_selector',
  data: {
    tags: [
      tag
      for n in std.range(0, std.length(tools.anodes) - 1)
      for tag in ['gauss%d' % n, 'wiener%d' % n]
    ],
  },
}, nin=1, nout=1);
local outgr = g.pipeline([ofanin, sp_tag_selector, outretagger, wcls_output.sp_signals, osink]);
// local outgr = g.pipeline([ofanin, outretagger, osink]);

local edge_selector(e) = std.startsWith(e.tail.node, "OmnibusSigProc:");
local fanout_factory(n,e) = { type:'FrameFanout', name:"splice%d"%n, data:{multiplicity: 2} }; // "2-wire" splice

local spliced_graph =
  if save_tradsp then
    g.splice(graph, outgr, edge_selector, fanout_factory)
  else
    graph;
```

- To prevent overmemory usage by spliced data pipeline, we need a filter so that only gauss/weiner results will be saved
- A new function named TagSelector is developed and tested

TagSelector for spliced pipeline



- The memory consumption measured using activity logger based on top command
- The TagSelector didn’t work as it is designed for

Plans

- Debugging TagSelector and make it work as expected

- Make pull request
  - Wire-Cell Toolkit: TagSelector
  - dunereco: updated jsonnet configruation files for PDHD and PDVD

- PDVD tasks
  - Working with Haiwang to solve the time offset issue at its source
  - Running a small PDVD sample production for the DNN training

Back Up