DUNE MC-Sample Generation

Gleb Sinev
Duke University

DUNE DAQ-Simulation Hack Days
October 17, 2016

DUNE Simulation

- DUNE uses LArSoft to fully simulate and reconstruct events

- Framework shared with other (mostly Fermilab-based) liquid-argon experiments

Creating MC Samples

- Setting up and running LArSoft
- LArSoft configuration files (FHICL)
- Using Fermigrid

Initial Setup

- LArSoft with DUNETPC

- On dunegpvm{01..10}.fnal.gov
  - Current LArSoft version:

    ```
    setup dunetpc v06_11_00 -q e10:prof
    ```

  - More on this in my back-up slides

Running LArSoft

- `lar -c configuration.fcl`
  `-n NEvents`
  `-s inputFile.root`
  `-o outputFile.root`
  `-T histogramFile.root`
- FHiCL (Fermilab Hierarchical Configuration Language) files are used to configure LArSoft
- `lar --help` to see other possible arguments

Monte-Carlo-Challenge FHiCL

- Most up-to-date available FHiCL files
- Most probably work
- Good to use as examples for your FHiCL files
- Located in `$DUNETPC_DIR/source/fcl`
- Here focus on DUNE far detector,
  also 35-ton and ProtoDUNE available

Far-Detector Geometries

- Make sure to use the same geometry for all FHiCL files in sim/reco chain!
- 12 APA geometry
  - `dune10kt_1x2x6` – standard
  - `dune10kt_3mmpitch_1x2x6` – 3-mm wire pitch
  - `dune10kt_45deg_1x2x6` – 45° wire angle
  - `dune10kt_r90deg_1x2x6` – APAs rotated by 90°
- Others available:
  - `dune10kt_dphase` – double phase, no photon simulation
  - `dune10kt_workspace` – 4APA, somewhat deprecated
  - `dune10kt` – full 10kt detector, no photon simulation

MCC Stages

- `gen`
- `g4`
- `detsim`
- `reco`
- `mergeana`

gen Stage

- Generate primary particles
- `$DUNETPC/source/fcl/dunefd/gen`
  - `atm`
  - `background`
  - `cosmic`
  - `genie`
  - `ndk`
  - `single`
  - `supernova`

g4 Stage

- Simulate detector materials with LArGeant (Geant4)
- `$DUNETPC/source/fcl/dunefd/g4`
  - `standard`
  - `supernova` – tuned for low-energy interactions

`detsim` Stage

- Simulate signals on wires and photon detectors
- `$DUNETPC/source/fcl/dunefd/detsim`
  - `standard`
  - `nozs` – no zero-suppression

`reco` Stage

- Reconstruct events
- `$DUNETPC/source/fcl/dunefd/reco`
  - `standard`
  - `nu`

`mergeana` Stage

- In MCC – merge multiple reco files
- Produce histograms and trees
- `$DUNETPC/source/fcl/dunefd/mergeana`

Final thoughts on FHiCL

- LArSoft configuration is quite flexible,
  you may have different stages or
  just one FHiCL file to produce all sim/reco
  (may be suboptimal)
- Good description of the language:
  https://cdcvs.fnal.gov/redmine/projects/fhicl/wiki
- More on FHiCL in my back-up slides

Using Fermigrid

Two ways to submit jobs:

- Jobsub
- project.py

Jobsub

- https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Submitting_Jobs_at_Fermilab
- Very flexible
- Need shell script with all commands run

project.py

- https://cdcvs.fnal.gov/redmine/projects/larbatch/wiki/User_guide
- Wrapper for Jobsub
- Controlled using XML file (examples available via `make_xml_mcc7.0.sh`)
- GUI available
- Used by DUNE MC production group

Storing Output

- 100s of jobs can produce TBs of data quickly
- Store output in dCache: https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Using_DUNE's_dCache_Scratch_and_Persistent_Space_at_Fermilab
  - `/pnfs/dune/scratch`
  - `/pnfs/dune/persistent`
  - `/pnfs/dune/tape_backed`

Conclusion

- That's a quick overview of how to create Monte-Carlo samples in DUNE
- You can also access files generated for MCC7: https://cdcvs.fnal.gov/redmine/projects/dune/wiki/DAQ-Simulations_MC_File_List
- Any questions?

Back-up Slides

Logging in

- Fermilab uses Kerberos for authentication
- Obtain Kerberos ticket with
  - **`kinit -f <username>@FNAL.GOV`**
- Log into one of DUNE General Purpose Virtual Machines
  - **`ssh -X <username>@dunegpvmXX.fnal.gov`** (XX is `01-10`)
  - Try using different XX from what others are using

Creating Workspace

- Make directory to work in
  - `mkdir -p /dune/app/users/<username>/tutorial-mc`
- Use `/dune/app/users/<username>` for code and other relatively small files, but don't store GBs of data there!

Setting up DUNE Software Environment

- For each new shell run
  - `source /grid/fermiapp/products/dune/setup_dune.sh`
  - To automate add the line above to `~/.bashrc`
  - Instead you can add it to another script that you run after opening a new shell session

Setting up LArSoft

- Check what versions are available
  - `ups list -aK+ dunetpc`
- Set up DUNETPC
  - **`setup dunetpc v06_11_00 -q e10:prof`**
- This sets up LArSoft and DUNE-specific code

FHiCL Language

- `parameter: value`
- `sequence: [ value1, value2, … valueN ]`
- `table:`

```
{
  parameter1: tableValue1
  parameter2: tableValue2
  …
  parameterM: tableValueM
              @table::someOtherTable
}
```

FHiCL Language

- `#include “fhicl_file.fcl”`
- `tableFromOtherTable: @local::otherTable`
- `outerTable.innerTable.parameter: value`

FHiCL Files

- With `#includes` it gets hard to understand what parameters you actually run
- Some tools exist to help with this
  - e.g. `fhicl-dump configuration.fcl`

Example of gen FHiCL file

`$DUNETPC/source/fcl/dunefd/gen/single/prod_eminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6.fcl`

```
#include "prodsingle_common_dunefd.fcl"

process_name: SinglesGen

outputs.out1.fileName: "prod_eminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6_gen.root"

services.Geometry: @local::dune10kt_1x2x6_geo
source.firstRun: 20000011

physics.producers.generator.PDG: [ 11 ]            # e-
physics.producers.generator.PosDist: 0             # Flat position dist.
physics.producers.generator.X0: [ 0 ]
physics.producers.generator.Y0: [ 0.0 ]
physics.producers.generator.Z0: [ 695 ]
physics.producers.generator.T0: [ 500.0 ]
physics.producers.generator.SigmaX: [ 360 ]      # x = (-3.6, 3.6)
physics.producers.generator.SigmaY: [ 600 ]      # y = (-6, 6)
physics.producers.generator.SigmaZ: [ 695 ]      # z = (0, 13.9)
physics.producers.generator.SigmaT: [ 500.0 ]       # In time
physics.producers.generator.PDist: 0               # Flat momentum dist. (0.1-2.0 GeV)
physics.producers.generator.P0: [ 2.55 ]
physics.producers.generator.SigmaP: [ 2.45 ]
physics.producers.generator.AngleDist: 0           # Flat angle dist.
physics.producers.generator.Theta0XZ: [ 0. ]       # y-azimuth
physics.producers.generator.Theta0YZ: [ 0. ]       # y-latitude
physics.producers.generator.SigmaThetaXZ: [ 180. ] # Quasi-isotropic
physics.producers.generator.SigmaThetaYZ: [ 90. ]
```

Edit to change the initial momentum

Example of g4 FHiCL file

`$DUNETPC/source/fcl/dunefd/g4/standard_g4_dune10kt_1x2x6.fcl`

```
1  #include "standard_g4_dune10kt.fcl"
2  process_name: G4
3  services.Geometry: @local::dune10kt_1x2x6_geo
4  services.PhotonVisibilityService: @local::dune10kt_1x2x6_photonvisibilityservice
5  services.LArG4Parameters.UseCustomPhysics: true
6  services.LArG4Parameters.EnabledPhysics: [ "Em",
7                                             "FastOptical",
8                                             "SynchrotronAndGN",
9                                             "Ion",
10                                            "Hadron",
11                                            "Decay",
12                                            "HadronElastic",
13                                            "Stopping",
14                                            "NeutronTrackingCut" ]
15
16
```

Remove to not simulate photons

Example of `detsim` FHiCL file

```
# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics: {
 producers: {
   daq:          @local::dune_detsim
   opdigi:       @local::dune35t_opdigi
   rns:          { module_type: "RandomNumberSaver" }
 }
 simulate: [ rns, daq, opdigi ]
 stream1:  [ out1 ]
 trigger_paths: [simulate]
 end_paths:     [stream1]
}

outputs: {
 out1: {
   module_type: RootOutput
   fileName:    "%ifb_detsim.root"
   dataTier:    "detector-simulated"
   compressionLevel: 1
 }
}

# Use fixed values instead of DB for pedestals.
services.DetPedestalService: @local::dune_fixedpeds

# DetSim flags.
physics.producers.daq.NoiseOn:     true
physics.producers.daq.PedestalOn:  true
physics.producers.daq.DistortOn:   false
physics.producers.daq.SuppressOn:  true

# DetSim services.
services.SimChannelExtractService: @local::scxgeneric
services.ChannelNoiseService:      @local::chnoiseold
services.PedestalAdditionService:  @local::padprovided
services.AdcDistortService:        @local::stuckbits
services.AdcSuppressService:       @local::zslegacy
services.AdcCompressService:       @local::cmpblock

# Disable bad channels.
#services.IChannelStatusService.BadChannels: [ ]
```

Remove to not simulate photon detectors

`$DUNETPC/source/fcl/dunefd/detsim/standard_detsim_dune10kt.fcl`

Changing FHiCL files

- Instead of modifying already existing files:
  - Create a new FHiCL file
  - Include the file you want to change
  - Change the parameters you want to change
- Example:
  - `$DUNETPC/source/fcl/dunefd/detsim/standard_detsim_dune10kt_1x2x6.fcl`
  - It includes `standard_detsim_dune10kt.fcl` and changes `process_name` and `services.Geometry`

```
1 #include "standard_detsim_dune10kt.fcl"
2 process_name: Detsim
3 services.Geometry: @local::dune10kt_1x2x6_geo
~
~
~
```