ldmx-sw Tutorial Course: Advanced Methods

Tom Eichlersmith

University of Minnesota

eichl008@umn.edu

May 27, 2021

Outline

LDMX

Pre-Requisites

Pre-Requisites

LDMX

- Familiarity with `ldmx-sw` already

Warning

If you are more inexperienced with `ldmx-sw`, much of this information may be out of your reach. That is okay. Seeing it early will help you learn it later if you need to use it.

Overview

Simulation Filters

LDMX

- Implemented as `UserActions`
- Used mainly for event filtering and track filtering
  - Event Filtering: Abort the sim event early to save time if criteria aren't met
  - Track Filtering: Make sure sim saves a specific particle information to output file
- Check out `Biasing` module for examples
- **Very flexible** — Handles to different points in the simulation
  - At **end** of each step
  - Choose which tracks are processed first ("classification")
  - A handle to when a stack is emptied ("stage")
  - Start and end of processing each track
  - Start and end of processing each event
  - Start and end of processing the run
  - Any mixture of the above

HistogramPool

- Exemplified by `EcalPN` analyzer in `DQM` module
- `get`: Get a histogram by name
- `create`: Make a new histogram by name and type
- You can tell the HistogramPool what kinds of 1D histograms to create in your python configuration by using `build1DHistogram` member function of a processor
  - Then you don't need to `create` those histograms in the Analyzer code

NtupleManager

- Only needs to be done in Analyzer code
- `create`: Make a new tree to put variables in
- `addVar`: Put a new variable name into a tree (template)
- `setVar`: Set the variable value during processing

Upcoming Changes

Will soon be using the same buffer for the NtupleManager and the Event Bus, so the NtupleManager will soon be able to handle vectors as well as individual values.

Skimming Events

LDMX

After using `setStorageHint` in your processor, you also need to tell the process that it should listen to your processor.

In your python configuration file:

```
myAna = ldmxcfg.Analyzer( "myAna" , "ldmx::MyAnalzer" )
# p is Process object created earlier
p.skimDefaultIsDrop() # drop events unless told otherwise
p.skimConsider(myAna.instanceName) # tell Process to listen to myAna
```

Drop/Keep Event Objects

LDMX

Maybe you know *for a fact* that you will *never* use the simulated hits later in your analysis chain and you want to save some space in your output file.

In your python configuration file:

```
# p is Process object created earlier
p.keep = [
    # don't put any collection with 'EcalSim' in the name into output file
    "drop EcalSim.*",
    ]
```

Drop/Keep Event Objects

LDMX

Another use-case would be re-running the digitization and reconstruction after some developments. You don't want to keep the old stuff, so you can drop it.

In your python configuration file:

```
# p is Process object created earlier
p.keep = [
    # don't save Digi and Rec of ending with the 'old' pass
    "drop .*Digi.*old", "drop .*Rec.*old"
    ]
```

Developing for ldmx-sw

git submodules

LDMX

▸ submodule Documentation

- Isolates different codes from each other ✓
- Requires more `git` nonsense (like `git submodule update`)

Basically

Think of a submodule as a totally separate repo. ldmx-sw **only** stores which commit of that repo it should put into that submodule (i.e. ldmx-sw only "references" the submodule).

```
1 # checkout trunk and the submodules pointed-to by trunk
2 ldmx checkout trunk
3 # checkout trunk but have Ecal be in developments
4 ldmx checkout trunk Ecal:developments
```

Code Formatting

LDMX

Using ▸ Google Style Guide and `clang-format`.

```
# need clang-format installed
find <directory> -type f -name "*.h" -o -name "*.cxx" \
  -exec clang-format -i --style=Google {} \;
```

Pull Request Workflow

LDMX

Template

`ldmx-sw` has a PR Template that walks you through the three big steps.

- Explain what you changed (and what it is supposed to fix and/or improve)
- Confirm that your changes don't break anything else (using the tests)
- Prove that your changes do what you say they do (depends on the changes)

Submodules

Since submodules are separate repositories, you will need separate PRs if you have changes in the submodule: One in the submodule for the changes and one in ldmx-sw to update the submodule reference.

Questions