Event Mixing for Larsoft

Larsoft Coordination Meeting
Sept. 13, 2016

H. Greenlee

Contents

- This talk is about MicroBooNE's implementation (code and scripts) for running cosmic overlay mc, including workflow and production tools.
  - Overview of event mixing in art.
  - MicroBooNE's strategy for event mixing.
  - Workflow/production tools.

Event Mixing Overview

- Definition of event mixing:
  - Primary stream consists of a standard art event source (e.g. RootInput module).
  - Another (non-source) module is responsible for reading from the secondary input stream and producing combined data products.
- Built-in art support event mixing.
  - EDFilter module MixFilter<T> handles actual I/O from secondary input stream.
  - User provides "detail" class T that is responsible for combining data products from the two input streams.
    - Detail class registers functions for mixing various data products.
    - Detail class optionally registers "getMixFile" class to choose next mix input file.

MicroBooNE Cosmic Overlay

- MicroBooNE's implementation of MixFilter<T> is called OverlayRawDataMicroBooNE (typedef MixFilter<OverlayRawDataDetailMicroBooNE>).
  - Deail class released in uboonecode product.
  - Detail class combines data products RawDigit and OpDetWaveform, and copies any other data products found in one input file (e.g. mc truth).
  - Also registers getMixFile method which is implemented to get secondary input files from a sam dataset.
  - One input is from mc. Other input is cosmic data.
    - Detail class can be configured such that either the mc or cosmic data is primary input.

Which Input Should be Primary?

- In general, for any event mixing scenario, the primary input should be the one that you are more interested in bookkeeping.
  - Workflow tools (project.py, pubs, poms), with art and sam, already provide significant support for bookkeeping the primary input stream (“for free”).
  - Art and MixFilter<T> provide zero support for bookkeeping the secondary input stream.
  - For above reasons, bookkeeping of secondary input stream will always be worse than primary input stream.
  - MicroBooNE chose to make the mc the primary input and cosmic data the secondary input.
    - Even though it might have been operationally easier to do the opposite.
    - Use sam to bookkeep the secondary input stream (rather than invent a whole 'nother bookkeeping system for secondary input).

Secondary Input SAM Project

- Mixing module detail class interacts with SAM system using art IFDH service (from ifdh_art ups product).
- Detail class is responsible for the following aspects of secondary sam project management.
  - Starting batch worker's consumer process.
  - Requesting next file(s).
    - Requested in form of streamable xrootd url.
  - Releasing file(s) and updating file status.
  - Stopping batch worker's consumer process.
  - Generating parentage sam metadata for secondary inputs.
- Workflow is responsible for:
  - Starting secondary sam project.
  - Stopping secondary sam project.

FCL Configuration of Mixing Module

- Specify sam dataset and sam project as fcl parameters of mising module OverlayRawDataMicroBooNE.

```
filters : {
  mixer: { module_type : OverlayRawDataMicroBooNE
           detail  : {
             SamDefname: prod_extunbiased_swizzle_inclusive_v3
             SamProject: greenlee_prod_extunbiased_swizzle_inclusive_v3_20160901_154140
           }
```

  - Sam fcl parameters (dataset definition name and project name) should be automatically generated by workflow.

Configuring Batch Projects for project.py

- Here is an example project.py xml merge step configuration.

```
<stage name="mix">
  <fcl>/uboone/data/users/greenlee/cosmic_overlay/run_DataOverlayMixer.fcl</fcl>
  <mixinputdef>greenlee_mix_test</mixinputdef>
  <outdir>/pnfs/uboone/scratch/users/greenlee/&release;/&name;/mix</outdir>
  <logdir>/pnfs/uboone/scratch/users/greenlee/&release;/&name;/mix</logdir>
  <workdir>/pnfs/uboone/scratch/users/greenlee/work/&release;/&name;/mix</workdir>
  <numjobs>20</numjobs>
  <datatier>raw</datatier>
  <defname>greenlee_&name;</defname>
  <maxfilesperjob>1</maxfilesperjob>
  <memory>4000</memory>
  <jobsub>--expected-lifetime=short</jobsub>
  <initsource>mix_sam.sh</initsource>
</stage>
```

  - <mixinputdef> is a newly defined xml element, which you use to specify the sam dataset containing secondary input data.
  - Use <initsource> xml element to run script mix_sam.sh (now included in ubutil package) to generate fcl wrapper.

Modifications to Larbatch

- Larbatch package contains scripts project.py and condor_lar.sh, used by several LArTPC experiments, and should be experiment-independent.
  - New stage xml element <mixnputdef> added to stage definition (stagedef.py).
  - Project.py generates dag to start sam project for secondary input stream (in addition to primary input stream, if from sam).
  - Name of sam project and name of sam dataset passed to batch script condor_lar.sh using command line options.
  - Condor_lar.sh parses secondary input command line options but doesn't do anything much with them (because experiment-specific).

Modifications to Ubutil

- Ubutil package contains changes that are MicroBooNE-specific, works with or plugs into hooks provided by larbatch scripts.
  - MicroBooNE Metadata extractor script extractor_dict.py merges primary and secondary parent files into single list.
  - Script mix_sam.sh plugs into hooks inside condor_lar.sh and generates experiment-specific wrapper fcl to pass sam parameters to module OverlayRawDataMicroBooNE.

Mixing Summary Data Product

- In addition to joining or copying data products, module OverlayRawDataMicroBooNE generates a MicroBooNE-specific data product called EventMixingSummary containing the event id (run, subrun, event number) of secondary event.
- We also want to store the time stamp of secondary event (because out databases are indexed by time stamp).
  - MixFilter interface does not currently make the time stamp of the secondary event available to mixing module.
  - Art feature request issue opened to make time stamp available.

SAM Bookkeeping

- SAM will automatically ensure that every process that requests a file within a given project (i.e. within a single batch submission) receives a different file (up to the maximum number of available files).
- It is also possible to define a kind of sam dataset (called a recursive dataset) that will only deliver a file once per campaign, even across multiple projects/ batch submissions.
  - Instructions available on uboonecode wiki:
    - https://cdcvs.fnal.gov/redmine/projects/uboonecode/wiki/Sam#Recursive-datasets
  - This is the kind of dataset we would probably want to use for mc production.

Status of Repositories

- Larbatch updated on feature branch feature/greenlee_mix.
  - Propose to merge to develop for this week's integration release.
  - Not a breaking change.

Summary

- Workflow changes needed to support MicroBooNE's model for event mixing (aka cosmic overlay) require changes in larbatch product (scripts project.py and condor_lar.sh), which are ready for release.

Backup

Previously Released Updates to Larbatch

- All file access goes through posix-like i/o layer (module larbatch_posix).
  - Can be configured to use either posix or grid i/o tools (e.g. ifdh).
  - Allows project.py to run on a node that doesn't have nfs access to bluearc or dCache.

Other Pending Updates to Larbatch

- Future updates to larbatch (Joel Mousseau).
  - Move most validation and bookkeeping (e.g. declare files to sam) from submitting node (project.py) to batch worker node (condor_lar.sh).
  - Allow condor_lar.sh to run multiple steps (multiple invocations of lar).
  - Combine experiment-independent aspects of metadata extraction into a common extensible implementation of extractor_dict.py.