HEPnOS status

Saba Sehrish

SciDAC 4 Collaboration meetin

Nov 6th, 2018

HEPnOS Status – in progress

- We have a hepnos docker image updated with the latest release of the hepnos code that includes `hepnos::Ptr`
- We are working on a demo program writing an association collection to the HEPnOS store.
  - read hits, tracks and association collection between hits and track from an *art*/ROOT file: `recob::Hit`, `recob::Track` and `art::Assns<recob::Hit, recob::Track>`
  - Write hits, tracks and association-like collection between hits and tracks to the hepnos store using `hepnos::Ptr`:
    ```
    vector<pair<hepnos::Ptr<recob::Hit>,
                hepnos::Ptr<recob::Track>>>
    ```
  - Currently working on reading back this data from the store: writing a test to write to/read back hand-made association collection

HEPnOS status – To do

- Complete the association demo
- Create a Docker image containing a recent version of dunetpc (and thus the full *art* framework, not just gallery).
- Write a demo program that writes a "large object" to HEPnOS.
  - RawDigits collection
- Run hepnos server outside the docker container, and use docker only for the client program

HEPnOS – Further future

- hepnos::Event is an analogue of art::Event and gallery::Event
  - this is an important start, for us to have something for user-code to interact with
- We need to start thinking about HEPnOS I/O for the *art* framework
  - one input module, one output module
  - Can not know about concrete user-defined data products
  - Instead, must interact with the abstract class art::EDProduct
  - Must also determine how to store the provenance information *art* requires, e.g.
    - ParameterSet objects used to configure jobs
    - ProcessHistory objects, reflecting the provenance of all data products
    - there are more…
    - We need to be able to recover all the necessary information, not to reproduce the art/ROOT file's storage mechanisms