Optimizing the access to one-to-many Association collection

Saba Sehrish

LArSoft Coordination Meeting

8/30/2016

Problem Overview

- The expectation is to come up with the design and implementation of a utility that will provide an efficient and easy to use interface for accessing one-to-many associations, only for the use case where:
  - The order in which associations are added to the collection carries relevant information and needs to be preserved
  - The association collection is ordered so that all the associations of a given object are contiguous

Different approaches to access art::Assns

1. Using FindMany* [Current]
   - Temporary container for the art::Ptrs to the R
   - Ordering not preserved
2. Using association data product [Current]
   - art::Assns<L, R>. E.g. art::Assns<recob::Track, recob::Hit>
   - Get association collection as a data product and loop over using index or range for
   - User has to know how to access L and R
   - User has to implement logic of moving form one L to another
3. Maintain a data structure to keep track of the associated objects [New]
   - Same as 2 except the logic of moving form one L to another
   - Additional data structure to main the index
4. Use range-v3 library to provide efficient access [New]

A utility function using the range-v3 library

```
for_each_associated_group(
                    /* association collection */,
                    /* a callable */);
```

This function performs the following:

- takes an association collection as input argument 1
- transforms it to a range of range objects
- calls the provided callable, the input argument 2, on the objects.

For example, given an `art::Assns<recob::Track, recob::Hit>`, it will transform this collection to a range representing range of art::Ptr<`recob::Hit`> for each track. The provided callable is invoked over this range for each track.

Demonstrating use of for_each_associated_group

- Toy Problem: Calculate the sum of SummedADCs for all the associated hits per track, and return a container with all the sums. In this example, ordering is not important.
- Assumptions:
  - Every recob::Track is represented in the association collection and in the same order it was inserted in the track collection.
  - All the associated recob::Hit for each recob::Track are contiguous in the collection.
- Recipe for using for_each_associated_group
  - Get the required association data product
  - Provide a lambda that implements the functionality to be executed on each group of the associated objects;
  - Call the function: `for_each_associated_group`

Example of user code

```
1.    typedef typename art::Assns<recob::Track, recob::Hit> th_assns;
2.    auto const & track_to_hit_assns = *e.getValidHandle<th_assns>
   (fTrackModuleLabel);

3.    std::vector<double> charge_per_track;
4.    auto fill_charge_per_track = [&charge_per_track](auto hits) {
5.       double sum_of_charges = 0.;
6.       for(auto h=begin(hits); h!=end(hits); ++h) {
7.          sum_of_charges+=(*h)->SummedADC();
8.       }
9.       charge_per_track.push_back(sum_of_charges);
10.   };
11.
12.   for_each_associated_group(track_to_hit_assns,
                                fill_charge_per_track);
```

Example of user code, making use of range-v3

```
1.    typedef typename art::Assns<recob::Track, recob::Hit> th_assns;
2.    auto const & track_to_hit_assns = *e.getValidHandle<th_assns>
   (fTrackModuleLabel);
3.    std::vector<double> charge_per_track;
4.    auto fill_charge_per_track = [&charge_per_track](auto hits) {
5.       charge_per_track.push_back(ranges::accumulate(hits |
   ranges::view::transform([] (auto h) {return h->SummedADC();}),
6.                                                      0.)
7.                                  );
8.    };
9.
10.   for_each_associated_group(track_to_hit_assns,
                                fill_charge_per_track);
```

Future work

- Next step is to use the for_each_associated_group utilitity in the example use cases:
  - Analysis tree codes
  - NeutrinoTrackingEff_module in larreco
  - Calorimtery_module in larana
- Performance measurements
- Work with art::Assns<L, R, D>

References

- Range-v3 Library
  - https://ericniebler.github.io/range-v3/
- Chris Green's presentation on art::Assns
  - https://indico.fnal.gov/getFile.py/access?sessionId=6&resId=10&materialId=0&confId=9928
  - https://indico.fnal.gov/getFile.py/access?sessionId=6&resId=9&materialId=0&confId=9928