INTT Weekly Meeting

Joseph Bertaux

Purdue University

February 7, 2024

Current State of Online Dead Channel Map

- My recent PR
  - Adds comparators for the `RawData_s`, `Online_s`, and `Offline_s` structs used to wrap position information from hits
    - This allows them to be easily used as keys for `std::map`'s and `std::set`'s
  - Adds member functions to the `InttCombinedRawDataDecoder` to populate such a set from a CDBTTree file
- Identification based on inclusion in an `std::set<InttNameSpace::RawData_s>`
  - Search is $\mathcal{O}(\ln(N))$ with N being the number of masked channels
- CDBTTree based loading; branches for the fields of `RawData_s`:
  - `felix_server` (ROC)
  - `felix_channel` (FEE)
  - `chip`
  - `channel`

Channel List Loading

Code snippet that shows how the CDBTTree is loaded. Note that `name` is an `std::string` passed as an argument, and `m_HotChannelSet` is a member variable of type `std::set<InttNameSpace::RawData_s>`

```
std::string database = CDBInterface::instance()->getUrl(name);
CDBTTree cdbttree(database);
cdbttree.LoadCalibrations();

m_HotChannelSet.clear();
Long64_t N = cdbttree.GetSingleIntValue("size");
for(Long64_t n = 0; n < N; ++n) {
        m_HotChannelSet.insert((struct InttNameSpace::RawData_s) {
                .felix_server  = cdbttree.GetIntValue(n, "felix_server"),
                .felix_channel = cdbttree.GetIntValue(n, "felix_channel"),
                .chip          = cdbttree.GetIntValue(n, "chip"),
                .channel       = cdbttree.GetIntValue(n, "channel")
        });
}
```

From the current combined raw data decoder

Hot Channel Masking

Since the `InttCombinedRawDataDecoder` now maintains a member set `m_HotChannelSet` of hot channels to mask, the only additional step is to add a guard clause inside the `process_event` member function

```
if(m_HotChannelSet.find(raw) != m_HotChannelSet.end())continue;
```

which has been added to the combined raw data decoder

Current State of Offline Dead Channel Map

- Relevant software modules:
  - Dead Map Loader
  - Dead Map Class
  - Simulation Macro
- These are based on an XML format
  - Strip ID is wrapped in a formatted string and stored as XML string variable
  - `Form("INTT_%d_%d_%d_%d", ladder_phi, ladder_z, strip_z, strip_phi)`
    - Offline/tracking convention
    - I believe this comment was left by Jin; here "`strip_z`" is most like the member field "`strip_y`" in the `Online_s`, and "`strip_phi`" is most like `Online_s`'s "`strip_x`"
  - Existing sPHENIX tools to produce such XML files quickly (phparameter/PHParameters)
- Pointed out that this is inefficient use of disk memory
  - (when compared to ROOT TTrees)

Offline Dead Channel Map Production

- Recently wrote a simple code that produces such xml files, example.cc
  - Note the dependencies on `phool` and `phparameter` libraries
- Haven't merged this to the main INTT branch, since it only servers to prototype things with minimal working examples

Misc. Progress

- For a period my analysis was setback b/c of an update to the event combining
  - Introduced stricter BCO matching against the GL1
  - Requires GL1 files in addition to INTT files
  - Usually corresponding BCOs (in INTT and GL1) were off by one beamclock
- Chris implemented a fix that allows for a BCO matching tolerance
  - `SetBcoRange` in the `SingleInttPoolInput` class
- I've incorporated and tested this in my macro
  - Doing `SetBcoRange(2)` seems to work

Misc. Progress (Cont'd)

- Wrote a `Fun4All` module to directly produce the normalized hitrates needed as input for my channel classification
  - Previous workflow was to produce event-based TTrees and analyze all hits, then iterate over the trees to compute the hitrates
  - Circuitous and wasteful use of disk memory
- Tested the hitrates with a few different values for `SetBcoRange()`, being 1, 2, 3, and 4.
- Found hitrates were identically the same for 2, 3, and 4
- Interestingly hitrates for 1 were slightly higher
  - In any case, matching seemed to stable with 2 beamclock tolerance (further increase did not change matching)
  - I will use 2 for further analysis work

Future Work

- Produce hot channel maps for both Online and Offline workflows
  - Am currently capable of producing the CDBTTrees used for the Decoder, or the XML files used in simulations
  - I can translate any existing maps people have to these formats
- Continue Lambda C analysis
- Help with the implementation of survey geometry in simulation
  - Will likely standardize survey data to CDBTTree as well
  - Mostly trivial changes

Future Work (Cont'd)

- Incorporating survey geometry into GEANT
    - Chris pointed out that the staves can be put into assembly volumes
    - Assembly volumes can act as intermediate/wrappers that represent the half-barrels
    - Then use the volumes to model the gap when closing
    - This will require extra analysis to find the best-fit barrel position, but
        - This will need to happen anyway to properly model inactive area (long term/future implementation)
        - Reduces the total amount of work in implementation to be worth doing
    - Assembly volume (or appropriate volume type) doesn't need to worry about overlap