Feature update to DetectorClocks service for data/simulation overlays

Adi Ashkenazi (MIT) and Wes Ketchum (FNAL)

Overview

- Propose adding a “rebase” functionality for G4RefTime in DetectorClocks to support data/simulation overlays

- Implemented in a similar manner as setting of the trigger time via trigger object
  - Default behavior should remain unchanged

- Feature branches
  - lardata: feature/wketchum_OverlayG4RefCorrection
  - lardataalg: feature/wketchum_OverlayG4RefCorrection

The problem, in general

- Recall that, generally, timings in DetectorClocksService are all referenced against a trigger time, with appropriate offsets tuned via fcl parameters
  - (Also, trigger time read in from a trigger object on the event)

- When overlaying simulation events onto data, there is no guarantee of consistent handling of timing if there is any skew in how the trigger times or offsets are done
  - Generally the timing and offsets allow for self-consistency in one definition of an event's timing scheme, but not robust to handling multiple timing schemes

The problem, in specific

- In MicroBooNE
  - In simulation, TriggerTime is a fixed time after start of particle generation (or readout? … doesn't matter, it's basically fixed) window
    - It's the same for every event, even if the art::Event's time is not the same
  - In data, TriggerTime is a time since the start of a run
    - Which is not the same for every event!

- We encounter an issue in matching particles:

```
/// Given G4 time returns electronics clock count [tdc]
virtual double TPCG4Time2Tick(double g4time) const override
{ return (G4ToElecTime(g4time) - doTPCTime()) /
fTPCClock.TickPeriod(); }
```

The problem, in even more specific

- That function tries to take G4 particles and find where they should be in TPC ticks
- Underneath…
  - `G4ToElecTime(g4time) = g4_time * 1.e-3 - fG4RefTime`
  - `doTPCTime() = TriggerTime() + TriggerOffsetTPC();`
- In uboone simulation, G4RefTime (coming from fcl) and TriggerTime() (coming from a trigger object created in a “triggersim” module) are tuned to work correctly together
- This breaks down if we inherit the trigger time in another way
  - Which we do from data in the overlays

Options

- There may be multiple ways to probe this
  - MicroBooNE could reconsider how it defines TriggerTime() in data
    - This does not feel feasible for us, and still locks us in to something
    - Other experiments should think carefully about this
  - We could try to switch overlays to using the TriggerTime() from simulation rather than data
    - Not dissimilar from the former
    - I think we tried this and things broke before → still locks us in to something
  - Modify the G4RefTime on an event-by-event basis
    - This is the route we went with

Modifying G4RefTime

- Why?
  - Feels like the right thing to do
    - G4RefTime is supposed to be offset of particle times to the trigger, which is the actual problem here
  - Can be done with probably minimal side effects
    - Allows us to continue reconstruction of the event like data with less worry (timing of simulated raw data onto real raw data already handled/validated in previous steps)

- Implication:
  - This is a fundamental change of G4RefTime: it must now vary event-by-event, rather than be constant from fcl

Modifying G4RefTime

- How?
  - Luckily, we have a good example for a mechanism to adjust DetectorClocks event-by-event: the trigger time
  - I've added similar functions:
    - setDetectorClocksStandardG4RefTimeCorrectionFromEvent (and the like)

```
virtual void RebaseG4RefTime(double sim_trig_time)
{ fG4RefTime = fG4RefTimeDefault - TriggerTime() +
sim_trig_time; }
```

  - Note, we keep the fcl as the "G4RefTimeDefault" so we can always recalculate against that
  - TriggerTime() here is from our data trigger module
  - sim_trig_time is a second trigger time read in from a new G4RefCorrTrigModuleName fcl parameter
    - By default, if not set or not there, will not call the rebase function

End of story?

- This works for comparing MCParticles to TPC reco objects in downstream analysis
- It (apparently) does not work for the BackTracker
  - As BackTracking is defined not by G4 times, but by SimChannel times …
  - How to consistently handle this is still being investigated: for now, we are forced to run BackTracking in its own art job with the DetectorClocks configured to use the simulation's trigger object for the TriggerTime()
  - Further investigation will tell…

- BUT
  - These branches are good to go in, and this is important for allowing downstream truth-matching

What is needed at some point…

- We should think about careful consideration of event timingin simulation and data specifically with overlays in mind
  - And specifically realizing that future “events” may have different structures/timing considerations
    - SN stream for DUNE?
- Note: this probably means careful thinking across the board
  - E.g. I don’t think we have hooks for setting the initial simulation time from the time of a data event, but that would have been another potential solution here …
- Opinions welcome!

Upcoming…

- Sorry Adi and I had to cancel this for today, but we would like to give an overview of the MicroBooNE simulation overlays
  - How it's done
  - Problems encountered and how we're trying to get around them
  - Ideas for how to roll out to LArSoft as a whole

- Aim for this in the new year!
  - But of course, feedback welcome now
  - *(My ignorance: Are other experiments doing simulation/data overlays already too? Would be good to share notes…)*