EDM4hep merging

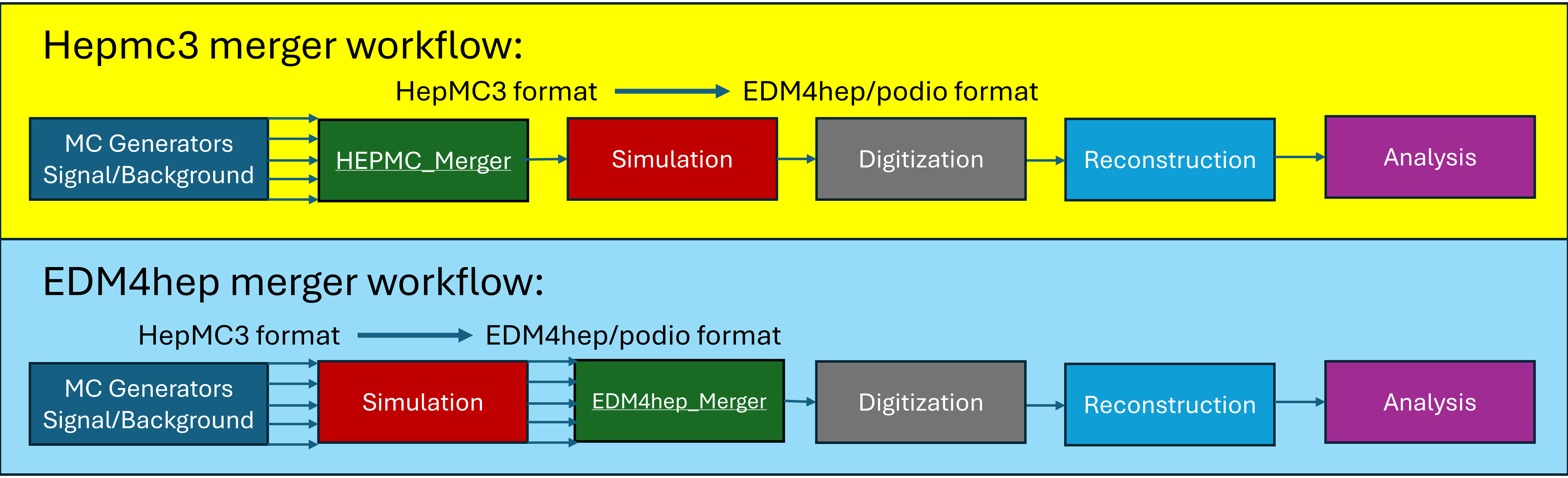
Merging dd4hep simulation output

5th November 2025
Physics and Detector Simulation Meeting
Simon Gardner

Contents

Try in eic shell:
EDM4hep_TimesliceGenerator

Motivation – MC background merging



- Digitization is the first stage MC samples interact
- Merging just before digitization allows the simulated events to be reused

Merging EDM4hep vs HepMC

Advantages

- Only need to simulate each MC event once – Simulation is slow.
- Can use exactly same events to:
    - Build background samples for different beam currents
    - Overlay different simulated signal events onto identical backgrounds
- GeneratorStatus of particles produced in simulation can be controlled.

Disadvantages

- Increased storage - Need to store raw simulation output and merged samples.
- HepMC merging is fast – If simulated events will only ever be used once would be faster
- EDM4hep merging can be slow due to data io rather than processing.
- Workflow becomes less linear
    - Waiting for multiple simulations to finish before reconstruction.
    - Persist simulation output for longer.
- Data per hit and associations make this trickier to build which is why this hasn't been previously available.

Features – Configuration Interfaces

Config File

```
# Example configuration for the standalone podio timeslice merger
output_file: merged_timeslices.root
max_events: 100
time_slice_duration: 2000.0 # in ns
bunch_crossing_period: 40.0 # in ns
introduce_offsets: true
sources:
  - name: signal
    input_files:
      - input3.root
    static_number_of_events: true
    static_events_per_timeslice: 1
    use_bunch_crossing: true
    generator_status_offset: 0
  - name: minbias
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 0.000083 # in GHz
    use_bunch_crossing: true
    generator_status_offset: 0
    repeat_on_eof: true
  - name: bethe_heitler
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 1.8 # in GHz # Check value
    use_bunch_crossing: true
    generator_status_offset: 7000
    repeat_on_eof: true
```

Command line

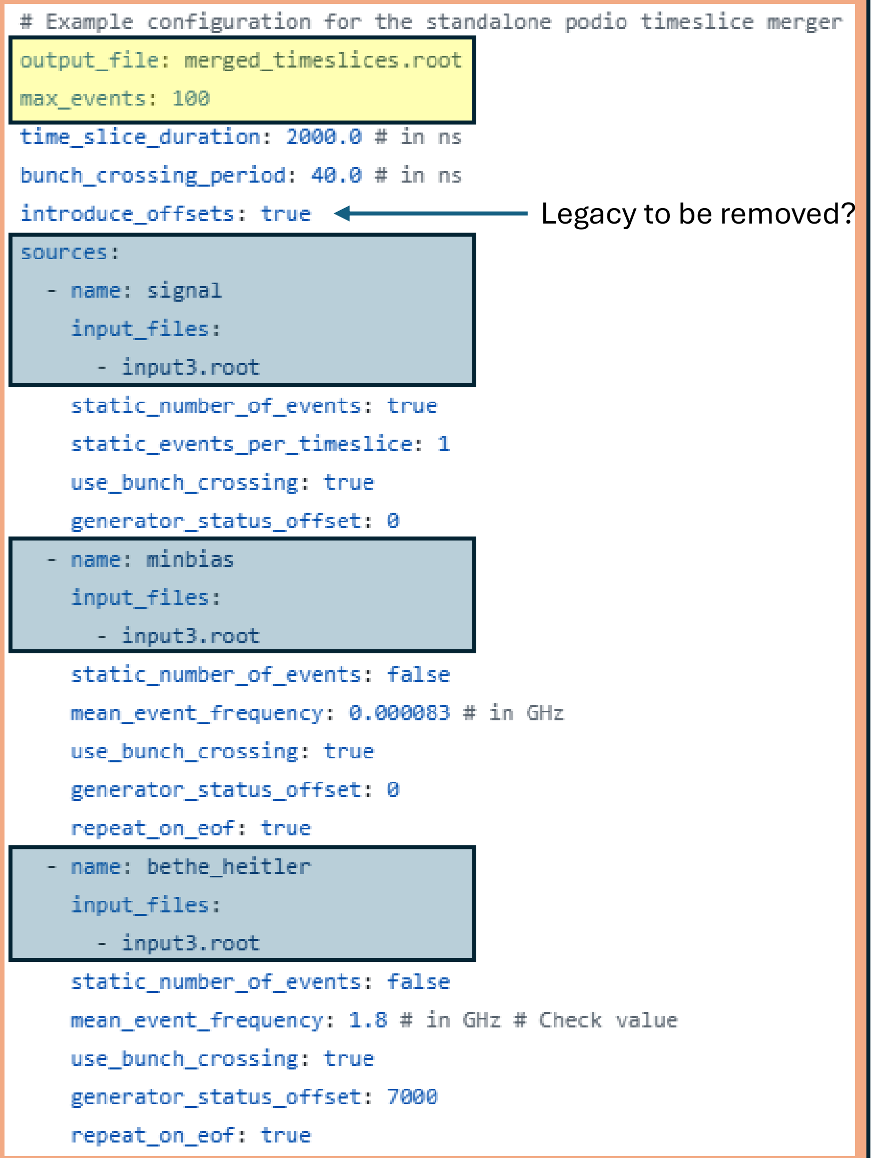
```
# Create signal and background sources with specific settings
./install/bin/timeslice_merger \
  --source:signal:input_files signal1.root,signal2.root \
  --source:signal:frequency 0.5 \
  --source:signal:static_events false \
  --source:background:input_files bg1.root,bg2.root \
  --source:background:static_events true \
  --source:background:events_per_slice 2 \
  --source:background:status_offset 1000
```

Config and command line

```
# Use config file but override specific source configuration
./install/bin/timeslice_merger --config config.yml --source:signal:input_files new_signal.root --source:signal:frequency 0.8
```

Features – I/O Configuration

Yml Config File



```yaml
# Example configuration for the standalone podio timeslice merger
output_file: merged_timeslices.root
max_events: 100
time_slice_duration: 2000.0 # in ns
bunch_crossing_period: 40.0 # in ns
introduce_offsets: true
sources:
  - name: signal
    input_files:
      - input3.root
    static_number_of_events: true
    static_events_per_timeslice: 1
    use_bunch_crossing: true
    generator_status_offset: 0
  - name: minbias
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 0.000083 # in GHz
    use_bunch_crossing: true
    generator_status_offset: 0
    repeat_on_eof: true
  - name: bethe_heitler
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 1.8 # in GHz # Check value
    use_bunch_crossing: true
    generator_status_offset: 7000
    repeat_on_eof: true
```

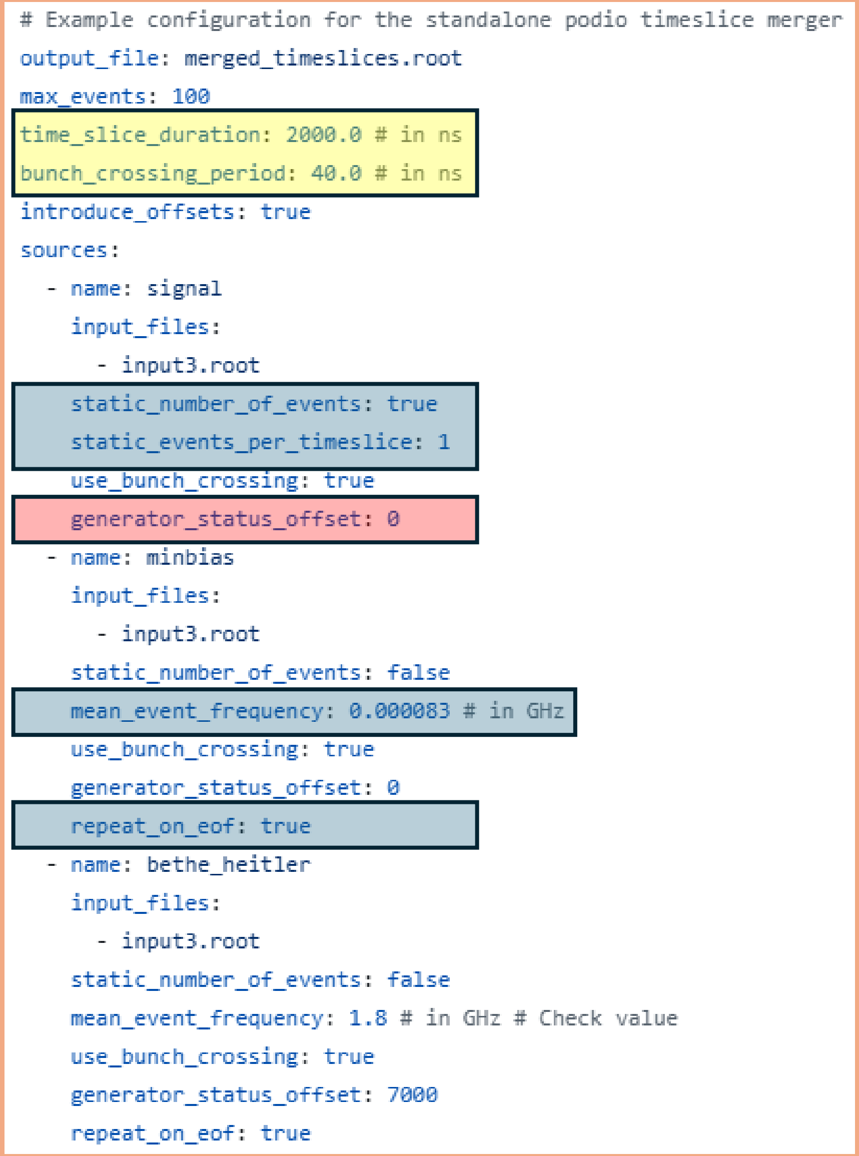
Legacy to be removed?

Output file - Hopefully self explanatory, no subdividing available at the moment.
Max events – Depends on number of events in sources and if looping enabled.

Sources – Any number of named sources can be added
Input Files – Any number of files can be passed to a source

Features – Frequency and Timing

Yml Config File



```yaml
# Example configuration for the standalone podio timeslice merger
output_file: merged_timeslices.root
max_events: 100
time_slice_duration: 2000.0 # in ns
bunch_crossing_period: 40.0 # in ns
introduce_offsets: true
sources:
  - name: signal
    input_files:
      - input3.root
    static_number_of_events: true
    static_events_per_timeslice: 1
    use_bunch_crossing: true
    generator_status_offset: 0
  - name: minbias
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 0.000083 # in GHz
    use_bunch_crossing: true
    generator_status_offset: 0
    repeat_on_eof: true
  - name: bethe_heitler
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 1.8 # in GHz # Check value
    use_bunch_crossing: true
    generator_status_offset: 7000
    repeat_on_eof: true
```

Global Timing

Time slice duration – Period over which an initial random time can be drawn from

Bunch Crossing period – Time to round random event time to. (10ns@5&10 GeV, 40ns@18GeV electron beam)

Sampling

Static number of events – Static/frequency based sample

Static events per timeslice – Actual number to sample

Mean event frequency – Frequency to sample from

Repeat on eof – Allows source to repeat

Generator status offset – Tag MCParticles from source with a GeneratorStatus offset

Features – Frequency and Timing

Yml Config File

```
# Example configuration for the standalone podio timeslice merger
output_file: merged_timeslices.root
max_events: 100
time_slice_duration: 2000.0 # in ns
bunch_crossing_period: 40.0 # in ns
introduce_offsets: true
sources:
  - name: signal
    input_files:
      - input3.root
    static_number_of_events: true
    static_events_per_timeslice: 1
    use_bunch_crossing: true
    generator_status_offset: 0
  - name: minbias
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 0.000083 # in GHz
    use_bunch_crossing: true
    generator_status_offset: 0
    repeat_on_eof: true
  - name: bethe_heitler
    input_files:
      - input3.root
    static_number_of_events: false
    mean_event_frequency: 1.8 # in GHz # Check value
    use_bunch_crossing: true
    generator_status_offset: 7000
    repeat_on_eof: true
```

```
- name: electron_synchrotron
  input_files:
    - input3.root
  static_number_of_events: false
  mean_event_frequency: 3.324 # in GHz
  use_bunch_crossing: true
  attach_to_beam: true
  beam_angle: -3.141592653589793 # in rad
  beam_speed: 0.299792458 # in m/ns
  beam_spread: 0.003 # in ns
  generator_status_offset: 2000
  repeat_on_eof: true
```

Source Timing

Use bunch crossing – Attach an event time to the bunch crossing time period. All that is needed for IP samples otherwise destroy afterburner.

Attach to beam – Adjust time by z offset.

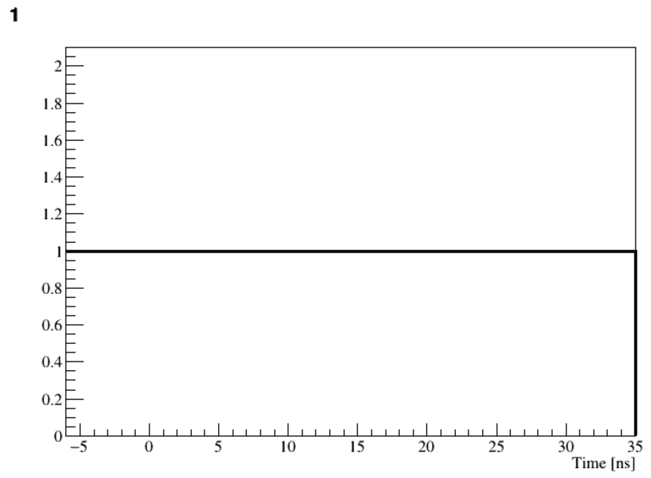
Beam angle – Angle the beam is approaching from.

Beam speed – Speed of the beam, correlate time with offset.

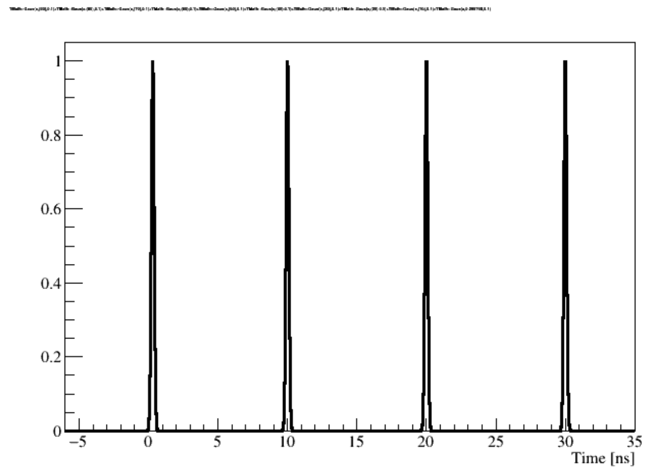
Beam spread – Gaussian time width of the bunch.

Time offsetting

Uniform



Bunch attached

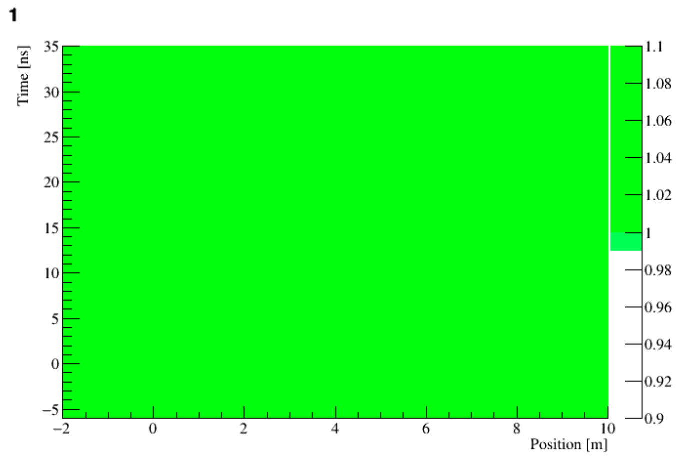


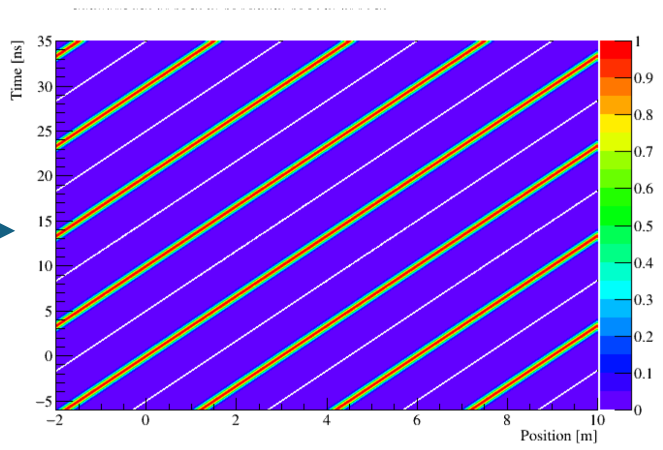
Afterburned events just need attaching to bunch crossing, already have time structure.

- Add uniform offset rounded to bunch crossing period.

Background events need attaching to bunch crossing, offsetting based on position and smearing by bunch shape.

- Add uniform offset rounded to bunch crossing period.
- Offset time based on position along beampipe and velocity
- Random offset by bunch shape (currently Gaussian 300/30ps)





Implementation

ROOT::TTreeReader – Simplest data driven approach which is also efficient.

- Loop over all MCParticle, SimTrackerHit, SimCalorimeterHit collections.
  - Builds a hard coded map of expected associations and names.
- Loop over event sources
  - Loads event from file
    - Create new time offset based on settings (and first generatorStatus==1 MCParticle position)
    - Loop over collections
      - Update MCParticles with GeneratorStatus offset
      - Updating time of MCParticle, SimTrackerHit, CaloHitContribution collections
      - Update association indices based on length of array after previous event.
      - Concatenate vector of data to end of combined vector.
    - Save event
- Abuses edm4hep::EventHeader to keep track of events which came from – Future dedicated data type.

Implementation Journey

- Entirely JANA based merging!
  - Stick a factory at the start of EICrecon which merges samples.
  - Built into the framework we are all familiar with.
  - Configuration and logging included.

- Not actually possible at the moment (2 months ago)
  - Managed to merge events from a single source into a timeframe using JEventUnfolder
  - Not evident how to source from multiple sources, event frames.
    - Collections at the same event level would get synchronised
    - Cannot merge from multiple event levels into one.
  - Quite slow ~1Hz per 20 signal events from one source into timeslice

Implementation Journey

- JANA based source merging followed by podio-merge-files
  - Source merging still in JANA
  - Simple tool for merging together the events from different sources

- Would require 3 step reconstruction
  - JANA merger, podio-merge-files, EICRecon
- **podio-merge-files appends events, doesn’t zip them…**

Implementation Journey

- Entirely podio based merging
  - Source merging still in JANA
  - Simple tool for merging together the events from different sources

- Still slow ~2Hz
  - When loading a frame from a data source podio constructs the C++ data model objects from the saved data
  - 1000s of MCParticles, associations, calorimeter hits contributions

Implementation Journey

- Entirely ROOT based merging
  - Do not need to know the details of the events and links, only need static offsets:
    - Time or the members
    - MCParticle generator status
    - Offset the relation indices by the previous length of the vector.
  - Output works now as an input to EICRecon

- Still slow? 10Hz (But much faster than simulating the same event again)
  - Feels a bit unsave
  - Probably still a way of making this much faster
  - Doesn’t seem to speed up when merging already merged samples together.
  - NB (Real events probably not this slow)

Proposed Improvements and open questions.

Organization

Tidy up repository

Move to eic GitHub organization

Request testers

Benchmark workflows

Implementation

Still feels very slow for what is actually being done

Investigate making podio/edm4hep feature for zipping of data at any level – Will require large revision.

Time Offsetting

Currently hits are kept on the primary event level – Sometime extending far beyond the timeframe.

Timeframe will have hits from events occurring earlier in previous timeframes.

Keep track of global clock offset relative to bunch crossings throughout sequential time slices.

To recreate a realistic timeslice, events need to be simulated outside of the period with understanding of how the DAQ will chop up the data from different detectors.

Benchmarking – to come

- Want to demonstrate the results of this are the same as the HepMC merger but more efficient over multiple samples.

Conclusions

- Ready for testing and to use locally.
- Requires a deeper revision of campaign resources before potentially adoptable as default.