中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

drift chamber multithreaded simulation with Gaudi Hive

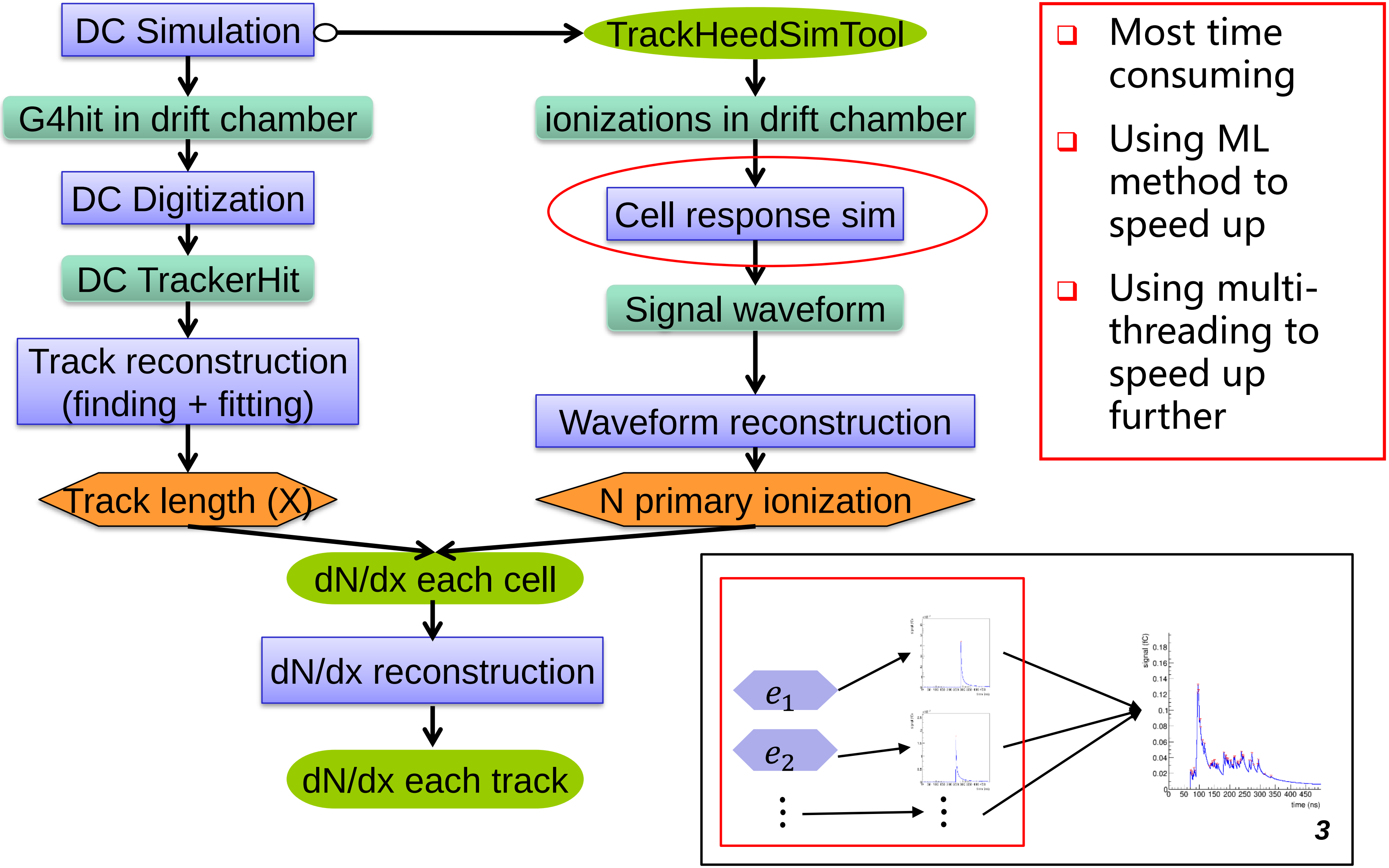
Wenxing Fang (IHEP) Weidong Li (IHEP), Tao Lin (IHEP), Jiaheng Zou(IHEP)

CEPC PhysDet meeting 2021.10.13

Motivation

- CEPC is a precise experiment
  - Higgs, W, Z, …
  - PID performance is important
- From previous study, the primary ionization counting (dN/dx) method has potential to get very good PID performance (<3% resolution)
- To prove that, the dN/dx method will be detailed studied for CEPC drift chamber. Need precise dN/dx simulation

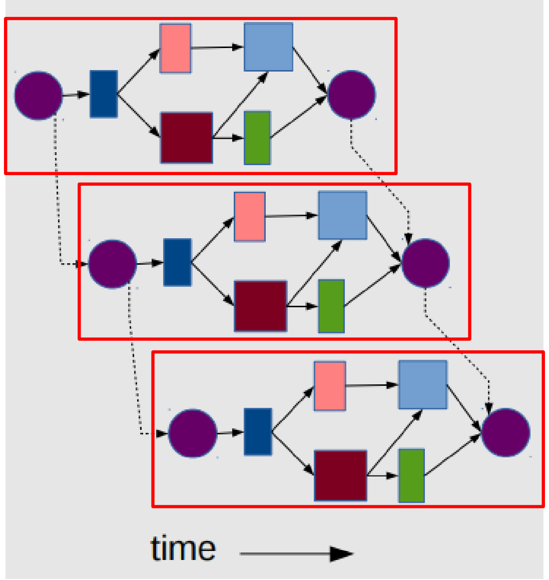
Schema of dN/dx study in CEPCSW



- ❑ Most time consuming
- ❑ Using ML method to speed up
- ❑ Using multi-threading to speed up further

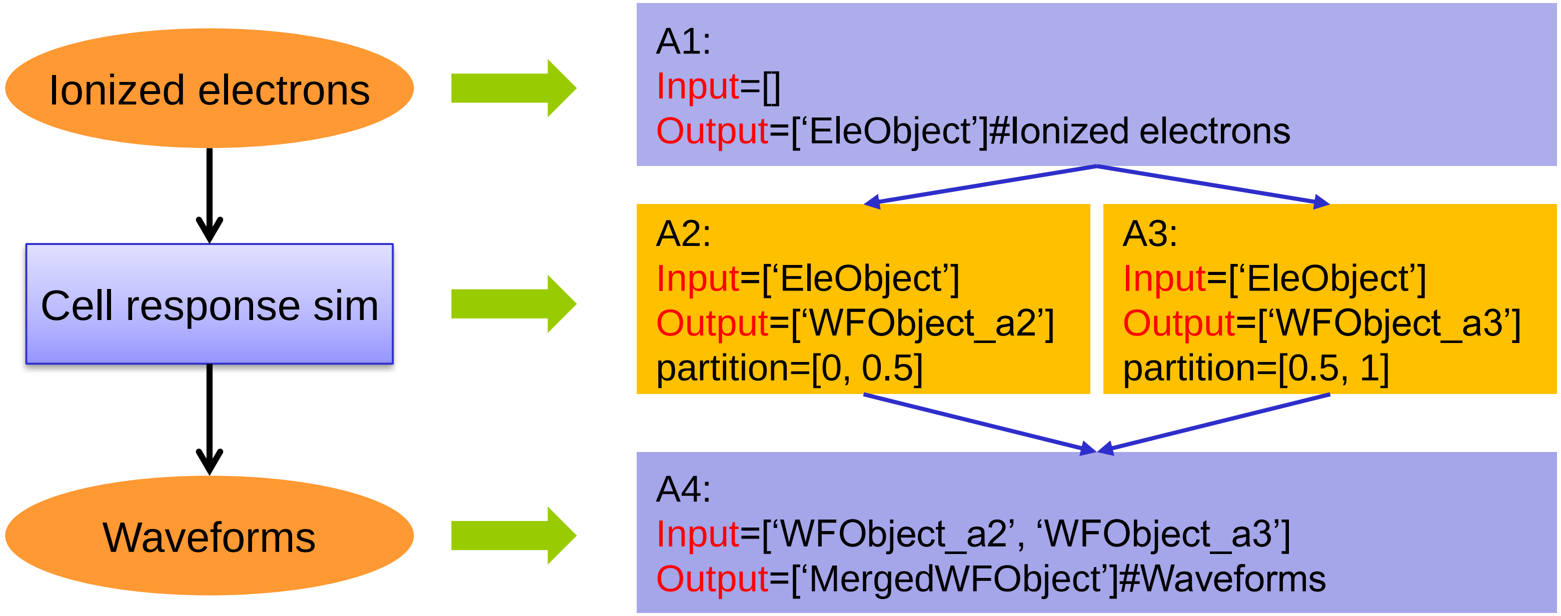
Gaudi Hive

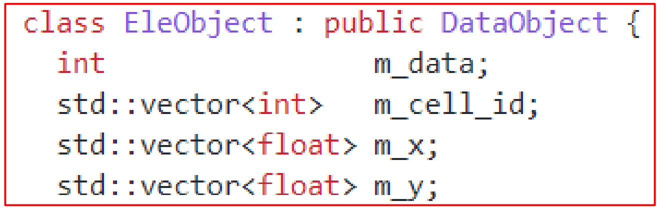
- Gaudi Hive: multi-threaded, concurrent extension to Gaudi
- Data Flow driven mechanism
  - Algorithms declare their data dependencies
    - build a directed acyclic graph - can be used for optimal scheduling
  - Scheduler automatically executes Algorithms as data becomes available
- Algorithms process events in their own thread
- Multiple algorithms and events can be executed simultaneously
- Algorithm Cloning
  - Multiple instances of the same Algorithm can exist, and be executed concurrently, each for different event



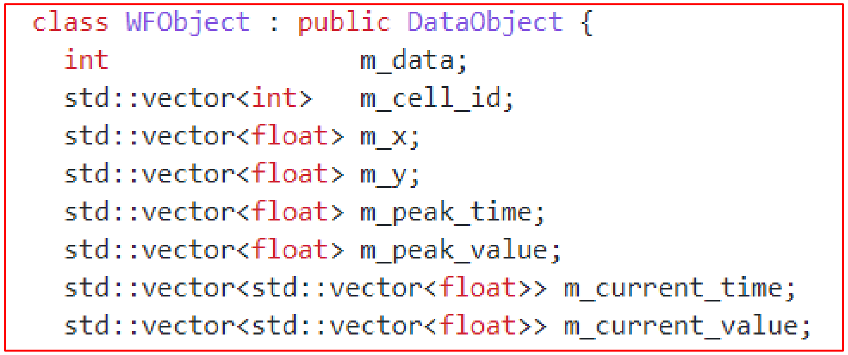
Example using dummy data object

- Performing the study using dummy data object





```
class EleObject : public DataObject {
  int                 m_data;
  std::vector<int>    m_cell_id;
  std::vector<float> m_x;
  std::vector<float> m_y;
```

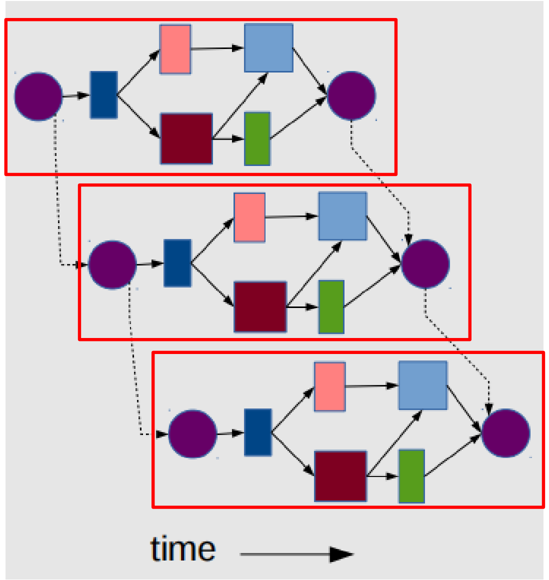


```
class WFObject : public DataObject {
  int                 m_data;
  std::vector<int>    m_cell_id;
  std::vector<float> m_x;
  std::vector<float> m_y;
  std::vector<float> m_peak_time;
  std::vector<float> m_peak_value;
  std::vector<std::vector<float>> m_current_time;
  std::vector<std::vector<float>> m_current_value;
```

- Working well

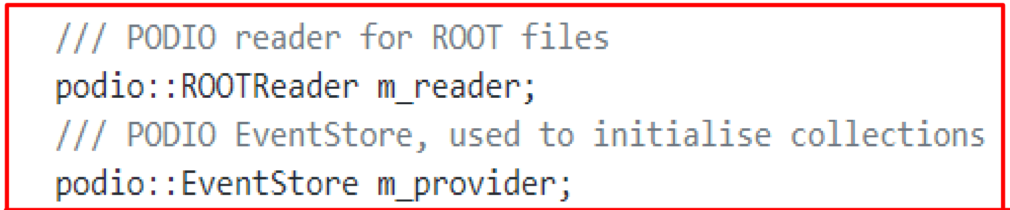
Example using dummy data object

- ❑ Some configurations:
  - ○ evtslots = 3 #number of events run in parallel
  - ○ whiteboard = HiveWhiteBoard("EventDataSvc", EventSlots=evtslots)
  - ○ scheduler = AvalancheSchedulerSvc(ThreadPoolSize=8)
  - ○ A1.Cardinality = 2 # number of instance of A1 after setting isClonable=true

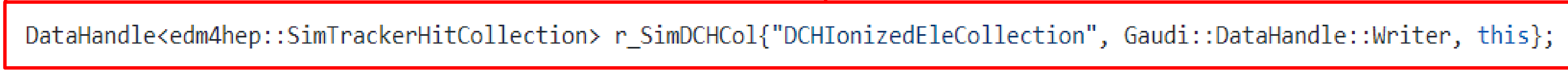


Example with podio input

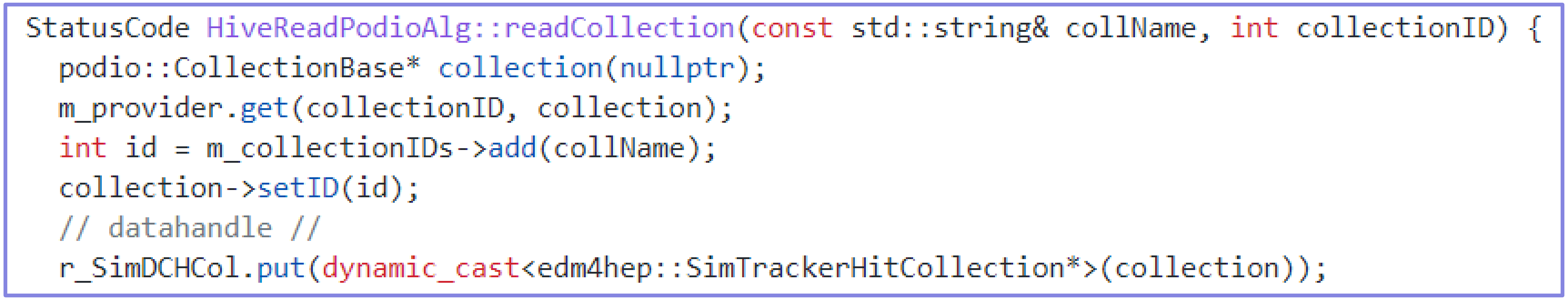
- Read podio data as input
- As the HiveWhiteBoard is used for event data service instead of PodioDataSvc. Currently, need create an algorithm to read podio data. People from key4hep is working on merging HiveWhiteBoard into PodioDataSvc



```
/// PODIO reader for ROOT files
podio::ROOTReader m_reader;
/// PODIO EventStore, used to initialise collections
podio::EventStore m_provider;
```



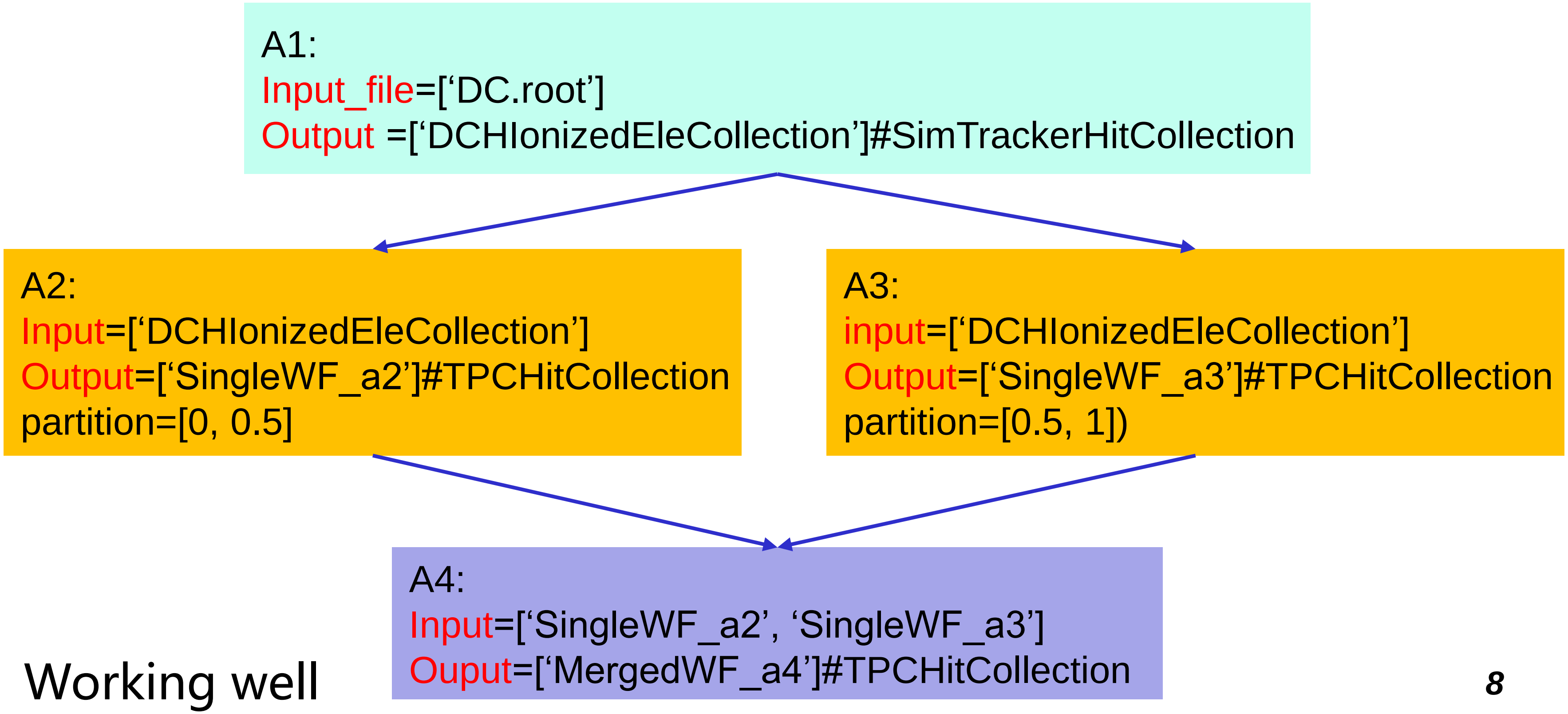
```
DataHandle<edm4hep::SimTrackerHitCollection> r_SimDCHCol{"DCHIonizedEleCollection", Gaudi::DataHandle::Writer, this};
```



```
StatusCode HiveReadPodioAlg::readCollection(const std::string& collName, int collectionID) {
  podio::CollectionBase* collection(nullptr);
  m_provider.get(collectionID, collection);
  int id = m_collectionIDs->add(collName);
  collection->setID(id);
  // datahandle //
  r_SimDCHCol.put(dynamic_cast<edm4hep::SimTrackerHitCollection*>(collection));
```

Example with podio input

- Read podio data as input and using edm4hep for EDM
- For this test, the edm4hep::SimTrackerHit is used for saving information of ionized electrons. The edm4hep::TPCHit is used for saving waveform information



```
A1:
Input_file=[‘DC.root’]
Output =[‘DCHIonizedEleCollection’]#SimTrackerHitCollection
```

```
A2:
Input=[‘DCHIonizedEleCollection’]
Output=[‘SingleWF_a2’]#TPCHitCollection
partition=[0, 0.5]
```

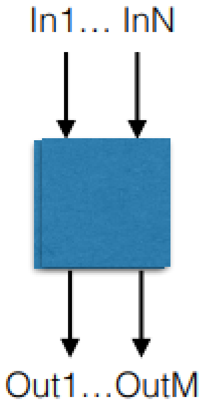
```
A3:
input=[‘DCHIonizedEleCollection’]
Output=[‘SingleWF_a3’]#TPCHitCollection
partition=[0.5, 1])
```

```
A4:
Input=[‘SingleWF_a2’, ‘SingleWF_a3’]
Ouput=[‘MergedWF_a4’]#TPCHitCollection
```

- Working well

Gaudi::Functional

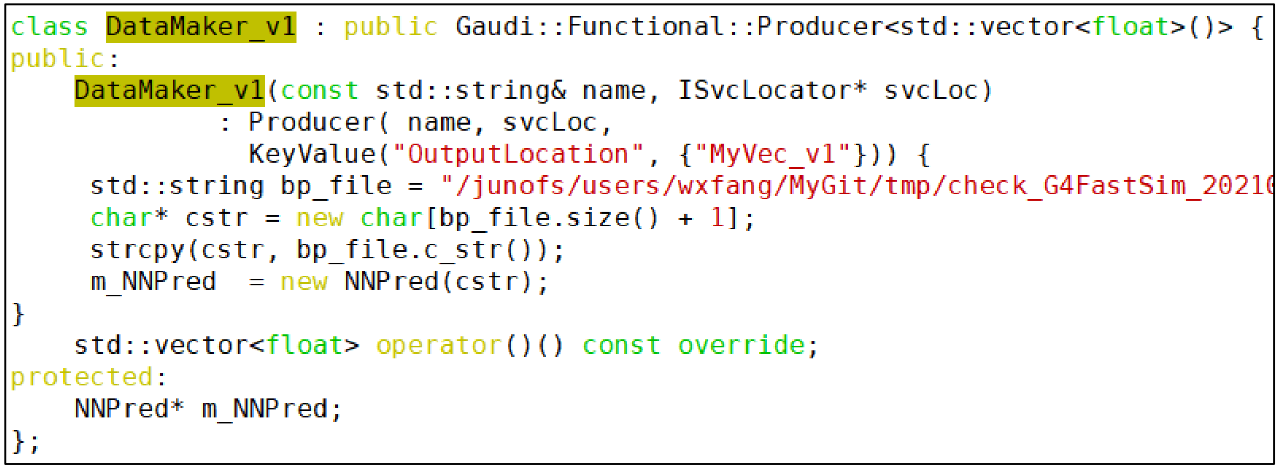
- Most algorithms look like "some data in" -> "some data out"
- Standardize the common pattern of getting data our of the TES, working on it, and putting it back in (in a different location).
  - Less code to write
  - More uniform code and easy to understand
  - Can be Re-Entrant, no need for clone, save memory
  - Multithreading friendly



- Patterns available:
  - Consumer, Producer, Filter, Transformer, MultiTransformer, ScalarTransformer, ...

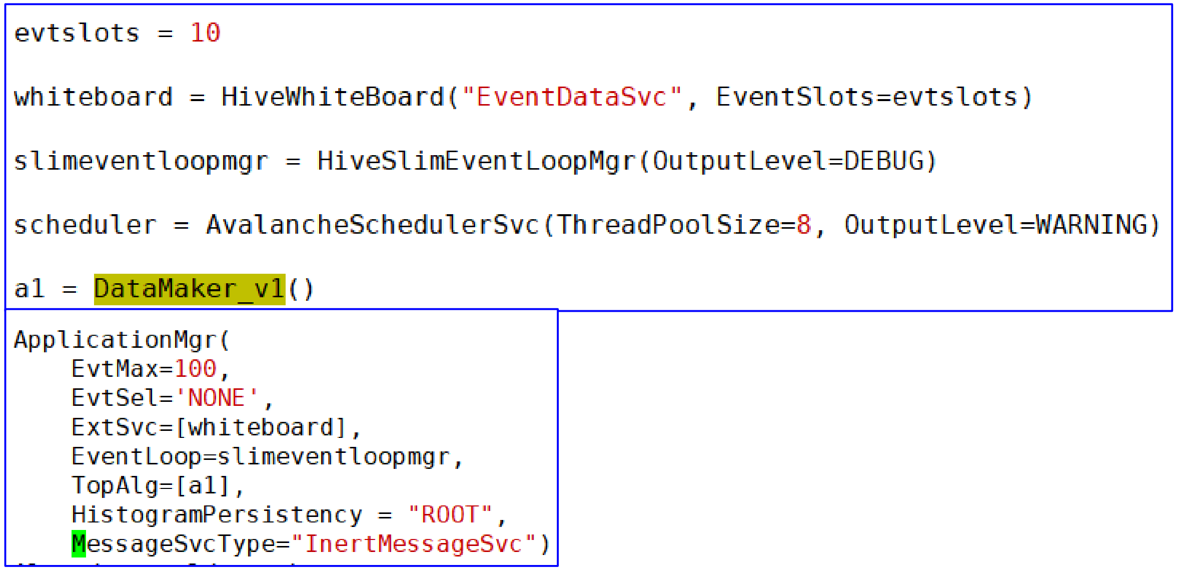
Re-Entrant test

- Gaudi::Functional Re-Entrant test

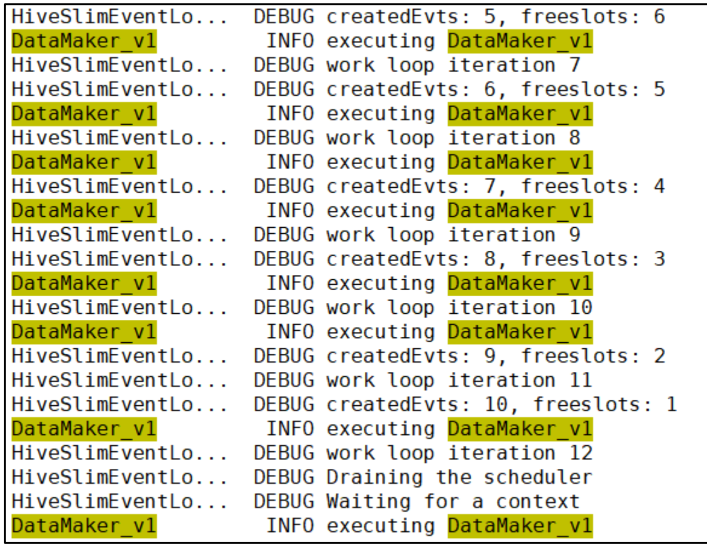


```cpp
class DataMaker_v1 : public Gaudi::Functional::Producer<std::vector<float>()> {
public:
    DataMaker_v1(const std::string& name, ISvcLocator* svcLoc)
             : Producer( name, svcLoc,
               KeyValue("OutputLocation", {"MyVec_v1"})) {
     std::string bp_file = "/junofs/users/wxfang/MyGit/tmp/check_G4FastSim_2021
     char* cstr = new char[bp_file.size() + 1];
     strcpy(cstr, bp_file.c_str());
     m_NNPred  = new NNPred(cstr);
}
    std::vector<float> operator()() const override;
protected:
    NNPred* m_NNPred;
};
```

- ❑ Gaudi::Functional is re-entrantable
- ❑ The pytorch model is re-entrantable

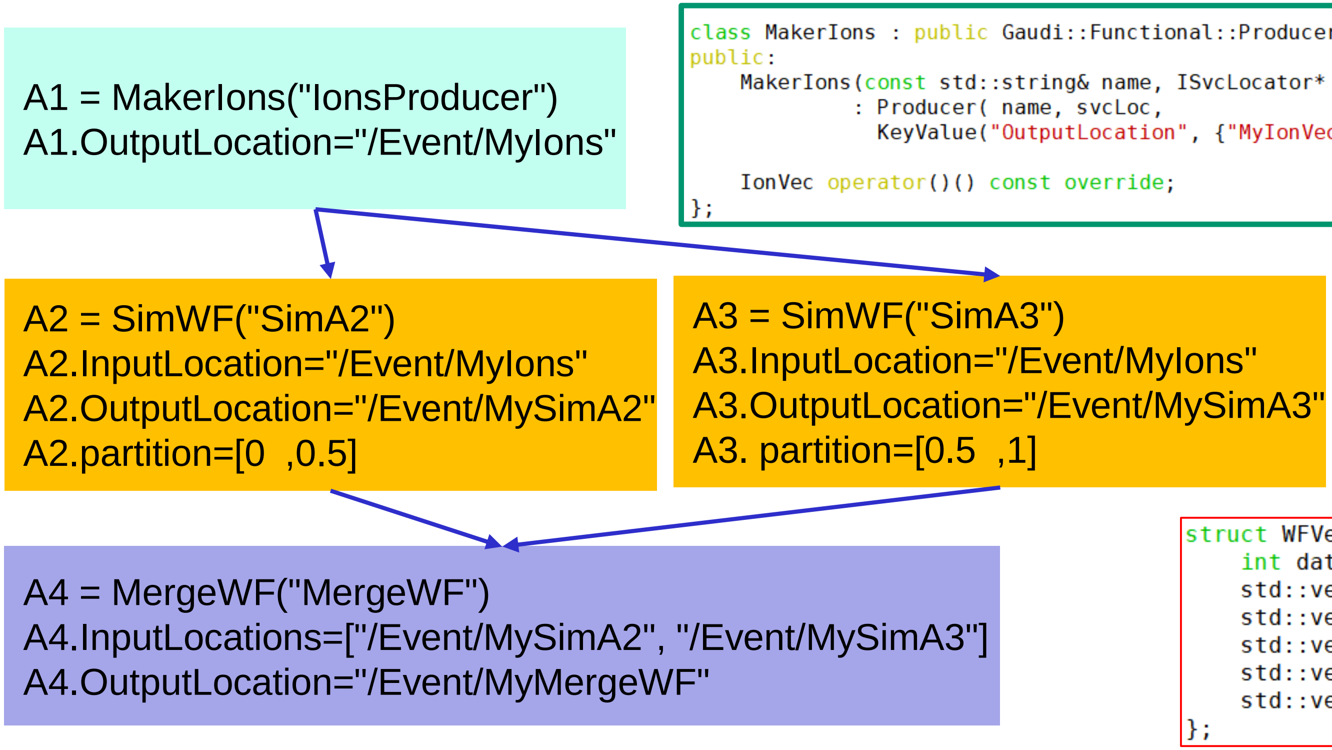


```python
evtslots = 10

whiteboard = HiveWhiteBoard("EventDataSvc", EventSlots=evtslots)

slimeventloopmgr = HiveSlimEventLoopMgr(OutputLevel=DEBUG)

scheduler = AvalancheSchedulerSvc(ThreadPoolSize=8, OutputLevel=WARNING)

a1 = DataMaker_v1()

ApplicationMgr(
    EvtMax=100,
    EvtSel='NONE',
    ExtSvc=[whiteboard],
    EventLoop=slimeventloopmgr,
    TopAlg=[a1],
    HistogramPersistency = "ROOT",
    MessageSvcType="InertMessageSvc")
```



```
HiveSlimEventLo...  DEBUG createdEvts: 5, freeslots: 6
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG work loop iteration 7
HiveSlimEventLo...  DEBUG createdEvts: 6, freeslots: 5
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG work loop iteration 8
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG createdEvts: 7, freeslots: 4
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG work loop iteration 9
HiveSlimEventLo...  DEBUG createdEvts: 8, freeslots: 3
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG work loop iteration 10
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG createdEvts: 9, freeslots: 2
HiveSlimEventLo...  DEBUG work loop iteration 11
HiveSlimEventLo...  DEBUG createdEvts: 10, freeslots: 1
DataMaker_v1         INFO executing DataMaker_v1
HiveSlimEventLo...  DEBUG work loop iteration 12
HiveSlimEventLo...  DEBUG Draining the scheduler
HiveSlimEventLo...  DEBUG Waiting for a context
DataMaker_v1         INFO executing DataMaker_v1
```

Using Gaudi::Functional



A1 = MakerIons("IonsProducer")
A1.OutputLocation="/Event/MyIons"

A2 = SimWF("SimA2")
A2.InputLocation="/Event/MyIons"
A2.OutputLocation="/Event/MySimA2"
A2.partition=[0 ,0.5]

A3 = SimWF("SimA3")
A3.InputLocation="/Event/MyIons"
A3.OutputLocation="/Event/MySimA3"
A3. partition=[0.5 ,1]

A4 = MergeWF("MergeWF")
A4.InputLocations=["/Event/MySimA2", "/Event/MySimA3"]
A4.OutputLocation="/Event/MyMergeWF"

```
class MakerIons : public Gaudi::Functional::Producer<IonVec()> {
public:
    MakerIons(const std::string& name, ISvcLocator* svcLoc)
             : Producer( name, svcLoc,
               KeyValue("OutputLocation", {"MyIonVec"})) {}

    IonVec operator()() const override;
};
```

```
struct IonVec{
    int data;
    std::vector<int> cell_id;
    std::vector<float> x;
    std::vector<float> y;
};
```

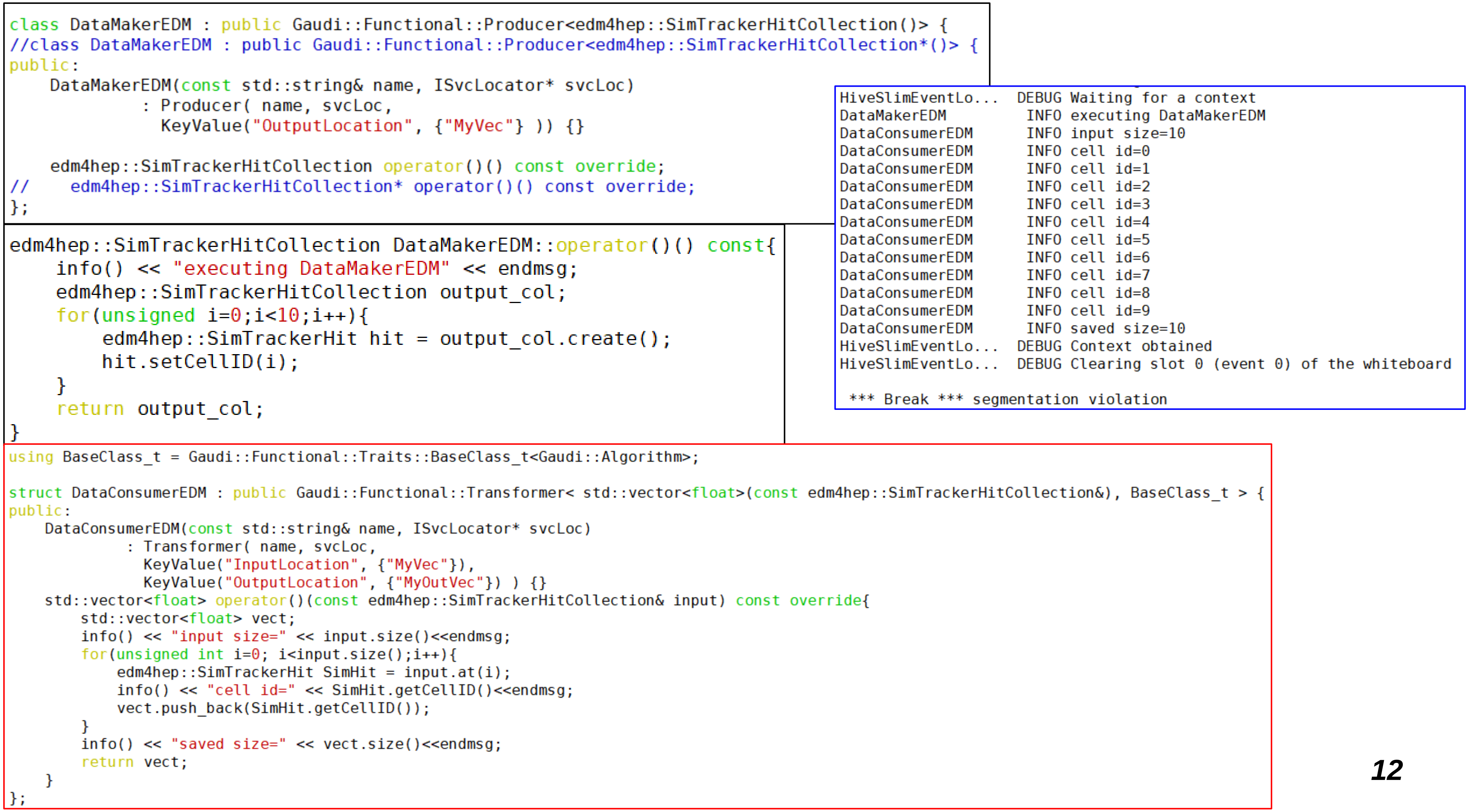
```
struct WFVec{
    int data;
    std::vector<int> e_id;
    std::vector<int> cell_id;
    std::vector<float> peak_time;
    std::vector<float> peak_value;
    std::vector<std::vector<float> > charges;
};
```

```
using BaseClass_t = Gaudi::Functional::Traits::BaseClass_t<Gaudi::Algorithm>;
struct SimWF final : Gaudi::Functional::Transformer<WFVec( const IonVec& ), BaseClass_t> {

  SimWF( const std::string& name, ISvcLocator* svcLoc )
      : Transformer( name, svcLoc, KeyValue( "InputLocation", "/Event/MyInt" ),
                     KeyValue( "OutputLocation", "/Event/MyFloat" ) ) {}
  WFVec operator()( const IonVec& input ) const override {
```

```
struct MergeWF final : Gaudi::Functional::MergingTransformer<WFVec( const Gaudi::Functional::vector_of_const_<WFVec>& ), BaseClass_t> {

  MergeWF( const std::string& name, ISvcLocator* svcLoc )
      : MergingTransformer( name, svcLoc, {"InputLocations", {}},{"OutputLocation", "/Event/MyConcatenatedIntVector"} ) {}

  WFVec operator()( const Gaudi::Functional::vector_of_const_<WFVec>& input ) const override {
```

Using Gaudi::Functional

❖ Using edm4hep data. Data can be accessed correctly. However, at the end of event, error happens



```
class DataMakerEDM : public Gaudi::Functional::Producer<edm4hep::SimTrackerHitCollection()> {
//class DataMakerEDM : public Gaudi::Functional::Producer<edm4hep::SimTrackerHitCollection*()> {
public:
    DataMakerEDM(const std::string& name, ISvcLocator* svcLoc)
             : Producer( name, svcLoc,
               KeyValue("OutputLocation", {"MyVec"} )) {}

    edm4hep::SimTrackerHitCollection operator()() const override;
//    edm4hep::SimTrackerHitCollection* operator()() const override;
};
```

```
edm4hep::SimTrackerHitCollection DataMakerEDM::operator()() const{
    info() << "executing DataMakerEDM" << endmsg;
    edm4hep::SimTrackerHitCollection output_col;
    for(unsigned i=0;i<10;i++){
        edm4hep::SimTrackerHit hit = output_col.create();
        hit.setCellID(i);
    }
    return output_col;
}
```

```
HiveSlimEventLo...  DEBUG Waiting for a context
DataMakerEDM         INFO executing DataMakerEDM
DataConsumerEDM      INFO input size=10
DataConsumerEDM      INFO cell id=0
DataConsumerEDM      INFO cell id=1
DataConsumerEDM      INFO cell id=2
DataConsumerEDM      INFO cell id=3
DataConsumerEDM      INFO cell id=4
DataConsumerEDM      INFO cell id=5
DataConsumerEDM      INFO cell id=6
DataConsumerEDM      INFO cell id=7
DataConsumerEDM      INFO cell id=8
DataConsumerEDM      INFO cell id=9
DataConsumerEDM      INFO saved size=10
HiveSlimEventLo...  DEBUG Context obtained
HiveSlimEventLo...  DEBUG Clearing slot 0 (event 0) of the whiteboard

 *** Break *** segmentation violation
```

```
using BaseClass_t = Gaudi::Functional::Traits::BaseClass_t<Gaudi::Algorithm>;

struct DataConsumerEDM : public Gaudi::Functional::Transformer< std::vector<float>(const edm4hep::SimTrackerHitCollection&), BaseClass_t > {
public:
    DataConsumerEDM(const std::string& name, ISvcLocator* svcLoc)
             : Transformer( name, svcLoc,
               KeyValue("InputLocation", {"MyVec"}),
               KeyValue("OutputLocation", {"MyOutVec"}) ) {}
    std::vector<float> operator()(const edm4hep::SimTrackerHitCollection& input) const override{
        std::vector<float> vect;
        info() << "input size=" << input.size()<<endmsg;
        for(unsigned int i=0; i<input.size();i++){
            edm4hep::SimTrackerHit SimHit = input.at(i);
            info() << "cell id=" << SimHit.getCellID()<<endmsg;
            vect.push_back(SimHit.getCellID());
        }
        info() << "saved size=" << vect.size()<<endmsg;
        return vect;
    }
};
```

Summary

- The Gaudi Hive is studied for multithreaded simulation of drift chamber
- User defined or edm4hep format data is supported in Gaudi Hive
- Using Gaudi::Functional instead of Algorithm have been tried, finding problems with edm4hep data, under investigation

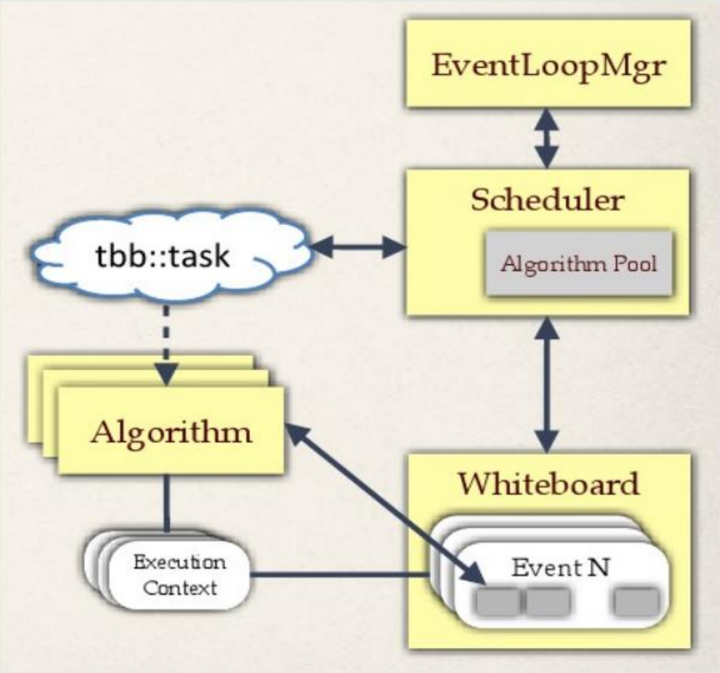
- Future plan:
  - Try to write the output to root files
  - Combining with Geant4 simulation
  - Integrating with k4FWCore, maybe develop a multithreading version of k4FWCore
  - Creating a prototype of CEPCSW based on GaudiHive
- Welcome to check the code: https://github.com/wenxingfang/DCMTSim

Back up

Gaudi Hive Operation

Charles Leggett
September 18 2015

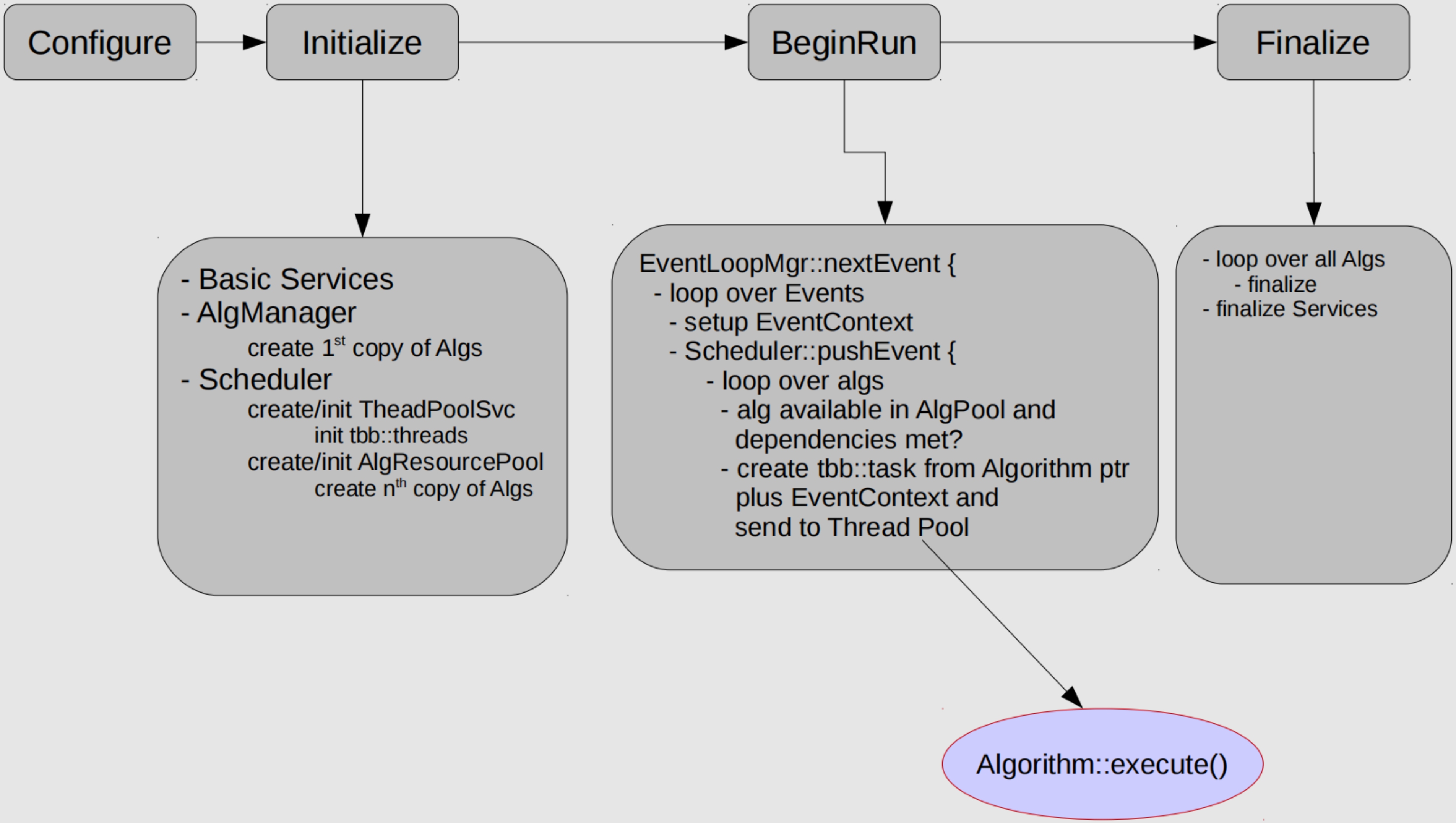
- Configuration, Initialization, Finalization are performed serially in "master" thread
  - only Algorithm::execute is concurrent
- Algorithms are scheduled when data becomes available
  - Algorithms must declare their inputs at initialization or dynamically with DataHandles
  - tbb::task wraps the pair (Algorithm*, EventContext)
- Several instances of the same Algorithm can co-exist
  - **cloning**: create new instance if can be scheduled, and all other instances busy
  - running on different events
  - managed by AlgoPool



- Multiple events are managed simultaneously
  - increases probability of scheduling an Algorithm
  - whiteboard DataStore is thread safe

Execution Flow

Charles Leggett
September 18 2015



Configure

Initialize

BeginRun

Finalize

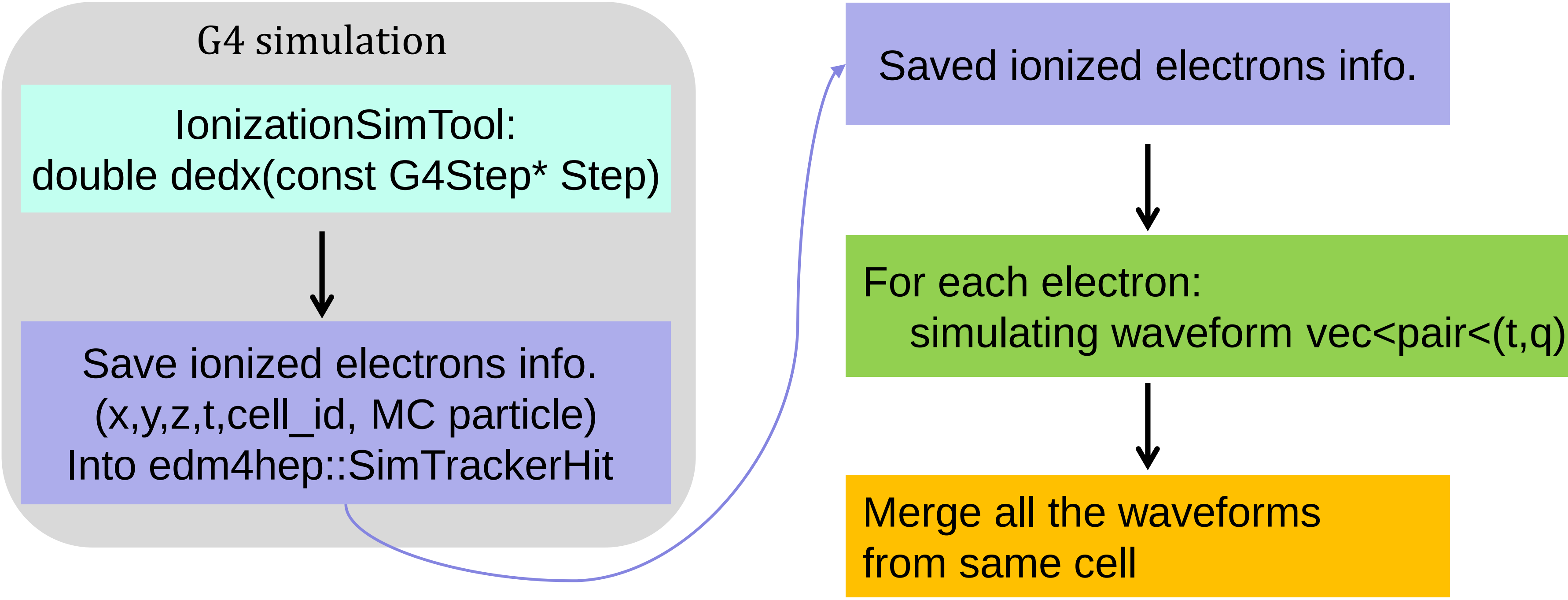
- Basic Services
- AlgManager
    create 1st copy of Algs
- Scheduler
    create/init TheadPoolSvc
        init tbb::threads
    create/init AlgResourcePool
        create nth copy of Algs

```
EventLoopMgr::nextEvent {
  - loop over Events
    - setup EventContext
    - Scheduler::pushEvent {
        - loop over algs
          - alg available in AlgPool and
            dependencies met?
          - create tbb::task from Algorithm ptr
            plus EventContext and
            send to Thread Pool
```

- loop over all Algs
    - finalize
- finalize Services

Algorithm::execute()

Ionization and waveform simulation



G4 simulation

IonizationSimTool:
double dedx(const G4Step* Step)

Save ionized electrons info.
(x,y,z,t,cell_id, MC particle)
Into edm4hep::SimTrackerHit

Saved ionized electrons info.

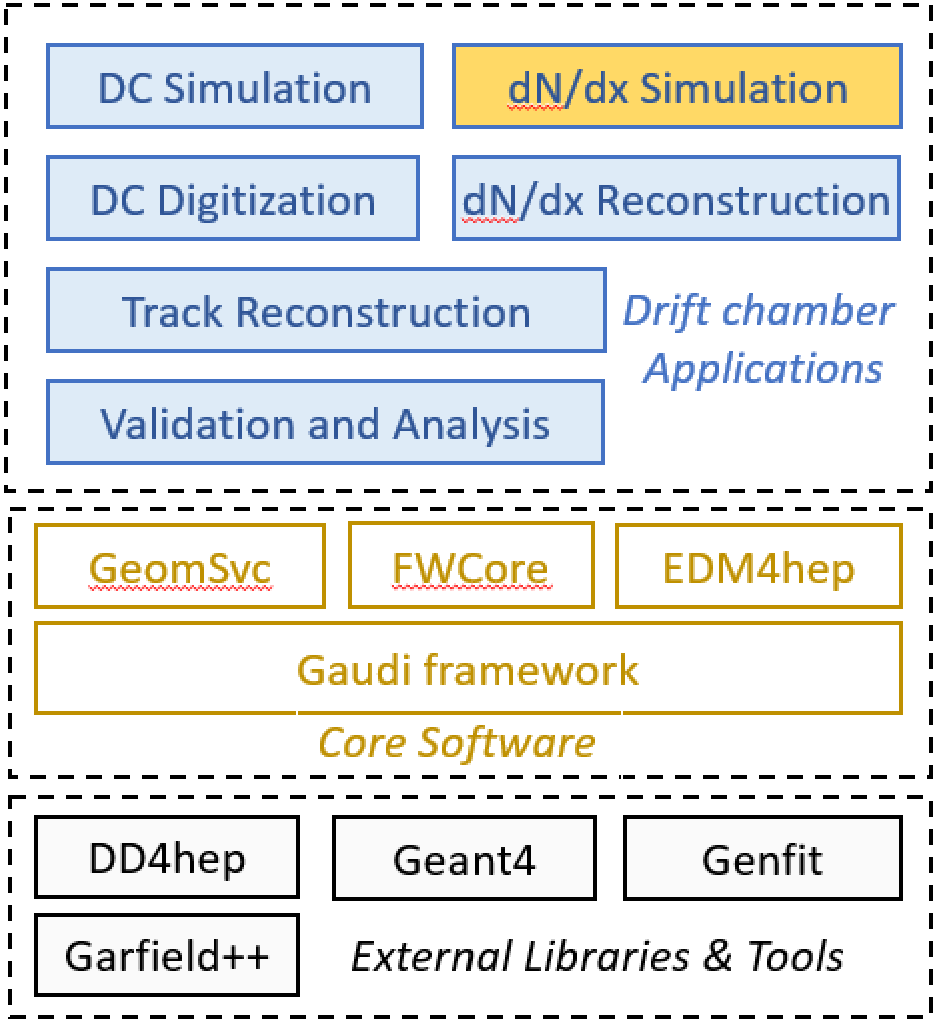
For each electron:
simulating waveform vec<pair<(t,q)

Merge all the waveforms
from same cell

- ❖ Most time consuming part:
  - ➢ Simulate waveform for each electron
  - ➢ For one cell, ~ 100 electrons:
    - o Using Garfield++: ~250 s
    - o Using ML fast simulation: ~1 s
- ❖ CEPC drift chamber is ~100 layers, for one track ~ 100 s. Need further speed up. Using multithreading or GPU technique.

CEPCSW for drift chamber

- Framework:
  - Gaudi
- EDM:
  - EDM4hep
  - FWCore
- Detector geometry and B field:
  - DD4hep
  - GeomSvc
- Drift chamber:
  - DC simulation (Geant4)
  - DC digitization
  - Track reconstruction (Genfit)
  - dN/dx simulation (Garfield++)
  - dN/dx reconstruction



https://github.com/cepc/CEPCSW