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

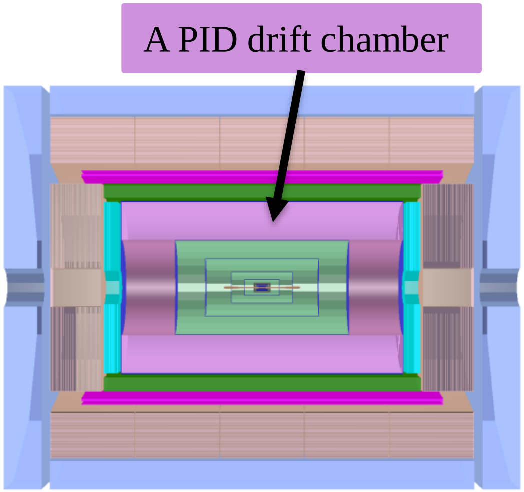
dN/dX study chain in the CEPCSW

Wenxing Fang (IHEP)
on behalf of the CEPC software working group

The CEPC PhysDet plenary meeting (2022.10.12)

Introduction

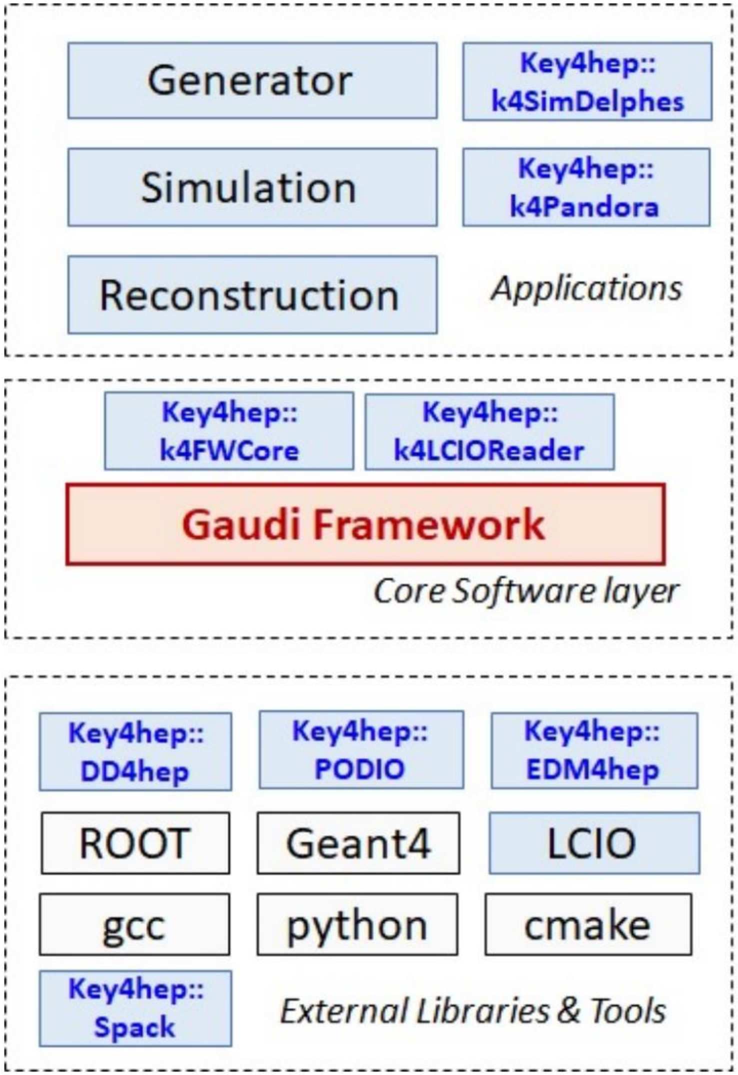
- CEPC aims to measure the Higgs boson precisely, and a good PID performance essential
- As shown previously, the dN/dx method has great potential for PID. In the 4th Concept CEPC detector design, the drift chamber is adopted for tracking and PID (mainly the dN/dx)



- To check the dN/dx performance in more detail, studying dN/dx using full simulation of the CEPC detector should be supported
- This talk will present the chain of dN/dx study in the CEPCSW

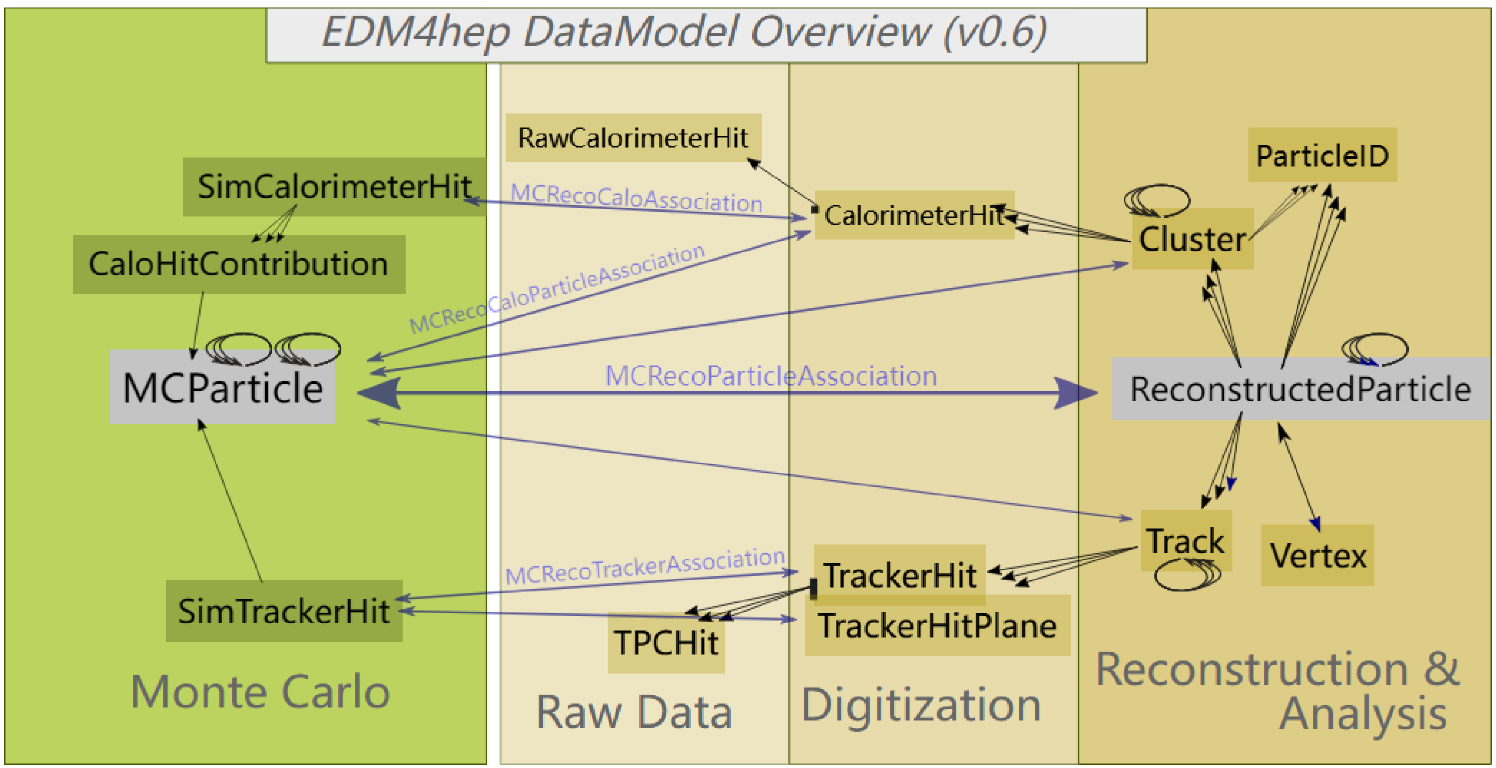
CEPCSW Software

- CEPCSW software structure
  - External libraries:
    - DD4hep: complete detector description (geometry, B field, Material, …). Consistent description (simulation, reconstruction, analysis)
    - EDM4hep: the generic event data model for HEP experiments (see next slide)
    - …
  - Core software:
    - Gaudi framework: defines interfaces to all software components and controls their execution
    - K4FWCore: data service for EDM4hep
  - Applications:
    - CEPC-specific software: generator, Gean4 simulation, reconstruction, and analysis



EDM4hep

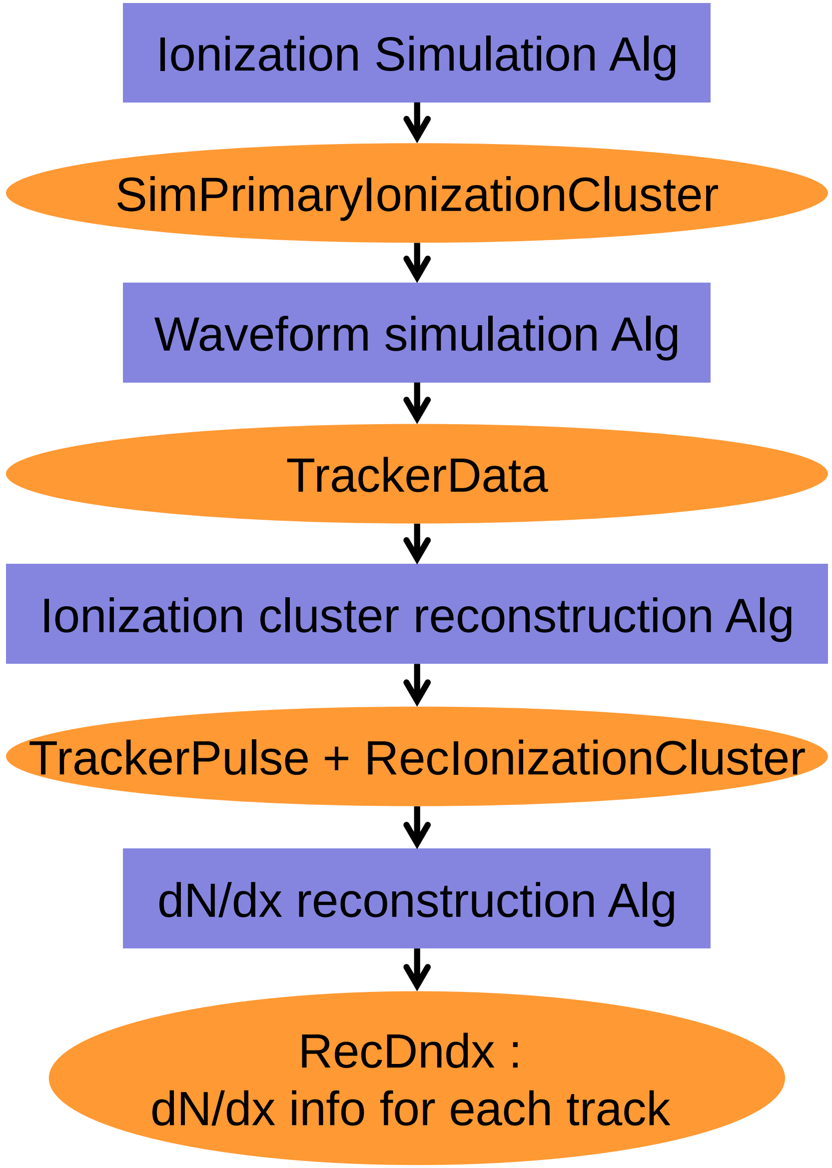
- Common EDM: ILC, FCC, CEPC, CLIC, …
- Efficiently implemented (fast data access, efficient memory usage)
- Support multi-threading
- Potentially heterogeneous computing
- Easy to generate the C++ code from a high-level description of the desired EDM (YAML file) using the podio



EDM4hep Extension

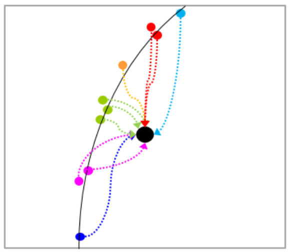
- Currently, the EDM4hep does not include the EDM for dN/dx study, we extended it by using the extension mechanism of podio (very convenient)
- Following EDMs are extended (more details in following slides):
  - SimPrimaryIonizationCluster
  - TrackerData
  - TrackerPulse
  - RecIonizationCluster
  - RecDndx
- The extended EDM is supposed to be used both for the drift chambers and the TPC

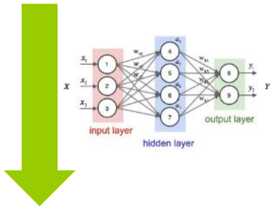
Chain of dN/dx study in CEPCSW

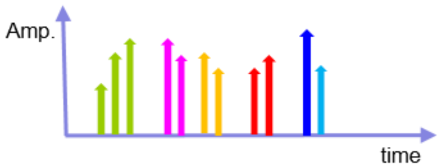


Ionization simulation

- The ionization simulation is done by combining Geant4 and TrackerHeed
  - TrackerHeed (from Garfield++) used for ionization process simulation
  - Geant4 for particle propagation (decay) in the detector, interaction with detector material, …
- Pulse simulation for each ionized electron
  - The Garfield++ simulation takes a long time
  - NN is used for fast simulation, simulating the time and amplitude of each pulse (ONNX runtime for inference)
- More details in this [talk]



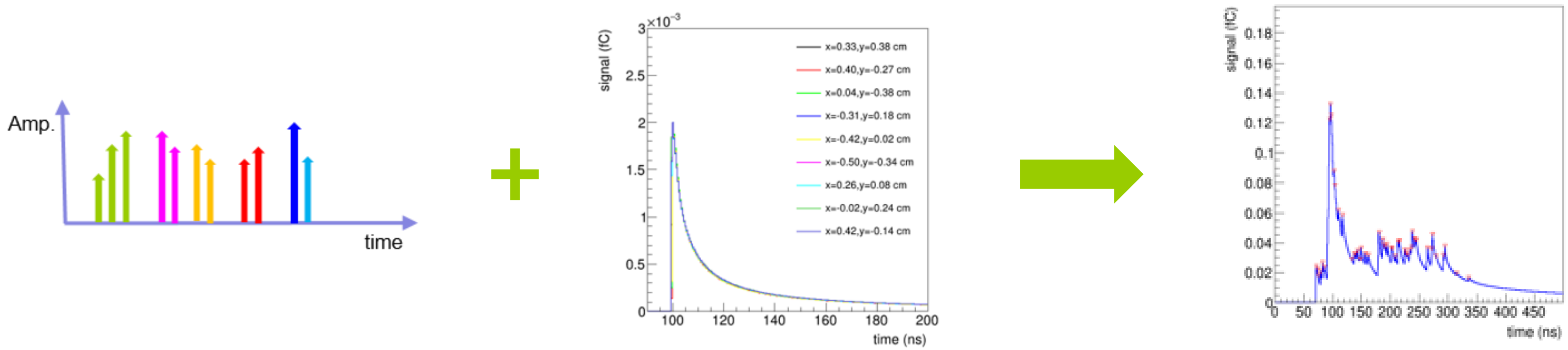





```
edm4dc::SimPrimaryIonizationCluster:
  Description: "Simulated Primary Ionizati
  Author : "Wenxing Fang, IHEP"
  Members:
    - unsigned long long cellID
    - float time
    - edm4hep::Vector3d position
    - int    type
  VectorMembers:
     - unsigned long long electronCellID
     - float electronTime
     - edm4hep::Vector3d electronPosition
     - float pulseTime
     - float pulseAmplitude
  OneToOneRelations:
    - edm4hep::MCParticle MCParticle
```

Waveform simulation

- From Garfield++ simulation, it was found that the normalized pulse shapes are quite similar, the differences between pulses are the time and amplitude
- Using the simulated pulse time and amplitude together with the pulse shape template, the waveform can be easily simulated

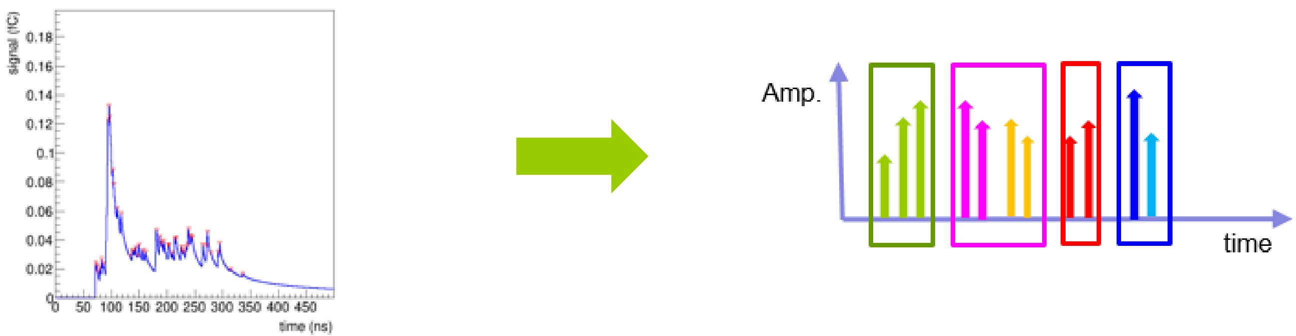


- To be more realistic, effects from the electronic noise and electronic response can be introduced to the waveform

```
edm4dc::TrackerData:
  Description: "TrackerData"
  Author : "Wenxing Fang, IHEP"
  Members:
      - unsigned long long cellID
      - float time
      - float interval
  VectorMembers:
      - float chargeValue
```

Ionization cluster reconstruction

- Using simulated waveform as input. Firstly, it reconstructs pulses (peak finding, derivative, deconvolution, NN, ...). Then it clustering the reconstructed pulses into several ionization clusters (time window, NN, ...)
- Outputs: reconstructed pulses and ionization clusters



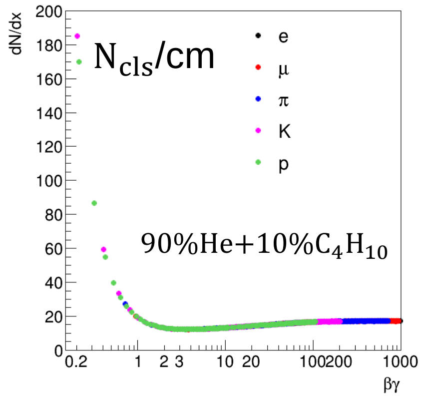

```
#----------  TrackerPulse
 edm4dc::TrackerPulse:
   Description: "Reconstructed Tracker Pulse"
   Author : "Wenxing Fang, IHEP"
   Members:
      - unsigned long long cellID       //cell
      - float time                      //time
      - float charge                    //char
      - int quality                     //qual
   VectorMembers:
      - float covMatrix                 //cova
   OneToOneRelations:
     - edm4dc::TrackerData trackerData  //Opti
```

```
#----------  RecIonizationCluster
 edm4dc::RecIonizationCluster:
   Description: "Reconstructed Ionization Cluster"
   Author : "Wenxing Fang, IHEP"
   Members:
      - unsigned long long cellID  //cell id.
      - float significance         //significance.
      - int type                   //type.
   OneToManyRelations:
      - edm4dc::TrackerPulse trackerPulse //the Tra
```

dN/dx reconstruction

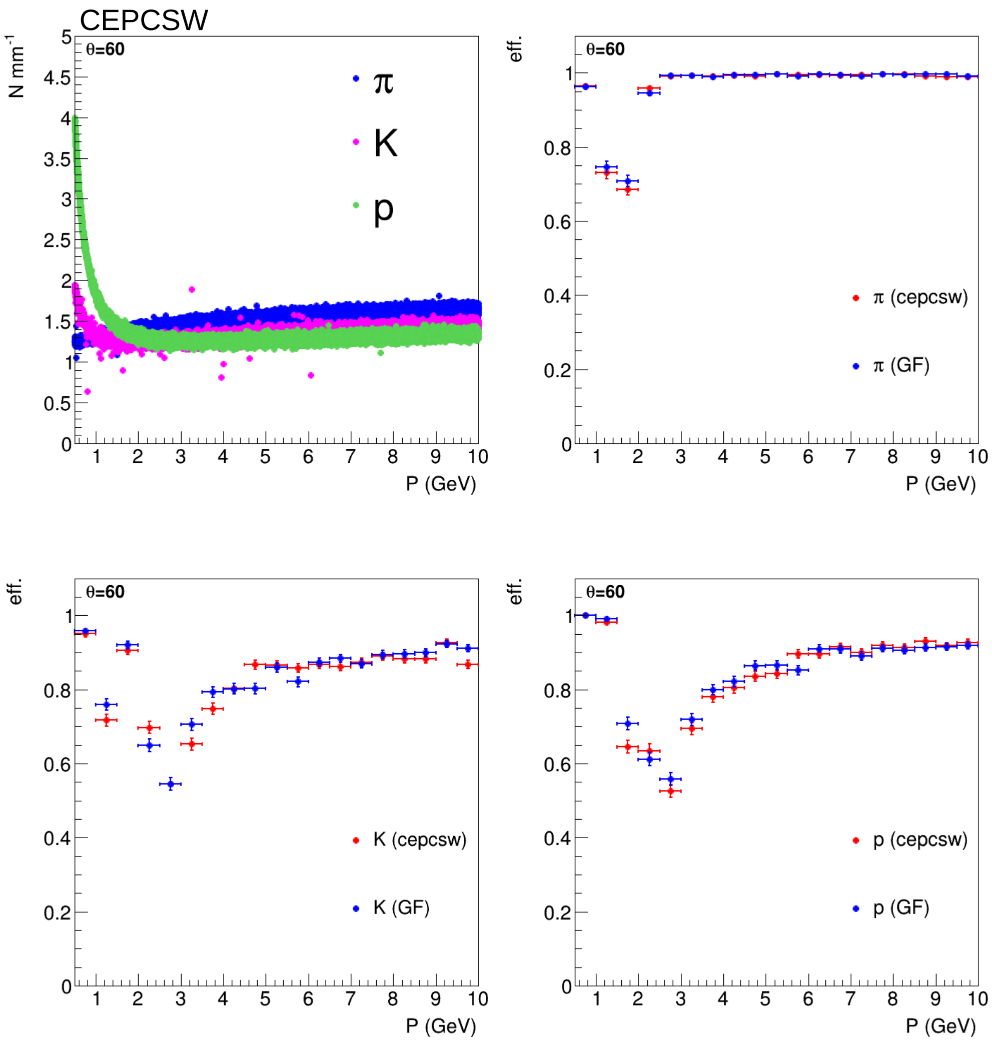
- Inputs: the reconstructed track and reconstructed ionization cluster
- From the reconstructed track, one can get the track length in each drift chamber cell (dX). And the reconstructed ionization cluster gives the number of clusters in each cell (dN)
- The dN/dx for each cell can be calculated. The truncated mean method could be used to calculate dN/dx for each track
- Output: RecDndx including the dN/dx, particle type, and chi for different particle hypotheses, …

```yaml
#-------------  RecDndx
edm4dc::RecDndx:
  Description : "dN/dx info of Track."
  Author : "Wenxing Fang, IHEP"
  Members :
    - float dNdx                    //the reconstructed dNdx.
    - float dNdxError               //error on the dNdx.
    - int particleType              //particle type, e(0),mu(1),pi(2),K(3),p(4).
    - int type                      //type.
  VectorMembers:
    - unsigned long long  cellID    //cell id.
    - float  N                      //number of reconstructed ionization cluster.
    - float  edep                   //energy deposite.
    - float  pathL                  //path length in [mm].
    - float  chi                    //chi for e(0), mu(1), pi(2), K(3), p(4).
    - float  dNdxExpect             //expected dNdx for e(0),mu(1), pi(2), K(3), p(4).
    - float  dNdxSigma              //expected sigma of the dNdx for e(0),mu(1), pi(2), K(3), p(4).
  OneToOneRelations:
    - edm4hep::Track track          //the corresponding track.
```



https://github.com/wenxingfang/CEPCSW/blob/master/Edm/edm4dc.yaml

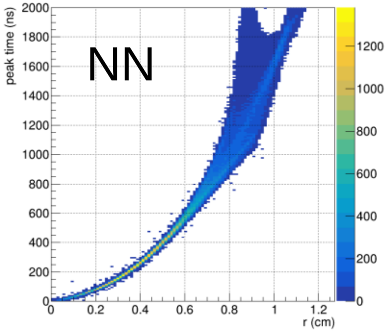
Preliminary dN/dx PID results



- Checked the dN/dx PID performance for gas ($90\%\mathrm{He}+10\%\mathrm{C_4H_{10}}$) using CEPCSW and Garfield++
- Using MC truth information (number of clusters, tracker length)
- The PID performance obtained in CEPCSW has good agreement with the standalone Garfield++ simulation

More be to studied

- Ionizations from secondary particles, backgrounds
- effect from track reconstruction
  - using a more realistic drift time (X-T) simulation



- Check the performance of different pulse and ionization cluster reconstruction algorithms
- Due to the space charge effect can not be simulated by Garfield++. This effect may be extracted from experimental data and considered in the dN/dx reconstruction stage

Summary

- The chain of dN/dx study in CEPCSW is presented
- To support the dN/dx study, the EDM4hep is extended and can be used both by drift chamber and TPC
- The preliminary results for dN/dx PID performance in CEPCSW are checked, they are in good agreement with the results from the standalone Garfield++ simulation

- More to be studied

Back up

Reminder

- Presented the talk at the last EDM4HEP meeting
- One comment is that if it is possible to also incorporate TPC, as TPC has a similar EDM
- Checked with MarlinTPC, it is similar to our design, so the TrackerData and TrackerPulse can be incorporated

EUDET-Report-2007-04

MarlinTPC

Data structure	Processor name	input/output collection name
`TrackerRawData`		TPCRawData
	TrackerRawDataToDataConverterProcessor	
`TrackerData`		TPCConvertedRawData
	PedestalSubtractorProcessor	
	TimeShiftCorrectorProcessor	
`TrackerData`		TPCData
	PulseFinderProcessor	
	ChannelMapperProcessor	
	CountsToPrimaryElectronsConverterProcessor	
`TrackerPulse`		TPCPulses
	HitTrackFinderTopoProcessor	
`TrackerHit`, `Track`		TPCHits, TPCTrackCandidates
	TrackSeederProcessor	
`Track`		TPCSeedTracks
	TrackFitterLikelihoodProcessor	
`Track`		TPCTracks

Table 1: Present MarlinTPC reconstruction processors

TPC

Better to rename the "edm4hep::TPCHit"

Public Member Functions

	TrackerRawDataImpl () Default Constructor - initializes all data to 0's.
virtual	**~TrackerRawDataImpl** () Destructor.
virtual int	**id** () const Returns an object id for internal (debugging) use in LCIO.
virtual int	**getCellID0** () const Returns the first detector specific (geometrical) cell id.
virtual int	**getCellID1** () const Returns the second detector specific (geometrical) cell id.
virtual int	**getTime** () const Returns the time.
virtual const EVENT::ShortVec &	**getADCValues** () const The measured ADC values.
void	**setCellID0** (int cellID0)
void	**setCellID1** (int cellID1)
void	**setTime** (int time)
void	**setADCValues** (const EVENT::ShortVec &adc) Set the ADC vector by copying the values.
EVENT::ShortVec &	**adcValues** () Allows direct access to the adc vector.

Protected Attributes

int	**_cellID0**
int	**_cellID1**
int	**_channelID**
int	**_time**

Public Member Functions

	TrackerDataImpl () Default Constructor - initializes all data to 0's.
virtual	**~TrackerDataImpl** () Destructor.
virtual int	**id** () const Returns an object id for internal (debugging) use in LCIO.
virtual int	**getCellID0** () const Returns the first detector specific (geometrical) cell id.
virtual int	**getCellID1** () const Returns the second detector specific (geometrical) cell id.
virtual float	**getTime** () const Returns the time.
virtual const EVENT::FloatVec &	**getChargeValues** () const The calibrated ADC values.
void	**setCellID0** (int cellID0)
void	**setCellID1** (int cellID1)
void	**setTime** (float time)
void	**setChargeValues** (const EVENT::FloatVec &charge) Set the charge vector by copying the values.
EVENT::FloatVec &	**chargeValues** () Allows direct access to the charge vector.

```
#----------  TPCHit
edm4hep::TPCHit:
  Description: "Time Projection Chamber Hit"
  Author : "F.Gaede, DESY"
  Members:
    - uint64_t cellID  //detector specific cell id.
    - int32_t quality           //quality flag for the hit.
    - float time              //time of the hit.
    - float charge            //integrated charge of the hit.
  VectorMembers:
    - int32_t rawDataWords          //raw data (32-bit) word at i.
```

Public Member Functions

	TrackerPulseImpl () Default Constructor - initializes all data to 0's.
	TrackerPulseImpl (const TrackerPulseImpl &) default copy constructor - use with care
TrackerPulseImpl &	**operator=** (const TrackerPulseImpl &) default assignment operator - use with care
virtual	**~TrackerPulseImpl** () Destructor.
virtual int	**id** () const Returns an object id for internal (debugging) use in LCIO.
virtual int	**getCellID0** () const Returns the first detector specific (geometrical) cell id.
virtual int	**getCellID1** () const Returns the second detector specific (geometrical) cell id.
virtual float	**getTime** () const The time of the pulse.
virtual float	**getCharge** () const The integrated charge of the pulse // FIXME: unit ?.
virtual const EVENT::FloatVec &	**getCovMatrix** () const Covariance matrix of the charge (c) and time (t) measurements.
virtual int	**getQuality** () const The quality bit flag of the pulse - use the defined constants for referring to the bits.
virtual EVENT::TrackerData *	**getTrackerData** () const Optionally the TrackerData that has been uesed to create the pulse can be stored with the pulse - NULL if none.
void	**setCellID0** (int cellID0)
void	**setCellID1** (int cellID1)
void	**setTime** (float time)
void	**setCharge** (float charge)
void	**setCovMatrix** (const float *cov)
void	**setCovMatrix** (const EVENT::FloatVec &)
void	**setQuality** (int quality)
void	**setQualityBit** (int bit, bool val=true)
void	**setTrackerData** (EVENT::TrackerData *corrData)

Protected Attributes

int	**_cellID0**
int	**_cellID1**
float	**_time**
float	**_charge**
int	**_quality**
EVENT::FloatVec	**_cov**
EVENT::TrackerData *	**_corrData**

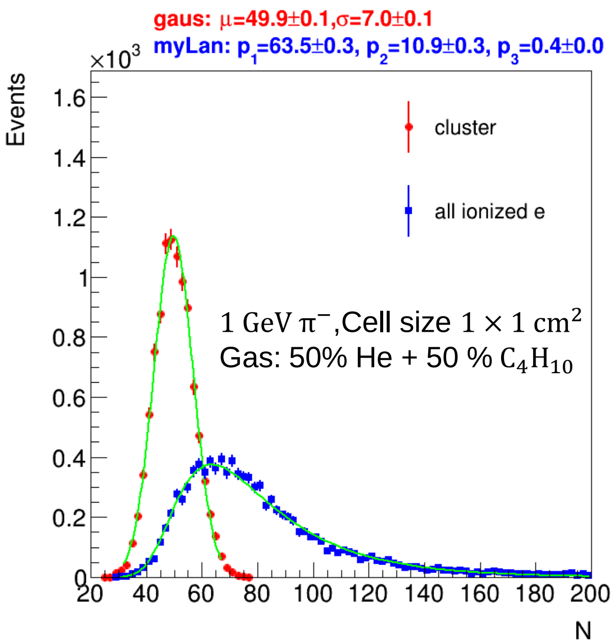
ilcsoft

Introduction

- As the dN/dx method has great potential for PID, studying dN/dx using full simulation of CEPC detector should be supported
- Try to develop the chain of dN/dx study based on CEPCSW
- CEPCSW is fully integrated with the key4hep, and the edm4hep is used for the event data model
- Currently, edm4hep does not include EDM for drift chamber study
- Try to develop a common EDM for the drift chamber based on PODIO

Motivation

- The particle identification is very important for CEPC flavor physics study. Good hadron separation up to 20 GeV is essential
- Traditionally: using dE/dx method
  - Due to the production of delta electron, the deposited energy follows Landau distribution
  - Resolution is ~6%
- New technique: using dN/dx (cluster counting) method
  - The number of primary ionization follows Poisson distribution
  - Resolution could reaches <3%
- The dN/dx technique will be widely explored in CEPC drift chamber detector



User extension data in EDM4hep

- As there is no waveform data format in EDM4hep yet, user extension data is a way to add additional data.
  - WIP: https://github.com/key4hep/EDM4hep/pull/117

Tao Lin

The proposed underlying data structure:

```
edm4hep::UserExt:
    Description: "A simple struct with user defined int/float/double"
    Author : "Tao Lin"
    VectorMembers:
      - int valI // data int
      - float valF // data float
      - double valD // data double
```

The proposed user APIs:

```
ud xyzi;
xyzi.reg("x", 1, 0)
    .reg("y", 1, 1)
    .reg("z", 1, 2)
    .reg("t", 2, 0)
    .reg("i", 0, 0);
```

Runtime Type definition

```
xyzi.from(usrexts[i], 0)
    .get("x", x)
    .get("y", y)
    .get("z", z)
    .get("t", t)
    .get("i", iii);
```

Getters

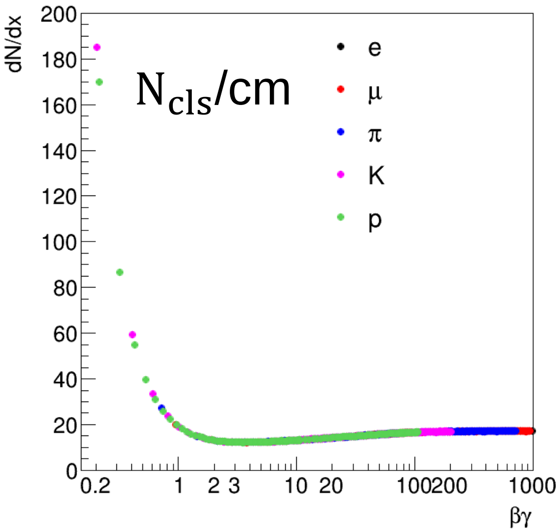
```
xyzi.put("x", x)
    .put("y", y)
    .put("z", z)
    .put("t", t)
    .put("i", i);
```

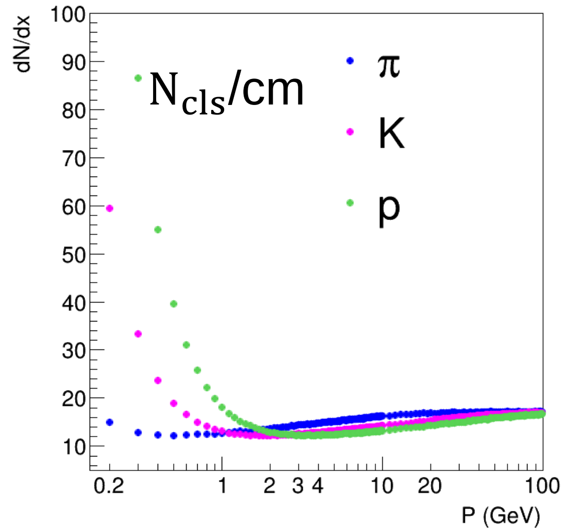
```
auto udv = usrexts.create();

xyzi.to(udv);
```

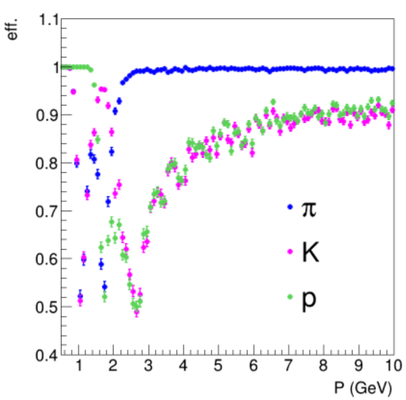
Setters

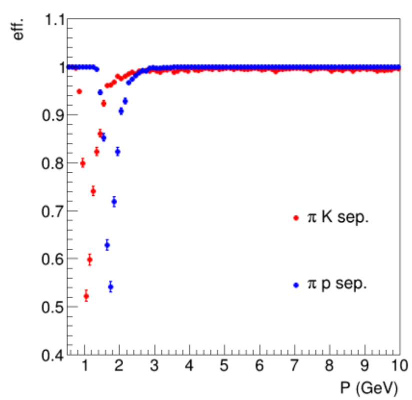
Garfield++ simulation

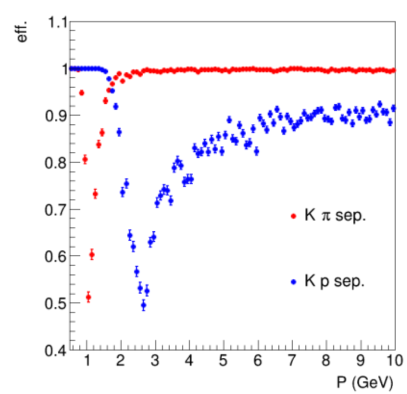
$90\%He+10\%C_4H_{10}$

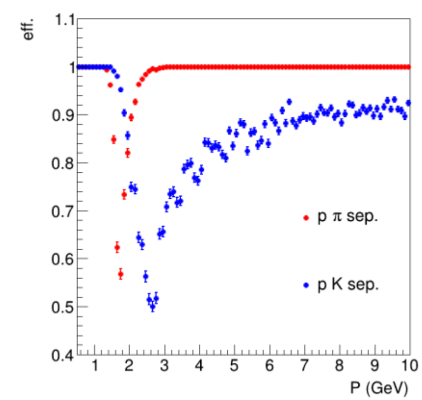












1 meter length