# EDM4hep: the Data Model in Key4HEP

Jiaheng Zou

2024.05.18

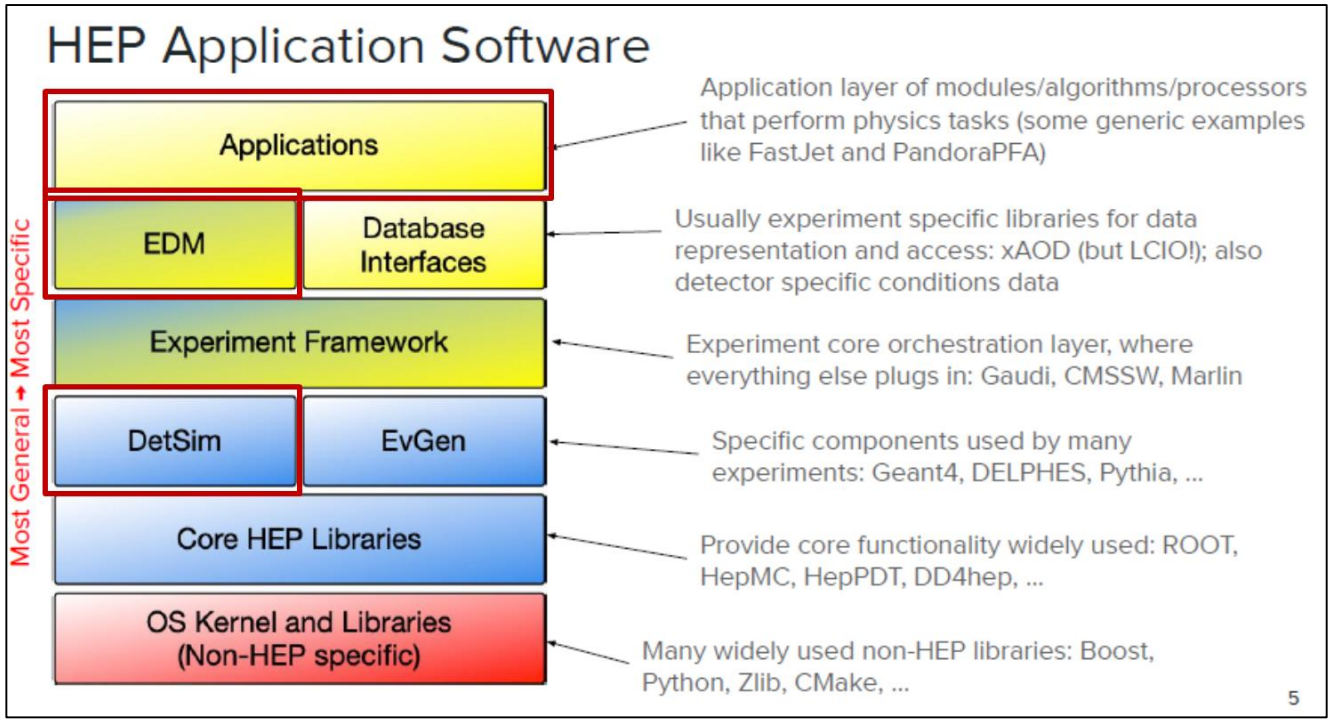
# Content

- The origin of Key4HEP and EDM4hep
- PODIO
- An explore to the EDM4hep

➢ My own understanding to the EDM4hep from the view of an user

# From CSS to Key4HEP

- The common view at the Bolognia workshop, June 2019
  - A common solution for future collider experiments, including CEPC, CLIC, FCC, ILC …
  - Maximize the sharing of software components between experiments
  - CSS: Common Software Stack
- Key4HEP
  - CHEP2019 (Nov. 2019)
  - A Turnkey Software Stack for HEP Experiment
- EDM
  - In the layer between FW and apps

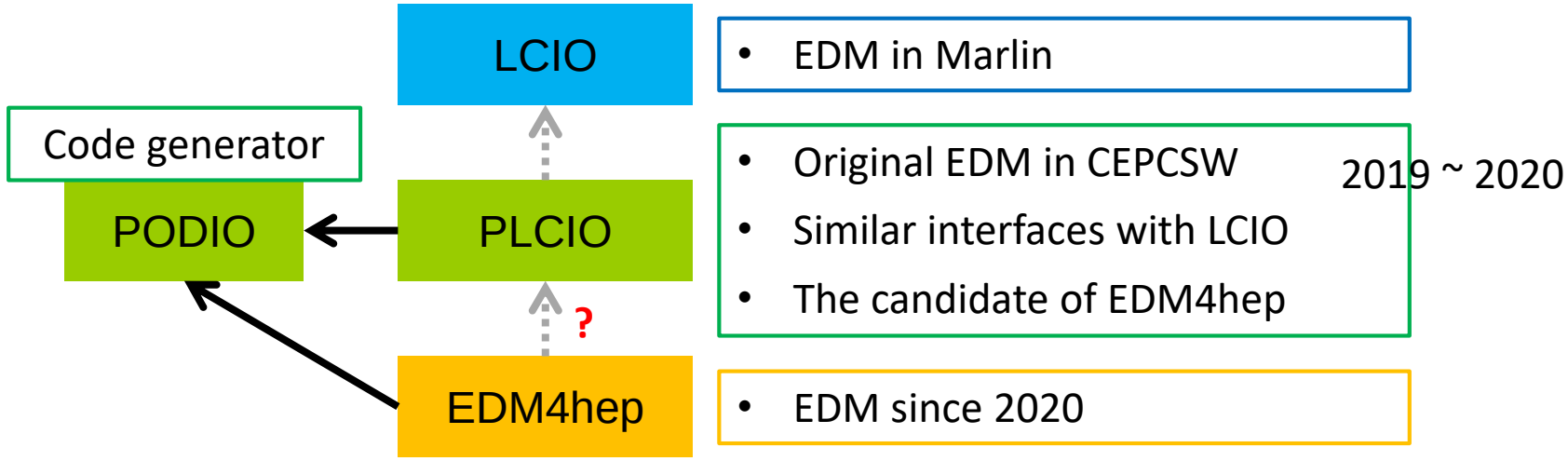



[Ref]: André Sailer, etc. , CHEP2019

https://indico.cern.ch/event/773049/contributions/3474763/attachments/1938664/3213633/191105_sailer_key4hep.pdf

# From PLCIO to EDM4hep

- PLCIO is an implementation of the LCIO event data model in PODIO
- PLCIO was used by CEPCSW before the first version of EDM4hep is released




- CEPCSW is the first user of PLCIO
  - Some missing classes – implemented by ourselves

# PODIO: an Event-Data Model toolkit

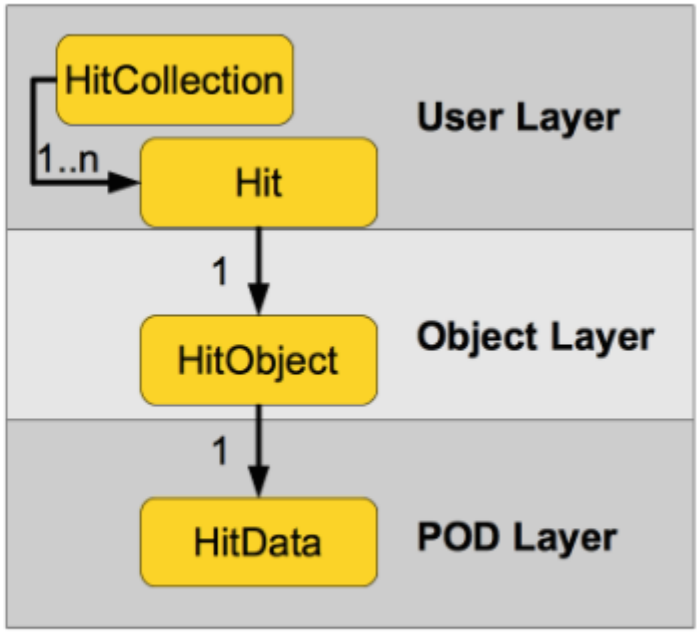
[Ref]: F. Gaede, etc. , CHEP2019

https://indico.cern.ch/event/773049/contributions/3473254/attachments/1939721/3215730/gaede_podio_chep19.pdf

PODIO is originally developed in context of the FCC study

Now it is an EDM toolkit in AIDA2020

- user layer (API):
  - handles to EDM objects (e.g. **Hit**)
  - collections of EDM object handles (e.g. **HitCollection**).
- object layer
  - transient objects (e.g. **HitObject**) handling *references* to other objects and *vector members*
- POD layer
  - the actual POD data structures holding the persistent information (e.g. **HitData**)



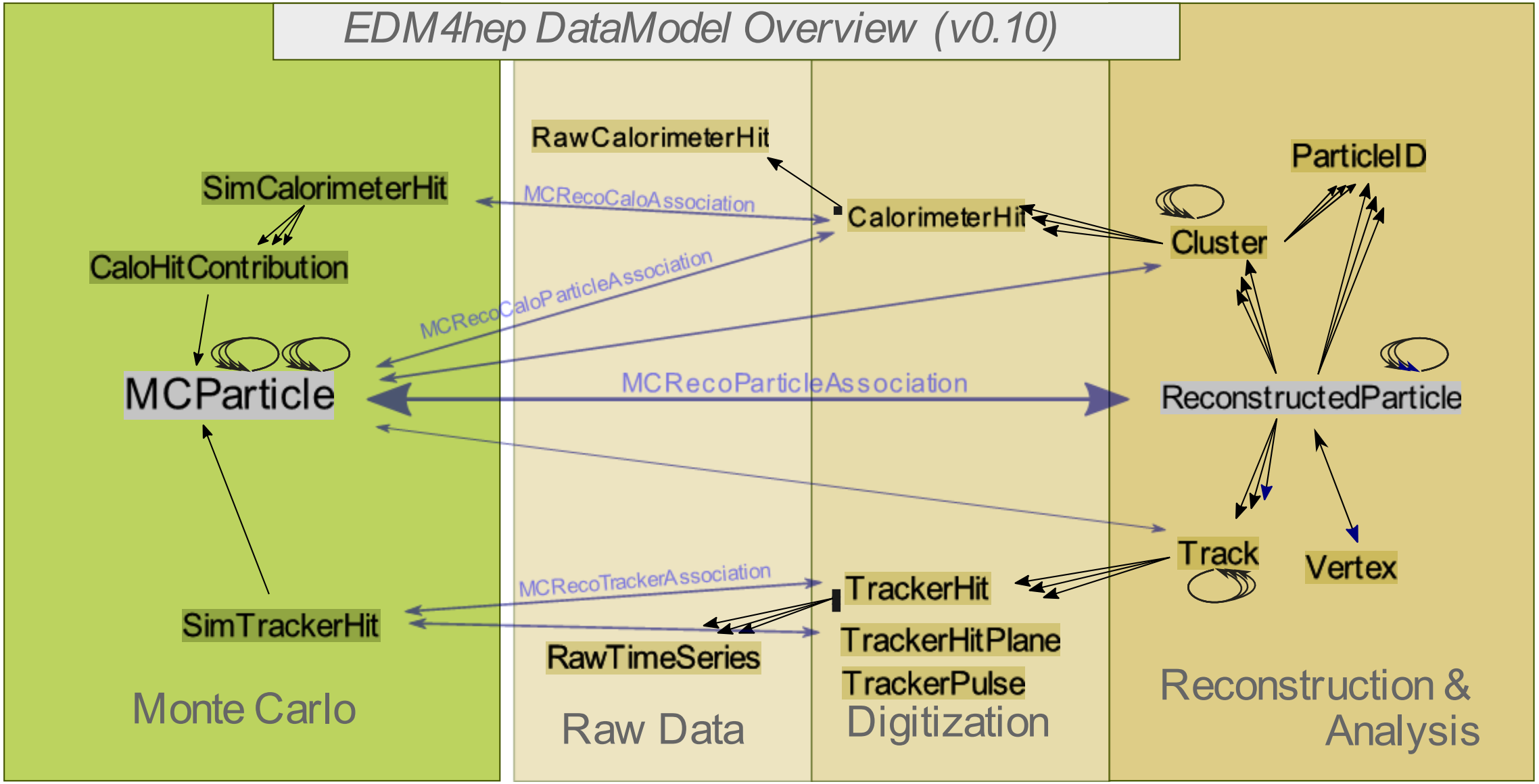

direct access to POD also possible - if needed for performance reason

# PODIO

- Generate code from yaml files
  - Basic type data members
  - Structs of basic types
  - References to other objects
  - Additional user code, such as member functions
- Different backend
  - ROOT (ROOT dictionaries are also generated automatically)
  - SIO

```
edm4hep::MCParticle:
  Description: "The Monte Carlo particle - based on the lcio::MCParticle."
  Author: "EDM4hep authors"
  Members:
    - int32_t PDG                              // PDG code of the particle
    - int32_t generatorStatus                  // status of the particle as defin
    - int32_t simulatorStatus                  // status of the particle from the
    - float charge                           // particle charge
    - float time [ns]                          // creation time of the particle i
    - double mass [GeV]                         // mass of the particle
    - edm4hep::Vector3d vertex [mm]                 // production vertex of the
    - edm4hep::Vector3d endpoint [mm]               // endpoint of the particle
    - edm4hep::Vector3d momentum [GeV]                // particle 3-momentum at
    - edm4hep::Vector3d momentumAtEndpoint [GeV]  // particle 3-momentum at
    - edm4hep::Vector3f spin                      // spin (helicity) vector of th
    - edm4hep::Vector2i colorFlow                  // color flow as defined by
  OneToManyRelations:
    - edm4hep::MCParticle parents // The parents of this particle
    - edm4hep::MCParticle daughters // The daughters this particle
  MutableExtraCode:
    includes: "#include <cmath>"
    declaration: "
    int32_t  set_bit(int32_t val, int num, bool bitval){ return (val & ~(1<<
    void setCreatedInSimulation(bool bitval) { setSimulatorStatus( set_bit(
    void setBackscatter(bool bitval) { setSimulatorStatus( set_bit( getSimul
    void setVertexIsNotEndpointOfParent(bool bitval) { setSimulatorStatus( s
```

https://github.com/key4hep/EDM4hep

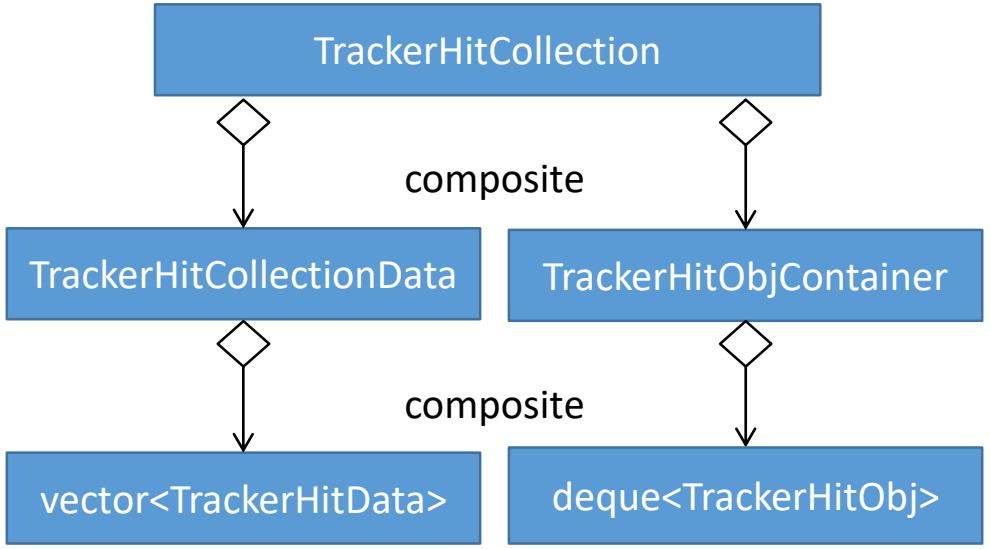


# An Explore to the EDM Classes (I)

- Take the TrackerHit as example
- User layer classes with friendly user interface
  - TrackerHitCollection
    - TrackerHit: read-only interfaces
    - MutableTrackerHit: data modification interfaces
    - For data safety (especially in multithreading context), do not use the mutable one unless necessary
  - Neither inheritance nor composition between these classes
    - TrackerHit & MutableTrackerHit objects are very thin that only hold a TrackerHitObj pointer
    - They are created on-the-fly when accessed

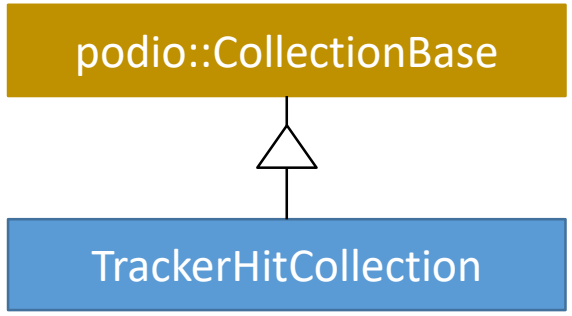
# An Explore to the EDM Classes (II)

- POD (Plain Old Data) layer
  - TrackerHitData
    - No base class
    - Same as a C struct with simple data members
    - A contiguous block of data in memory
  - TrackerHitCollectionData
    - The TrackerHitData objects are copied into a contiguous block of buffer for writing
  - Better I/O performance is expected with the contiguous block of data
- Object layer
  - Used to handle the relations between the data objects

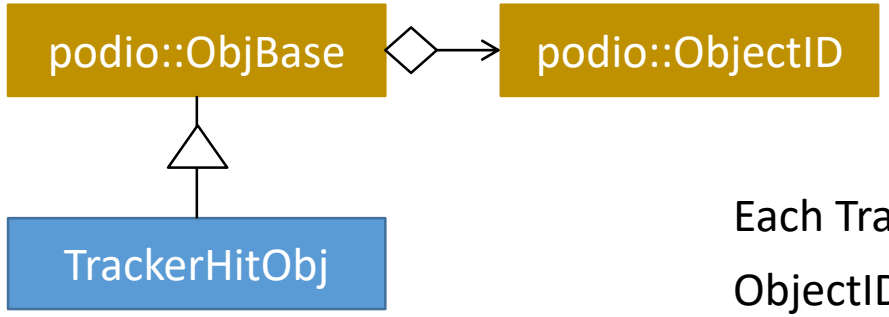


# An Explore to the EDM Classes (III)

- Class inheritance




Managed by the EventStore (or Frame in the latest podio)




- Collection ID: a hash ID for each collection
- Index: the (vector) index of the data object in the collection

Each TrackerHitObj has a unique ObjectID

ObjectID is used to keep the relations between data objects

# Relations and Associations

- Internal relations
  - OneToOneRelations, such as the ReconstructedParticle to its start Vertex
  - OneToManyRelations, such as the ReconstructedParticle to its sub components including clusters and tracks
  - Stored in the data members of its object layer or POD layer
- External associations, take MCRecoParticleAssociation as example
  - A OneToOneRelations to a ReconstructedParticle
  - A OneToOneRelations to a MCParticle
  - Equivalent to an external association between the ReconstructedParticle and the MCParticle
  - A many to many relation can be present by an association collection

# Experiment Specific Data Models

- Generate code from a customized yaml file with PODIO
  - The yaml syntax is straightforward and simple to understand
  - The new EDM classes can be used together with EDM4hep as an extension
- Software sharing is disrupted
  - Specific data models and the algorithms derived from them cannot be shared between experiments
  - Reuse the data models in EDM4hep if possible
  - Contribute to EDM4hep if possible

# Summary

- EDM4hep is the official EDM in the Key4HEP project
  - Implemented based on PODIO
    - Without complex class inheritance, better performance is expected
    - Friendly user layer is provided
  - Shared by many future collider experiments
  - Recent improvements for multithreading
- More potential possibilities with the POD layer ?
  - Parallelization with the contiguous block of data in memory
  - Easier data transfer between heterogeneous devices (GPU …)