EDM4hep and podio - The event data model of the Key4hep project and its implementation

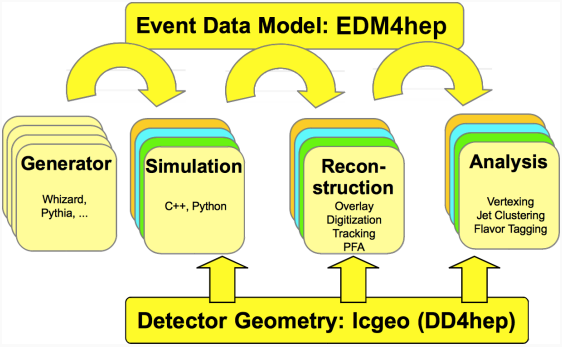
tCSC 2022

Thomas Madlener

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

May 4, 2022

The EDM at the core of HEP software



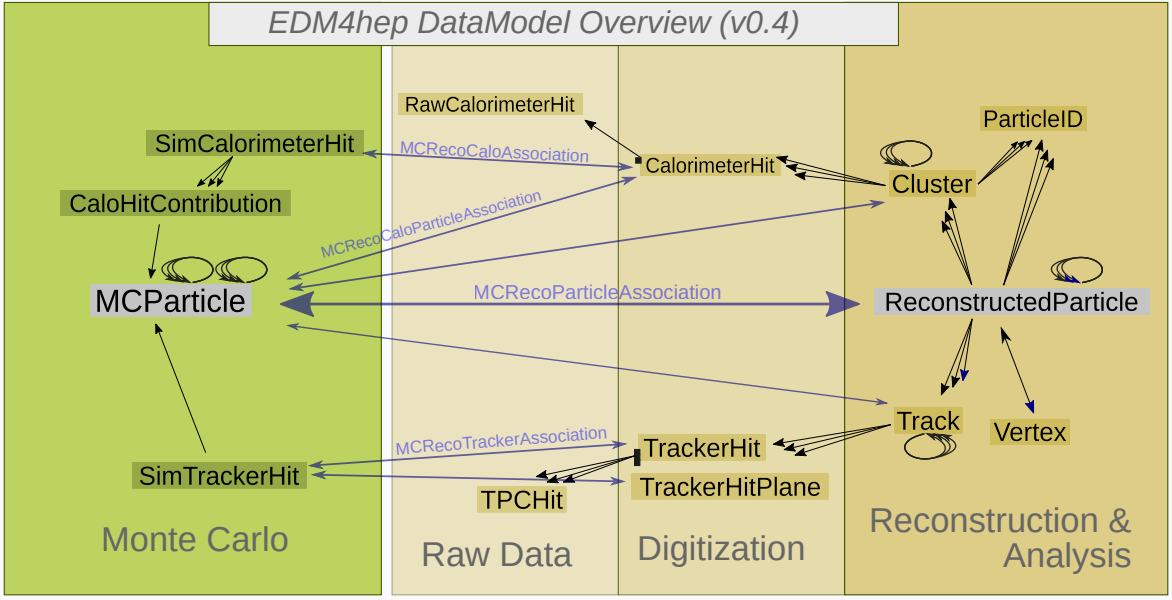
- Different components of HEP experiment software have to talk to each other
- The event data model defines the language for this communication
- Users express their ideas in the same language

Goals for EDM4hep

key4hep/EDM4hep

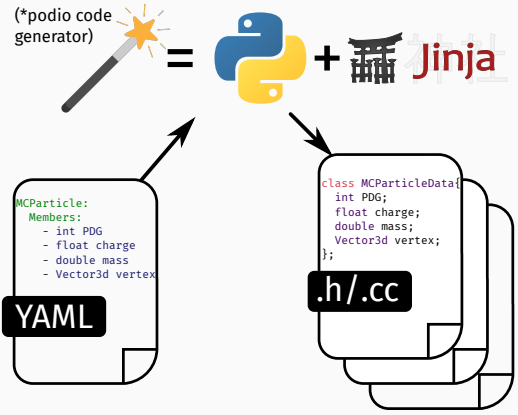
- The Key4hep project aims to define a **common software stack for all future collider projects**
- EDM4hep is the common EDM that can be used by all communities in the Key4Hep project
  - **ILC, CLIC, FCC-ee & FCC-hh, CEPC, EIC, ...**
- Efficiently implemented, support multi-threading and potentially heterogeneous computing

EDM4hep schema



podio as generator for EDM4hep

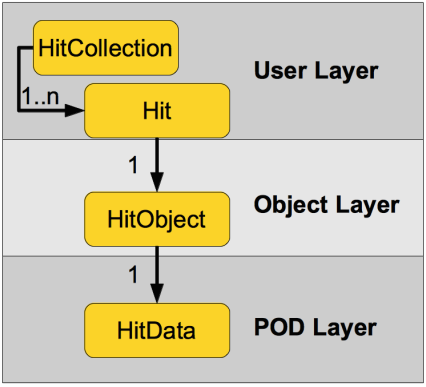
- Traditionally HEP c++ EDMs are heavily Object Oriented
- Use `podio` to generate thread safe code starting from a high level description
- Provide an easy to use interface to the users



AIDASoft/podio

The three layers of podio

- podio favors **composition over inheritance** and uses **plain-old-data (POD)** types wherever possible
- Layered design allows for efficient memory layout and performant I/O implementation



podio - features of generated code

- c++17 code with “value semantics”

```cpp
auto particles = MCParticleCollection();
// ... fill ...

for (auto mc : particles) {
  auto mass = mc.getMass();
  for (auto p : mc.getParents()) {
    auto pid = p.getPDG();
  }
}
```

- Python bindings via PyROOT

```python
particles = MCParticleCollection()
# ... fill ...

for mc in particles:
  mass = mc.getMass()
  for p in mc.getParents():
    pid = p.getPDG()
```

- Easy to read ROOT files

```python
d = ROOT.RDataFrame('events', 'events.root')
h = (d.Define('abs_pdg', 'abs(Particle.PDG)')
      .Define('mu_sel', 'abs_pdg == 13')
      .Define('mu_px',
              'Particle.momentum.x[mu_sel]')
      .Histo1D('mu_px'))
h.DrawCopy()
```

Summary & Outlook

Summary

- EDM4hep is the common EDM for all future collider projects
- It is generated via podio from a high level description

Next steps / TODO list

- Schema evolution for EDMs
- `podio::Frame` - a generalied “event”
- Support for usage on heterogenous resources
- ...


Backup

podio - datamodel definition

```yaml
components:
  edm4hep::Vector3f:
    Members: [float x, float y, float z]

datatypes:
  edm4hep::ReconstructedParticle:
    Description: "Reconstructed Particle"
    Author : "F.Gaede, DESY"
    Members:
      - edm4hep::Vector3f     momentum   // [GeV] particle momentum
      - std::array<float, 10> covMatrix  // energy-momentum covariance
    OneToOneRelations:
      - edm4hep::Vertex startVertex // start vertex associated to this particle
    OneToManyRelations:
      - edm4hep::Cluster clusters  // clusters that have been used for this particle
      - edm4hep::ReconstructedParticle particles // associated particles
    ExtraCode:
      declaration: "bool isCompund() const { return particles_size() > 0; }\n"

  edm4hep::ParticleID:
    VectorMembers:
      - float parameters // hypothesis params
```

*extracted from edm4hep.yaml

- Reusable components
- Fixed sized arrays as members
- *VectorMembers* for variable sized array members
- 1 – 1 and 1 – N relations
- Additional user-provided code

CMake interface for projects using podio

```cmake
find_package(PODIO)

# generate the c++ code from the yaml definition
PODIO_GENERATE_DATAMODEL(edm4hep edm4hep.yaml headers sources IO_BACKEND_HANDLERS "ROOT;SIO")
# compile the core data model shared library (no I/O)
PODIO_ADD_DATAMODEL_CORE_LIB(edm4hep "${headers}" "${sources}")
# generate and compile the ROOT I/O dictionary
PODIO_ADD_ROOT_IO_DICT(edm4hepDict edm4hep "${headers}" src/selection.xml)
# compile the SIOBlocks shared library for the SIO backend
PODIO_ADD_SIO_IO_BLOCKS(edm4hep "${headers}" "${sources}")

# Install the created targets
install(TARGETS edm4hep edm4hepDict edm4hepSioBlocks)
```

- Easy to use functions for integrating a podio generated EDM into a project
- Split into core EDM library and I/O handling for different backends
  - Pick what you need
  - I/O handling parts dynamically loaded by podio on startup (searching `LD_LIBRARY_PATH`)

The I/O in podio

- I/O operations are based on collections
- Concatenate the different parts into arrays of PODs
- I/O backends only have to be able to read / write arrays of PODs. The rest is handled by podio