lcio2edm4hep converter and applications

Thomas Madlener

for the Key4hep developers

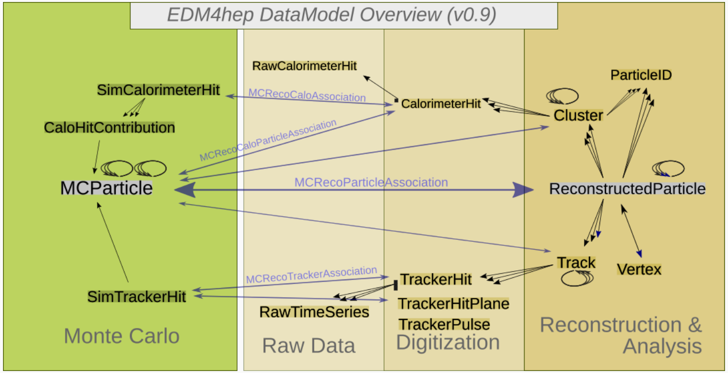
ECFA Higgs Factories

Topical meeting on Reconstruction

July 11, 2023

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

EDM4hep - The common EDM for Key4hep



- Based on `LCIO` and `FCC-edm`
  - Focus on usability in analysis
- Quite stable over the last two years
- Addition of datatypes for **CEPC drift chamber study**
- Can easily be extended
  - Used by EDM4eic
  - Main purpose: prototyping
- Generated via `podio`

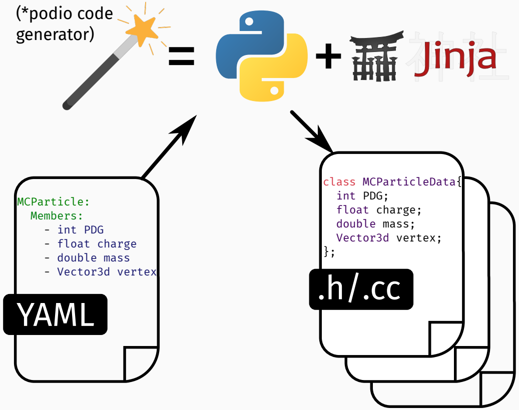
key4hep/EDM4hep

edm4hep.web.cern.ch

AIDASoft/podio

The podio EDM toolkit

- Implementing a performant event data model (EDM) is non-trivial
- Use `podio` to generate code starting from a high level description
- Provide an easy to use interface to the users
- Main customers
  - key4hep/EDM4hep
  - eic/EDM4eic
- **Finishing schema evolution for v1.0**



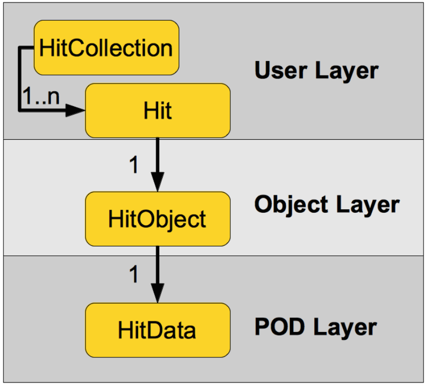
```
MCParticle:
  Members:
    - int PDG
    - float charge
    - double mass
    - Vector3d vertex
```

```
class MCParticleData{
  int PDG;
  float charge;
  double mass;
  Vector3d vertex;
};
```

AIDASoft/podio

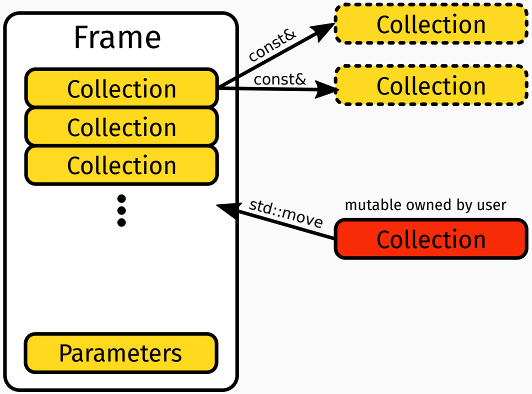
The three layers of podio

- 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



The `Frame` - A generalized (event) data container

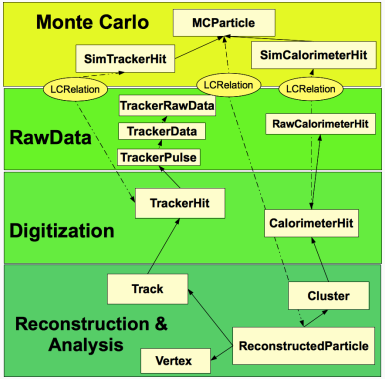
- Replaces deprecated `EventStore`
- *Type erased* container aggregating all relevant data
- Defines an *interval of validity* / category for contained data
  - Event, Run, readout frame, ...
- Easy to use and thread safe interface for data access
  - Immuatable read access only
  - Ownership model reflected in API
- Decouples I/O from operating on the data



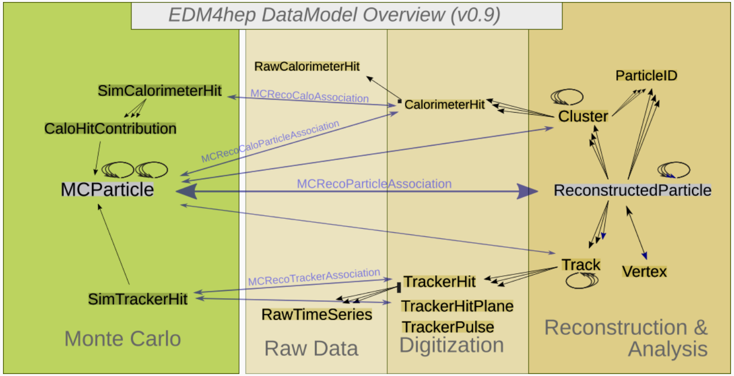
```
template<typename CollT>
const CollT& get(const std::string& name) const;

template<typename CollT, /*enable_if*/>
const CollT& put(CollT&& collection,
                 const std::string& name);
```

LCIO vs EDM4hep (at the highest level)

LCIO



EDM4hep



- Since EDM4hep is based on LCIO the high-level structure is very similar
- Largest differences between the two are due to their implementations
- LCIO is 20 years old now. A lot of time to develop tools for it.

LCIO vs EDM4hep - High level differences

	LCIO	EDM4hep
call syntax	pointer semantics (`->`)	value semantics (`.`)
code layout	inheritance (`LCObject`)	composition
mutability	always*	creation only
external relations	`LCRelation`	dedicated `Association` types
event container	`LCEvent`	`podio::Frame` + `edm4hep::EventHeader`
run container	`LCRunHeader`	`podio::Frame`
event contents	missing collections allowed	all events with same collections†
parameters	collections & `LCEvent`	`podio::Frame` only
file format	`slcio` (SIO)	`.root` (default), SIO available

*"If you know what you are doing"

†Technically a requirement of the ROOT backend

Why another converter?

- key4hep/k4LCIOReader already provides reading of LCIO files into the Key4hep (Gaudi) world
  - Used in key4hep/k4MarlinWrapper
- **Wanted a standalone executable** (no Gaudi or Marlin)
- **Using the `podio::Frame`**
- Support all necessary functionality (e.g. subset collections)
- Easier to use shared library
- Complete overhaul of pre-existing functionality
  - Major effort from Finn Johannsen (DESY project student)
  - Shared library in key4hep/k4EDM4hep2LcioConv

How to use the standalone converter

- README
- Simplest case (complete events in LCIO)

```
lcio2edm4hep input.slcio output.edm4hep.root
```

- Almost simple case (e.g. non-complete events in LCIO)
  - Need some help to "patch" collections on-the-fly, e.g.

```
SETSpacePoints                TrackerHit
RecoMCTruthLink               LCRelation[ReconstructedParticle,MCParticle]
```

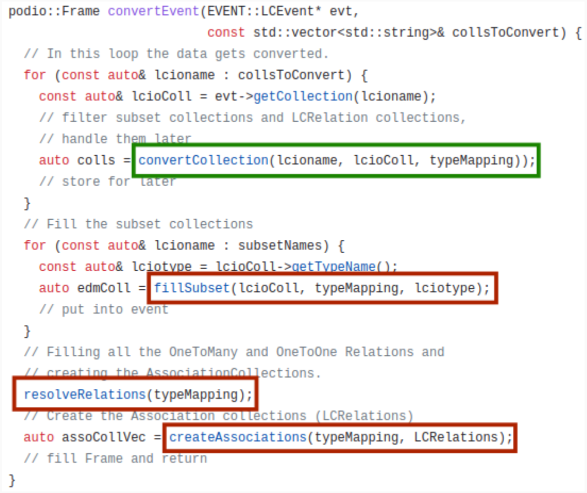
  - Don't write this yourself!

```
check_missing_cols --minimal input.slcio > patch.txt
lcio2edm4hep input.slcio output.edm4hep.root patch.txt
```

- `patch.txt` file can also be used to select a subset of collections to convert

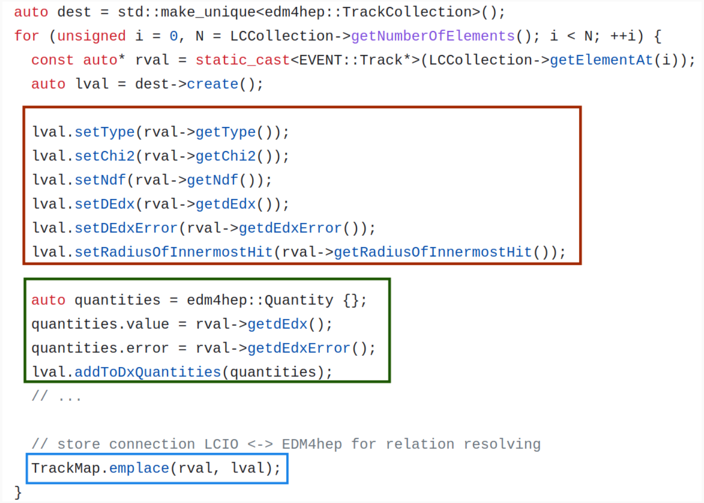
Using the shared library

- Conversion in two steps
    1. Convert data
    2. Resolve inter-object relations
- High-level functions delegate to type specific ones
- Ongoing work to integrate this into `k4MarlinWrapper`



```
podio::Frame convertEvent(EVENT::LCEvent* evt,
                          const std::vector<std::string>& collsToConvert) {
  // In this loop the data gets converted.
  for (const auto& lcioname : collsToConvert) {
    const auto& lcioColl = evt->getCollection(lcioname);
    // filter subset collections and LCRelation collections,
    // handle them later
    auto colls = convertCollection(lcioname, lcioColl, typeMapping));
    // store for later
  }
  // Fill the subset collections
  for (const auto& lcioname : subsetNames) {
    const auto& lciotype = lcioColl->getTypeName();
    auto edmColl = fillSubset(lcioColl, typeMapping, lciotype);
    // put into event
  }
  // Filling all the OneToMany and OneToOne Relations and
  // creating the AssociationCollections.
  resolveRelations(typeMapping);
  // Create the Association collections (LCRelations)
  auto assoCollVec = createAssociations(typeMapping, LCRelations);
  // fill Frame and return
}
```

LCIO to EDM4hep conversion example

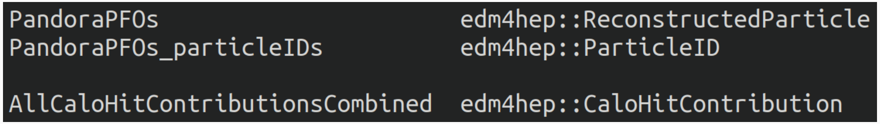


```cpp
auto dest = std::make_unique<edm4hep::TrackCollection>();
for (unsigned i = 0, N = LCCollection->getNumberOfElements(); i < N; ++i) {
  const auto* rval = static_cast<EVENT::Track*>(LCCollection->getElementAt(i));
  auto lval = dest->create();

  lval.setType(rval->getType());
  lval.setChi2(rval->getChi2());
  lval.setNdf(rval->getNdf());
  lval.setDEdx(rval->getdEdx());
  lval.setDEdxError(rval->getdEdxError());
  lval.setRadiusOfInnermostHit(rval->getRadiusOfInnermostHit());

  auto quantities = edm4hep::Quantity {};
  quantities.value = rval->getdEdx();
  quantities.error = rval->getdEdxError();
  lval.addToDxQuantities(quantities);
  // ...

  // store connection LCIO <-> EDM4hep for relation resolving
  TrackMap.emplace(rval, lval);
}
```

- Majority of cases is trivial
- Some minor differencs need treatment
  - EDM4hep with generalized dQ/dx treatment
  - `CellID`s always 64 bits in EDM4hep

Some visible differences after conversion

- ParticleID
  - Part of `ReconstructedParticle` and `Cluster` in LCIO
  - Dedicated type + relations in EDM4hep
  - → Additional collection in EDM4hep output
- CaloHitContribution
  - Part of `SimCalorimeterHit` in LCIO
  - Dedicated type in EDM4hep
  - → Additional (global) collection in EDM4hep output
- Transparent for reconstruction / analysis
- `TPCHit` (LCIO) → `RawTimeSeries` (EDM4hep)



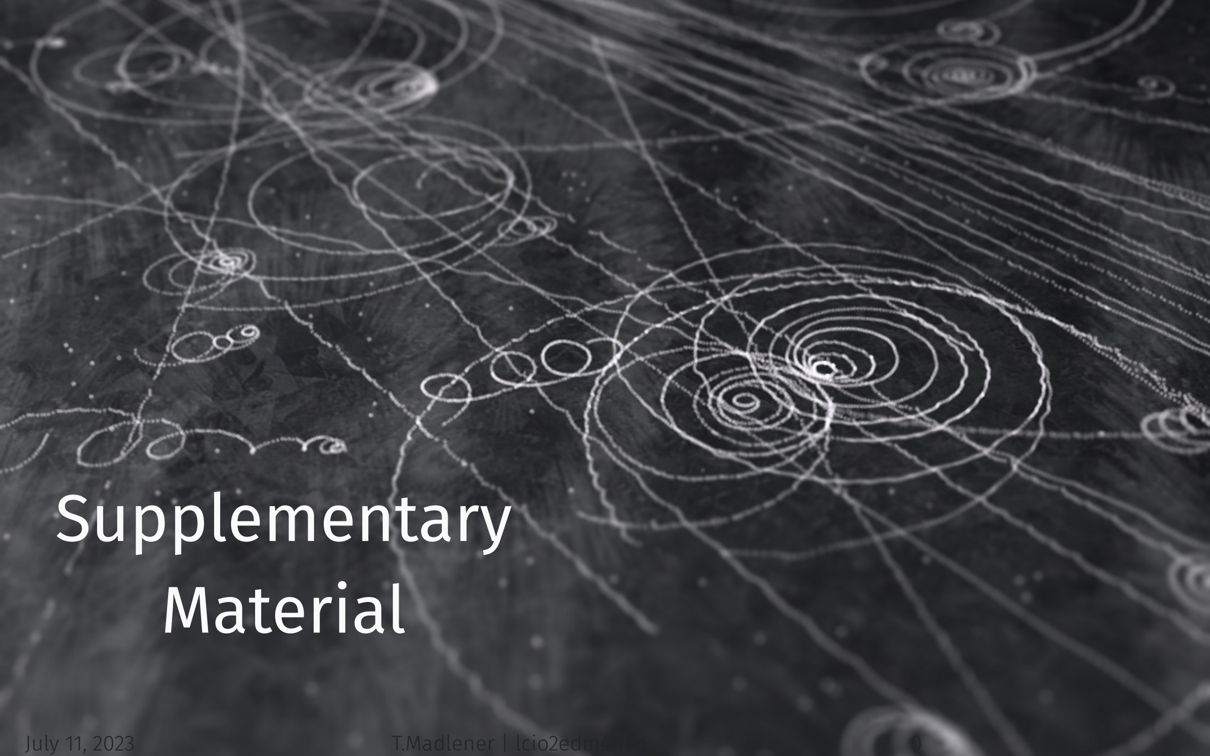
```
PandoraPFOs                      edm4hep::ReconstructedParticle
PandoraPFOs_particleIDs          edm4hep::ParticleID

AllCaloHitContributionsCombined  edm4hep::CaloHitContribution
```

Differences that need more work

- EDM4hep is quite a bit more restrictive compared to LCIO
  - Hyrum's Law is a thing, especially in HEP
  - Mutability of objects that are read
  - Possibilities to store data outside the EDM
- No generic Associations in EDM4hep (similar to `LCRelations`)
  - Can only convert those that are present
- Some features will not be available in EDM4hep
  - E.g. collection parameters; wild mix of *collection level metadata* and *event level collection related data* in LCIO files
- Some things will potentially need conceptual changes

Summary

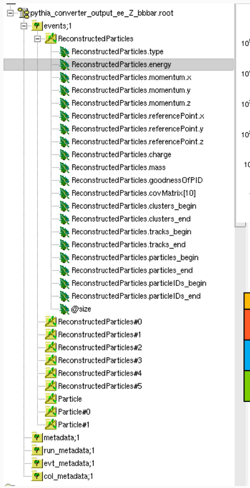
- New LCIO-to-EDM4hep conversion library & standalone `lcio2edm4hep` available
- Will be used in `k4MarlinWrapper` as well
- Covers everything we discovered during development
- Some thing still need work
- **Feedback is extremely valuable! If you find an issue, let us know!**
- Also let us know about things you want to have



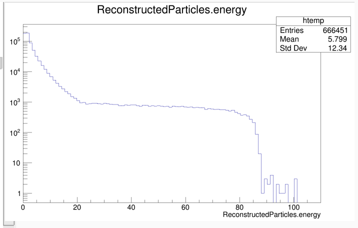
Supplementary Material

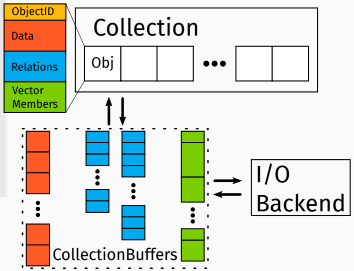
podio supports different I/O backends

- Default **ROOT** backend
  - POD buffers are stored as branches in a `TTree`
  - Files can be interpreted **without EDM library**(!)
  - Can be used in `RDataFrame` or with `uproot`
- Alternative **SIO** backend
  - Persistency library used in `LCIO`
  - Complete events are stored as binary records
- Adding more I/O backends is possible



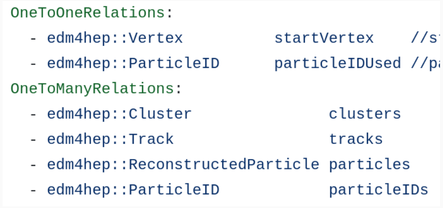
```
pythia_converter_output_ee_Z_bbbar.root
  events;1
    ReconstructedParticles
      ReconstructedParticles.type
      ReconstructedParticles.energy
      ReconstructedParticles.momentum.x
      ReconstructedParticles.momentum.y
      ReconstructedParticles.momentum.z
      ReconstructedParticles.referencePoint.x
      ReconstructedParticles.referencePoint.y
      ReconstructedParticles.referencePoint.z
      ReconstructedParticles.charge
      ReconstructedParticles.mass
      ReconstructedParticles.goodnessOfPID
      ReconstructedParticles.covMatrix[10]
      ReconstructedParticles.clusters_begin
      ReconstructedParticles.clusters_end
      ReconstructedParticles.tracks_begin
      ReconstructedParticles.tracks_end
      ReconstructedParticles.particles_begin
      ReconstructedParticles.particles_end
      ReconstructedParticles.particleIDs_begin
      ReconstructedParticles.particleIDs_end
      @size
    ReconstructedParticles#0
    ReconstructedParticles#1
    ReconstructedParticles#2
    ReconstructedParticles#3
    ReconstructedParticles#4
    ReconstructedParticles#5
    Particle
    Particle#0
    Particle#1
  metadata;1
  run_metadata;1
  evt_metadata;1
  col_metadata;1
```



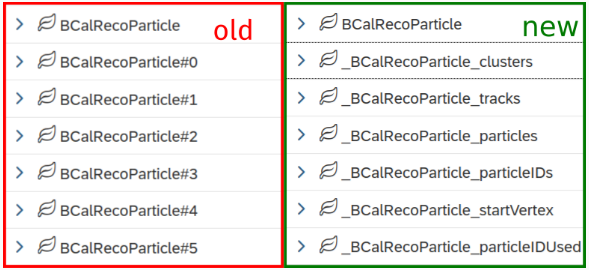


Other recent developments

- More legible branch names for relations
- Stable collection IDs based on collection names
- Ongoing efforts to have EDM4hep in coffea
  - First version based on ILD DST files



```
OneToOneRelations:
  - edm4hep::Vertex            startVertex    //s
  - edm4hep::ParticleID        particleIDUsed //p
OneToManyRelations:
  - edm4hep::Cluster                  clusters
  - edm4hep::Track                    tracks
  - edm4hep::ReconstructedParticle particles
  - edm4hep::ParticleID               particleIDs
```



old	new
BCalRecoParticle	BCalRecoParticle
BCalRecoParticle#0	_BCalRecoParticle_clusters
BCalRecoParticle#1	_BCalRecoParticle_tracks
BCalRecoParticle#2	_BCalRecoParticle_particles
BCalRecoParticle#3	_BCalRecoParticle_particleIDs
BCalRecoParticle#4	_BCalRecoParticle_startVertex
BCalRecoParticle#5	_BCalRecoParticle_particleIDUsed