Converting algorithms from iLCSoft to Gaudi

Summer Student Program 2022

Violeta Vicente Cantero

Supervisors: Plácido Fernández Declara and Benedikt Hegner

Overview

- Motivation of the project
- Marlin and Gaudi frameworks
- LCIO and EDM4hep
- Stages of my project
- Process of learning
- Results and comparison
- Next weeks work
- Conclusions
- Thanks

Motivation of the project: removing the Wrapper!

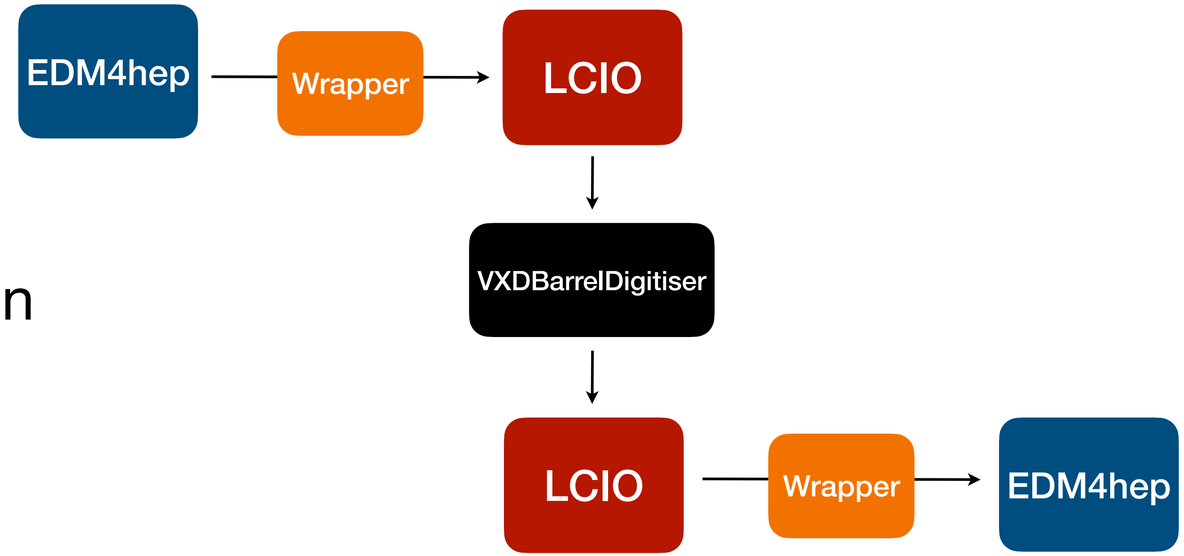
Marlin algorithms are currently being executed through a wrapper

Downsides

1. Overhead
2. Keeps the original code intact

   → not ready for parallel execution

3. No integration with Gaudi
4. No interoperability with Key4hep

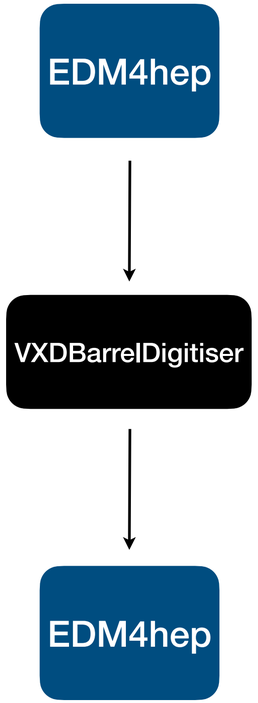


Motivation of the project: removing the Wrapper!

Marlin algorithms are currently being executed through a wrapper

Goals

- **better integration** with Gaudi
- **removing overheads** from conversions (EDM and framework)
- opportunity to **modernize the code**, using safer coding techniques



What do we mean by modernizing?

- **Modern C++**
- **Functional programming** where possible
- Use as many **Key4hep elements as possible** for future experiments: Gaudi, EDM4hep

Marlin and Gaudi frameworks

Analysis, simulation and reconstruction frameworks

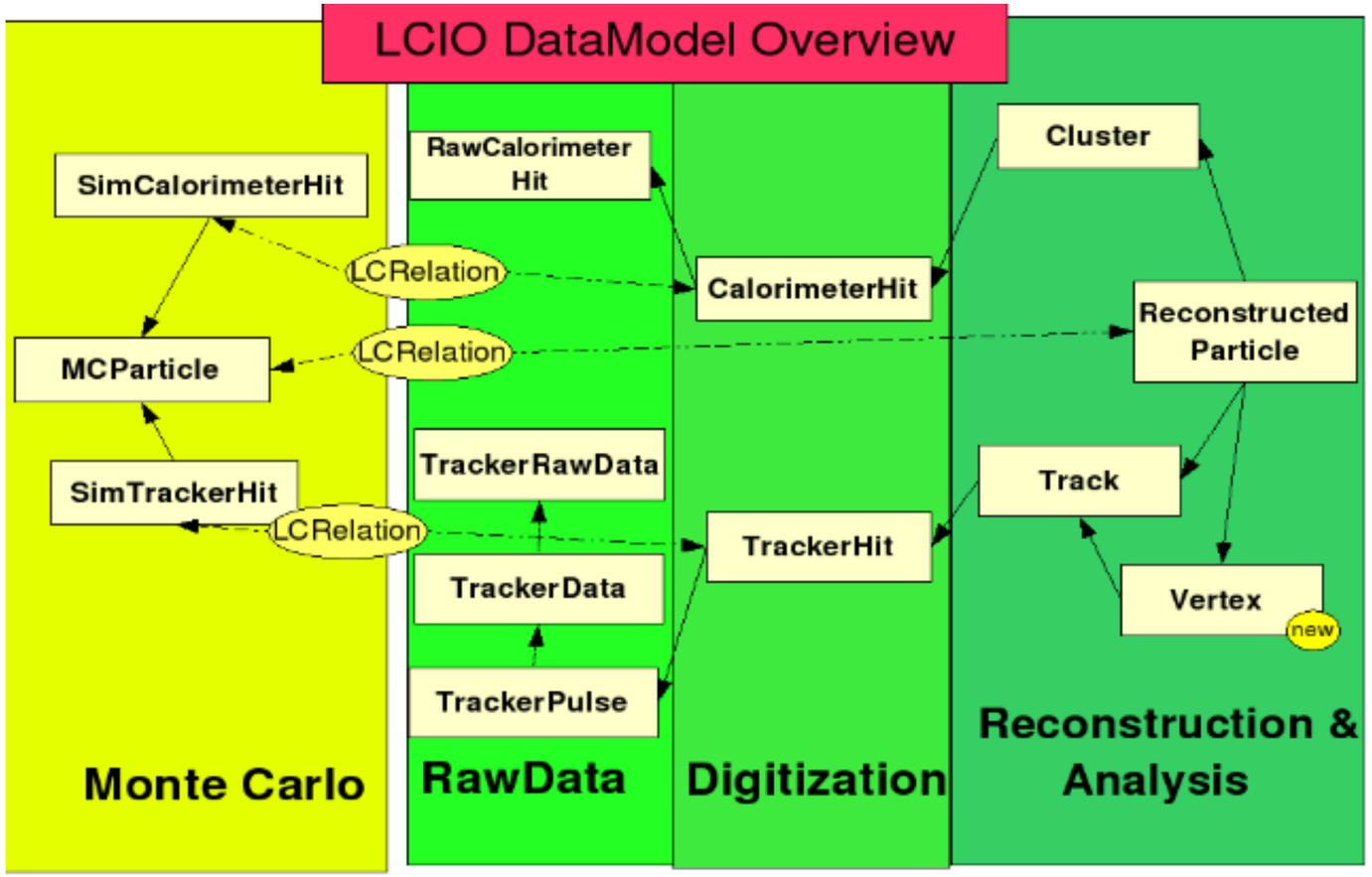
- **Marlin** (Modular Analysis and Reconstruction for the LINear Collider)
  - used by the linear collider community over 15+ years
- **Gaudi** used by LHCb, ATLAS, Key4hep and other experiments

	Marlin	Gaudi
Language	C++	C++
Working unit	Processor	Algorithm
Config. language	XML	Python
Set-up function	init	initialize
Working function	process	execute
Wrap-up function	end	finalize
Transient Data Format	LCIO	anything

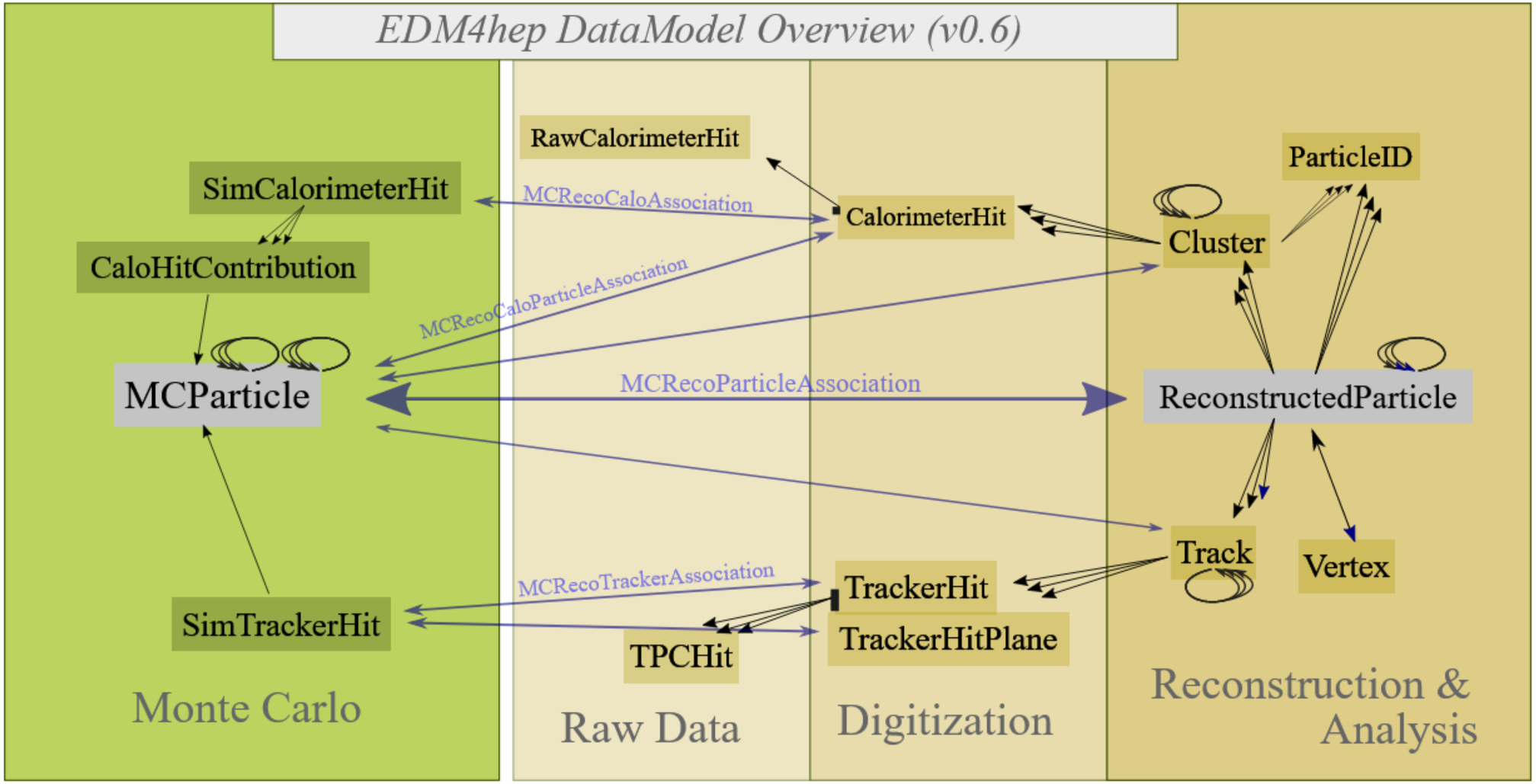
LCIO and EDM4hep

Comparison of both Event Data Models

- **LCIO** (Linear Collider I/O)
- **EDM4hep** (Event Data Model 4 hep)
- Very similar, as EDM4hep was inspired by LCIO
- EDM4hep is evolving, and reaching V1.0



EDM4hep


EDM4hep DataModel Overview (v0.6)
RawCalorimeterHit
SimCalorimeterHit
MCRecoCaloAssociation
CalorimeterHit
ParticleID
Cluster
CaloHitContribution
MCRecoCaloParticleAssociation
MCParticle
MCRecoParticleAssociation
ReconstructedParticle
Track
Vertex
MCRecoTrackerAssociation
TrackerHit
SimTrackerHit
TrackerHitPlane
TPCHit
Monte Carlo
Raw Data
Digitization
Reconstruction & Analysis

1. Choosing an algorithm to port

DDPlanarDigiProcessor from Marlin

- **CLICReconstruction**: simulation software for the Compact Linear Collider

- DDPlanarDigiProcessor **produces TrackerHitsCollections**, which are outputed in a file for later use in algorithms or analysis

CLICRec processors include:

- Digitisers
- Vertexing
- Pfoselector
- Track pattern recognition
- Track fitting
- Jet clustering
- Flavour tagging

2. Process of learning

Overcoming some difficulties

- Impeded by **lack of documentation** of some of the the components such as EDM4hep or even Gaudi

- Very **old documentation** of Gaudi ⟶ diving into the code
  running
  testing
  googling around

- **Asking** colleagues, relying on my supervisors

- Absence of one of my supervisors ⟶ **less opportunities to ask**

3. Learning Gaudi

New event processing and EDM

Gaudi's structure:

- **Separation of Algorithms and Data**
- **Processing structure**: initialize(), execute() and finalize()
- Order of execution: **all** initialize() first, then execute()

  → important for parallelizing
- **Graph structure** for the Algorithms

4. First steps

Learning "the basics"

1. Get **Key4hep stack running**
2. **Get familiar** with Key4hep software and tools and **understand** how they interact
3. Learn what a **digitiser** does
4. Learn **Gaudi, bash, modern C++, Cmake, make, ninja, ROOT**...

5. Hands-on the algorithm!

Parsing processor' parameters in a Gaudi-friendly way

As many Properties as parameters needed

Marlin

```
// DDPlannarDigiProcessor.h
  bool _isStrip;
  FloatVec _resV ;
```

```
// DDPlannarDigiProcessor.cc
registerProcessorParameter( "IsStrip",
                            "whether hits are 1D strip hits",
                            _isStrip,
                            bool(false) );

FloatVec resUEx ;
resUEx.push_back( 0.0040 ) ;

registerProcessorParameter( "ResolutionU" ,
                            "resolution in direction of u - either one per layer or one for all layers "  ,
                            _resU ,
                            resUEx) ;
```

Gaudi

```
//GaudiDDPlannarDigiProcessor.h
  Gaudi::Property<bool> m_isStrip{this, "IsStrip", {}};
  Gaudi::Property<std::vector<float>> m_resU{this, "ResolutionU", {}};
```

5. Hands-on the algorithm!

Output collections

- **Marlin** structure have specializations of a **LCCollection vector**

```
// DDPlanarDigiProcessor.cc
// Output collections
registerOutputCollection( LCIO::TRACKERHITPLANE,
                          "TrackerHitCollectionName" ,
                          "Name of the TrackerHit output collection"  ,
                          _outColName ,
                          std::string("VTXTrackerHits") ) ;
```

- **Gaudi uses DataHandlers**: useful to acces data, read and write it, etc.

```
//Gaudi_DDPlanarDigiProcessor.h
DataHandle<edm4hep::TrackerHitPlaneCollection> m_TrackerHitHandle{"VXDTrackerHits", Gaudi::DataHandle::Writer, this};

//Gaudi_DDPlanarDigiProcessor.cpp
declareProperty("VXDTrackerHits", m_TrackerHitHandle, "Dummy Hit collection (output)");
edm4hep::TrackerHitPlaneCollection* trkhitVec = m_TrackerHitHandle.createAndPut();
```

5. Hands-on the algorithm!

Adapt external components call from my algorithm

1. **Global::EVENTSEEDER** ⟶ Key4FWCore::UniqueIDGenSvc

```
// From Global::EVENTSEEDER->registerProcessor(this);
// To
m_uniqueIDService = serviceLocator()->service("UniqueIDGenSvc");
m_uniqueIDService->getUniqueID(1, 2, name());
```

2. **HistogramFactory** in Marlin ⟶ ROOT Histograms
   - Flexibility, allowed by ROOT
   - Gaudi independent

3. **Streamlog** ⟶ yet to decide, currently with std::cout

5. Hands-on the algorithm!

Adapt external components calls from my algorithm

4. LCIO::UTIL::CellIDEncoder ⟶ dd4hep::DDSegmentation::BitFieldCoder

Marlin

```
// DDPlanarDigiProcessor
CellIDEncoder<TrackerHitPlaneImpl> cellid_encoder( lcio::LCTrackerCellID::encoding_string() , trkhitVec ) ;

CellIDDecoder<SimTrackerHit> cellid_decoder( STHcol) ;
int layer  = cellid_decoder( simTHit )["layer"];
cellid_decoder( simTHit ).valueString()
```

Gaudi

```
//GaudiDDLPlanarDIgiProcessor.cpp
std::string cellIDEncodingString = m_generalSimTrackerHitHandle.getCollMetadataCellID(sth_coll->getID());

auto& collmd = m_podioDataSvc->getProvider().getCollectionMetaData(trkhitVec->getID());
collmd.setValue("CellIDEncodingString", cellIDEncodingString); // encoding

// Creating the instance of the encoder to decode
dd4hep::DDSegmentation::BitFieldCoder bitFieldCoder(cellIDEncodingString);
```

5. Cmake files also had to be updated

Results and comparison

CLICRec with both the Wrapper and my new algorithm

Marlin

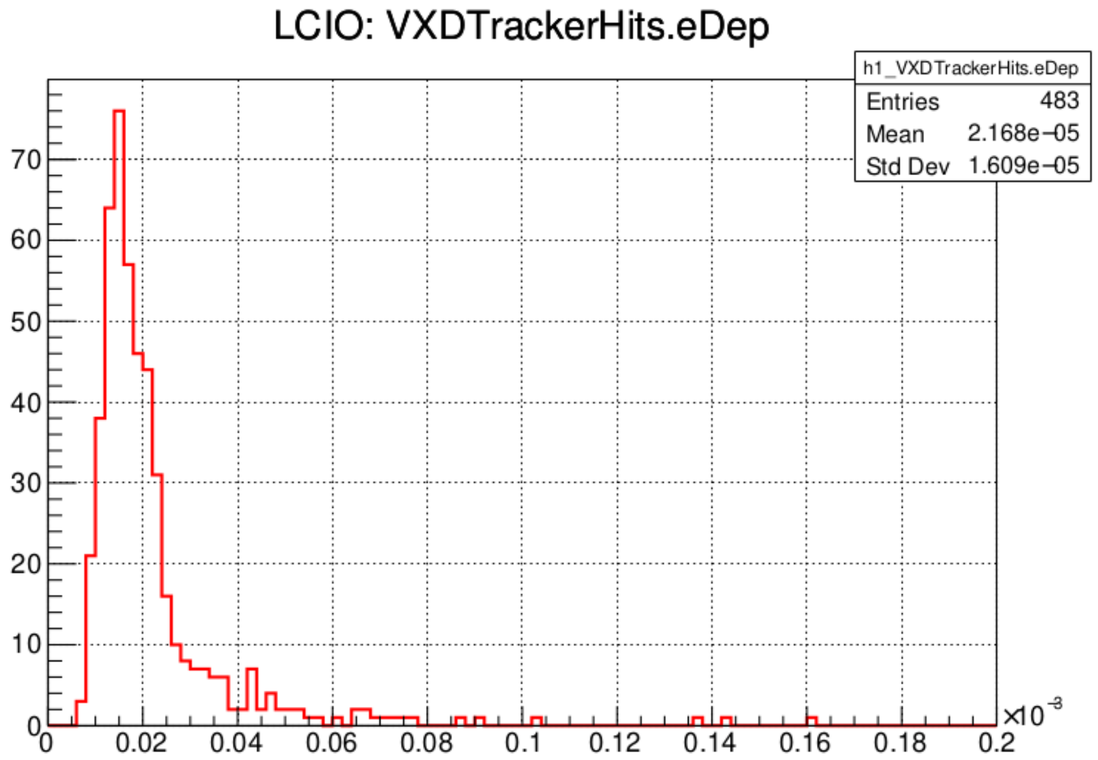
```
# Write output to EDM4hep
from Configurables import PodioOutput
out = PodioOutput("PodioOutput", filename = "my_MarlinOutput.root")
out.outputCommands = ["keep *"]


algList.append(inp)
algList.append(MyAIDAProcessor)
algList.append(EventNumber)
algList.append(InitDD4hep)
algList.append(Config)
algList.append(OverlayFalse)  # Config.OverlayFalse
algList.append(VXDBarrelDigitiser)
algList.append(out)

from Configurables import ApplicationMgr
ApplicationMgr( TopAlg = algList,
                EvtSel = 'NONE',
                EvtMax   = 3,
                ExtSvc = [evtsvc],
                OutputLevel=WARNING
              )
```

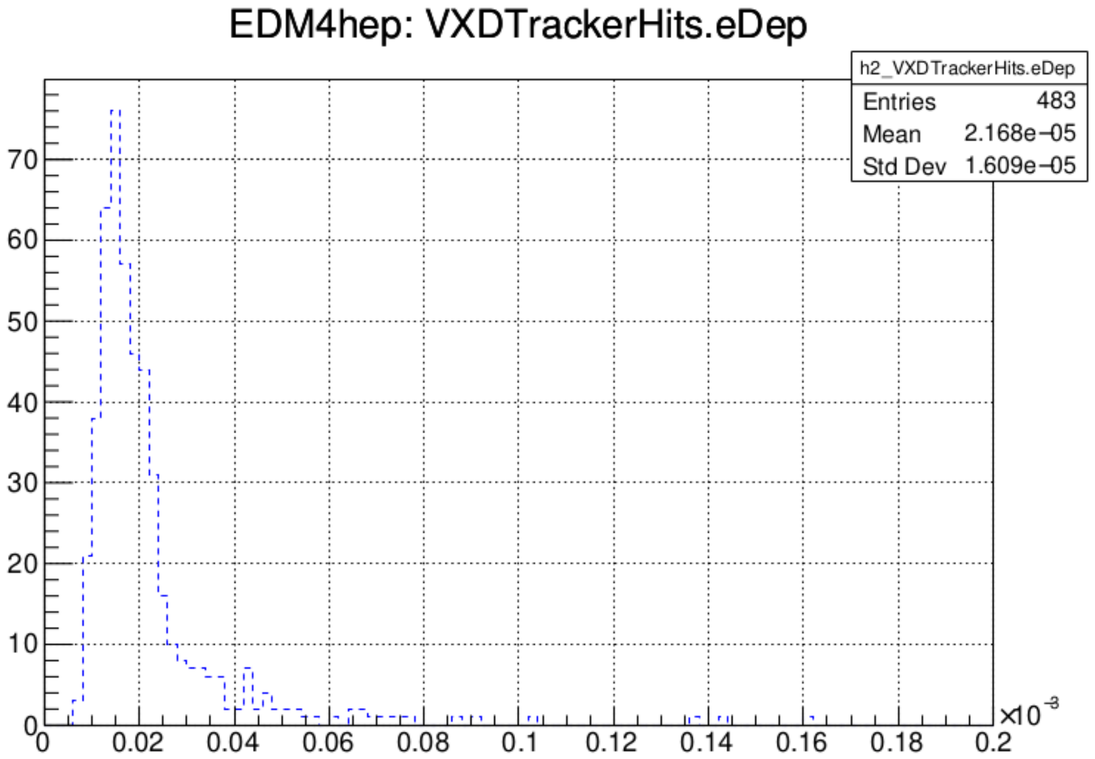
Gaudi

```
# Write output to EDM4hep
from Configurables import PodioOutput
out = PodioOutput("PodioOutput", filename = "my_GaudiOutput.root")
out.outputCommands = ["keep *"]


algList.append(inp)
algList.append(MyAIDAProcessor)
algList.append(EventNumber)
algList.append(InitDD4hep)
algList.append(Config)
algList.append(OverlayFalse)  # Config.OverlayFalse
algList.append(DDPlanarDigiProcessor)
algList.append(out)

from Configurables import Gaudi__Histograming__Sink__Root as RootHistoSink
RootHistSvc("RootHistSvc").OutputFile = "Gaudi_histo.root"

from Configurables import ApplicationMgr
ApplicationMgr( TopAlg = algList,
                EvtSel = 'NONE',
                EvtMax   = 3,
                ExtSvc = [evtsvc, RootHistoSink()],
                OutputLevel=WARNING,
                HistogramPersistency="ROOT",

              )
```

Results and comparison

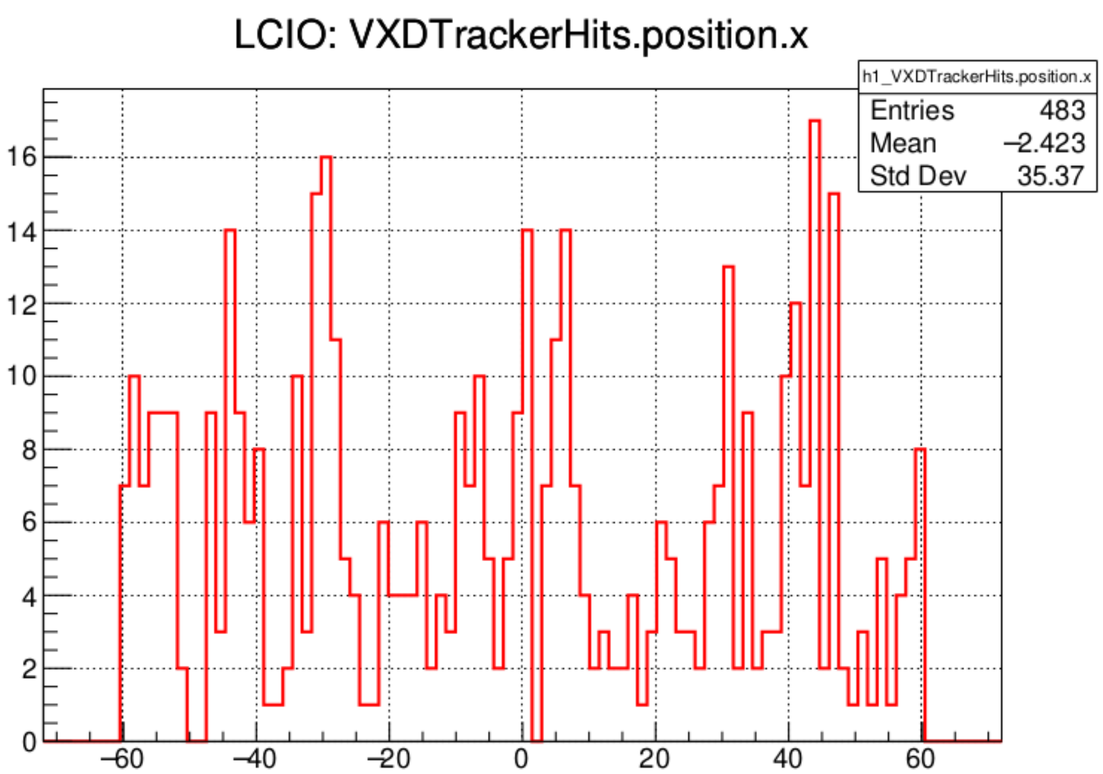
CLICRec with both the Wrapper and my new algorithm

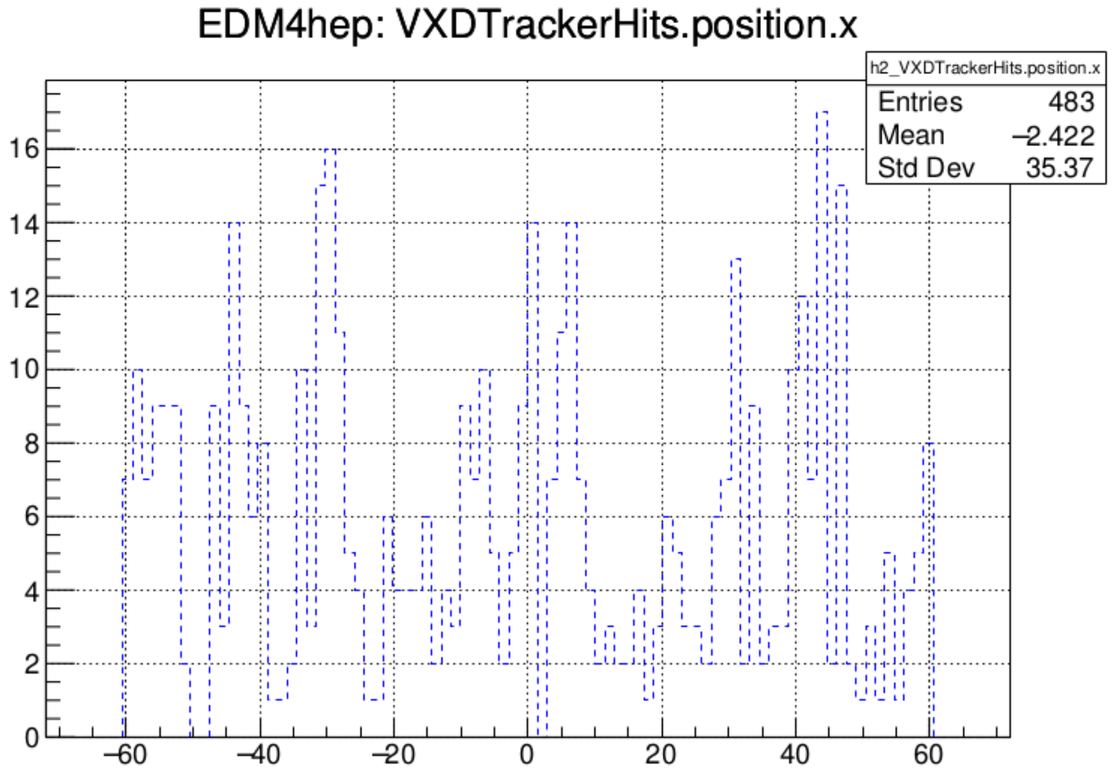




Results and comparison

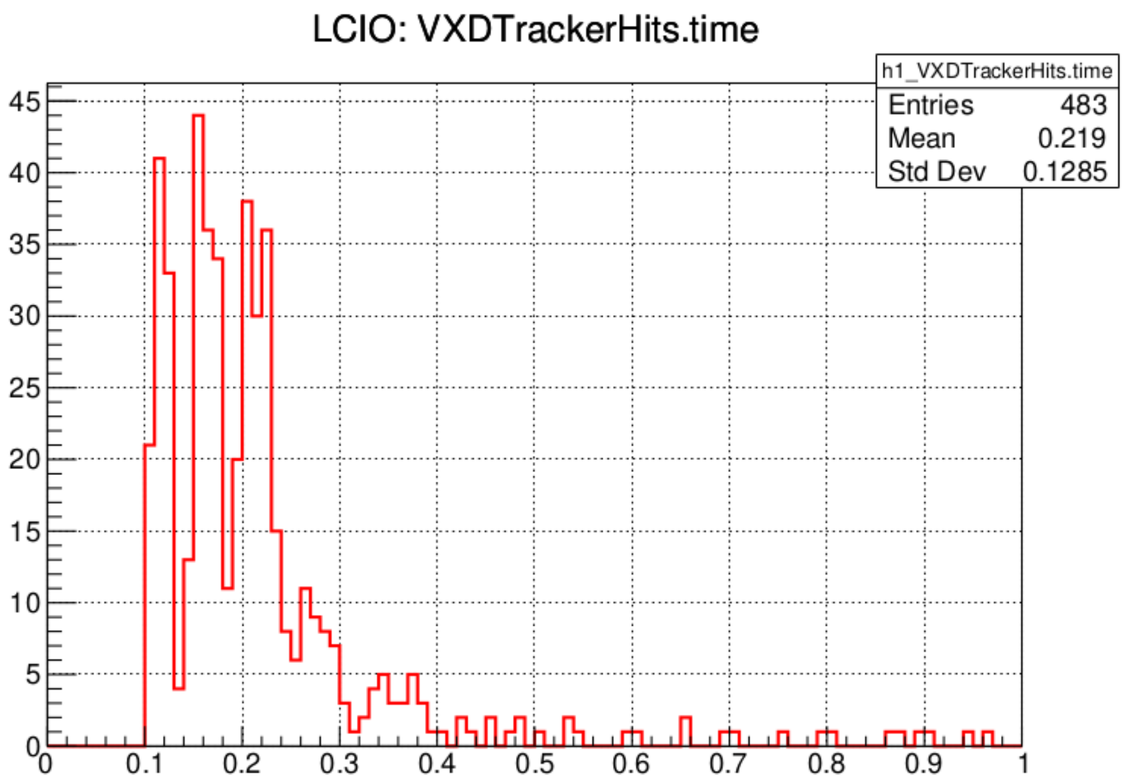
CLICRec with both the Wrapper and my new algorithm

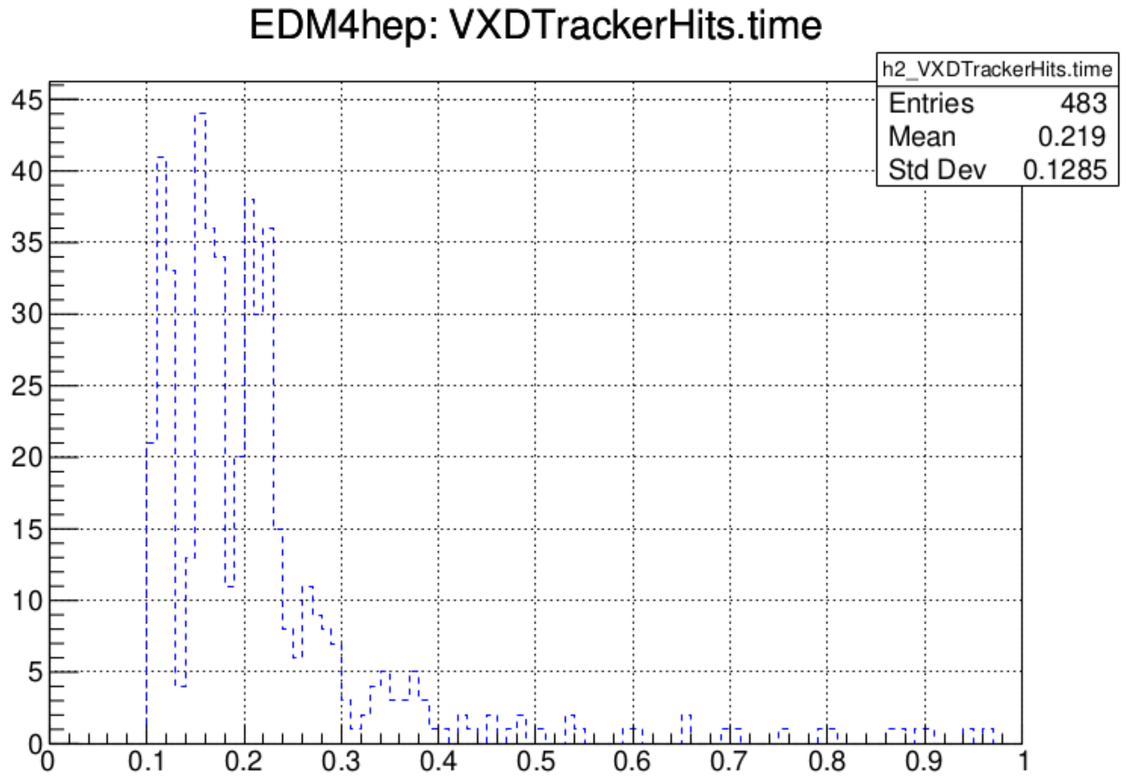




Results and comparison

CLICRec with both the Wrapper and my new algorithm





Next weeks work

- **Streamlog** (logging library used in Marlin)
- Optimize and prepare the algorithm to run in parallel with **Gaudi::Functional**

parallel processing ready:
  - const functions
  - functional
  - no global state
  - use Gaudi::Functional components

Conclusions

- **Great learning of a real world scientific software stack**
- **Real impact**: algorithm to be used in Key4hep as a drop-in replacement for the one in Marlin
- This first algorithm **leads the way** for other algorithms to follow
- **Still some work to do!**

Thanks!