Implementation of the LCFI+ algorithm into key4hep

FCC-ee topical meeting on vertexing - combined Phys Performance + Software

Plácido Fernández Declara, André Sailer

February 10, 2021

CERN

k4MarlinWrapper

- Part of the Key4hep project: https://github.com/key4hep/
- *Marlin* Processors functionality made available in Key4hep through the *Gaudi* framework.
- It contains the necessary interfaces to deal with *Marlin* formats to be run from *Gaudi* algorithms.
  - Wrapper around Marlin Processors
  - XML steering file to Python options file converter
  - EDM4hep to LCIO event converter in memory
- Marlin source code is kept intact, and can be called on demand.

Configuration and running

- Config and running done via Python file as with the Gaudi Framework.
- Processor parameters defined for each instance, and list algorithms configured.
- On algorithm initialization of Marlin Processors, the `MARLIN_DLL` environment variable is used to load the necessary libraries.

```
VertexFinder = MarlinProcessorWrapper("VertexFinder")
VertexFinder.OutputLevel = WARNING
VertexFinder.ProcessorType = "LcfiplusProcessor"
VertexFinder.Parameters = [
    "Algorithms", "PrimaryVertexFinder", "BuildUpVertex", END_TAG,
    "BeamSizeX", "6.4E-3", END_TAG,
    "BeamSizeY", "28E-6", END_TAG,
    "BeamSizeZ", "12", END_TAG,
    "BuildUpVertex.AVFTemperature", "5.0", END_TAG,
    ...
    "MCPCollection", "MCParticle", END_TAG,
    "MCPFORelation", "RecoMCTruthLink", END_TAG,
    "MagneticField", "2.0", END_TAG,
    "PFOCollection", "ReconstructedParticle", END_TAG,
    "PrimaryVertexCollectionName", "PrimaryVertices", END_TAG,
    ...
    ]
algList.append(VertexFinder)
```

XML to Python converter

- A converter from XML steering file to Python options file is available as a Python script.
- It produces the list of Gaudi algorithms, including optional Processors.
  - These are left as commented algorithms that need to be manually uncommented by the user.
  - A comment is also included to indicate its configuration.
  - `# algList.append(MyFastJetProcessor) # Config.OverlayNotFalse`
- It now includes *Constants* parsing from the XML
  - It lists the `CONSTANTS =` to be modified by the user
  - These are replaced in the processors with String substitution: `"%(DD4hepXMLFile_subPath)s" % CONSTANTS`
  - It now supports lists of arguments in the constants as well
- `Marlin -x` can create a steering file containing all the parameters for the known processors. This can be converted to python.

EDM4hep to LCIO conversion

- Conversion between EDM4hep and LCIO needed to run with Marlin Processors
- LCIO to EDM4hep conversion available here: https://github.com/key4hep/k4LCIOReader
- Uses *k4FWCore* to read the input collections indicated in the options file
  - https://github.com/key4hep/k4FWCore

```
from Configurables import k4DataSvc, ToolSvc,
    MarlinProcessorWrapper, EDM4hep2LcioTool
theFile= 'edminput.root'
evtsvc = k4DataSvc('EventDataSvc')
evtsvc.input = theFile

from Configurables import PodioInput
inp = PodioInput('InputReader')
inp.collections = [
    'ParticleIDs',
    'ReconstructedParticles',
    'EFlowTrack'
]
inp.OutputLevel = DEBUG
algList.append(inp)

ToolSvc.LogLevel = DEBUG
```

EDM4hep to LCIO conversion

- Converter implemented in *k4MarlinWrapper* as a Gaudi Tool
- Configured in the options file indicating which processor needs to use the Tool to convert the EDM4hep event to LCIO format
- Events are read through the DataHandle from k4FWCore; these are converted and registered in the Transient Event Store (TES) to make it available to the framework.

```
MyFastJetProcessor = MarlinProcessorWrapper("MyFastJetProcessor")
...
MyFastJetProcessor.Conversion = [
    # type                                    EDM4hep Collection        LCIO Collection
    "edm4hep::TrackCollection",               "EFlowTrack",             "Tracks",
    "edm4hep::ReconstructedParticleCollection", "ReconstructedParticles", "PFOCollection"
]
MyFastJetProcessor.addTool(EDM4hep2LcioTool())
...
algList.append(MyFastJetProcessor)
```

Development of EDM4hep to LCIO conversion

- During conversion some Collections depend on each other.
- i.e. A `ReconstructedParticle` contains a `ConstVertex`, which then links to a `ReconstructedParticle` again.
- Some inconsistencies were found between EDM4hep and LCIO.
- i.e. EDM4hep `vertex.getAlgorithmType()` returns a `int`, whereas LCIO vertex expects a `std::string`.
- The track subdetectorHitNumbers() needs to be resized manually to accommodate the hits and avoid memory errors.
- PR for the converter tool in Key4hep: https://github.com/key4hep/k4MarlinWrapper/pull/26

Conclusions

- The converter tool is able to run the ”VertexFinder” LCFI+ Processor, converting the EDM4hep collections to LCIO *on the fly*.
- Conversion of the output collection back to EDM4hep needs to be integrated/implemented.
- More collection types to be converted are being supported.
- Feedback from real world usage appreciated.