Key4hep: Current status and recent developments

Juan Miguel Carceller

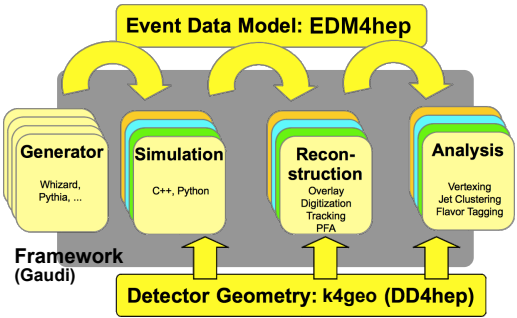
on behalf of the Key4hep team

CERN

October 12, 2023

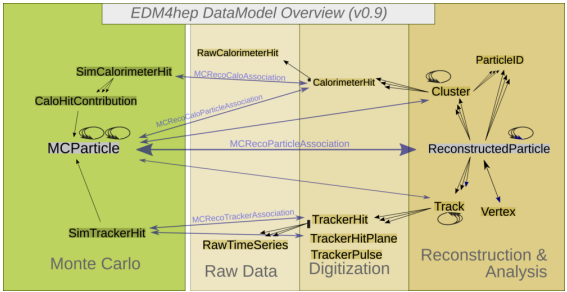
Key4hep

- Turnkey software for future accelerators
- Share components to reduce maintenance and development cost and allow everyone to benefit from its improvements
- Complete data processing framework, from generation to data analysis
- Community with people from many different experiments: C^3, CEPC, CLIC, EIC, FCC, ILC, Muon Collider, etc.



The Key4hep Event Data Model: EDM4hep

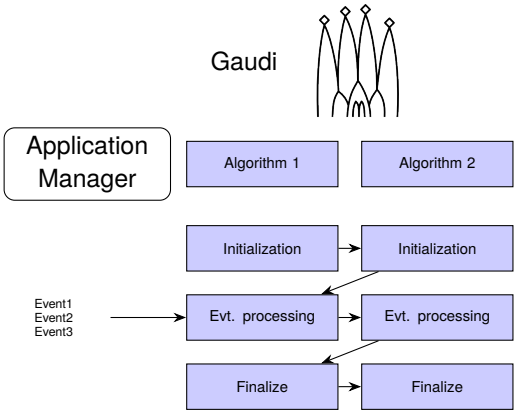
- Data Model used in key4hep, it is the language that all components must speak
- From a specification in a yaml file, and using podio, the C++ code containing all the classes and methods is generated



- Classes for physical objects, for example: MCParticle
- Associations between these, for example: between MCParticle and a ReconstructedParticle
- Adapt based on the news of the collaborators. Example: RawTimeSeries previously was TPCHit

The Key4hep Framework

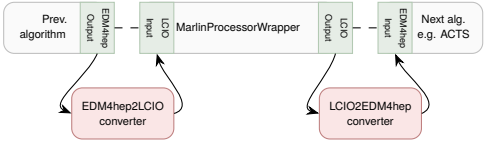
- Gaudi based core framework:
  - k4Gen for integration with generators
  - k4SimGeant4 for integration with Geant4
  - k4SimDelphes for integration with Delphes
  - k4geo for detector models, previously lcgeo
  - k4FWCore provides the interface between EDM4hep and Gaudi
  - k4MarlinWrapper to call Marlin processors
  - …

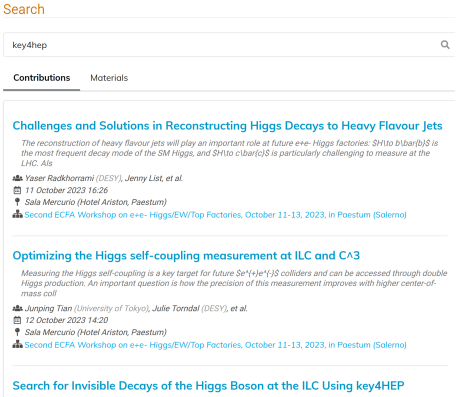


- Used by LHCb, ATLAS, Key4hep and others

The Key4hep Framework

- Key4hep provides the interface and the glue to make the different pieces talk to each other
- Components are picked up from other places (for example Gaudi for the event processing framework)
- iLCSoft can be used thanks to the k4MarlinWrapper, we benefit from lots of years of development and tested software
- Results on detector studies, analyses, etc





The Key4hep Stack

- Software provided in “stacks” deployed on cvmfs
- More than 500 packages (most are dependencies)
- Nightly builds in `/cvmfs/sw-nightlies.hsf.org` with the latest version of the key4hep packages and other packages. CentOS 7, AlmaLinux 9 and Ubuntu 22.04 supported
- Releases in `/cvmfs/sw.hsf.org` with tagged versions of the packages
- Easy setup with cvmfs:

```
source /cvmfs/sw-nightlies.hsf.org/key4hep/releases/setup.sh   # Latest nightly
source /cvmfs/sw.hsf.org/key4hep/releases/setup.sh             # Latest release
```

- Questions, problems, complaints and anything else related to packages happens in https://github.com/key4hep/key4hep-spack

New developments

Podio and EDM4hep

- Schema evolution: it is possible to modify our definitions in EDM4hep and still be able to read old data
- Schema evolution is working and has been added to podio, and by extension, to EDM4hep [T. Madlener and B. Hegner]
  - Renaming, adding a new member, removing a member is now possible
- New RNTuple backend to write RNTuples (experimental new format for ROOT files)
- Python bindings for EDM4hep

```
import edm4hep
particle = edm4hep.MCParticle() # default initialized particle
particle.getCharge() # 0.0
```

- Together with the podio bindings it is possible to read or generate data and save it to a file, all from python
- Few minor changes in the model itself $\rightarrow$ very stable

Frame reading and writing in Key4hep

- The Frame (from podio) is a data container where collections can be stored
- Key feature: support for multithreading
- Typically represents an event but can be anything else
- A backend decides how it is written to a file (ROOT files with ROOT TTrees most of the time)
- Automatic Frame reading and writing was introduced to the key4hep Gaudi algorithms
  - Previously, files were being saved to an Event Store (didn't support multithreading)
  - Now, files produced are read and written as Frames
  - Changes were transparent; users didn't need to modify their code

Simple interface with `get` and `put`

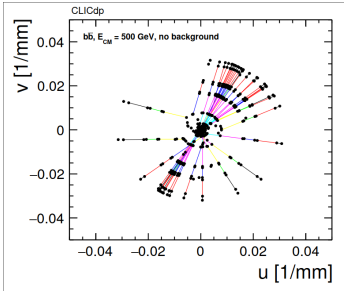
```
frame.get("MCParticleCollection");
frame.put(std::move(coll), "NewCollection");
```

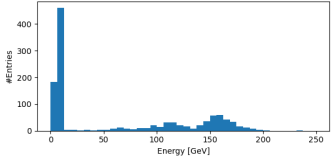
Also in python:

```
from podio.root_io import Reader
reader = Reader('myfile.root')
events = reader.get('events')
for frame in events:
    coll = frame.get('MCParticleCollection')
```

Integrations and native algorithms

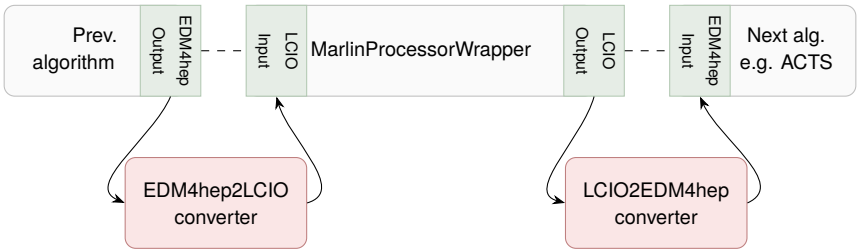
- There is ongoing work on either native tools or integration with other tools
  - Integration with ACTS (L. Reichenbach)
  - Integration of the Pandora Particle Flow Algorithm (S. Sasikumar)
  - Digitisation algorithm in native key4hep (J.M. Carceller)
  - Overlay in native key4hep (Y. Khrabatyn, summer student at CERN)





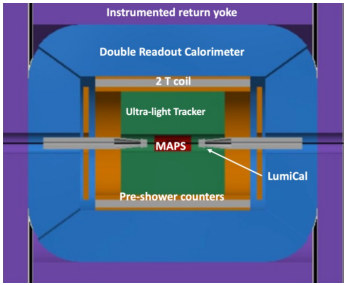
LCIO Converters

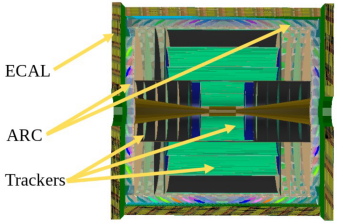
- There has been an overhaul of the EDM4hep - LCIO converters
  - Fixes some issues: associations were broken when using EDM4hep input in the conversion from LCIO to EDM4hep
- Current status
  - Marlin processors can be used in Gaudi using a ‘MarlinProcessorWrapper‘
  - EDM4hep input can be used and not worry about Marlin processors taking LCIO input and giving LCIO output
  - Standalone converter `lcio2edm4hep` to convert files



Detector and Reconstruction studies

- New detectors have been added recently to k4geo:
  - IDEA
  - ALLEGRO
  - CLD with the ARC subdetector
  - IDEA Vertex Detector
- See A. Tolosa's talk and A. Sailer's talk



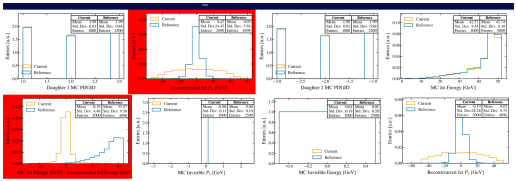


The Key4hep stack

- Nightly builds now support CentOS 7, AlmaLinux 9 and Ubuntu 22.04 (previously it was only CentOS 7)
- Support for these three OSes will also extend to new releases
- Several recent fixes:
  - Graphic visualizations (Geant4, DD4hep) now work well in the nightlies
  - Nightly validation, broken nightlies are not deployed on cvmfs
  - Improvements in CI so that it is possible to rebuild a nightly at a certain point in time
  - New packages added to the stack

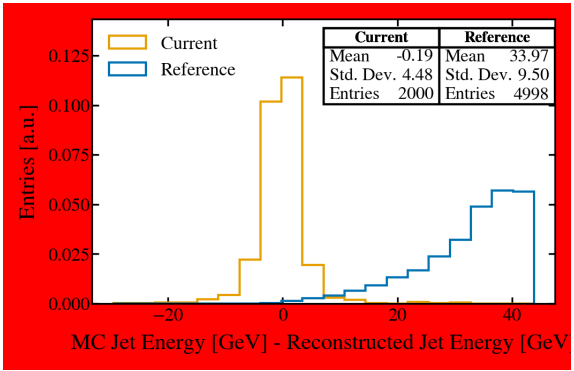
Key4hep Validation: Simulation and Reconstruction

- Check the simulation and reconstruction chain
- Run daily, use the latest key4hep nightlies
- Run a simulation with DD4hep, then reconstruction, then analysis scripts and then make plots
- Results are compared to a reference sample
- Plots are deployed to WebEOS (static webpage)
- https://key4hep-validation.web.cern.ch/
- Work in progress, no documentation yet



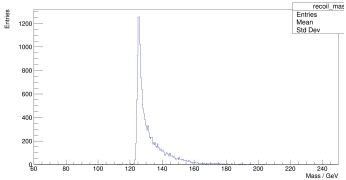
Key4hep Validation: Simulation and Reconstruction

- Example of a plot using a release with a bug as the reference one



Key4hep Tutorial

- Key4hep tutorial on Tuesday
- Several topics covered
  - EDM4hep
  - LCIO EDM4hep converters
  - Algorithms in Key4hep using Gaudi
  - Plotting from files
- Documentation will be kept online
  https://github.com/key4hep/key4hep-tutorials
- Feel free to ask questions / report issues about the tutorials in person or by mail or github



Outlook: Functional algorithms

- One of the key features of the Frame is support for multithreading
- Current algorithms don't typically work in a multithreaded environment
  - If there is internal state several threads could change it at the same time
- Functional algorithms is the recommended way in Gaudi
  - Algorithms can not modify an internal state
- Working with EDM4hep collections using functional algorithms is now possible

Summary

- Lots of progress in key4hep in different areas
- Many more improvements, impossible to list them all!
- More to come, expect more integrations and native algorithms in key4hep, as well as bug fixes and quality of life improvements
- Very wide community with people from different experiments
- Many thanks to all the developers, users and bug reporters!

Acknowledgements

- CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)
- European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 101004761.

Backup

Key4hep Builds: Nightlies

- Nightlies were updated
- Two build types depending if the build is done from scratch or not
- Builds from scratch will get the latest packages and let us know that the changes in spack don't break our builds - Updates every O(weeks)
- Daily builds that use as upstream the builds from scratch, they only build the packages that have changed (or those that depend on a package that has changed)
- Three new hidden files in each release
  - `.scratch`: If it is there, it is a build from scratch (all packages)
  - `.spack-commit`: Which commit of spack was used to build this release
  - `.key4hep-spack-commit`: Which commit of `key4hep-spack` was used to build this release

Key4hep Builds: Tests

- New usability tests are being added: compilation, ROOT, python, python packages, whizard, key4hep tools
- These tests come from experience, mainly from what people use that one day doesn't work, to make sure it doesn't repeat

```
cat > ee.sin <<EOF
    process ee = e1, E1 => e2, E2
    sqrts = 360 GeV
    n_events = 10
    sample_format = lhef
    simulate (ee)
EOF
run_test "whizard test" "whizard -r ee.sin"
```

podio: RNTuple backend

TTree based

```
ROOTFrameWriter writer(filename);
writer.writeFrame(frame);
writer.finish();
```

```
ROOTFrameReader reader{};
reader.openFile(filename)
auto event = podio::Frame(reader.readEntry("events", 0));
```

RNTuple based

```
ROOTNTupleWriter writer(filename);
writer.writeFrame(frame);
writer.finish();
```

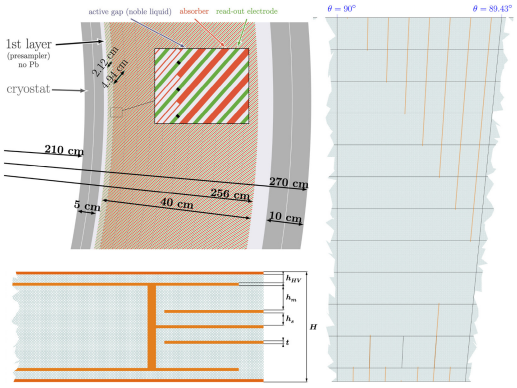
```
ROOTNTupleReader reader{};
reader.openFile(filename)
auto event = podio::Frame(reader.readEntry("events", 0));
```

- For the future:
  - Comparisons between the RNTuple and TTree-based backends: reading and writing speed, file size
  - Python bindings for the RNTuple writer and reader

Pandora

- Liquid Argon (LAr) detectors are being studied for future experiments (e.g. FCC)
- Swathi will study jet energy resolution for IDEA-LAr when a full simulation for IDEA is implemented

- Currently studying LAr with CLD (with full simulation)
- While adding the LAr calorimeter inside CLD overlaps were found so changes in the geometry had to be done
- Implementing a Gaudi algorithm to use Pandora PFA with Gaudi



Other improvements

- Many other small (or not) improvements
- Fixes and improvements in k4run in k4FWCore: #93, #110, #126, #131, #132, #143, #137, #138 and #140
- New packages added to the stack and / or nightlies (for example, opendatadetector) as requested
- Improvements in the documentation
- New tutorials

Functional algorithms in Key4hep

```cpp
struct ExampleFunctionalConsumer final
    : Gaudi::Functional::Consumer<void(const edm4hep::MCParticleCollection& input), BaseClass_t> {
  ExampleFunctionalConsumer(const std::string& name, ISvcLocator* svcLoc)
      : Consumer(name, svcLoc, KeyValue("InputCollection", "MCParticles")) {}

  void operator()(const edm4hep::MCParticleCollection& input) const override {
    // Do something with the input
  }
};
```

example exctracted from k4FWCore