Key4hep Software Stack for Detector Studies

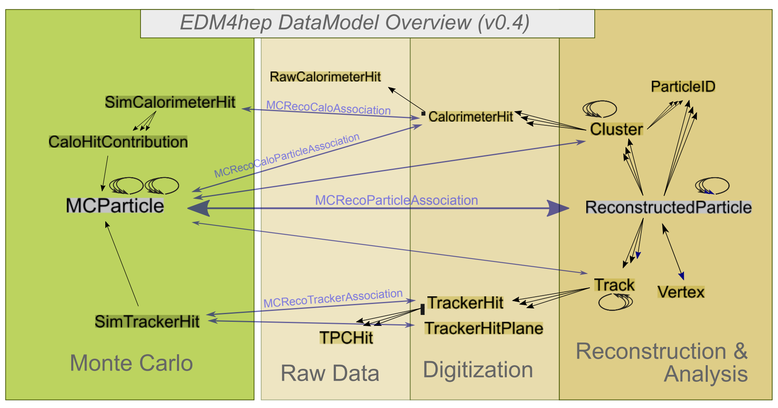
Placido Fernandez Declara for the Key4hep team
CERN, Switzerland
placido.fernandez@cern.ch

Introduction

Detector optimisation and physics performance studies are an integral part of the development of future collider experiments. The Key4hep project aims to design a common set of software tools for future, or even present, High Energy Physics projects. Based on the iLCSoft and FCCSW frameworks an integrated solution for detector simulation, reconstruction and analyses is being developed. It shows the seamless integration of fast simulation with Delphes and the LCFIplus vertexing processor from iLCSoft, demonstrates the execution of multi-threaded simulation of a drift chamber in the CEPC experiment, and shows the status of the multi-threaded execution of the iLCSoft processors in the Gaudi framework making use of the k4MarlinWrapper.

Event Data Model: EDM4hep

- Common Event Data Model between components
- podio based [1]
- Fast, efficient, multithreaded and transparent I/O



Simulation: k4SimGeant4

- Geant4 simulation coherent with Key4hep
- Common Gaudi interface for simulation and reconstruction
- Outputs EDM4hep format

Detector model: DD4hep

- Complete detector description: Geometry, materials, visualization, readout, alignment, calibration
- Single source of information for simulation, reconstruction and analysis
- Plug-in mechanism for customization
- Industry standard: used by FCC, CEPC, ILC, CLIC, EIC, LHCb, CMS...

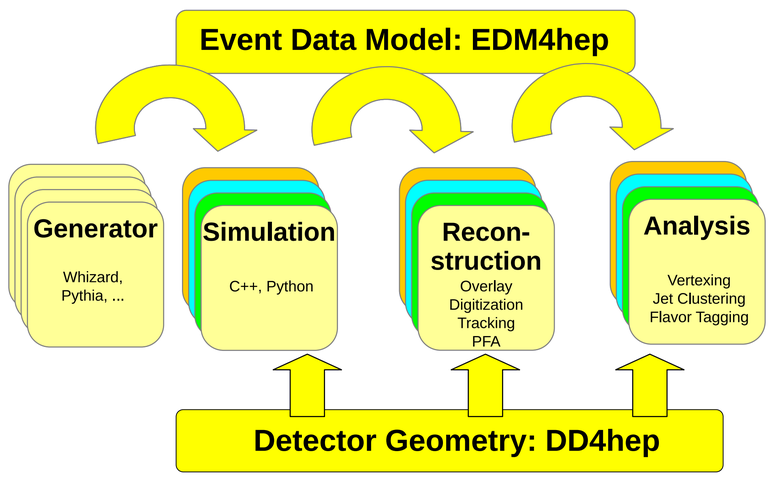
Package manager: Spack

- Package manager developed by the HPC community, OS independent, builds all packages from source
- Deals with different configurations of the same package
- Key4hep stack is built with Spack
- Deployed to CVMFS: `/cvmfs/sw.hsf.org/key4hep/setup.sh`

References

[1] github.com/aidasoft/podio.
[2] Delphes. doi.org/10.5281/zenodo.4896637.
[3] Lcfiplus. doi.org/10.5281/zenodo.5714029.
[4] github.com/key4hep/key4DCMTSim.
[5] k4MarlinWrapper. doi.org/10.5281/zenodo.5584540.
[6] indico.cern.ch/event/855454/contributions/4596547.
[7] indico.cern.ch/event/855454/contributions/4596738.

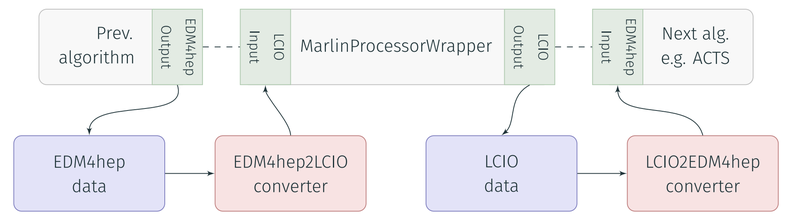
Key4hep



Framework: k4FWCore

- Core framework functionality
  - Data service for podio collections
  - Overlay for backgrounds
- Gaudi-based framework for Full/Fast Simulation and Reconstruction

k4MarlinWrapper



- Run Marlin Processors as Gaudi Algorithms
- Steering file converter (XML → Python)
- Input and Output support for LCIO and EDM4hep
- In-memory on-the-fly converters between LCIO ↔ EDM4hep

Simulation: k4SimDelphes

- Uses Delphes for simulation and reconstruction
- Available as standalone executables
- Quick way to get your hand dirty and do physics with EDM4hep

Delphes and LCFIplus

- Group of algorithms to perform vertexing, jet finding and flavour tagging.
- Uses iLCSoft algorithms: different EDMs are used to run in Key4hep through in-memory conversion and interfaces
  - Generation uses Delphes [2] and outputs to EDM4hep format
  - LCFI+ [3] uses LCIO: on input EDM4hep is converted to LCIO; on output LCIO is converted back to EDM4hep.

Multi-threaded simulation for CEPC drift chamber

- To validate the design of the CEPC detector, a machine-leaning based simulation tool was developed to simulate detector response inside the cell of the drift chamber [4]. The fast simulation tool has also been tested with Gaudi Hive:
  - Case 1 with Gaudi Algorithm: concurrent simulation ran successfully.
  - Case 2 with Gaudi Functional: a memory error occurred and is still being investigated.

iLCSoft parallel execution in Key4hep

- iLCSoft steering files can run multiple events in parallel by using Gaudi Hive [5]
  - Using `HiveWhiteBoard`, `HiveSlimEventLoopMgr` and the `AvalancheSchedulerSvc` from Gaudi.
  - Algorithms in the same event run sequentially.
- Adaptations are needed to enhance k4FWCore to support multithreaded. More testing is needed for concurrent access and writing of the EDM.

Next steps

- Work on podio for schema evolution and multithreading
- Validation of parallel execution of components of Key4hep and EDM4hep
- Further integration of CLUE [6] and ACTS [7] into Key4hep for coherent usage
- Documentation, how to contribute and get started: https://cern.ch/key4hep

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 871072 and 101004761. This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)