The Key4hep turnkey software stack for future colliders

EPS-HEP conference 2021

Placido Fernandez Declara (CERN) for the Key4hep software group

July 29, 2021

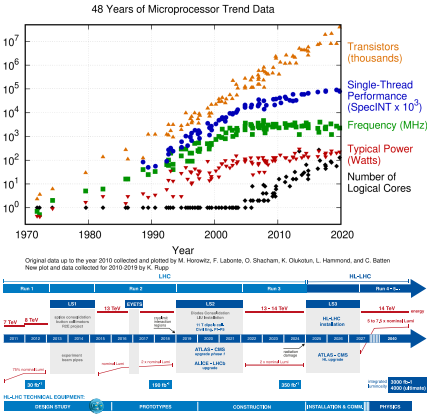
CERN

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

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

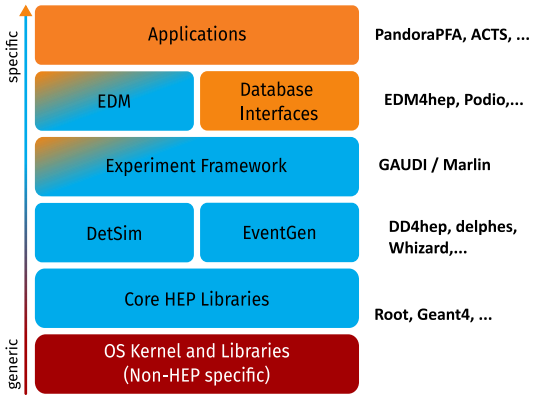
Software challenges on HEP

- Future detector studies rely on well maintained software: proper studies of detector concepts, their physics reach and limitations
- Long Lifetimes – Shift of priorities throughout the evolution of an experiment
  - Conceptual and design work: quick iterations relying on simulation
  - Production quality: requires stability & continual updates
  - Data preservation
  - Avoid amassing "technical debt"
  - Computing performance and efficiency
  - New technological developments and paradigms



HEP Software stack

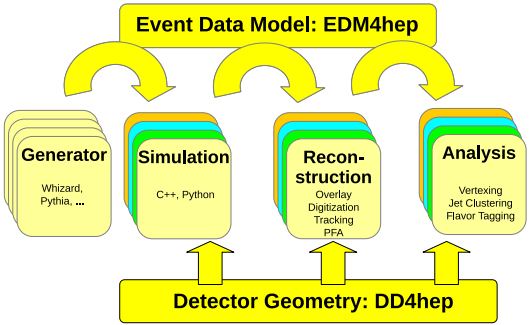
- Pieces of software are not living in isolation: interaction between components
- Compatibility between different elements doesn't come for free
  - Common standards can help a lot
- Building a consistent stack of software for an experiment is highly non-trivial
  - Benefits can be gained from using common approaches



Key4hep Goals

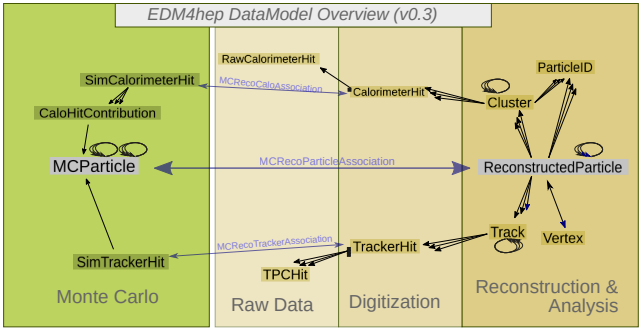
Software stack that connects and extends packages to provide a complete data processing framework, comprising fast and full simulation, reconstruction and analysis.

- Contributions from FCC, CLIC, ILC, CEPC
  - Re-use existing tools where possible
- EDM4hep: based on LCIO and FCC-edm
- Gaudi: framework
- DD4hep: geometry information
- Spack: package manager
- Ease of use for librarians, developers and users
- Provide examples, documentation, templates and common practices



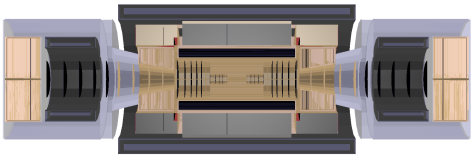
- To facilitate interoperability, different components should talk the same language
- In HEP this is the Event Data Model: describes structure of HEP data

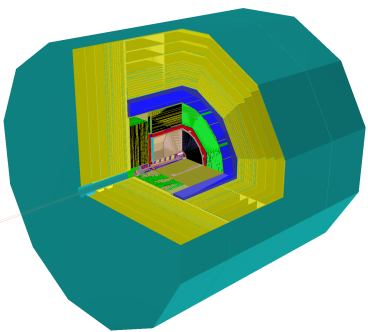
- Based on LCIO and FCC-edm
- Focus on: fast, efficient, multithreaded and transparent I/O system
- Implemented with PODIO - *vCHEP 2021: EDM4hep and podio*



DD4hep - Detector Description Toolkit for HEP

- Originally developed for ILC and CLIC but with all of HEP in mind
- Provides a complete detector description
  - Geometry, materials, visualization, readout, alignment, calibration, ...
- From a single source of information
  - Used in simulation, reconstruction, analysis
- Comes with a powerful plug-in mechanism that allows customization
- More or less “industry standard” by now
  - ILC, CLIC, FCC, CEPC, EIC, ...
  - LHCb and CMS to use DD4hep for Run3
- `ddsim` already directly outputs EDM4hep
  - when writing ...`edm4hep.root` files





Software Framework for HEP: Gaudi

- Traditionally HEP has not done too well with sharing efforts towards a common experiment framework. Notable exceptions are:
  - *Marlin* developed and used by ILC and CLIC
  - *Gaudi*[1], originally developed by LHCb, now also by ATLAS
    - Also used by FCCSW and other experiments (HARP, BESIII, Minerva, Daya Bay, ...)
- Key4hep has decided to adapt `Gaudi` as its base framework
  - Several Key4hep components are developed on top of it
  - Contribute to its development where necessary
- Integration and migration of iLCSoft algorithms into Key4hep with the help of a `Marlin`→`Gaudi` wrapper
  - Allows to use `Marlin` processors within the `Gaudi` framework

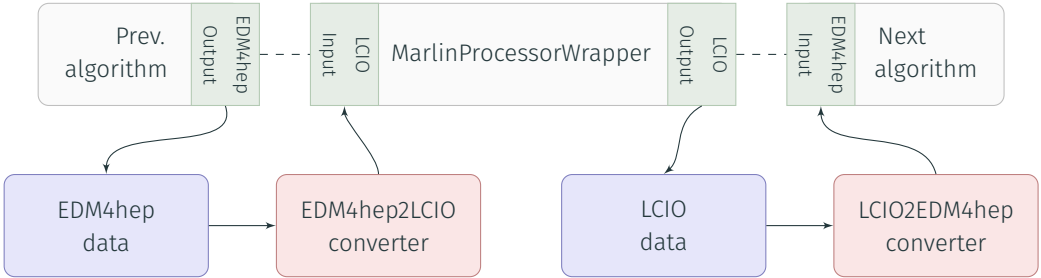
[1] https://cern.ch/gaudi

k4MarlinWrapper

- Run *Marlin* Processors through *Gaudi* Algorithms, without changes to *Marlin*
- Enhanced *Marlin* XML to *Gaudi* Python steering files converter
  - Optional execution markers
  - *Constants* parsing: propagated through the steering file
- Input and output support for LCIO and EDM4hep
  - Read events with *LcioEvent* or *PodioInput* using *k4DataSvc*
  - Write events with wrapped *LCIOOutputProcessor* or *PodioOutput*
- Uses cases with *k4MarlinWrapper*:
  - CLIC reconstruction: full CLIC processors sequence
  - LCFI+ algorithm: vertex and jet finding, and flavour tagging
    - Different EDMs mixed to run the sequence: EDM converters
  - SCT integration: use k4MarlinWrapper in Aurora framework

k4MarlinWrapper: EDM converters

- In-memory on-the-fly conversion for LCIO ↔ EDM4hep
- Implemented as Gaudi Tools, can be attached to any *MarlinProcessorWrapper*
- LCIO → EDM4hep conversion achieved through k4LCIOReader
- Metadata conversion is being implemented
- Time measurements for the converters



k4SimDelphes - First steps towards physics

- k4SimDelphes uses delphes to do the simulation and reconstruction, and creates output files in EDM4hep format
- Quick way to get your hands dirty and do some physics with EDM4hep
- Currently available as standalone executables
  - E.g. `DelphesPythia8_EDM4HEP`, `DelphesSTDHEP_EDM4HEP`, ...
- Part of a coherent approach to generation / simulation in Key4hep
  - Ideally no difference between the different approaches of simulating a detector response
  - Work on full integration is ongoing with a prototype

Key4hep Framework

- k4FWCore
  - Core Key4hep framework providing core functionality, e.g.
    - Data Service for PODIO collections
    - Overlay for backgrounds
  - Recently switched to Gaudi v35
- k4-project-template
  - template repository showing how to build new components on top of the core Key4hep framework
- Ongoing work to collaborate more with Gaudi ecosystem (Gaussino)
- Ongoing work to integrate more components (ACTS, ...)

Software Infrastructure

- Documentation
  - key4hep.github.io/key4hep-doc (main documentation)
  - cern.ch/edm4hep (doxygen code reference)
- Spack package manager to build whole stack from source
  - Emphasis on dealing with multiple configurations of same package
  - Several versions, compilers, external library version, ...
- Automated builds and continuous integration (CI) where possible
  - Regular nightly builds of the complete stack
- Distribution via CVMFS
  - Latest release can be found at `/cvmfs/sw.hsf.org/key4hep`
- Release early and release often
  - Make fixes available early
  - Discover problems and collect feedback as early as possible

Key4hep adaptations

- CEPC
  - Started with iLCSoft; adopted Gaudi, EDM4hep, DD4hep
  - Developed Geant4 based sim. Implemented k4LCIOReader
- FCCSW
  - Gaudi and DD4hep based; Adaptation to EDM4hep
  - FCCSW re-arranged into Key4hep components: k4FWCore, k4SimDelphes, k4RecCalorimeter, fccDetectors, and others
- iLCSoft
  - ILC and CLIC: keep existing software chains, enabled by k4MarlinWrapper
  - Port algorithms to Gaudi in the future
- SCT
  - Adaptation to EDM4hep in progress
  - Integration with k4MarlinWrapper to enable Marlin algorithms

Summary and outlook

- Core components of Key4hep already -or close to- a first production quality release
  - Provides framework, simulation and reconstruction
- Opportunity for HEP community to develop a common software ecosystem, for the first time.
  - Leverage synergies between communities and existing experiments to develop and maintain common core tools
- Projects have started migrating their software to Key4hep: FCCSW, ILC, CLIC & CEPC

Backup

Key4hep common tooling

- Not only common software HEP tools, but also general software tools and practices.
- Templates provided for new packages to follow common structure with folders.
- Common use of build systems: CMake, gcc, clang.
- Encourage use of common software analysis tools: sanitizers.
- Encourage uniform consistent code style and practices: clang format, clang tidy.
- Common testing tools: Catch2, Pytest.
- Unified building and deployment with Spack and CVMFS.
- Moving towards common interfaces, frameworks, packages, versions, etc.
- Modern tooling: compilers, language standards,

Spack for Key4hep

- Spack is a package manager
  - Does not replace CMake, Autotools, ...
  - Comparable to apt, yum, homebrew, ...
  - Independent of operating system
  - Builds all packages from source
- Originaly developed by the HPC community
  - Emphasis on dealing with multiple configurations of the same package
    - Different versions, compilers, external library versions, ...
  - Several versions of the same package can coexist on the same system
- The whole Key4hep software stack can be built from scratch using spack
- Spack allows different workflows for setting up consistent software stacks
  - Currently testing which one fits our purposes the best