MC generators

FCC Software HandsOn tutorial

October 20, 2022
G Ganis, CERN-EP

MC Generators

See also W. Kilian's, ECFA kick-off, June 2021

Needs

- High energy (√s > hZ threshold) e+e- generators
- Generators at Z peak, WW
- Heavy Flavour decays, including taus

Examples of heavily used codes for LC

- Whizard, MadGraph5_aMC, PhysSim, Pythia6, ...

Areas of work (not exhaustive)

- Recovery of LEP generators, but still state of art for Z peak, WW
  - KKMC family, BHLUMI, BHWIDE, Babayaga, … interfacing work ~~in progress~~ mostly done
- Hadronization “tunes” for e+e- (Pythia, Herwig, …) partly done
  - Eg. cannot import Pythia6 tune (from LEP) to Pythia8
- Interfaces with up-to-date decay codes (EvtGen, …) work needed

Monte Carlo Generators in key4hep

- A Monte Carlo generator is a package
- Key4hep includes already many generators as packages
  - Initial list derived from LCG stacks, so sort of LHC oriented
  - But several e+e- additions available: Whizard, KKMCee, BabaYaga, BHLUMI, …
    - Including wrappers for better user experience
- What does it mean “adding a generator to key4hep”?
  - Required information for inclusion in the package manager
    - Source location, minimal documentation on how to build and required dependencies, default configuration files, tests, ...
  - Key4hep infrastructure will
    - Build in shared installation mode
    - Run built-in tests, if any
    - Install in distributed shared file system

List of MC Generators currently available in key4hep

From spack upstream

form 4.2.1
crmc 1.6.0
photos 3.64
qd 2.3.13
evtgen 2.1.0
lhapdf 6.3.0
mcutils 1.3.5
qgraf 3.4.2
rivet 3.1.4
hepmc 2.06.11
vbfnlo 2.7.1
syscalc 1.1.7
thepeg 2.2.1
chaplin 1.2
heputils 1.3.2
looptools 2.15
njet 2.1.1
pythia6 6.4.28
sherpa 2.2.11
hepmc3 3.2.4
collier 1.2.5
gosam 2.0
herwig3 7.2.1
madgraph5amc 2.8.1
openloops 2.1.2
pythia8 8.306
recola 2.2.3
yoda 1.9.0

From key4hep-spack

guinea-pig 1.2.2rc
whizard 3.0.1
KKMCee 5.00.01
BHLUMI 4.04-linuxLHE
Babayaga fcc-1.0.0

Levels of interoperability

- Level 0 - *Common Data Formats*
  - Maximal interoperability, even on different hardware
- Level 1 - *Callable Interfaces*
  - Defined for one or more programming languages
  - Implementation quality of interfaced components important
  - Required to define plugins
- Level 2 - *Introspection Capabilities*
  - Software elements to facilitate the interaction of objects in a generic manner such as Dictionaries and Scripting interfaces
  - Language bindings, e.g. PyROOT
- Level 3 - *Component Level*
  - Software components are part of a *common framework*, optimal interplay
  - Common configuration, log and error reporting, plug-in management, ...

Common Data Formats

Recommended

- HEPMC
  - Generic framework for MC generator event record encoding and manipulation
  - Version 3 complete re-write of previous version and generally adopted
  - Provide tools for managing other formats, e.g. LHEf files
- EDM4hep
  - Common multipurpose event data model
  - Contains relevant data structures

Accepted

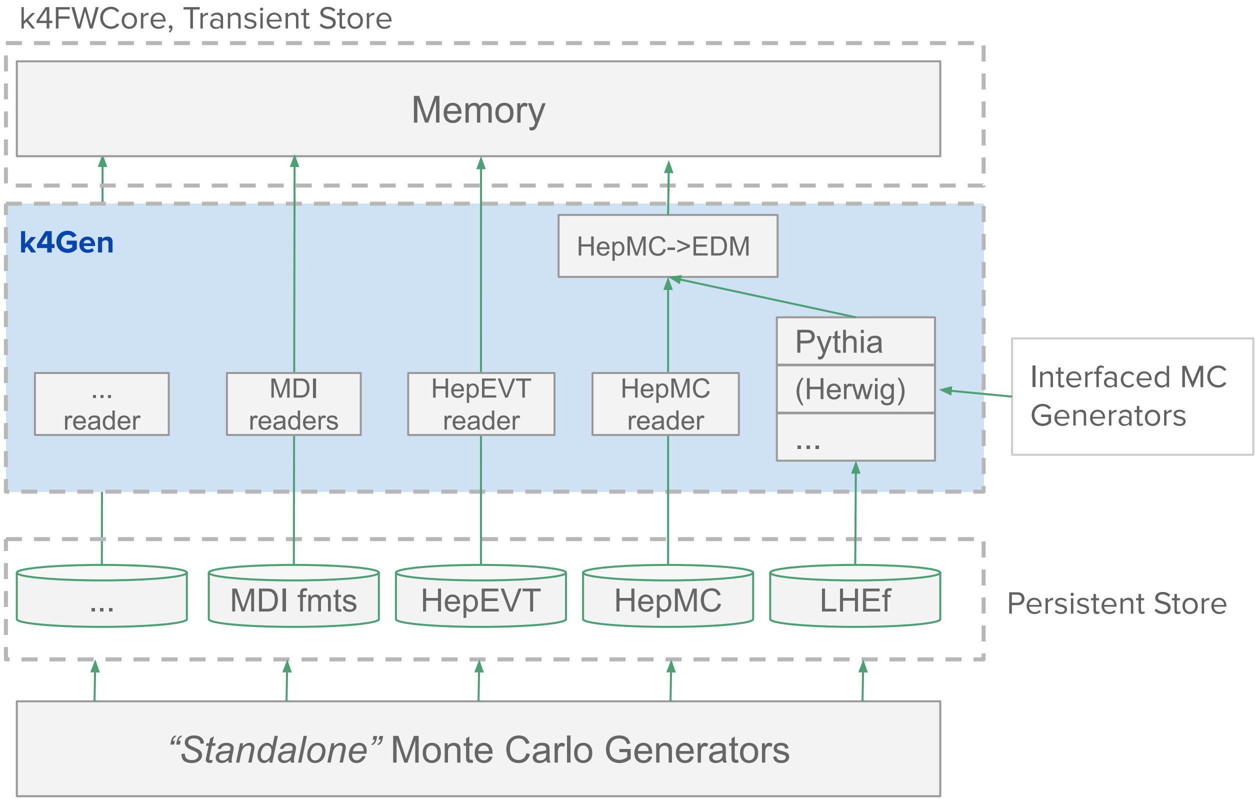
- LHEf
  - Format agreed for LHC, generally OK for e+e-
  - Some issues for FCC (see FCC presentation)
- HepEVT
  - Format used by old fortran generators
- Potentially any format completely documented

Managing interoperability through Gaudi

- Components tailored for specific tasks
  - Tools for inputting and managing MC outputs
- What is required
  - Readers for data formats
    - Currently available: HepEVT, HepMC, LHEf, ...
  - Level-1 interfaces for MC
    - Only currently available Pythia8
    - Place where to put an interface to Herwig

The repository is k4Gen, and includes also modules to perform actions on the MC events, such as vertex smearing, particle filters, or basic MC tools, such particle guns

Managing interoperability



Hands-on

- Main page (added Whizard for LHEf)
- Steps
    - Generating ditaus with KKMCee
        - HepMC to EDM4hep conversion
    - Generating ditaus with Pythia8
    - Generating ditaus with Whizard
        - LHEf to EDM4hep conversion
    - Looking at the produced files: the MCParticle class
        - Creating flat ntuples with FCCAnalyses
        - Comparing distributions