FCC Analyses and RDataFrame

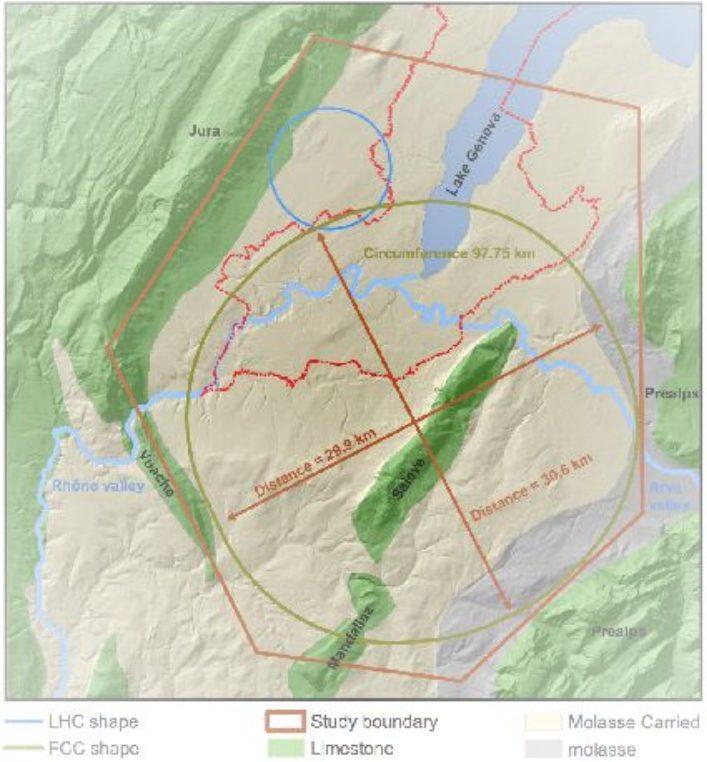
From the past (Heppy) to the future (EDM4hep)

PPP Developers Meeting October 2019

Oct 17, 2019

Valentin Volkl for the FCC Software Group
Univ. Innsbruck / CERN

Future Circular Collider Design Study



Conceptual Design Report based on

- 150 TB of full simulation data
- 100 TB of Delphes events

produced with Gaudi-based framework FCCSW

FCC-hh(ee) Framework

Fully integrated analysis chain
Links between various steps

1. GridPack producer
   - Makes MG5_aMC@NLO GridPacks, need to understand if other GEN can do it
2. LHE Producer
   - Produce LHE files on condor queues, either from step 1) or on the fly. Need a detailed review of the possibilities offered by other GEN (this has started as we have just seen)

Creates a database of LHE events

3. FCCSW
   - Runs Pythia8 parton shower+hadronisation and Delphes with FCC detector
     http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_fccee_v01.php
     Also need to support later other parton showers

Creates a database of FCC events

4. Analysis pre-selection
   - Python framework produces flat ROOT trees

Use the events in the database to produce analyses templates

5. Analysis final selection
   - Python framework for optimising analysis cut flows and producing

Use the events produced at pre-selection to create stacked plots

6. Final analyses (custom, but we have limit setting tool for instance)

FCC-hh(ee) Framework

Fully integrated analysis chain
Links between various steps

1. fcc-physics-events.web.cern.ch

Home | About | Contact | 100TeV FCC Physics | 27TeV HE-LHC Physics | Full Simulation FCChh | FCCee Physics | Stat

Delphes FCC Physic events v0.2

Search for names..

NO	NAME	NEVENTS	NWEIGHTS	NFILES	NBAD	NEOS	SIZE (GB)	OUTPUT PATH	MAIN PROCESS	FINAL STATES	CROSS SECTION (PB)
1	mgp8_pp_hh_lambda096_5f_htautau	980,000	0	98	1	99	45.63	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda096_5f_htautau/	HH, H->bb, H undec., kl = 0.96		1.47359
2	mgp8_pp_hh_lambda097_5f_htautau	1,000,000	0	100	0	100	46.55	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda097_5f_htautau/	HH, H->bb, H undec., kl = 0.97		1.46195
3	mgp8_pp_hh_lambda098_5f_htautau	990,000	0	99	0	99	46.13	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda098_5f_htautau/	HH, H->bb, H undec., kl = 0.98		1.45039

4. Analysis pre-selection
   - Python framework produces flat ROOT trees

5. Analysis final selection
   - Python framework for optimising analysis cut flows and producing

6. Final analyses (custom, but we have limit setting tool for instance)

Use the events in the database to produce analyses templates

Use the events produced at pre-selection to create stacked plots

FCC-hh(ee) Framework

Fully integrated analysis chain
Links between various steps

1. GridPack producer

Home | About | Contact | 100TeV FCC Physics | 27TeV HE-LHC Physics | Full Simulation FCChh | FCCee Physics | Stat

Delphes FCC Physic events v0.2

Search for names..

NO	NAME	NEVENTS	NWEIGHTS	NFILES	NBAD	NEOS	SIZE (GB)	OUTPUT PATH	MAIN PROCESS	FINAL STATES	CROSS SECTION (PB)
1	mgp8_pp_hh_lambda096_5f_htautau	980,000	0	98	1	99	45.63	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda096_5f_htautau/	HH, H->bb, H undec., kl = 0.96		1.47359
2	mgp8_pp_hh_lambda097_5f_htautau	1,000,000	0	100	0	100	46.55	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda097_5f_htautau/	HH, H->bb, H undec., kl = 0.97		1.46195
3	mgp8_pp_hh_lambda098_5f_htautau	990,000	0	99	0	99	46.13	/eos/experiment/fcc/hh/generation/DelphesEvents /fcc_v03/mgp8_pp_hh_lambda098_5f_htautau/	HH, H->bb, H undec., kl = 0.98		1.45039

4. Analysis pre-selection
   - Python framework produces flat ROOT trees

Use the events in the database to produce analyses templates

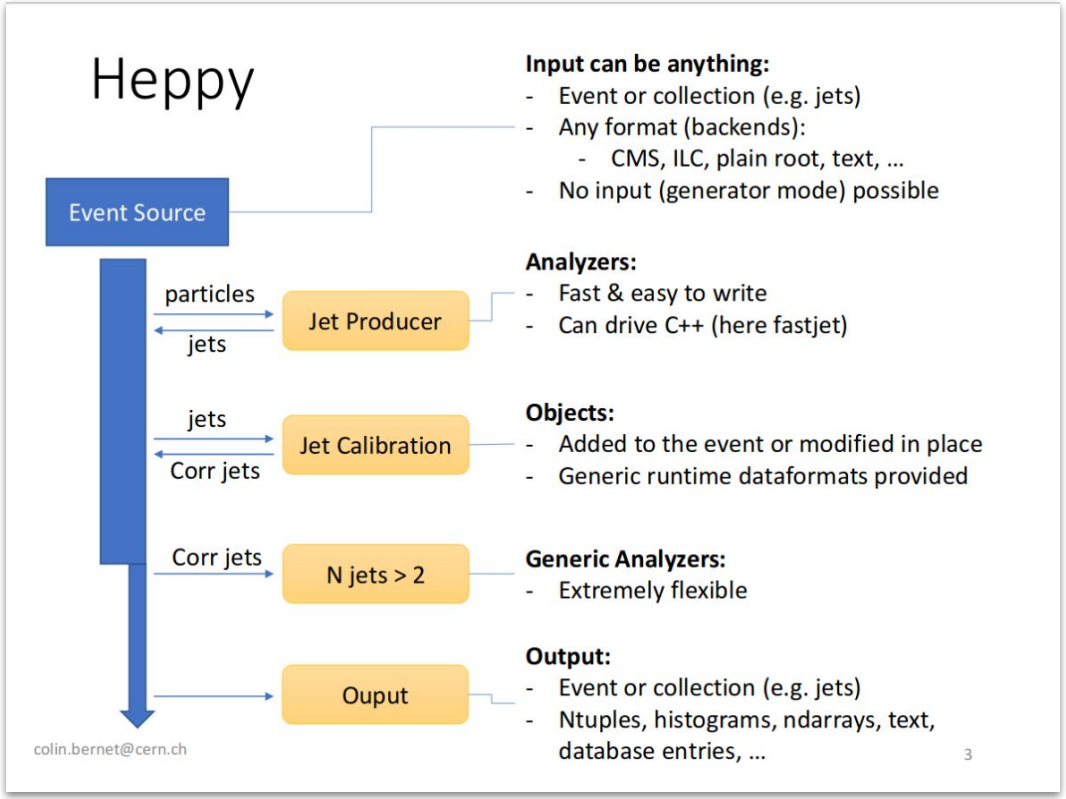
5. Analysis final selection
   - Python framework for optimising analysis cut flows and producing

Use the events produced at pre-selection to create stacked plots

6. Final analyses (custom, but we have limit setting tool for instance)

HEPPY

- See **Colin's Talk** giving a general overview



Problems: Performance and Maintainance

- Original Authors no longer around
- Pure Python Loop ➔ Slow! Using tth_4l analysis as a test case:

```
~/$ heppy_loop.py output analysis.py
    event 100 (17.8 ev/s)
    event 200 (21.9 ev/s)
    event 300 (24.6 ev/s)
    event 400 (26.1 ev/s)
    ….
```

Problems: Performance and Maintainance

- Clear that event loop needs to be moved to C++.
  - Coincided with first RDataFrame implementations
  - Spent last summer rewriting one analysis
- Speedup of 30x (6 seconds vs 3 minutes for one file on eos containing 7000 events)
- Not yet running with MultiThreading

HEPPY

```
def process(self, event):
  sqrts = self.cfg_ana.sqrts
  to_remove = getattr(event, self.cfg_ana.to_remove)
  recoil_p4 = TLorentzVector(0, 0, 0, sqrts)
  for ptc in to_remove:
     recoil_p4 -= ptc.p4()
  recoil = Recoil(0, 0, recoil_p4, 1)
  setattr(event, self.cfg_ana.output, recoil)
```

RDataframe:

```
struct recoil {
  recoil(float arg_sqrts) : m_sqrts(arg_sqrts) {};
  float m_sqrts = 240.0;
  std::vector<fcc::ParticleData> operator()
(std::vector<fcc::ParticleData> in) {
    std::vector<fcc::ParticleData> result;
    auto recoil_p4 = TLorentzVector(0, 0, 0, m_sqrts);
    for (auto & v1: in) {
      TLorentzVector tv1;
      tv1.SetXYZM(;
      recoil_p4 -= tv1;
    }
      auto recoil_fcc = fcc::ParticleData();
      recoil_fcc.core.p4.px = recoil_p4.Px();
      recoil_fcc.core.p4.py = recoil_p4.Py();
      recoil_fcc.core.p4.pz = recoil_p4.Pz();
      recoil_fcc.core.p4.mass = recoil_p4.M();
      result.push_back(recoil_fcc);
      return result;
  }
};
```

Changing the Heppy “backend”

- Heppy seems to have found a nice “syntax” for the analyses though
  - That and the sheer number of analyses to port cited as arguments against dataframe
- Started to investigate if configuration files can be kept
- fccflatprod

```
sys.modules["heppy"] = __import__("fccflatprod")
```

  - Each analyzer then maps to one Define

Changing the Heppy “backend”

- Heppy seems to have found a nice “syntax” for the analyses though
  - That and the sheer number of analyses to port cited as arguments against dataframe
- Started to investigate if configuration files can be kept
- fccflatprod

```
sys.modules["heppy"] = __import__("fccflatprod")
```

Main issues:

- Python lambdas in filters
- heppy idiosyncracies in I/O
- Analyzers that modify data in place

EDM4HEP

- Effort to converge to common datamodel for Future Collider Communities (CLIC/ILC, FCC and CEPC, and wider community where useful!)
- HSF Project
  - https://github.com/HSF/EDM4HEP
- Content close to more mature LCIO edm
  - Already ported to podio (“plcio”) by Frank Gaede
- Rapidly approaching first prototype
- FCC will certainly migrate as soon as possible

Implementation Podio

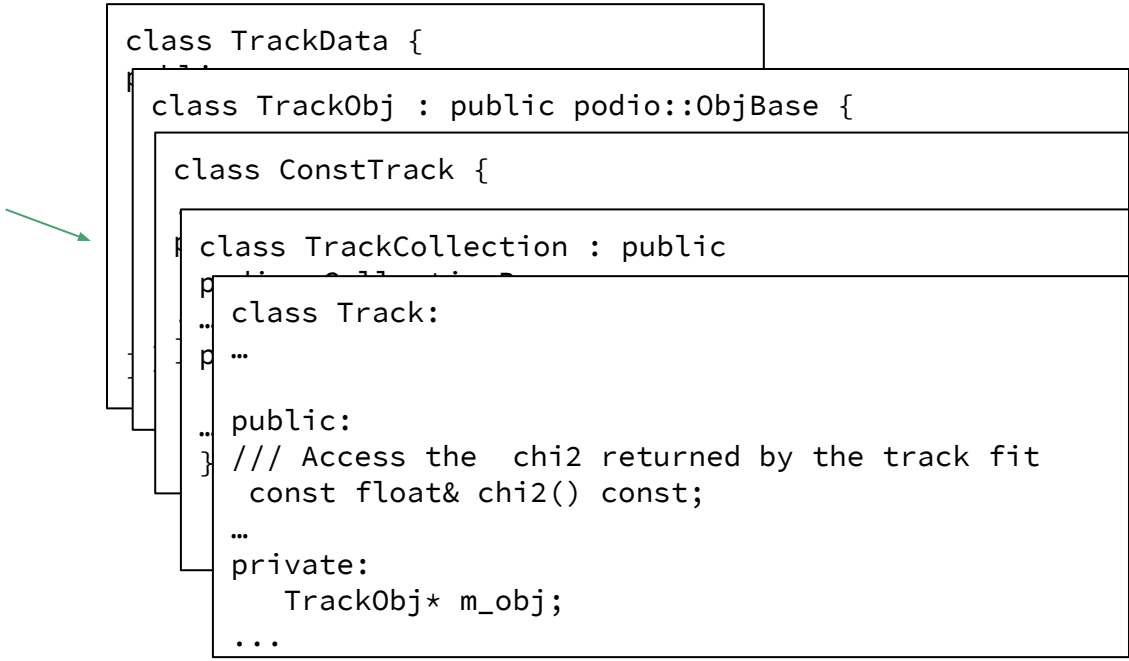
```
fcc::Track:
  Members:
   - float chi2 // from track fit
   - unsigned ndf //
   - unsigned bits // stores flags
  OneToManyRelations :
   - fcc::PositionedTrackHit hits
   - fcc::TrackState states
```



```
class TrackData {
public:
  float chi2;  ///< chi2
  unsigned ndf;  ///
  unsigned bits;  ///< Stores flags
  unsigned int hits_begin;
  unsigned int hits_end;
  unsigned int states_begin;
  unsigned int states_end;
};
```

Implementation: PODIO

```
fcc::Track:
  Members:
    - float chi2 // from track fit
    - unsigned ndf //
    - unsigned bits // stores flags
  OneToManyRelations :
    - fcc::PositionedTrackHit hits
    - fcc::TrackState states
```



Plans for the Future

- Working with the **Plain Old Data** is sufficient for basically all of the current FCC usecases for heppy
- Constructing a DataFrame from the plain root file is a feature
- Could distribute a library + dictionary of lambdas replacing analyzers
- Investigating **Bamboo**