FCCAnalyses status and plans

Juraj Smieško (CERN)

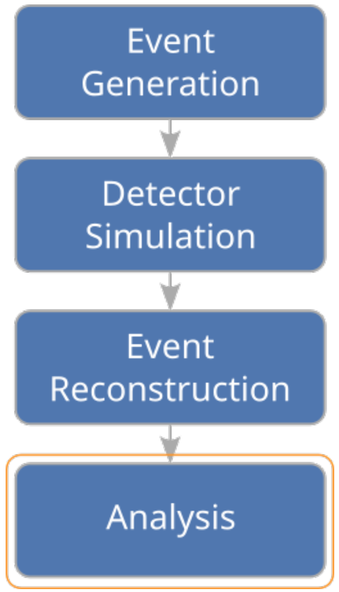
8th FCC Physics Workshop

CERN

13–16 January 2025

FCCAnalyses Overview

Analysis framework inside Key4hep ecosystem build
on top of the ROOT RDataFrame

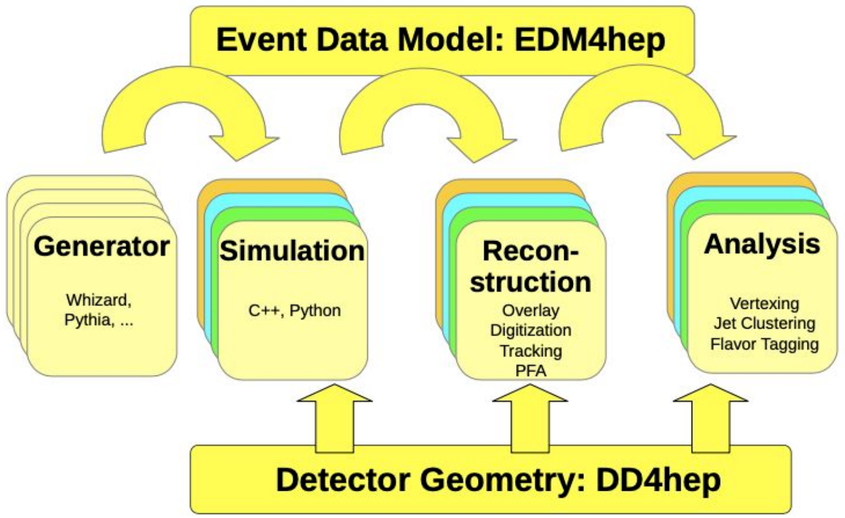


- Manages input samples
  - Remote pickup of required information
- Has standard library of functions
  - Many C++ HEP frameworks available
- Runs the dataframe
  - Local or remote execution
- Helps with histograms/plots
  - Export of results to other tools
- Registry for the analyses
  - Dedicated place for all case-studies

Key4hep

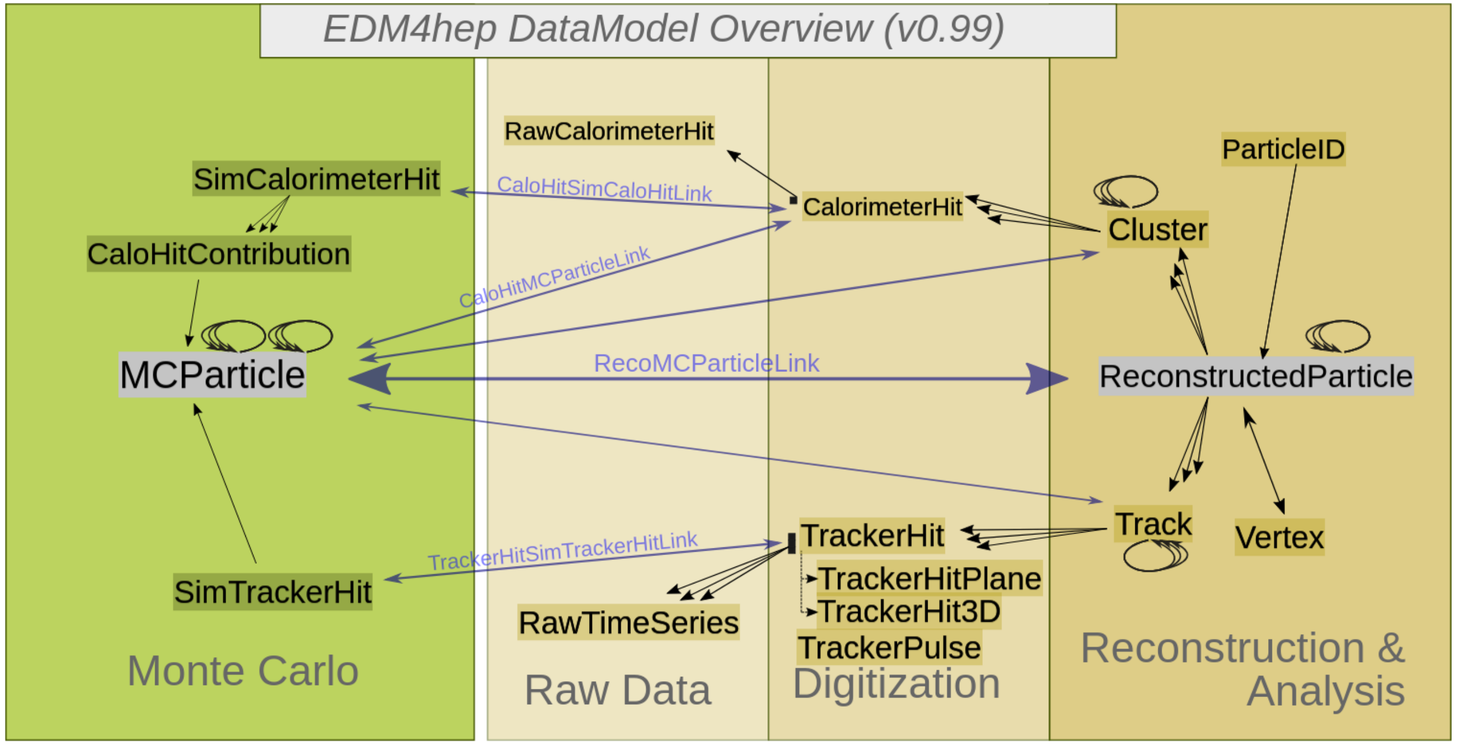
Coherent set of packages, tools, and standards for different collider Concepts

- Common effort from FCC, CLIC/ILC, EIC, CEPC, Muon Collider, ...
  - Preserves and adds onto existing functionality from iLCSoft, FCCSW, CEPCSW, ...
  - Builds on top of the experience from LHC experiments and results of targeted R&D (AIDA, ...)
  - Many institutes involved: CERN, DESY, IHEP, INFN, IJCLab, ...
- Each project rebases its stack on top of Key4hep
- Having common building blocks enables synergies across collider communities
- Main ingredients:
  - Event data model: EDM4hep, based on PODIO, AIDA project
  - Event processing framework: Gaudi, used in LHCb, ATLAS, ...
  - Detector description: DD4hep, AIDA project
  - System to build, test and deploy: Spack, suggested by HSF + CVMFS



EDM4hep

Common language for processing and persistifying data



- Specification in a single YAML file
  - Describes standard data structures and relations between them
- Generated by PODIO (developed as part of AIDA R&D)
- Challenge: efficiency and thread safeness

- Created by consensus
- Trade-off between being generic and preserve compactness
- Moving towards first stable LTS version (v1.0)

Versions of FCCAnalyses

Or how to get and run FCCAnalyses

- Latest version of FCCAnalyses can be obtained from:
  - GitHub:
    `git clone git@github.com:HEP-FCC/FCCAnalyses.git`
  - Key4hep nightlies stack:
    `source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh`
- Latest released version of FCCAnalyses is `v0.10.0` and can be obtained from:
  - GitHub:
    `git clone --branch v0.10.0 git@github.com:HEP-FCC/FCCAnalyses.git`
  - Key4hep release stack:
    `source /cvmfs/sw.hsf.org/key4hep/setup.sh`
- Specialized version for `winter2023` samples can be obtained only from:
  - GitHub:
    `git clone --branch pre-edm4hep1 git@github.com:HEP-FCC/FCCAnalyses.git`
- Recommendation:
  FCCAnalyses can be run without compiling — `fccanalysis` command is part of the Key4hep stack

Centrally produced samples

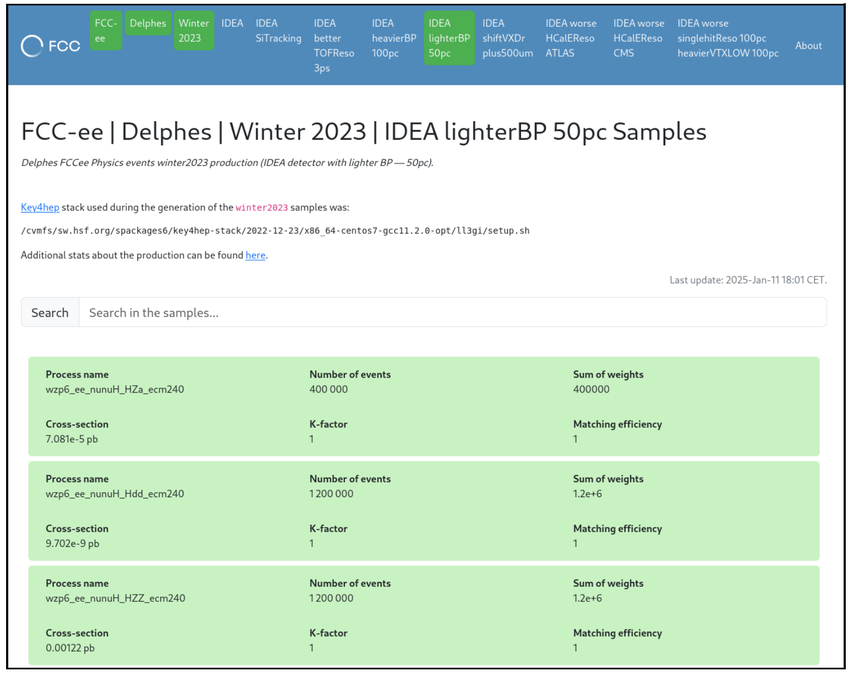
Management of centrally produced samples

- Samples include generator level files, parametrized simulation and Fullsim samples for FCC-ee and FCC-hh
- Samples stored on EOS at:

  `/eos/experiment/fcc/<accelerator-type>/generation/`

  `/eos/experiment/fcc/prod/fcc/<accelerator-type>/`
  - Ongoing transition from EventProducer to iLCDirac (FCC configurations)
- Detailed information about the samples published on FCC Physics Events website
- FCCAnalyses framework automatically picks up the sample information from YAML and JSON files

  The interface under overhaul, to allow the information to be consumed also by other analysis solutions
- Old samples moved to tape (FCC-hh v02, v03, v04)
  - Sample list will be kept on the website



FCC Physics Events

EOS Analysis Space

Various intermediate files of common interest can be stored centrally

FCC-ee space is located at:
`/eos/experiment/fcc/ee/analyses_storage/...`

in four sub-folders:

- `BSM`
- `EW_and_QCD`
- `flavor`
- `Higgs_and_TOP`

Access and quotas:

- Read access is granted to anyone
- Write access needs to be granted: Ask your convener :)
- Total available quota for all four sub-directories is 140TB
  - Currently 37T used
  - Quota is allocated based on actual needs

Standard library of analyzers

Set of self-contained functions/functors operating on ROOT dataframe

- Many functions/functors to run on dataframe columns provided from outside of FCCAnalyses (usually low level)
  - ROOT RVec, EDM4hep, ral
- FCCAnalyses provides more specialized ones in its standard library

  A lot missing due to many input/output objects and their combinations
- Analyzers can depend on following C++ frameworks
  - ROOT — together with RDataFrame
  - ACTS — track reconstruction tools (not fully supported)
  - ONNX — neural network exchange format
  - FastJet — jet finding package
  - DD4hep — detector description
  - Delphes — fast simulations
- Fork model of FCCAnalyses creates many copies of analyzers, which are shared among different groups
  - Call: Upstream your functions/functors

- The operations on the dataframe happen with small stateless functions:

```
float getMass(const ROOT::VecOps::RVec<edm4hep::Reconst
  ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<doubl

  for (auto & p: in) {
    ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<dou
    tmp.SetPxPyPzE(p.momentum.x, p.momentum.y, p.moment
    result+=tmp;
  }

  return result.M();
}
```

- or with structs, which have internal state:

```
/// Get the number of particles in a given hemisphere (
/// wrt to axis). Returns 3 values: total, charged, neu
struct getAxisN {
public:
  getAxisN(bool arg_pos=0);
  ROOT::VecOps::RVec<int> operator() (const ROOT::VecOp
                                      const ROOT::VecOp
private:
  bool _pos; /// Which hemisphere to select, false/0=cc
};
```

Towards better defined analysis script

Analysis encapsulated into a class

```python
class Analysis():
    def __init__(self, cmdline_args):
        self.process_list = {
            'p8_ee_WW_ecm240': {'fraction': 0.5, 'chunks': 7}
        }
        self.n_threads = 4
        ...

    def analyzers(self, dframe):
        dframe2 = (
            .Define('selected_muons',
                    f'ReconstructedParticle::sel_pt({muon_pt})(muons)')
            .Define('selected_muons_pt',
                    'ReconstructedParticle::get_pt(selected_muons)')
        ...
        return dframe2

    def output(self):
        branch_list = [
            'selected_muons_pt',
            'selected_muons_y',
        ...
        return branch_list
```

- In both (staged or histmaker) styles various attributes are used to adjust behavior of the running of the analysis script
- It is not clear which are needed and when
- The attributes are being documented in `fccanalysis-script`, `fccanalysis-final-script` and `fccanalysis-plots-script` manual pages
- In order to better define the script interface all attributes are being moved into the analysis class
  - Not yet done for `final` and `plots` stages
- For now old style of analysis is still supported

Analysis CLI arguments

Analysis script can use parameters provided from the command-line.

- Command line arguments are fed into the `Analysis` class
- They need to be parsed by the script itself
- All arguments provided after `--` (double dash) are considered to belong to the script

```python
def __init__(self, all_cmdline_args):
    parser = ArgumentParser()
    parser.add_argument('--muon-pt', default='10.', type=float,
                        help='Minimal pT of the mouns.')
    self.args, _ = parser.parse_known_args(cmdline_args['remainder
...
    # Select muons by pT
    .Define('selected_muons',
            f'ReconstructedParticle::sel_pt({self.args.muon_pt})(m
```

The anatomy of the `fccanalysis` command line interface:

```
fccanalysis <global-args> <sub-command> <sub-command-args> <analysis-script> -- <script-args>
```

Example:

```
fccanalysis -vv run --n-threads 4 my_fcc_analysis.py -- --pt-min 40
```

`--` (double dash) will be introduced in PR#422

Submit sub-command

Extracting submission machinery from analysis execution

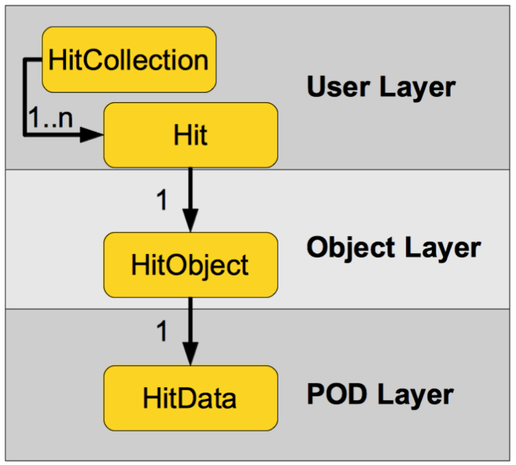
- Done in order to allow for other forms of distributed computing other than HTCondor
- Improves also current HTCondor submission machinery
- Will allow running of Histmaker style analyses on HTCondor
- Usage: `fccanalysis submit ana_script.py`
- Almost ready to be merged: PR#422

PODIO Datasource

Preserving EDM4hep relationships in RDataFrame

- Collection objects in PODIO/EDM4hep can be accessed on several layers
- Highest layer provides one-to-many and many-to-many relationships
- Easy access to the related objects greatly improves writing and understanding of the analyzer
- Price for this convenience is performance

  Alternative: NTuple creation in Gaudi algorithm



Enabling PODIO Datasource in the analysis:

- Use analyzers which take as input EDM4hep collections:

  `edm4hep::ReconstructedParticleCollection` ,

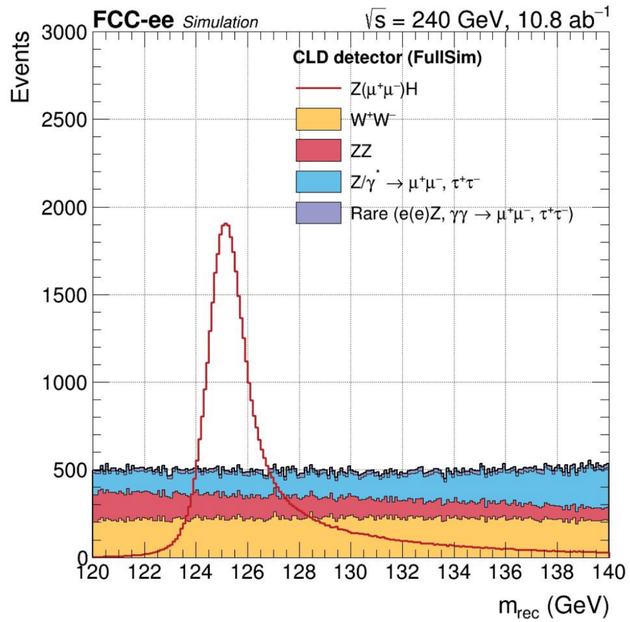
  `edm4hep::RecoMCParticleLinkCollection` , ...
- Instruct FCCAnalyses to use `podio::DataSource` :

  `self.use_data_source = True` or `fccanalysis run --use-data-source ana_script.py`

Recent improvements

Improvements from users are highly welcome!

- Variable event weights implemented in the context of FCC-hh
- TMVAHelper: support for BDT/MVA using XGBoost
- Export of Combine datacards from the Final stage/Histmaker outputs
- Harmonization of (meta)data exchange between individual stages
  - Includes exports of cut-flows in TeX and JSON format
- Plotting improvements
- Logging / print-out support for C++ analyzers
- Running in SWAN

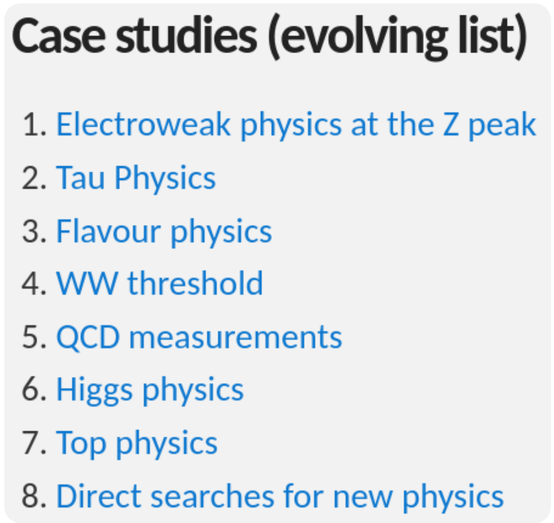


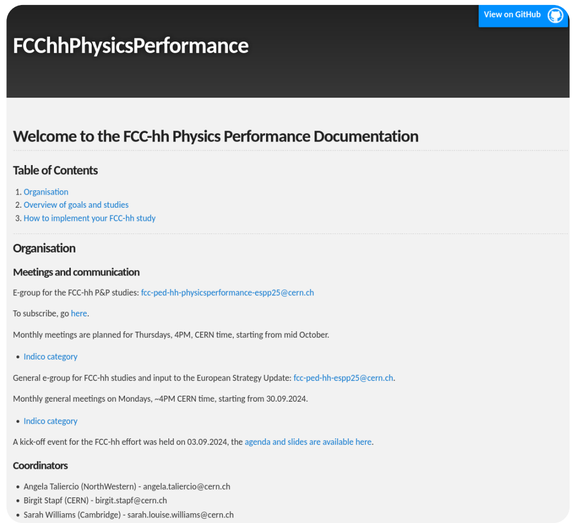
source: Jan Eysermans

Analysis registry

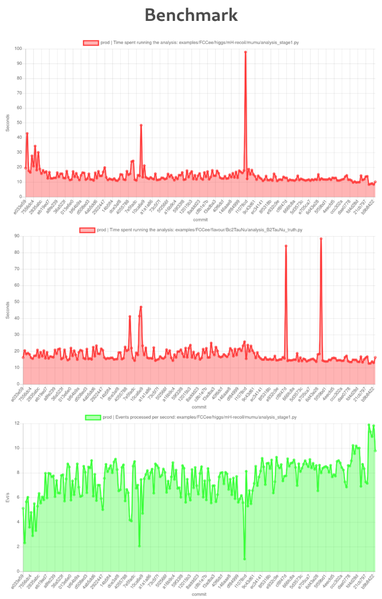
Central registry for the FCC case studies

- Two repositories:
  - FCCeePhysicsPerformance lists FCC-ee case studies
  - FCChhPhysicsPerformance contains FCC-hh physics performance documentation
- Experimentally one can create analysis package for analysis specific code
- Rudimentary and in need of overhaul
  - Based on the post FSR needs





Tests and benchmarks



Ensuring correctness and performance

- Tests of the FCCAnalyses framework are done daily
  - Test suite is published in FCCTests
  - Goal is to ensure FCCAnalyses be available in the Key4hep nightlies stack
  - Every test run inside independent subprocesses
  - Tests need access to LXPlus like environment
    - Missing HTCondor testing
- Benchmarks of FCCAnalyses is done after every merge of a PR
  - Benchmarks run on small set of events
  - There is no guaranteed machine to run on
  - Benchmarks of Higgs mass recoil example on LXPlus:
    20k–45k evt/s

All the Links

Contacts

- FCC Analysis Mattermost channel
- FCCAnalyses section at FCC Software forum
- FCC-PED SW Analysis mailing list:
  `FCC-PED-SoftwareAndComputing-Analysis@cern.ch`

Documentation

- FCCAnalyses website, FCC Software website
- Code reference for the analyzers:
  https://hep-fcc.github.io/FCCAnalyses/doc/latest
  - Provides details about implementation of individual analyzers
- Manual pages:
  `man fccanalysis` , `man fccanalysis-script` , `man fccanalysis-<subcommand>` , ...
  - Info about individual (sub)commands directly in the terminal
- FCC Tutorials: https://hep-fcc.github.io/fcc-tutorials/
  - Focused on providing a tutorial on a specific topic

Broader Plans

- Focus on distributed computing
  - Allow running on GRID, Slurm or other platforms
- Make FCCAnalyses able to run on non LXPlus machines
- Improvement to sample management
  - Make it consumable by other analysis solutions
- Better ML support

  Depending not only on the ROOT capabilities
- Expand plotting facilities
  - Many products in the Python HEP space
  - Integrate existing tools or make FCCAnalyses more interoperable
- Improve performance of EDM4hep relationship handling
- Streamline NTuple production mechanism

Conclusions

- Core functionalities are becoming more stable, many features are missing
- The framework used by the majority of FSR case studies
- Fullsim analyses are possible, more analyzer adjustments needed
- Push to make FCCAnalyses compilation free continues
- Looking forward for EDM4hep 1.0 — new campaign of centrally produced samples
- Semi regular meetings happen on Wednesdays 04:00 PM
  - Informal meeting to debug and discuss FCCAnalyses issues