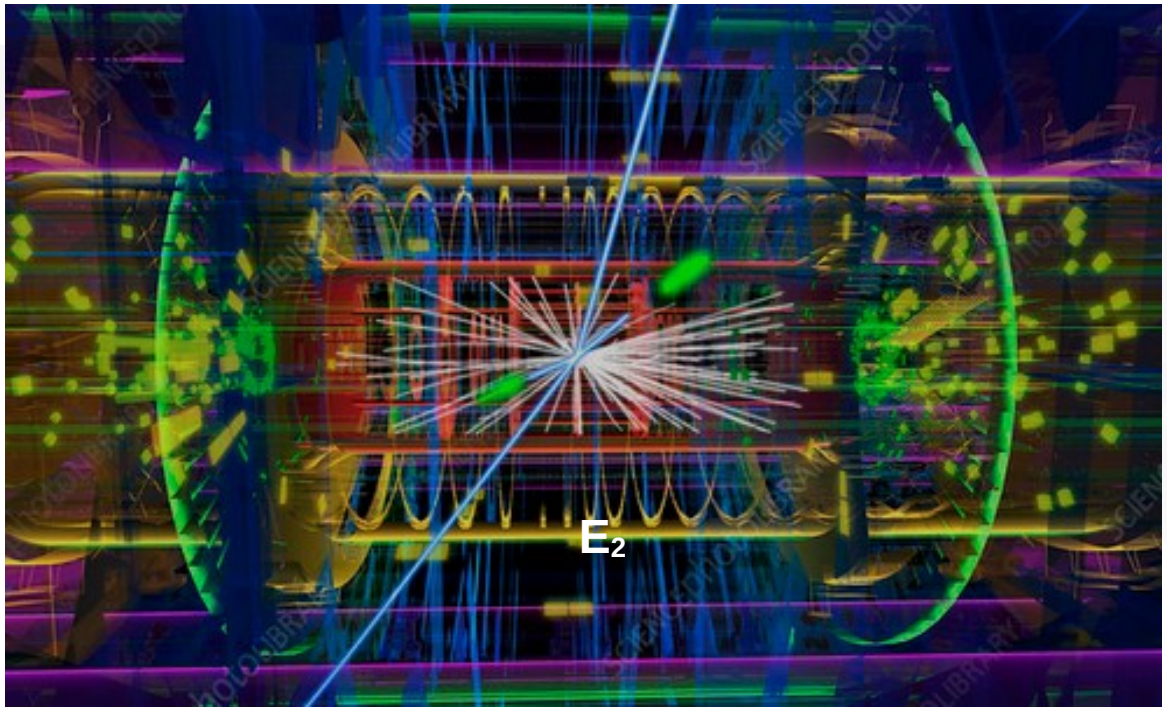
FCCPana – lightweight C++ analysis program for fast and full simulations for FCCee

S.Chekanov (ANL)

May 6, 2024, CERN

Software general meeting



Why 100% CPP code?

- FCCAnalyses heavy relies on Python users (I'm not .. even after publishing Python books!)
- Cannot complain much about Python, but we know its limitation
- Being looking for C++ code with an integration of external libraries from L3/HERA/ATLAS for ML/constrained fits
- Decided to put together a simple C++ code (<100k total size)
- Used some code from ATLAS analysis (used in ~5 papers) and made some simplifications
- First version of this code can be find:
  - https://github.com/chekanov/FCCPana
  - Tested on 24 cores. Works well

Some useful features

- Uses **key4hep** setup
- Runs on any EDM (ROOT) files
- Small (100KB), simple and fast! (Note: FCCAnalysis is 80 MB)
- Can run on multiple cores (tested on 24 cores)
- Can define systematics + multiple outputs via main.in input file
- Can attach additional C++ libraries (simply put them to inc/ or src/)
- Unified object definition
- Works for both Delphes IDEA and CLD files without code change
- Flexible. Program any analysis logic (even 4-level nested loops with arbitrary selections!) using variables listed in analysis.h

Small example

```
git clone https://github.com/chekanov/FCCPana
cd FCCPana
source setup.sh  # setup key4hep
make             # compile
mkdir -p data/IDEA_DELPHES
# Add 2 EDM files for testing (DElphes)
cd data/IDEA_DELPHES
wget https://mc.hep.anl.gov/asc/hepsim/events/ee/240gev/py8_ZH_idea/rfast053/py8_ZH_idea_1.root
wget https://mc.hep.anl.gov/asc/hepsim/events/ee/240gev/py8_ZH_idea/rfast053/py8_ZH_idea_2.root
cd ../../
./A_RUN    # run on 2 cores (or modify how many cores you need)
```

- Similarly, run full simulations (CLD).
- Copy files, define in A_RUN the directory with the EDM files, and run!
- Find merged outputs inside “out”

Program structure

- All EDM-specific header files are in “aux” and predefined for fast and full simulation
- If data changes, there is a tool to recreate such header files
- program “knows” how to handle different data record using pre-processor statement

Program structure:

- Define a histogram name in **inc/Histo.h**
- Any global parameters are in **inc/Global.h**
- Initialize the histogram in **src/Histo.cxx**
- Apply selection cuts for your events in **src/CutEvent.cxx**
- Event loop in **src/Loop.cxx** - main analysis program. Do anything you want

Not sure what variable to use in **src/Loop.cxx**? Look at analysis.h. It lists all variables from the input EDM file (it changes depending on fact or full simulation)

Inside src/Loop.cxx

- All variables from Delphes and full simulations are moved to unified representation of events
- Currently the program fills 8 invariant masses from fast and full simulations (for truth and reco)

```
      // truth
      for(Int_t  i = 0; i < MCParticles_; i++){
       if ( MCParticles_generatorStatus[i] != 1) continue;
        TLorentzVector tl;
        float e=sqrt( MCParticles_momentum_x[i]* MCParticles_momentum_x[i]
+MCParticles_momentum_y[i]* MCParticles_momentum_y[i]
+MCParticles_momentum_z[i]*MCParticles_momentum_z[i]
+MCParticles_mass[i]*MCParticles_mass[i]);

tl.SetPxPyPzE(MCParticles_momentum_x[i],MCParticles_momentum_y[i],MCParticles_momentum_z
[i],e);
        LParticle p;
        p.SetP(tl);
        int pdg=MCParticles_PDG[i];
        p.SetType( pdg  );
        p.SetStatus(  MCParticles_simulatorStatus[i] );
        p.SetParent( 0 );
        p.SetCharge( MCParticles_charge[i]  );
        if (abs(pdg)==11) true_electrons.push_back(p);
        if (abs(pdg)==13) true_muons.push_back(p);
        if (abs(pdg)==22) true_photons.push_back(p);
    };
```

Available vectors with objects:

jets, Ljets, Bjets, electron, muons, photons and truth level counterparts

Each element: extended TLorenzVector (+ any parameter you want!)

Where it can fail

- The code does not “understand” previous iteration of EDM files (<2023) since they have “#” in the names (to indicate leading and subleading objects)
- ROOT (and C++) does not allow this.
- TTree::MakeClass creates headers which cannot be compiled
- The issue was brought to the ROOT team. Philippe Canal fixed this problem, replacing “#” with “_”. But this feature has not been tested (need dev branch of ROOT)
- Meanwhile, use `/cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh` to create EDMs without “#”

Conclusion

- https://github.com/chekanov/FCCPana
- Try to use it. If it fails, let me know
- If there are significant changes in the event record of EDM, I will show you how to re-generate header file to reflect such change