# Simulation framework in the CEPCSW prototype

Tao Lin

(on behalf of CEPCSW group)

IHEP

IAS HEP 2020, Hong Kong

16-17 Jan 2020

# Outline

**During R&D stage, CEPC seeks a lightweight & unified data processing system. A simulation framework is developed based on Gaudi and DD4hep.**

- Motivation
- The status of simulation framework prototype
  - Event data model, geometry management, physics generator interface, detector simulation, software performance
- The development plan of the simulation framework
  - Coherent full/fast simulation, parallelized simulation
- Summary

# Motivation

# Motivation

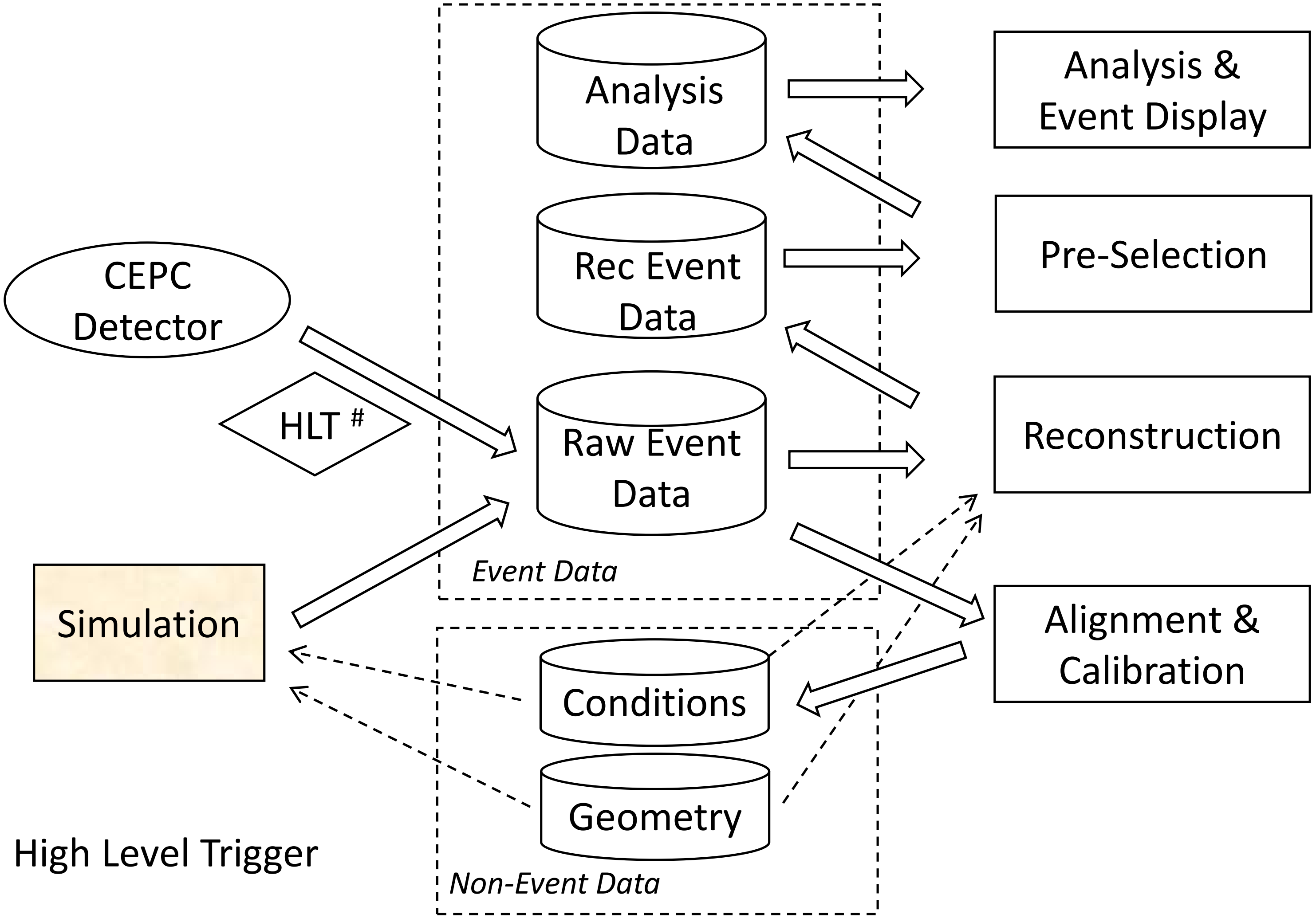
CEPC data volume and computing challenges

| | | Estimated Data Volume |
|---|---|---|
| **Short term** | R&D: detector design | ~PB/year |
| **Long term** | Higgs/W factory | 1.5~3 PB/year |
| | Z factory | 0.5~5 EB/year |

Requirements on simulation:

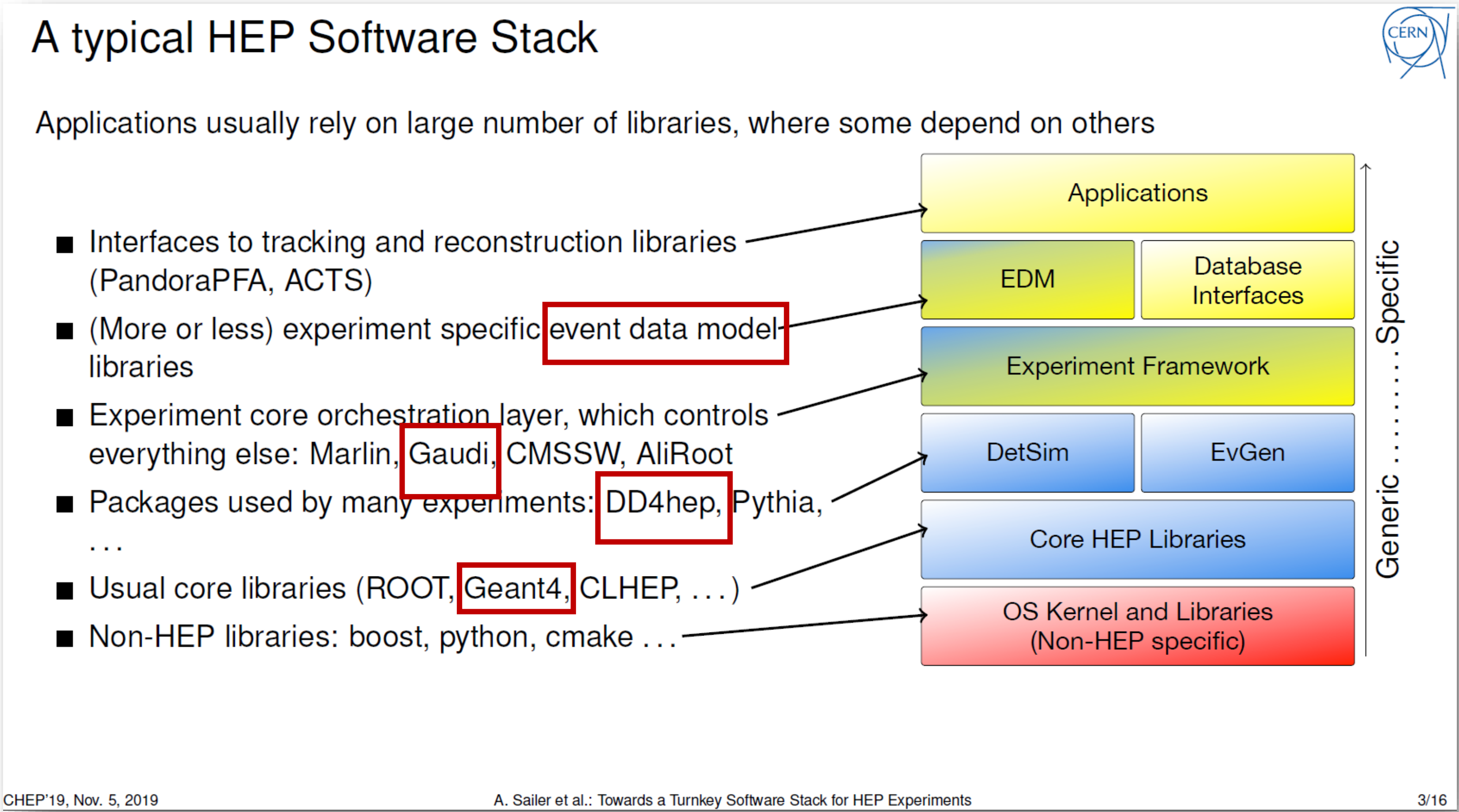
- R&D stage: fast development and iteration.
  - Detector performance studies using full and fast simulation.
  - geometry management & geometry update
- Operation: extensible, efficient and sustainable.
  - Modern software engineering, parallelism

# Data processing chain



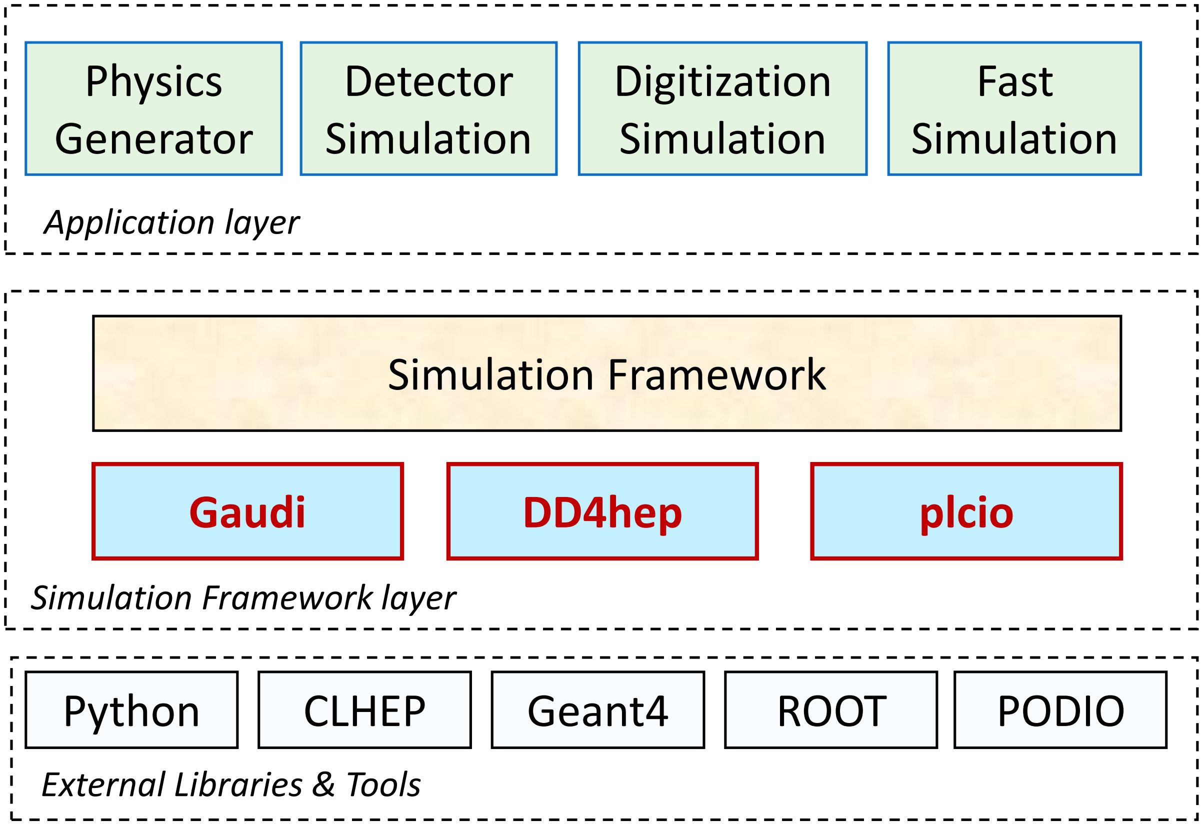

# HLT: High Level Trigger

# KEY4HEP: Common software stack




[Andre Sailer, CHEP2019, https://indico.cern.ch/event/773049/contributions/3474763/](https://indico.cern.ch/event/773049/contributions/3474763/)

# Simulation Software Stack

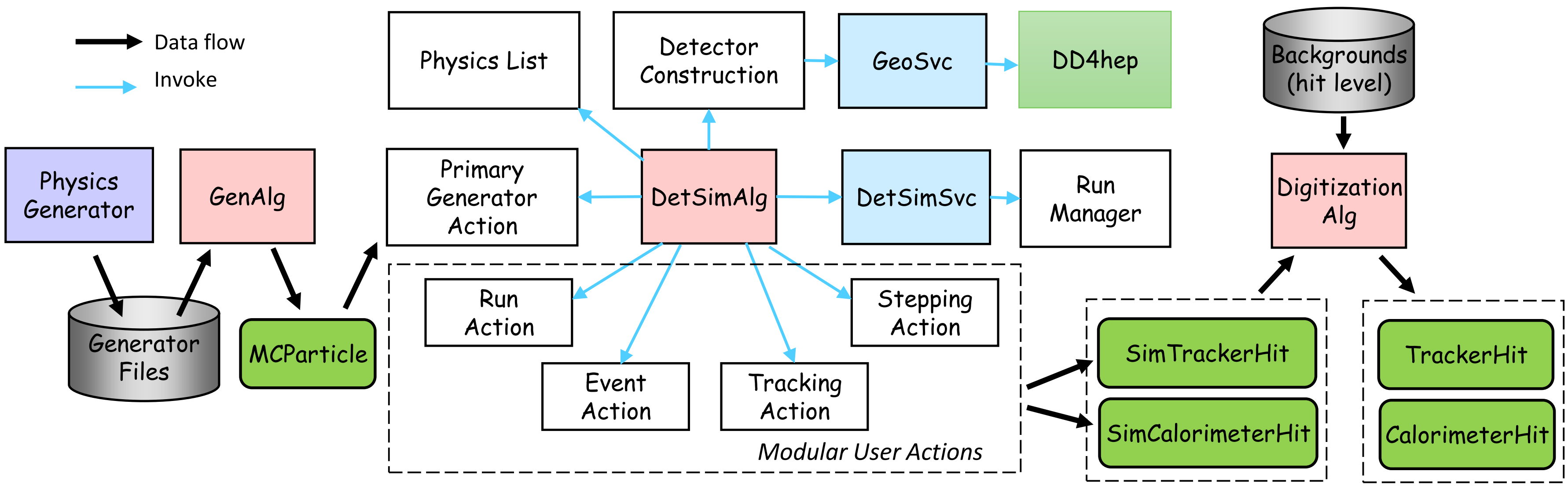



Simulation framework provides flexible integration of the applications (physics generator, detector simulation, digitization).

# Status

# Design of simulation framework

- Based on Gaudi and DD4hep.
  - Reuse part of interfaces defined in FCCSW.
- In the current prototype, “Tracker” is setup.




Core Packages: Generator, Simulation/DetSimInterface, Simulation/DetSimCore
Other packages: Detector, Simulation/DetSimGeom, Simulation/DetSimAna

# Event data model

- Based on plcio.
  - MCParticle
  - SimTrackerHit
  - SimCalorimeterHit
  - TrackerHit
  - CalorimeterHit

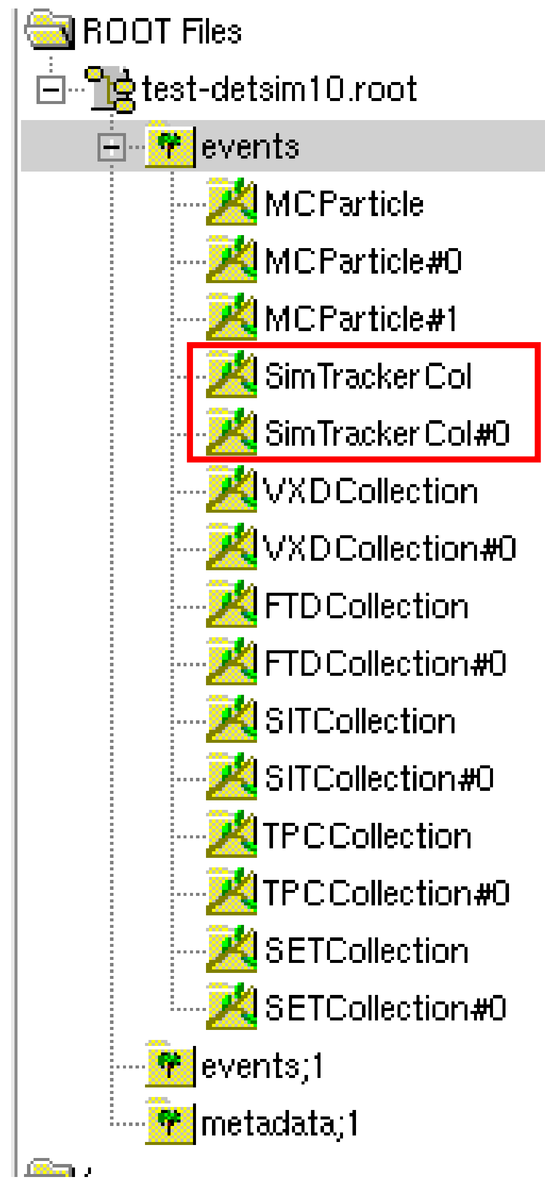
Hit

Digi

- Following collections are available in output:
  - MCParticle
  - VXDCollection
  - SITCollection
  - TPCCollection
  - SETCollection

SimTrackerHit

- Keep compatible with Mokka.




For other unknown.

# Unified Geometry

- **Configurable geometry** with DD4hep's XML files: support multiple options of detectors and beam test geometry.
- GeoSvc provides the interface to access the DD4hep.
  - `dd4hep::Detector* lcdd();`
- Then DD4hep is used to construct the detector in simulation.
  - Based on Chengdong's version (CEPC workshop, Oxford, 2019).
  - Package: Detector/DetCEPCv4
  - Geometry with VXD only: CepC_v4-onlyVXD.xml (Default)
  - Geometry with Tracker: CepC_v4-onlyTracker.xml
  - VXD: compact/vxd07.xml + src/tracker/VXD04_geo.cpp
- In detector simulation, a DetElemTool is used to convert the DD4hep geometry to Geant4 geometry.
  - Package: Simulation/DetSimGeom/
  - Allow the traditional Geant4 geometry construction.

# Identifier

- ID for each detector component.
- Keep compatible between Mokka and DD4hep.

```xml
<readouts>
  <readout name="VXDCollection">
    <id>system:5,side:-2,layer:9,module:8,sensor:8</id>
  </readout>
</readouts>
```

- CellID0 and CellID1: 32bit for each
- The bitmap starts from low to high

1110 000000101 11 00001 high <- low
layer(9) system(5)
Module(8) side(-2: sign value)

```cpp
const unsigned ILDCellID0::subdet = 0 ;
const unsigned ILDCellID0::side   = 1 ;
const unsigned ILDCellID0::layer  = 2 ;
const unsigned ILDCellID0::module = 3 ;
const unsigned ILDCellID0::sensor = 4 ;

const int ILDDetID::NOTUSED    =   0 ;
const int ILDDetID::VXD        =   1 ;
const int ILDDetID::SIT        =   2 ;
const int ILDDetID::FTD        =   3 ;
const int ILDDetID::TPC        =   4 ;
const int ILDDetID::SET        =   5 ;
const int ILDDetID::ETD        =   6 ;
const int ILDDetID::ECAL       =  20 ;
const int ILDDetID::ECAL_PLUG  =  21 ;
const int ILDDetID::HCAL       =  22 ;
const int ILDDetID::HCAL_RING  =  23 ;
const int ILDDetID::LCAL       =  24 ;
const int ILDDetID::BCAL       =  25 ;
const int ILDDetID::LHCAL      =  26 ;
const int ILDDetID::YOKE       =  27 ;
const int ILDDetID::COIL        = 28 ;
const int ILDDetID::ECAL_ENDCAP= 29 ;
const int ILDDetID::HCAL_ENDCAP= 30 ;
const int ILDDetID::YOKE_ENDCAP= 31 ;
```

LCIO/src/cpp/src/UTIL/ILDConf.cc

# Physics generator interface

- Implemented as a major algorithm with multiple small GenTools.
  - The major algorithm creates the event object and passes it to GenTools.
  - A GenTool then adds or modifies the MC particle in the event.
- Several GenTools are ready for use.
  - GtGunTool: a particle gun, supporting multiple particles.
  - StdHepRdr: reads StdHep format data.
  - SLCIORdr: reads LCIO format data.
- Configure in the job option file.
  - GtGunTool: Particle name/PDGcode, momentum, direction
  - Readers: Input

# Physics list

- Default physics list: "QGSP_BERT"
  - Same as Mokka (Control::PhysicsListName = "QGSP_BERT")
- Can be changed easily by setting
  - detsimalg.PhysicsList = "QGSP_BERT_HP";
- Using Geant4's G4PhysListFactory to instance the real physics list.

```
nlists_hadr = 23;
G4String ss[23] = {
  "FTFP_BERT","FTFP_BERT_TRV","FTFP_BERT_ATL","FTFP_BERT_HP","FTFQGSP_BERT",
  "FTFP_INCLXX","FTFP_INCLXX_HP","FTF_BIC", "LBE","QBBC",
  "QGSP_BERT","QGSP_BERT_HP","QGSP_BIC","QGSP_BIC_HP","QGSP_BIC_AllHP",
  "QGSP_FTFP_BERT","QGSP_INCLXX","QGSP_INCLXX_HP","QGS_BIC",
  "Shielding","ShieldingLEND","ShieldingM","NuBeam"};
```

```
nlists_em = 12;
G4String s1[12] = {"","_EMV","_EMX","_EMY","_EMZ","_LIV","_PEN",
                   "__GS","__SS","_EM0","_WVI","__LE"};
```

# Sensitive detector

- The DDG4's sensitive detectors are registered automatically.
  - Done in the IDetElemTool::ConstructSDandField.
  - See: Simulation/DetSimGeom/src/AnExampleDetElemTool.cpp
- The volumes marked as sensitive are associated with SD using the "type" name in the compact file.
  - Supporting two types: tracker and calorimeter.

Compact XML file:

```
<detector name="VXD" type="VXD04" vis="VXDVis" id="ILDDetID_VXD"
limits="Tracker_limits" readout="VXDCollection" insideTrackingVolume="true">
```

Logs:

```
Compact         INFO  ++ Converted subdetector:VXD of type VXD04 [tracker]
```

```
ToolSvc.AnExamp...   INFO Type/Name: tracker/VXD
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_00
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_01
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_02
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_03
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_04
ToolSvc.AnExamp...   INFO  -> Adding SiActiveLayer_05
```

Associate tracker SD and SiActiveLayer_XXX.

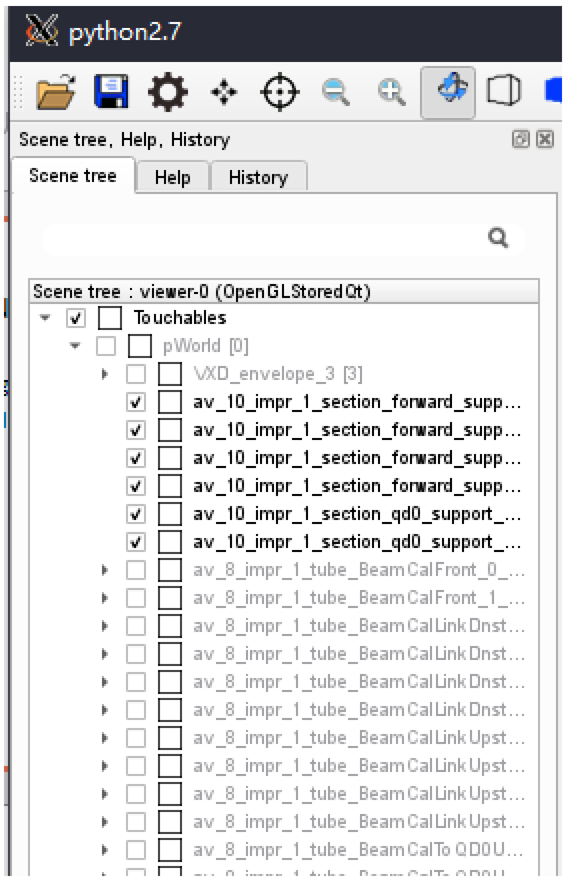
# Modular user actions

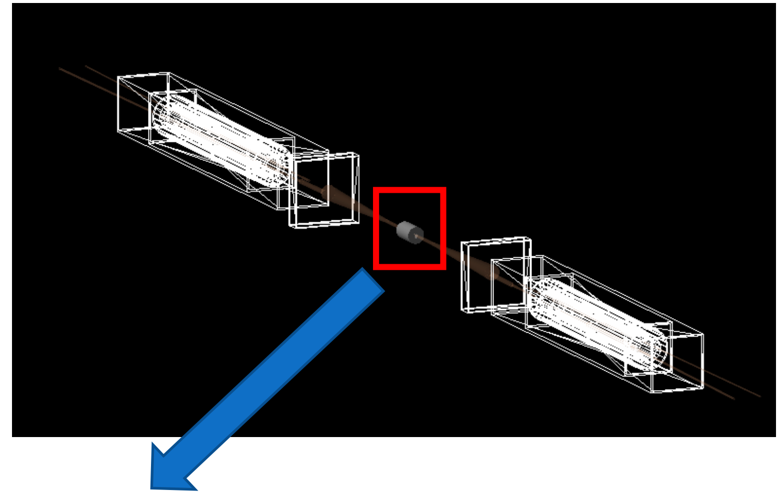
- Implemented as IAnaElemTool
  - Path: Simulation/DetSimAna
  - The registered tools will be invoked by the Geant4 user actions automatically (Run, Event, Tracking, Stepping Actions).
- ExampleAnaElemTool
  - An example to convert the Geant4's hit objects to plcio's hit objects.
  - All the necessary collections are defined here.

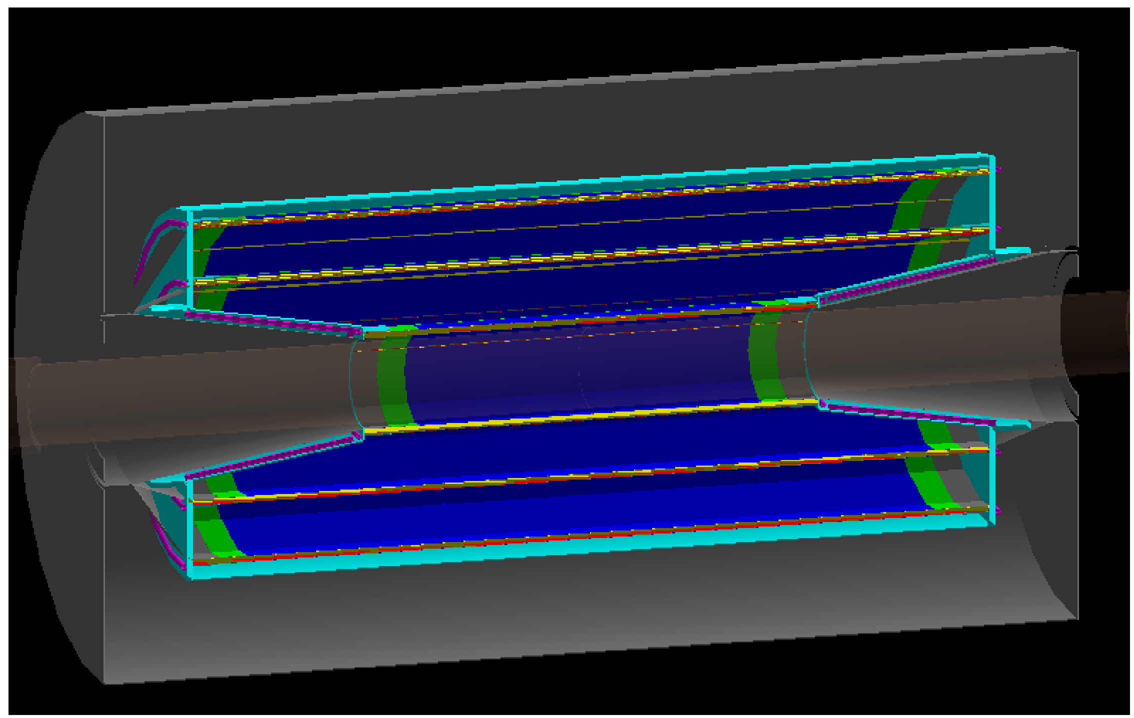
```
DataHandle<plcio::SimTrackerHitCollection> m_trackerCol{"SimTrackerCol",
        Gaudi::DataHandle::Writer, this};
DataHandle<plcio::SimTrackerHitCollection> m_VXDCol{"VXDCollection",
        Gaudi::DataHandle::Writer, this};
DataHandle<plcio::SimTrackerHitCollection> m_FTDCol{"FTDCollection",
        Gaudi::DataHandle::Writer, this};
DataHandle<plcio::SimTrackerHitCollection> m_SITCol{"SITCollection",
        Gaudi::DataHandle::Writer, this};
DataHandle<plcio::SimTrackerHitCollection> m_TPCCol{"TPCCollection",
        Gaudi::DataHandle::Writer, this};
DataHandle<plcio::SimTrackerHitCollection> m_SETCol{"SETCollection",
        Gaudi::DataHandle::Writer, this};
```

# Visualization

- Simulation framework is working with Geant4's Qt-based UI.





# Job option

- Job option is a python script, which defines the standard simulation chain.
  - Command services: Geometry Service, Random Number Service, Event Data service, I/O service
  - Algorithms: Generator interface algorithm, Detector simulation algorithm.
- Use gaudirun.py to run it. (It can be improved in the future.)

```
./run gaudirun.py '$EXAMPLESROOT/options/tut_detsim.py'
```

```
detsimalg = DetSimAlg("DetSimAlg")

# detsimalg.VisMacs = ["vis.mac"]

detsimalg.RunCmds = [
#    "/tracking/verbose 1",
]
detsimalg.AnaElems = [
    # example_anatool.name()
    "ExampleAnaElemTool"
]
detsimalg.RootDetElem = "WorldDetElemTool"

from Configurables import AnExampleDetElemTool
example_dettool = AnExampleDetElemTool("AnExampleDetElemTool")
```

<= If need visualization, uncomment it

<= Part of Geant4 macros are support

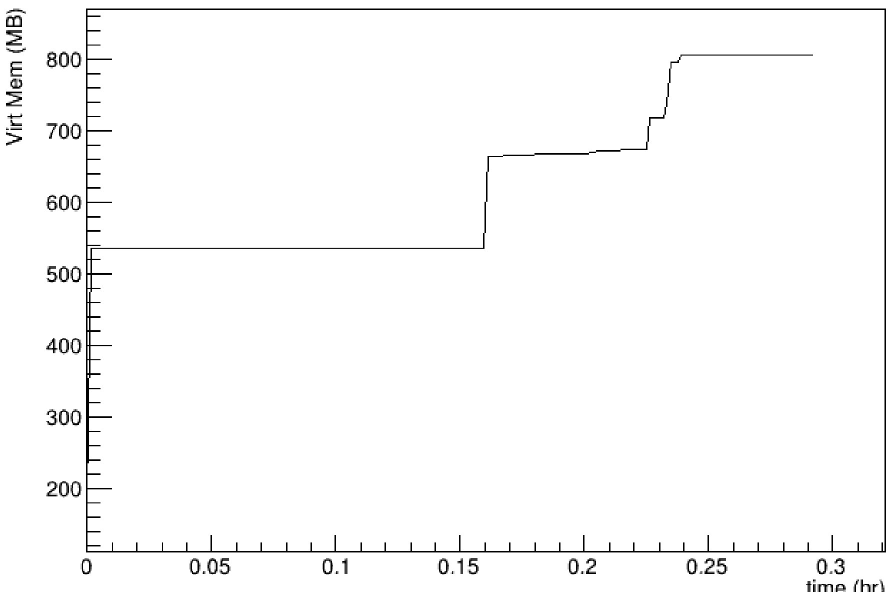
<= User actions

<= Geometry

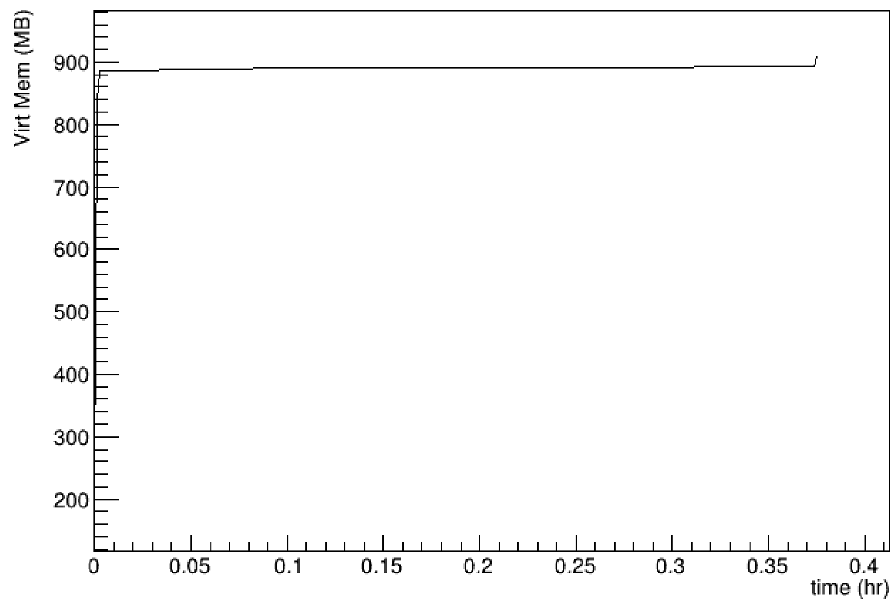
# Software performance

- Measure two cases
  - Physics generator only
  - Physics generator + detector simulation

CPU: Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz
Memory: 128 GB




Input file:
CEPC250/2fermions/E250.Pbhabha.e0.p0.whizard195/bhabha.e0.p0.00001.stdhep
Total events: 200,000



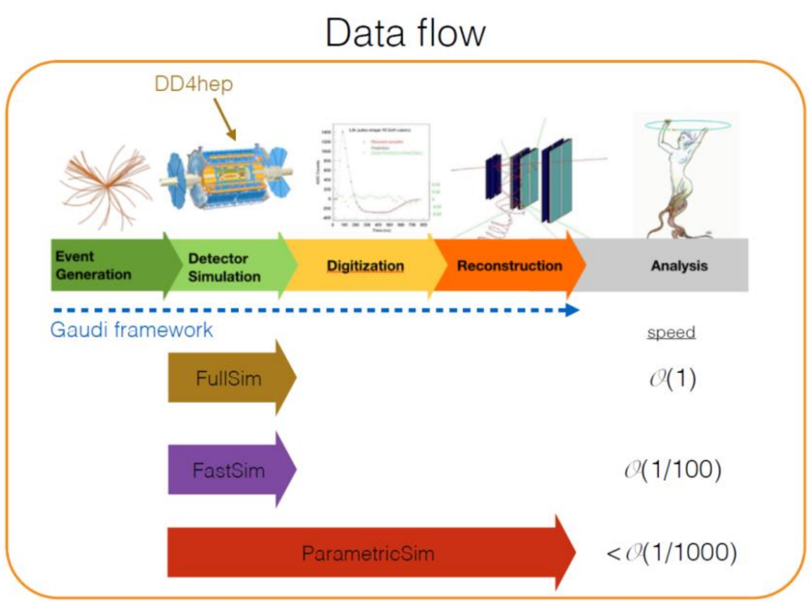

Geometry： VXD only
Total events: 1000

# Development Plans

# Coherent full/fast simulation

- Three types of simulation
  - Full Simulation (Geant4), O(1)
  - Fast Simulation, O(1/100)
  - Parametric simulation, O(1/1000)
- Plan to develop a coherent simulation framework
  - Allow mixing of full and fast simulation.
  - When a particle enters a certain detector region, user-defined simulation tool will be used.
- Fast simulation tool development plan
  - Track: hit level fast simulation
  - Calorimeter: frozen shower, GAN


ATLAS ISF simulation

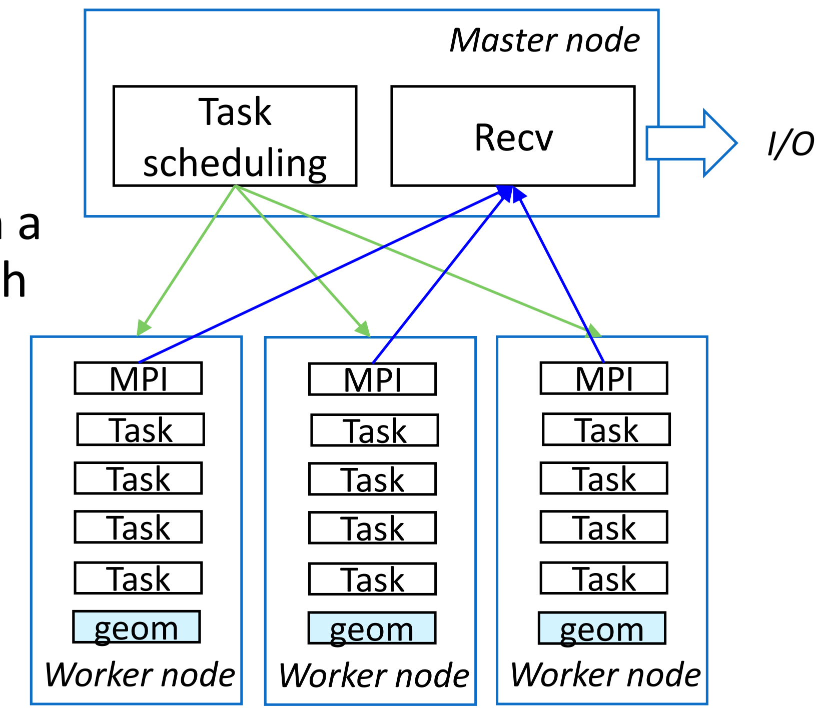



*Prototyping a coherent framework for full, fast and parameteric detector simulation for the FCC project*

J. Hrdinka *et al.* PoS EPS-HEP2015 (2015) 248

# Parallelized simulation

- New computing trends
  - **Multi- & Many-core**: MT (one-node)
  - **HPC**: MPI and MT (multi-node)
  - **Memory usage:** via shared information, such as geometry.
  - **I/O performance**: especially when a lot of threads write data, need high performance filesystem.
- Different levels parallelization
  - Event level
  - Track level
- Plan to develop MPI- and MT-based simulation
  - MPI is used for communication between nodes.
  - MT is used inside one node.



# Summary and plans

- A simulation framework prototype is developed.
  - Configurable Geometry with DD4hep's XML files: support multiple options of detectors and beam test geometry.
  - Physics generator: Integrate with external physics generators easily.
  - Modular user actions to collect data in simulation: Save more information other than the event data model.

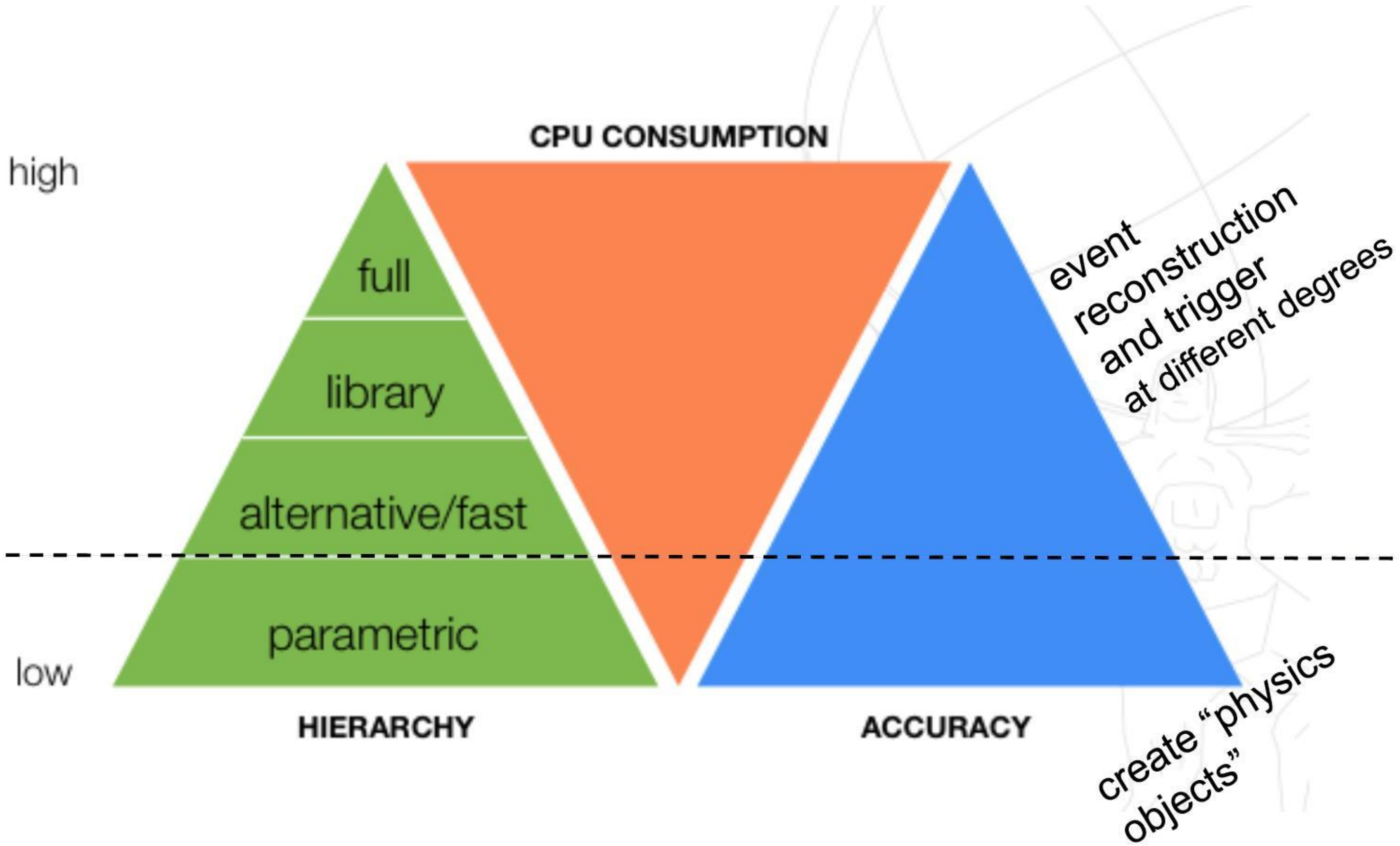
| **Sub-components** | **Plans** |
|---|---|
| Generators | Generators on the fly |
| Event Data Model | MCTruth correlation |
| Geometries & fields | TPC, calo, magnetic field, **different options** |
| Digitization | **MC hit level event mixing** |
| Fast simulation | Integration, Parameterization, Machine Learning |
| Validation & Production | stress testing, performance testing, MC data challenges etc. |
| Parallelism | Gaudi, Geant4 10.x, MT&MPI (multi-nodes) |

Git Repo: http://cepcgit.ihep.ac.cn/cepc-prototype/CEPCSW

# Thank you!

# Backup slides

- Full/Fast simulation

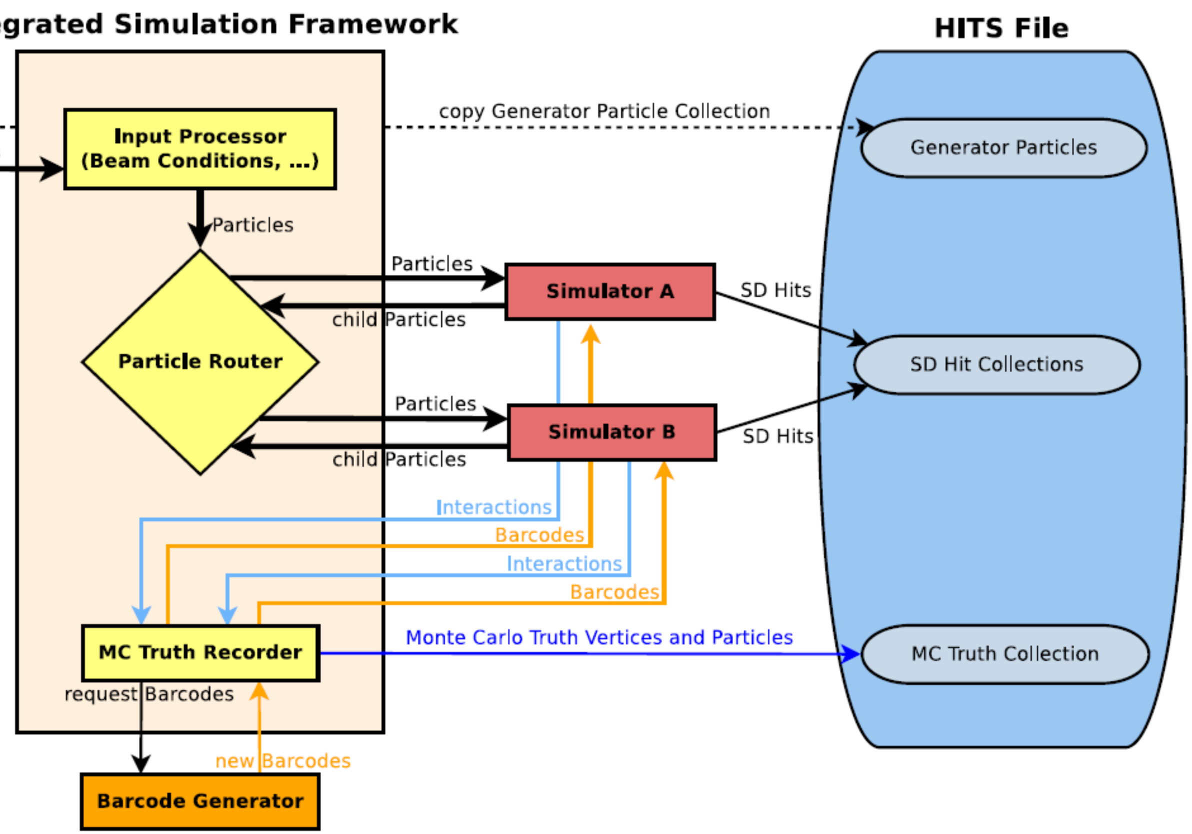


# Fast simulation in FCCSW

- Geant4 region-based
  - Several interfaces are implemented as Tools, which are then invoked by Geant4 Fast simulation model.
  - ISimG4RegionTool: Associate regions and fast simulation models.
    - SimG4FastSimCalorimeterRegion (GFlashShowerModel)
    - SimG4FastSimTrackerRegion (sim::FastSimModelTracker)
- Delphes-based
  - Algorithm “DelphesSimulation”
  - [https://cp3.irmp.ucl.ac.be/projects/delphes](https://cp3.irmp.ucl.ac.be/projects/delphes)

# Fast simulation in ATLAS

- ATLAS ISF (Integrated Simulation Framework)




Allow mixing of full and fast simulation within each physics event.


http://physik.uibk.ac.at/hephy/theses/diss_er.pdf

# Job option

- Job option is a python script.
- Users need to copy the standard one.
- Use gaudirun.py to run it. (It can be improved in the future.)

```
./run gaudirun.py '$EXAMPLESROOT/options/tut_detsim.py'
```

```
from Gaudi.Configuration import *

##############################################################################
# Random Number Svc
##############################################################################
from Configurables import RndmGenSvc, HepRndm__Engine_CLHEP__RanluxEngine_

# rndmengine = HepRndm__Engine_CLHEP__RanluxEngine_() # The default engine in Gaudi
rndmengine = HepRndm__Engine_CLHEP__HepJamesRandom_() # The default engine in Geant4
rndmengine.SetSingleton = True
rndmengine.Seeds = [42]  <= Random number seed

# rndmgensvc = RndmGenSvc("RndmGenSvc")
# rndmgensvc.Engine = rndmengine.name()
```

# Job option (Event Data & GeoSvc)

```
#############################################################################
# Event Data Svc
#############################################################################
from Configurables import CEPCDataSvc
dsvc = CEPCDataSvc("EventDataSvc")


#############################################################################
# Geometry Svc
#############################################################################

# geometry_option = "CepC_v4-onlyTracker.xml"
geometry_option = "CepC_v4-onlyVXD.xml"

if not os.getenv("DETCEPCV4ROOT"):
    print("Can't find the geometry. Please setup envvar DETCEPCV4ROOT." )
    sys.exit(-1)

geometry_path = os.path.join(os.getenv("DETCEPCV4ROOT"), "compact", geometry_option)
if not os.path.exists(geometry_path):
    print("Can't find the compact geometry file: %s"%geometry_path)
    sys.exit(-1)

from Configurables import GeoSvc
geosvc = GeoSvc("GeoSvc")
geosvc.compact = geometry_path
```

<= DD4hep XML file

The geometry file under DetCEPCv4 is loaded.

# Job option (Physics Generator)

```
##############################################################################
# Physics Generator
##############################################################################
from Configurables import GenAlgo
from Configurables import GtGunTool
from Configurables import StdHepRdr
from Configurables import SLCIORdr
from Configurables import HepMCRdr
from Configurables import GenPrinter

# gun = GtGunTool("GtGunTool")
# gun.Particles = ["pi+"]
# gun.Energies = [100.] # GeV

# gun.ThetaMins = [] # rad; 45deg
# gun.ThetaMaxs = [] # rad; 45deg

# gun.PhiMins = [] # rad; 0deg
# gun.PhiMaxs = [] # rad; 360deg

stdheprdr = StdHepRdr("StdHepRdr")
stdheprdr.Input = "/cefs/data/stdhep/CEPC250/2fermions/E250.Pbhabha.e0.p0.whizar

# lciordr = SLCIORdr("SLCIORdr")
# lciordr.Input = "/cefs/data/stdhep/lcio250/signal/Higgs/E250.Pbbh.whizard195/E

# hepmcrdr = HepMCRdr("HepMCRdr")
# hepmcrdr.Input = "example_UsingIterators.txt"

genprinter = GenPrinter("GenPrinter")

genalg = GenAlgo("GenAlgo")
# genalg.GenTools = ["GtGunTool"]
genalg.GenTools = ["StdHepRdr"]
# genalg.GenTools = ["StdHepRdr", "GenPrinter"]
# genalg.GenTools = ["SLCIORdr", "GenPrinter"]
# genalg.GenTools = ["HepMCRdr", "GenPrinter"]
```

<= Particle Gun

<= StdHep Reader

<= LCIO Reader

<= Enable the GenTools

# Job option (Detector simulation)

```
###############################################################################
# Detector Simulation
###############################################################################
from Configurables import DetSimSvc

detsimsvc = DetSimSvc("DetSimSvc")

# from Configurables import ExampleAnaElemTool
# example_anatool = ExampleAnaElemTool("ExampleAnaElemTool")

from Configurables import DetSimAlg

detsimalg = DetSimAlg("DetSimAlg")

# detsimalg.VisMacs = ["vis.mac"]
```

<= If need visualization, uncomment it

```
detsimalg.RunCmds = [
#    "/tracking/verbose 1",
]
```

<= Part of Geant4 macros are support

```
detsimalg.AnaElems = [
    # example_anatool.name()
    "ExampleAnaElemTool"
]
```

<= User actions

```
detsimalg.RootDetElem = "WorldDetElemTool"

from Configurables import AnExampleDetElemTool
example_dettool = AnExampleDetElemTool("AnExampleDetElemTool")
```

<= Geometry

# Job option (I/O & AppMgr)

```
#################################################################################
# POD I/O
#################################################################################
from Configurables import PodioOutput
out = PodioOutput("outputalg")
out.filename = "test-detsim10.root"
out.outputCommands = ["keep *"]

#################################################################################
# ApplicationMgr
#################################################################################

from Configurables import ApplicationMgr
ApplicationMgr( TopAlg = [genalg, detsimalg, out],
                EvtSel = 'NONE',
                EvtMax = 10,
                ExtSvc = [rndmengine, dsvc, geosvc],
)
```

<= Output file name

<= Total events

# Schematic view of Tracker

From CEPC CDR

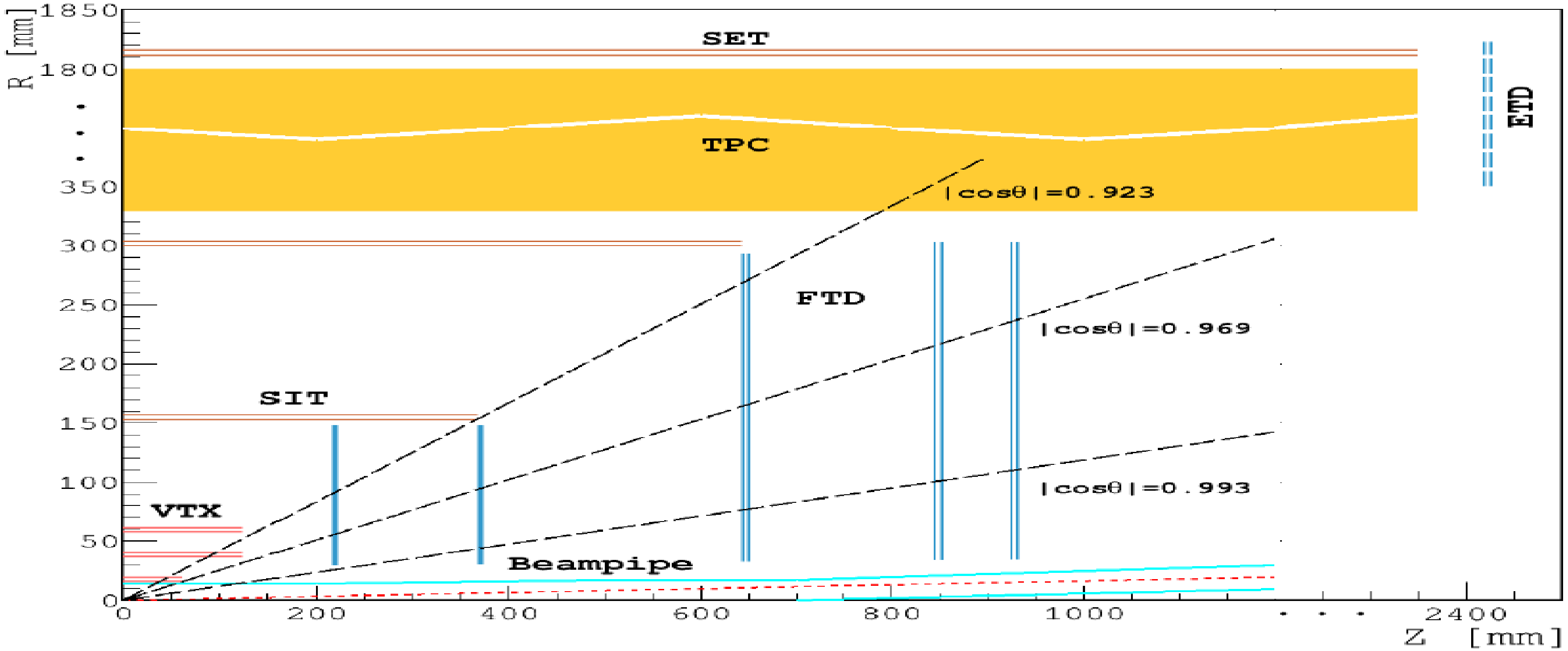



**Figure 4.1:** Preliminary layout of the tracking system of the CEPC baseline detector concept. The Time Projection Chamber (TPC) is embedded in a Silicon Tracker. Colored lines represent the positions of the silicon detector layers: red lines for the Vertex Detector (VTX) layers; orange lines for the Silicon Inner Tracker (SIT) and Silicon External Tracker (SET) components of the silicon tracker; gray-blue lines for the Forward Tracking Detector (FTD) and Endcap Tracking Detector (ETD) components of the silicon tracker. The cyan lines represent the beam pipe, and the dashed red line shows the beam line position with the beam crossing angle of 16.5 mrad. The ETD line is a dashed line because it is not currently in the full simulation. The radial dimension scale is broken above 350 mm for display convenience.