INTERNATIONAL WORKSHOP
ON THE HIGH ENERGY CIRCULAR ELECTRON POSITRON COLLIDER
CEPC 2025
November 6-10, 2025 Guangzhou, China

CEPC

Status and Plan for the Simulation Software in CEPCSW

Tao Lin

Workshop on CEPC 2025

Guangzhou

Nov 10, 2025

Outline

- CEPCSW: Key4hep based software
- Status of simulation software
- R&D of CEPC-on-Gaussino prototype
- Summary

Outline

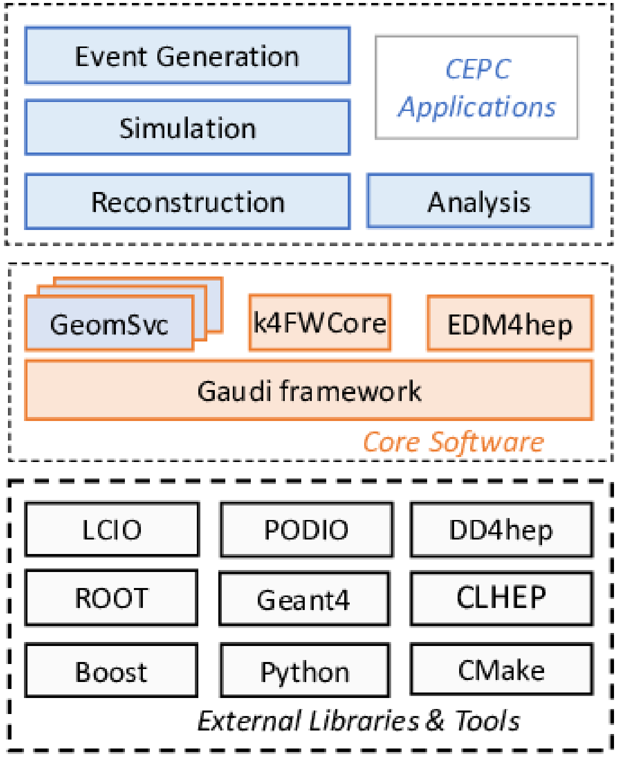
- CEPCSW: Key4hep based software
- Status of simulation software
- R&D of CEPC-on-Gaussino prototype
- Summary

CEPCSW: CEPC Software

- **The CEPC experiment is among the first to integrate with Key4hep**
  - A common software stack designed for future HEP experiments such as CEPC, CLIC, FCC, and ILC

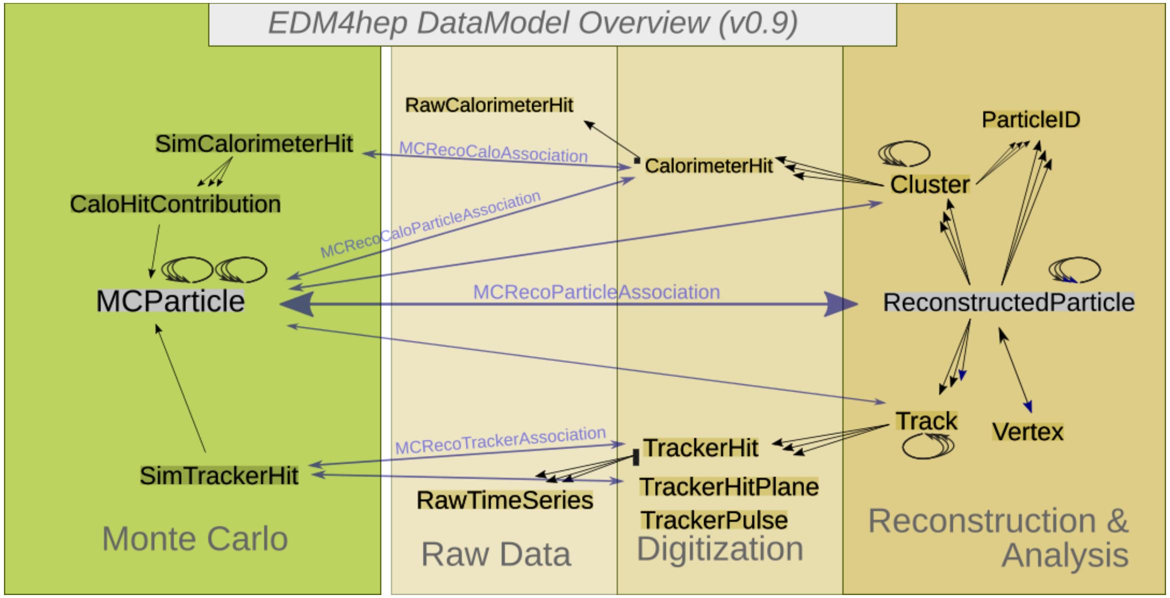
https://code.ihep.ac.cn/cepc/CEPCSW

- **Multi-layer structure**
  - Applications: simulation, reconstruction, analysis
  - Core software
  - External libraries



- **The key components of core software:**
  - Gaudi: defines interfaces to all software components and manages execution flow
  - EDM4hep: generic event data model
  - DD4hep: unified detector geometry description
  - CEPC-specific components: GeomSvc, simulation framework, RDFAnalysis, beam background mixing, fast simulation, machine learning interface, etc.

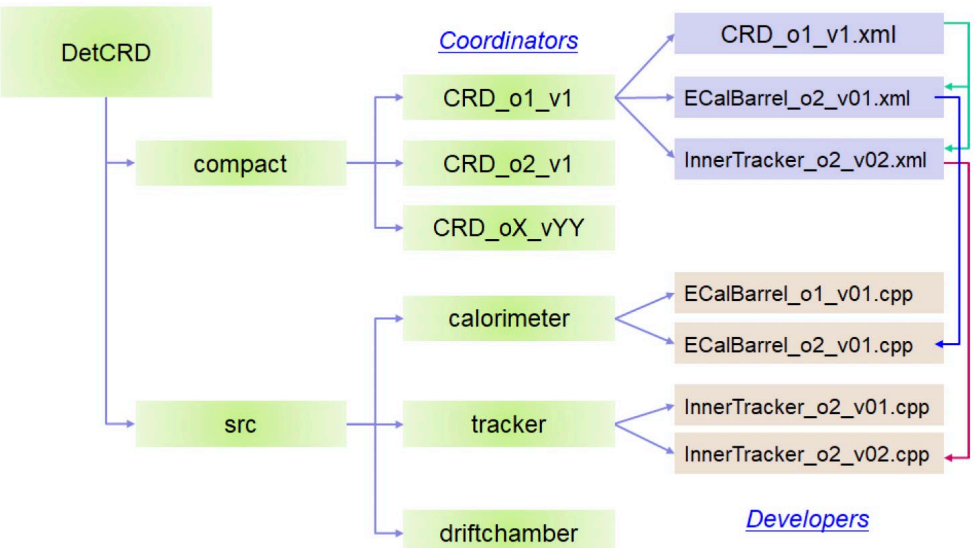
Event Data Model

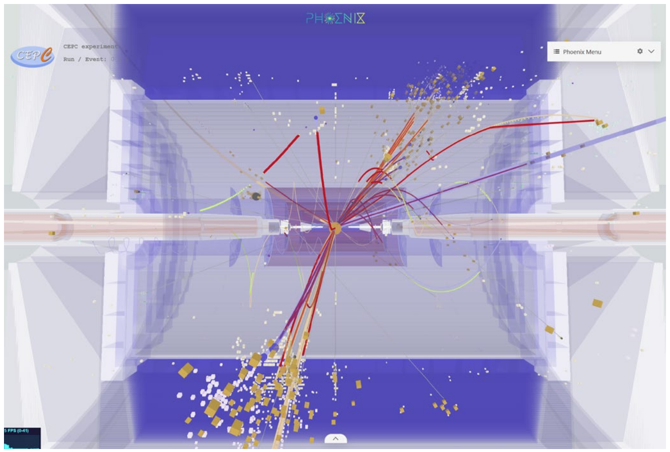


- **EDM4hep has been adopted as the official event data model**
  - provide a unified framework for managing event data across different HEP experiments
  - supports multiple backend data formats, including ROOT and LCIO
- **A number of CEPC-specific classes have been added to EDM4hep**

Detector Description

- **The DD4hep toolkit offers detector information to various applications from a unified source**
  - Each detector element can be created using XML-based compact detector descriptions and C++-based detector constructors
  - Any detector element can be enhanced with extensions, e.g. supplying extra information for reconstruction algorithms
- **A geometry service manages the DD4hep instance in CEPCSW**

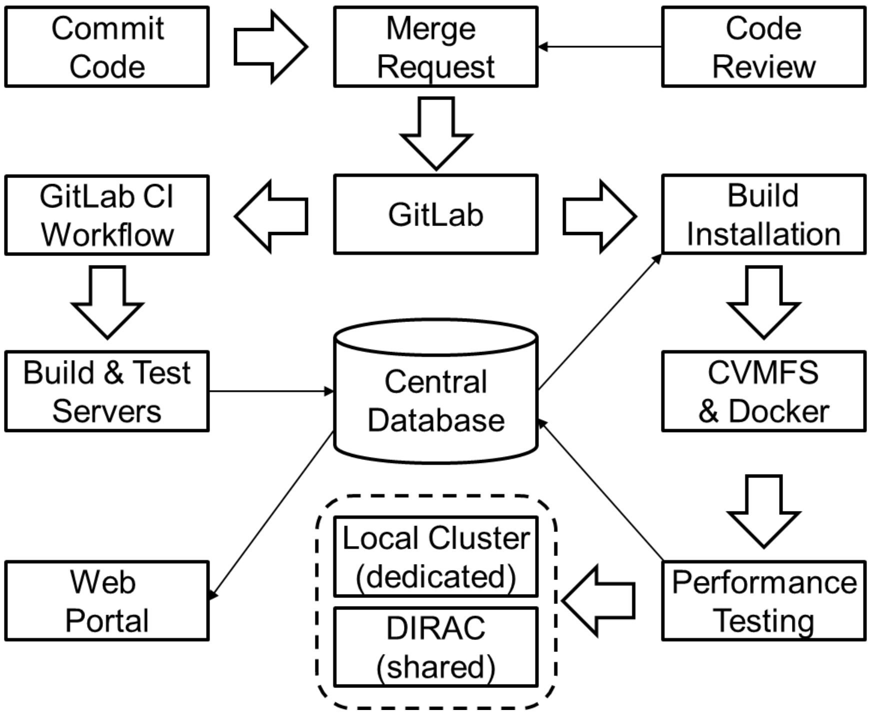




Event display in Phoenix

Software development and validation

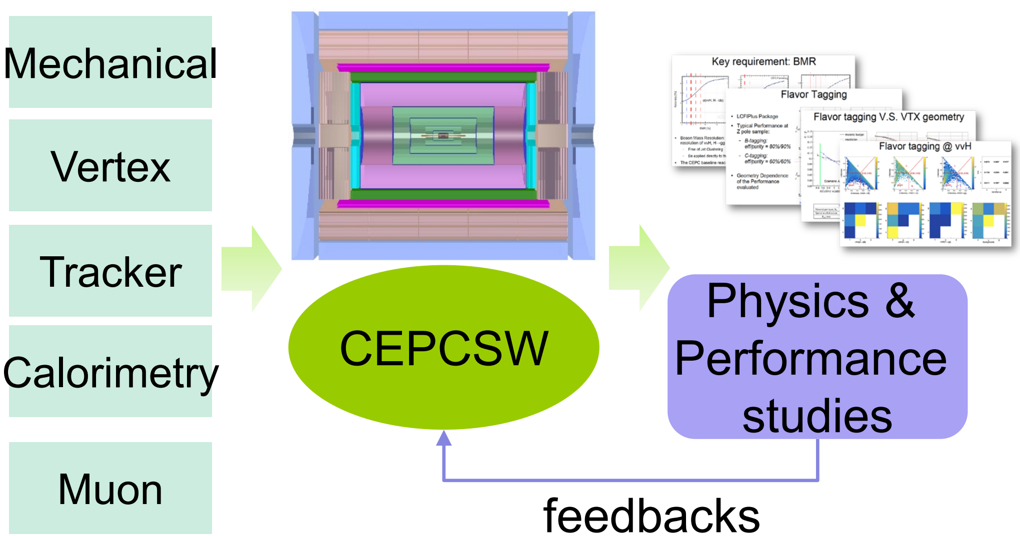
- **Modern software development practices are applied in CEPCSW, following best practices from HEP Software Foundation (HSF).**
- **Version control & CI/CD**
  - Git/GitHub/GitLab
  - GitLab Runners
- **Configuration & building**
  - CMake
  - Make and Ninja
- **Software distribution**
  - CVMFS
  - Container: Docker and Apptainer
- **Documentation**
  - Markdown and Sphinx+RTD
    https://cepcsw-docs.readthedocs.io/en/latest/



Automated validation infrastructure

Software release

- **We support the rapid iterations of the reference detector design.**
  - New version scheme based on date: tdr *YY.MM.NN*
  - External libraries are frozen at LCG 105 at AlmaLinux 9



Release	Timeline	Release Count
tdr24.3	March	2
tdr24.4	April	2
tdr24.5	May	1
tdr24.9	Sept	2
tdr24.10	Oct	1
tdr24.12	Dec	1
tdr25.1	Jan	3
tdr25.3	March	8
tdr25.5	May	1
tdr25.6	June	1
tdr25.8	Aug	1

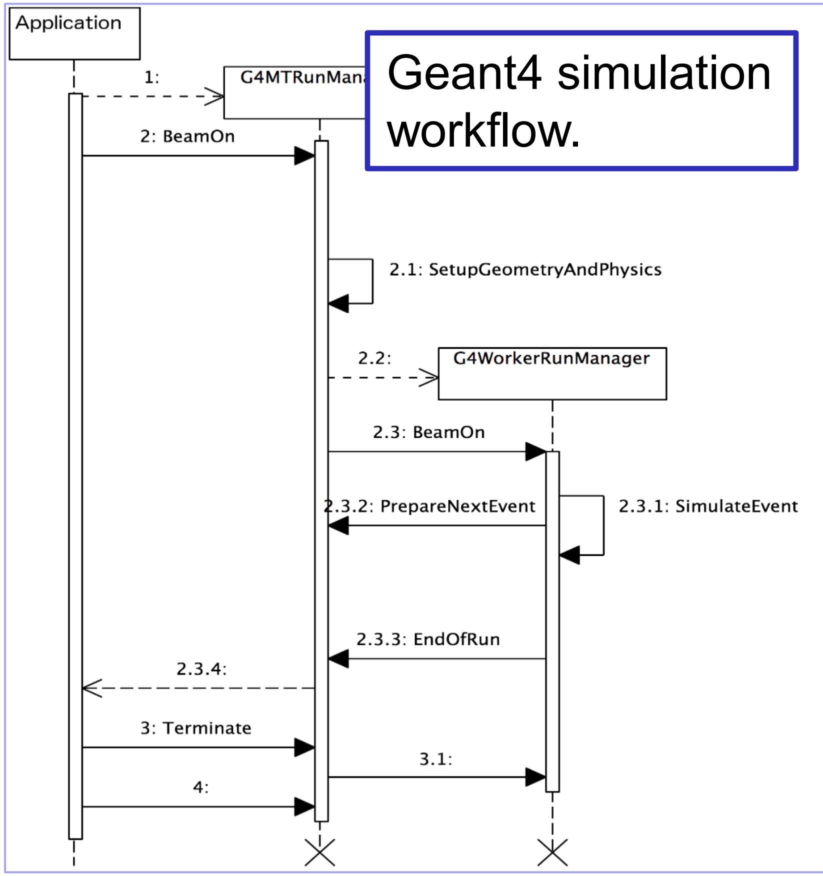
Outline

- CEPCSW: Key4hep based software
- Status of simulation software
- R&D of CEPC-on-Gaussino prototype
- Summary

Simulation framework in CEPCSW

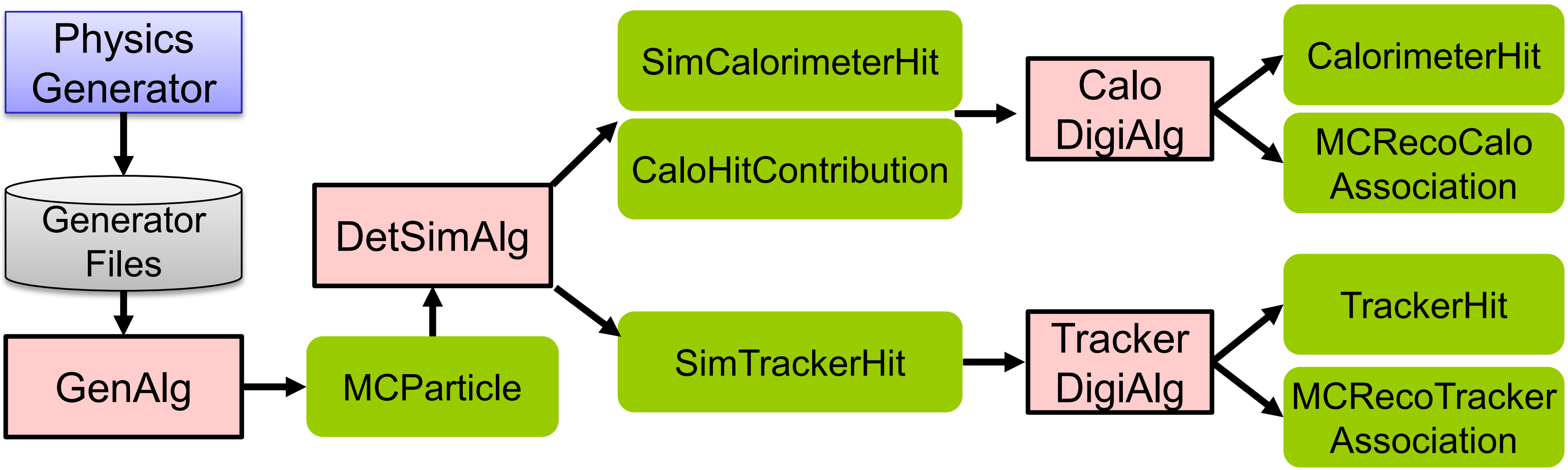
- Simulations have become critical for
  - the design of detectors
  - the development of reconstruction algorithms
- Three stages in a simulation chain
  - Physics generator: produce primary particles.
  - Detector simulation: produce hits.
  - Digitization: produce digits.
- The simulation framework provides the abilities to run the simulation chain easily.
  - Customize generation, detector design, physics list, etc.

Geant4 simulation workflow.



For example, Geant4 implements its own run managers to control the simulation workflow. Need to make Geant4 work with the other algorithms.

Simulation workflow

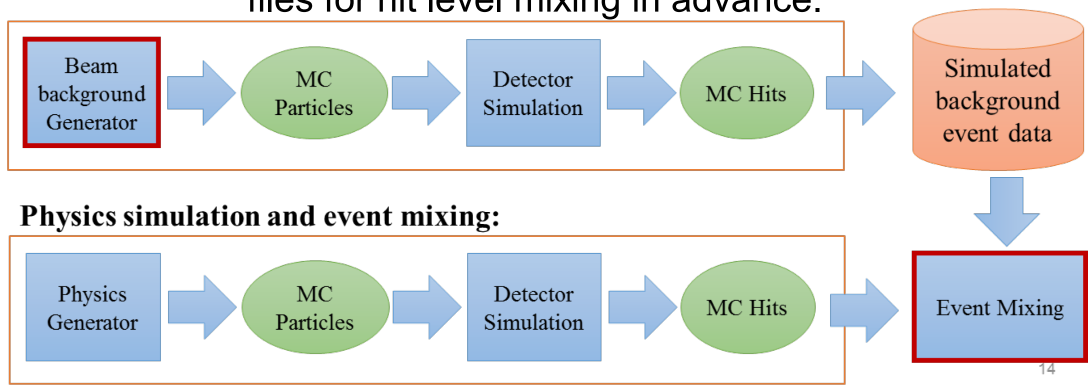


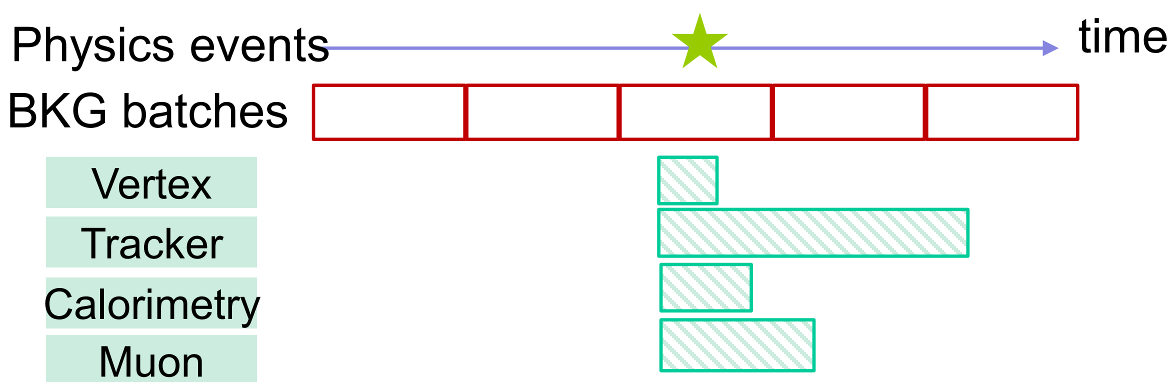
- **The workflow comprises three stages:**
  - Physics event generation: WHIZARD, Pythia6/8, MDI background, particle gun
  - Detector simulation: Geant4 + DD4hep/DDG4
  - Digitization: silicon tracker, TPC, ECAL, HCAL, Muon
- **Beam-induced background: particle level mixing, hit level mixing**

Beam-induced background simulation

- **Execution flow**
  - Before digitization, keep same EDM
- **Two modes of mixing have been implemented**
  - Particle level mixing: used by MDI study
  - Hit level mixing: used by performance study
- **Hit level mixing algorithm**
  - **BackgroundLoader:** loads background files with podio::Frame, then filters hits within a configurable time window.
  - **BackgroundEvent:** transient object that holds all background hit collections
  - **Time mode:** configurable as uniform or fixed
  - **Time window:** configurable per sub-detector

MDI group prepares different types of background files for hit level mixing in advance.

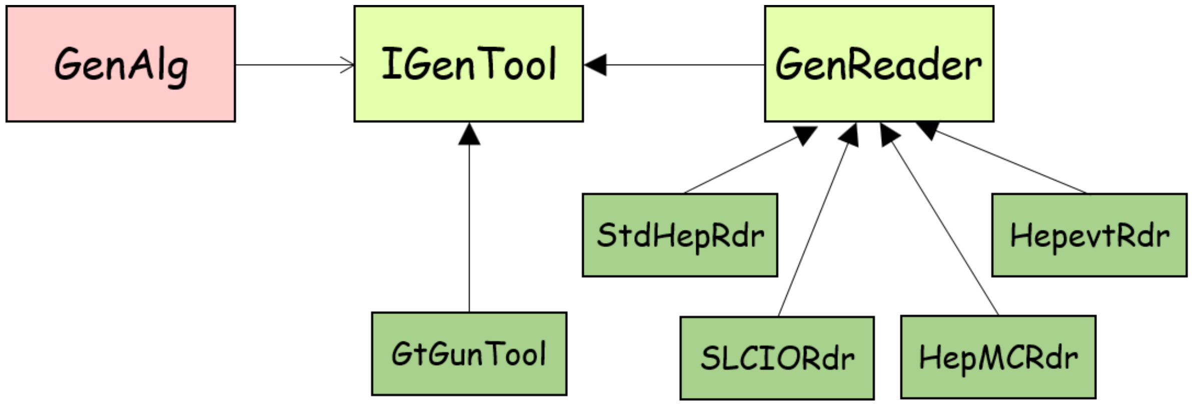




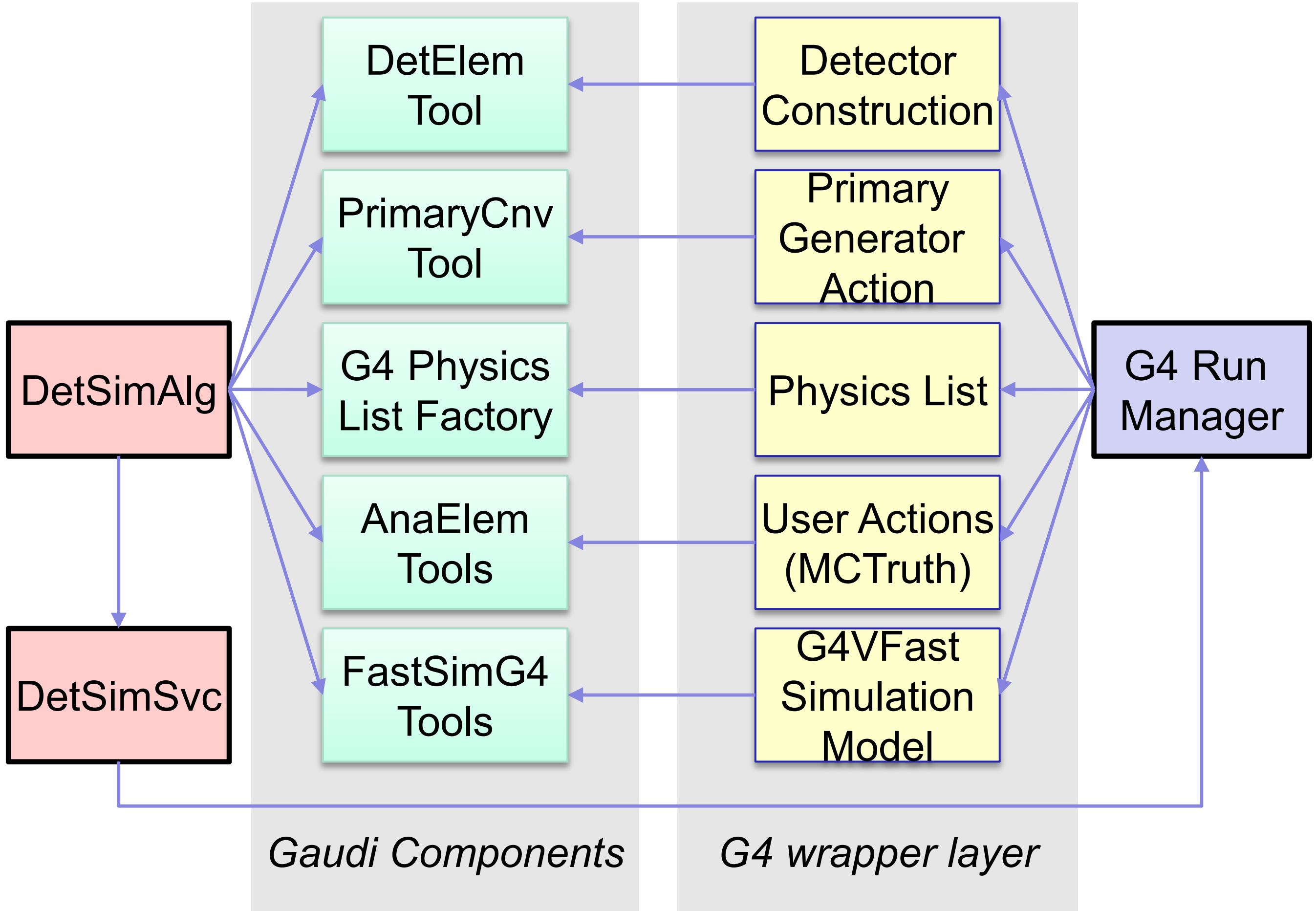
OOW (out of window): stores additional MC truth hits outside the defined time window, enabling dedicated algorithm studies.

Physics generator interface

- Modular design to support different types of generators
  - Different formats: StdHep, HepEvt, LCIO, HepMC etc.
  - Particle gun: customized generation of single/multiple particles
  - Particle level mixing: extend the GenTool and add "time" information



Detector simulation



Within CEPCSW, a lightweight simulation framework has been developed to enable seamless integration of Geant4 into Gaudi framework

MC truth

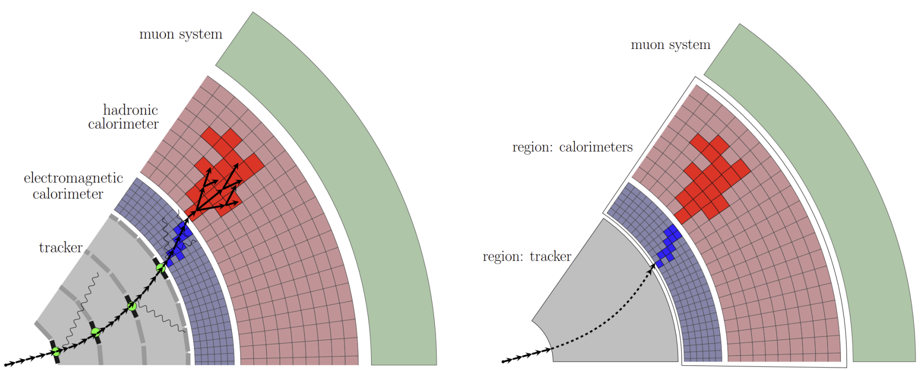
- **MC truth information can help developers and analyzers understand the detailed processes.**
  - Particle – particle relationship at generation and simulation stage
  - Hit – particle relationship at simulation stage

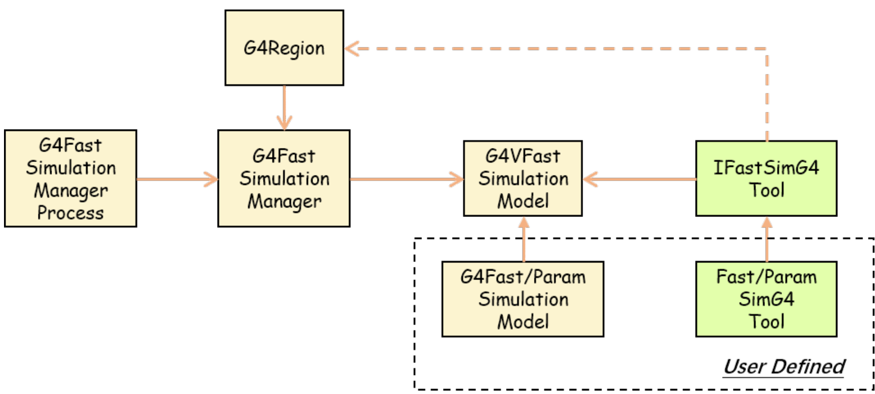
- **Particle – particle relationship**
  - Truth information from physics generators
  - Interesting physics processes in simulation: decay and gamma conversion
  - Part of secondaries created inside the tracker system if the initial kinetic energy of a secondary track is higher than a threshold

- **Hit – particle relationship**
  - A tracker/calo hit can be associated with a primary track or a secondary track in simulation
  - Easy to know a hit is from signals or backgrounds.

Fast simulation

- Geant4 simulation could be time consuming. The idea of fast simulation is to replace the consuming part with a fast algorithm, such as parameterization method.
- Region based fast simulation is adopted
  - When a particle enter a region, fast simulation will be triggered by Geant4.
  - Machine learning inference could be integrated via ONNX.



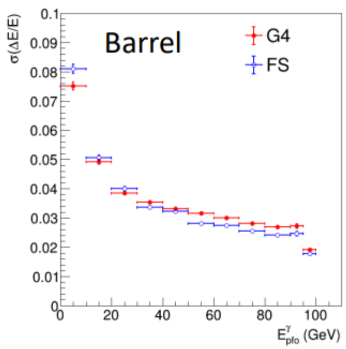


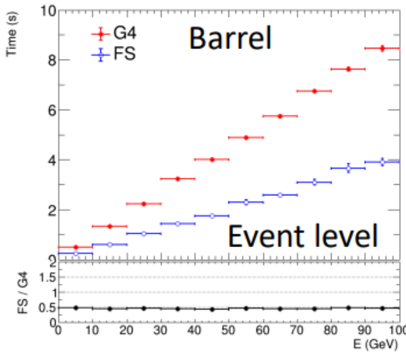
A Zaborowska 2017 J. Phys.: Conf. Ser. 898 042053

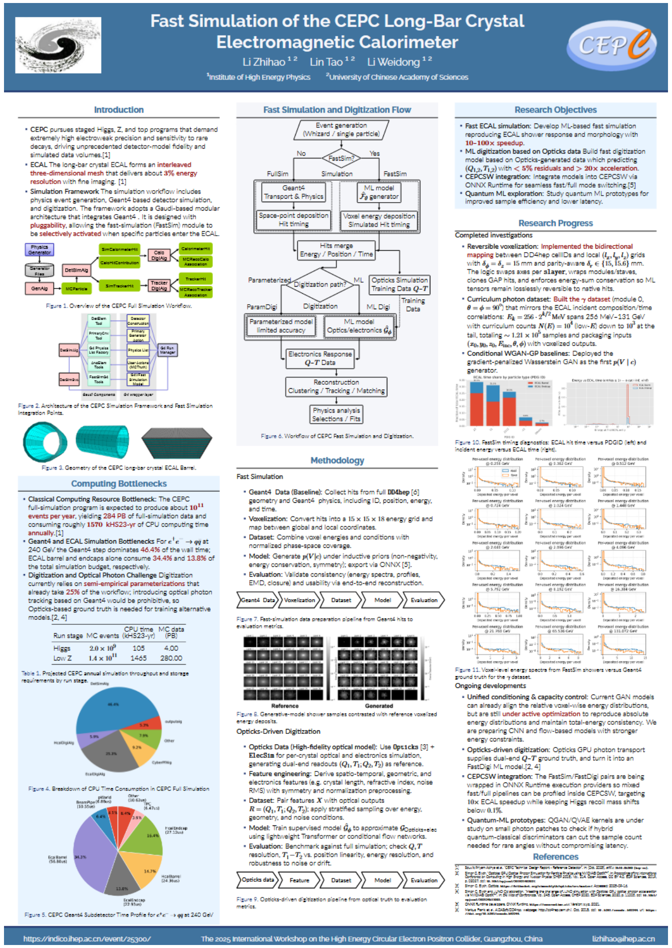
Fast simulation

Zhihao Li, Poster #D10

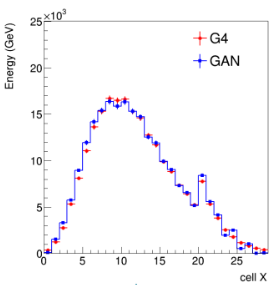
- Traditional method: shower parameterization, frozen shower,

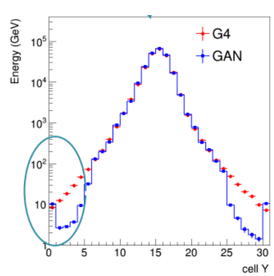


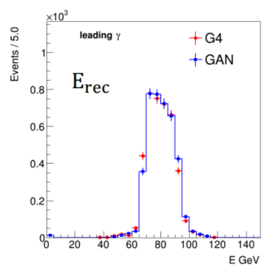


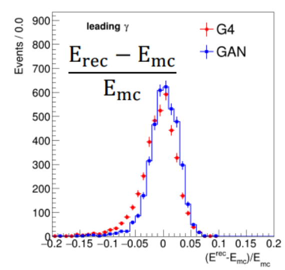


- ML-based method: GAN, VAE, NF, Diffusion, …









Wenxing Fang

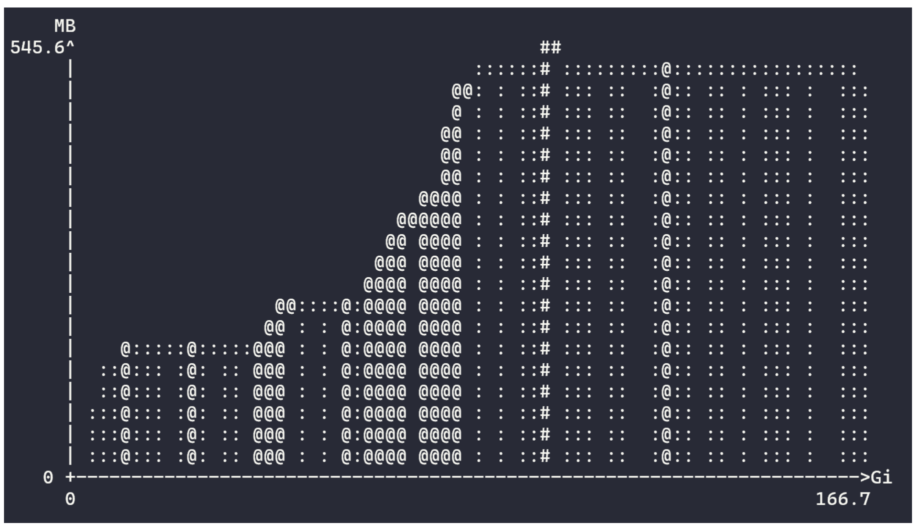
Outline

- CEPCSW: Key4hep based software
- Status of simulation software
- R&D of CEPC-on-Gaussino prototype
- Summary

Challenges in simulation framework

- Due to the more precise geometries and physics, more memory will be used.
- An efficient solution is using the multi-threaded techniques, which requires the multi-threading design.

Memory usage (serial version)



The RSS memory is about **950MB at initialization stage**.

Two different approaches for Gaudi and Geant4.

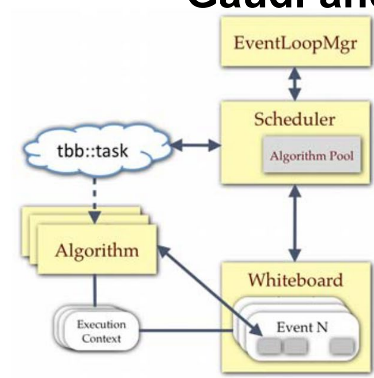


Figure 1. General structure of the GaudiHive framework

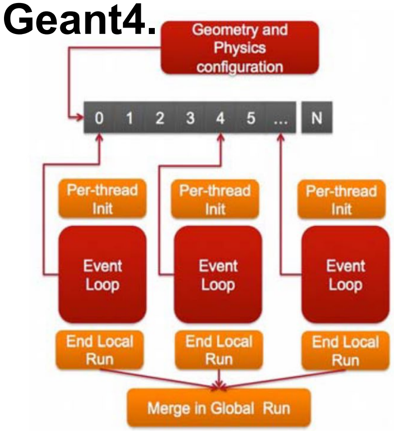
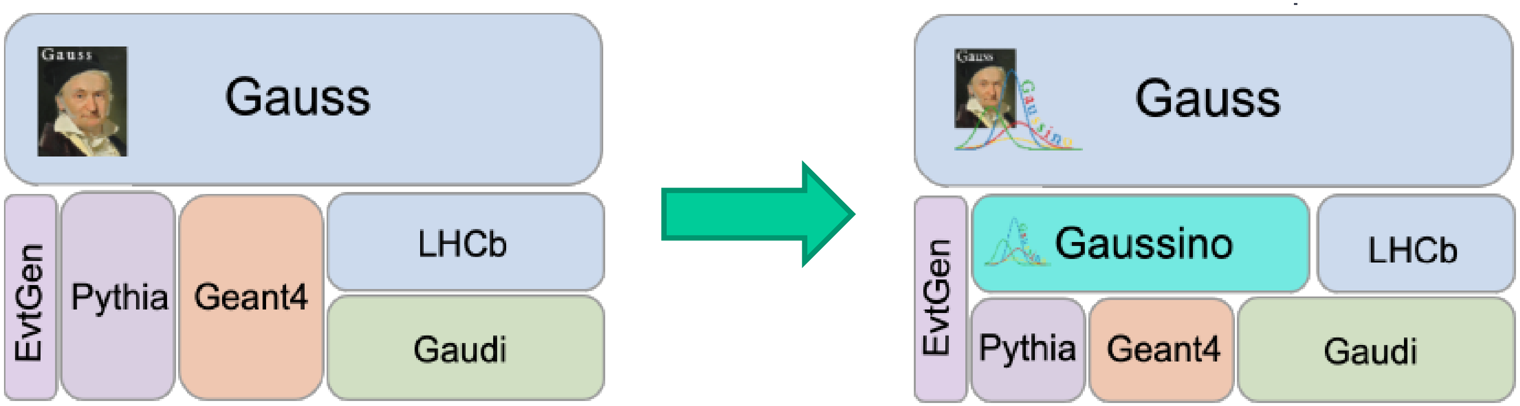


Figure 2. A sketch of the structure of a Geant4-MT simulation

A Di Simone and on behalf of the ATLAS Collaboration 2017 J. Phys.: Conf. Ser. 898 042010

Question: Develop a new one or adopt an existing one?

- Evolution of the simulation framework from LHCb
  - The underlying framework is moving to Gaudi Functional and Gaudi Hive
  - Better support for multi-threading, machine learning, fast simulation methods
  - Gauss-on-Gaussino is a new version of LHCb simulation framework
- Gaussino is being added to Key4hep by extracting experiment-independent parts from Gauss.



Gaussino simulation framework

The idea of Gaussino in LHCb

- LHCb Upgrade in Run3 very challenging for software and computing!
  - large increase in luminosity, i.e. Linst : $4 \times 10^{32} \rightarrow 2 \times 10^{33} cm^{-2} s^{-1}$ and pileup: $1.1 \rightarrow 7.6$
  - full software trigger with high signal purity
  - analysis directly on trigger output
  - **simulation will continue to dominate the CPU needs**

- Modernization of the whole software
  - **Multi-threading**
  - Better use of multi-processor CPUs
  - **Reduce memory usage**
  - Optimize cache performance
  - **Remove dead code**
  - Move to modern data structures
  - Enable code vectorization
  - Enable algorithmic optimization
  - HLT1 reconstruction on GPUs

Simulation software upgrade also needed!
- $\sim$ **15 years old**
- Adapt **to change** in LHCb common software, e.g. use of DD4Hep
- Exploit new feature of external HEP simulation software, e.g. in Geant4
- Combine multi-threaded Gaudi and Geant4
- Need of extensive palette of fast simulations

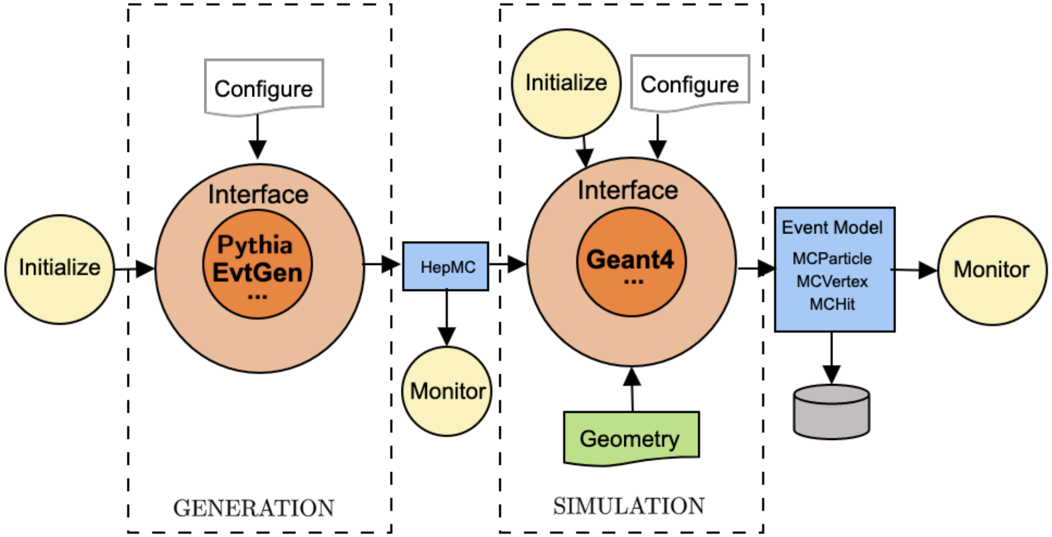
Separate simulation core functionality

LHCb Collaboration, Upgrade Software and Computing TDR, CERN-LHCC-2018-007

G. Corti CEPC 2025 - 7 November 2025 - Guangzhou, China Gaussino 4

Core components of Gaussino

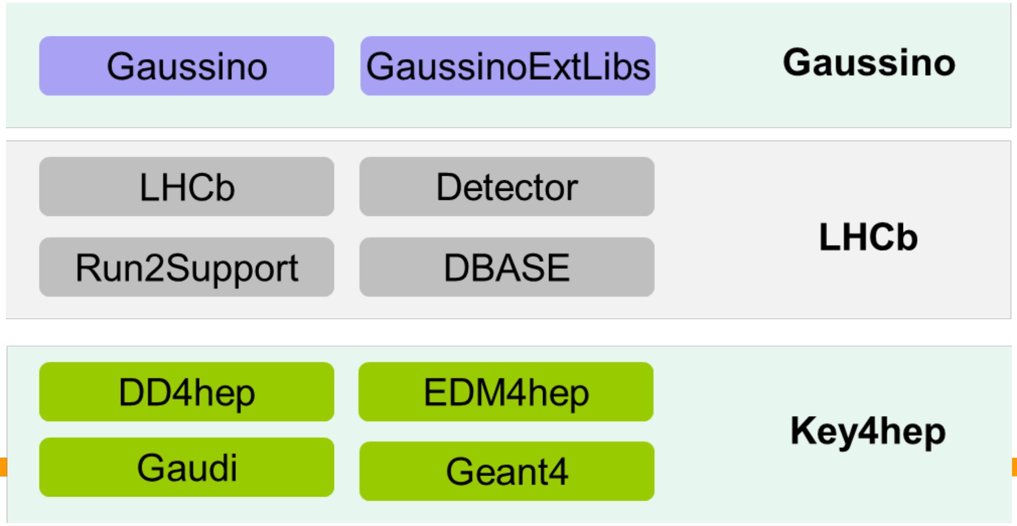
- Gaussino is a thread-safe simulation framework based on Gaudi Functional and provides interfaces to Pythia and Geant4.
- Modular design
  - Gaudi Functional Algorithms
  - Gaudi Tools
- Four components
  - Generation of events
  - The detector simulation
  - Geometry service
  - Monitoring & output
- Easy to configure by customizing the algorithms, services and tools



- Generation: `Generation` and `ParticleGun`
  - The input is **LHCb GenHeader**
  - The output is **HepMC GenEvent**
- Detector simulation: `GiGaAlg`
  - The input is **HepMC GenEvent**
  - The output is **G4Event** and **MC truths**

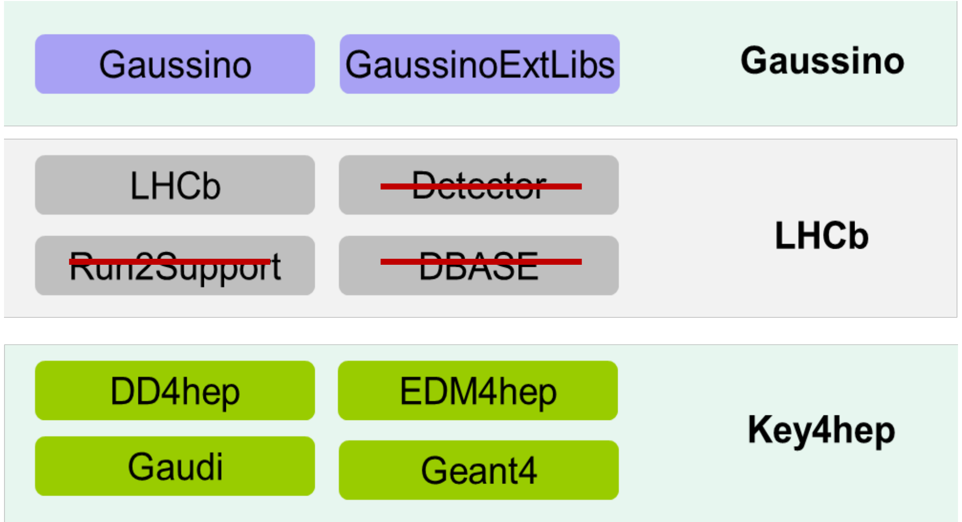
CEPC-on-Gaussino prototype (1)

- Gaussino still depends on LHCb software and removing the dependencies is on going
  - Some existing works: MR !1493 in Gaudi by Graeme.
- Development of CEPC-on-Gaussino was planned with the following three steps
  - ✓ **Using the original version having the dependency on the LHCb software**
  - ✓ **Creating a modified version with less LHCb dependency**
  - ☐ Directly using the Key4hep version without LHCb dependency (not available at the moment)



CEPC-on-Gaussino prototype (2)

- It will be too heavy to build the prototype with the whole LHCb software.
- CMakeLists.txt in LHCb and gaussinoextlibs are modified to build necessary libraries.
- After optimization, about 20 libraries are built in LHCb and GaussinoExtLibs.

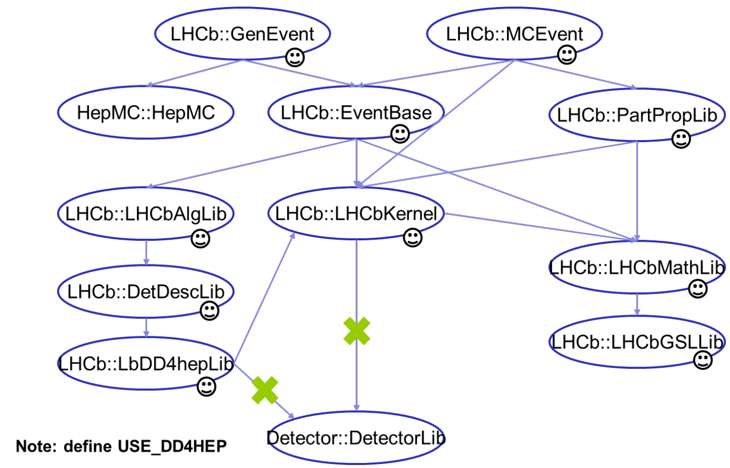



```
[lint@lxslc705]$ ls -l LHCb/InstallArea/lib/
total 24712
drwxr-xr-x 3 lint physics      18 Dec 12 03:54 cmake
-rw-r--r-- 1 lint physics   37218 Dec 12 03:54 DAQEventDict_rdict.pcm
-rw-r--r-- 1 lint physics    3053 Dec 12 03:54 EventBaseDict_rdict.pcm
-rw-r--r-- 1 lint physics    7512 Dec 12 03:54 GaudiGSLMathDict_rdict.pc
-rw-r--r-- 1 lint physics    8364 Dec 12 03:54 GenEventDict_rdict.pcm
-rw-r--r-- 1 lint physics     370 Dec 12 03:54 LHCb.components
-rw-r--r-- 1 lint physics     751 Dec 12 03:54 LHCb.confdb
-rw-r--r-- 1 lint physics   32768 Dec 12 03:54 LHCb.confdb2
-rw-r--r-- 1 lint physics   22277 Dec 12 03:54 LHCbDict.rootmap
-rw-r--r-- 1 lint physics   28815 Dec 12 03:54 LHCbMathDict_rdict.pcm
-rwxr-xr-x 1 lint physics  424208 Dec 12 03:54 libDAQEventDict.so
-rwxr-xr-x 1 lint physics 3229632 Dec 12 03:54 libDAQEventLib.so
-rwxr-xr-x 1 lint physics  236840 Dec 12 03:54 libEventBaseDict.so
-rwxr-xr-x 1 lint physics 1171648 Dec 12 03:54 libGaudiGSLLib.so
-rwxr-xr-x 1 lint physics  269464 Dec 12 03:54 libGaudiGSLMathDict.so
-rwxr-xr-x 1 lint physics 1836888 Dec 12 03:54 libGaudiGSL.so
-rwxr-xr-x 1 lint physics 1221432 Dec 12 03:54 libGenEventDict.so
-rwxr-xr-x 1 lint physics  442888 Dec 12 03:54 libGenEvent.so
-rwxr-xr-x 1 lint physics 3061496 Dec 12 03:54 libLHCbKernel.so
-rwxr-xr-x 1 lint physics 2534464 Dec 12 03:54 libLHCbMathDict.so
-rwxr-xr-x 1 lint physics 6642152 Dec 12 03:54 libLHCbMathLib.so
-rwxr-xr-x 1 lint physics  363000 Dec 12 03:54 libMCEvent.so
-rwxr-xr-x 1 lint physics  509928 Dec 12 03:54 libPartPropDict.so
-rwxr-xr-x 1 lint physics 1107720 Dec 12 03:54 libPartPropLib.so
-rwxr-xr-x 1 lint physics 2050024 Dec 12 03:54 libPartProp.so
-rw-r--r-- 1 lint physics    7934 Dec 12 03:54 PartPropDict_rdict.pcm
```

```
[lint@lxslc705]$ ls -l gaussinoextlibs/InstallArea/lib/
total 46028
drwxr-xr-x 3 lint physics       29 Dec 12 03:54 cmake
-rw-r--r-- 1 lint physics     2940 Dec 12 03:54 G__DDG4Python_rdict.pcm
-rw-r--r-- 1 lint physics    23520 Dec 12 03:54 G__DDG4_rdict.pcm
-rw-r--r-- 1 lint physics     1763 Dec 12 03:54 G__DDPython_rdict.pcm
-rw-r--r-- 1 lint physics      208 Dec 12 03:54 libDDG4LCIO.components
lrwxrwxrwx 1 lint physics       19 Oct 24 11:51 libDDG4LCIO.so -> libDDG4LCIO.so.1.25
-rwxr-xr-x 1 lint physics  3452704 Dec 12 03:54 libDDG4LCIO.so.1.25
-rw-r--r-- 1 lint physics    16549 Dec 12 03:54 libDDG4Plugins.components
lrwxrwxrwx 1 lint physics       22 Oct 24 11:51 libDDG4Plugins.so -> libDDG4Plugins.so.1.25
-rwxr-xr-x 1 lint physics 15688056 Dec 12 03:54 libDDG4Plugins.so.1.25
-rw-r--r-- 1 lint physics      220 Dec 12 03:54 libDDG4Python.components
lrwxrwxrwx 1 lint physics       21 Oct 24 11:51 libDDG4Python.so -> libDDG4Python.so.1.25
-rwxr-xr-x 1 lint physics   818528 Dec 12 03:54 libDDG4Python.so.1.25
lrwxrwxrwx 1 lint physics       15 Oct 24 11:51 libDDG4.so -> libDDG4.so.1.25
-rwxr-xr-x 1 lint physics 23030168 Dec 12 03:54 libDDG4.so.1.25
lrwxrwxrwx 1 lint physics       19 Oct 24 11:51 libDDPython.so -> libDDPython.so.1.25
-rwxr-xr-x 1 lint physics   422488 Dec 12 03:54 libDDPython.so.1.25
lrwxrwxrwx 1 lint physics       20 Oct 24 11:49 libHepMC3search.so -> libHepMC3search.so.4
-rwxr-xr-x 1 lint physics   246384 Dec 12 03:54 libHepMC3search.so.4
-rw-r--r-- 1 lint physics   363832 Dec 12 03:54 libHepMC3search-static.a
lrwxrwxrwx 1 lint physics       14 Oct 24 11:49 libHepMC3.so -> libHepMC3.so.3
-rwxr-xr-x 1 lint physics   923296 Dec 12 03:54 libHepMC3.so.3
-rw-r--r-- 1 lint physics  2108684 Dec 12 03:54 libHepMC3-static.a
drwxr-xr-x 3 lint physics       27 Dec 12 03:54 python3.9
```

CEPC-on-Gaussino prototype (3)

- **Event Data Model**
  - reuse GenEvent and MCEvent from the LHCb project
  - A minimum number of packages are selected
  - Non-required dependencies were removed
- **Detector description**
  - use the DD4hep, so that no dependent on LHCb detector description library
  - Use DD4hepCnvSvc as Geometry Service to load CEPC tracker detector.
- **Detector response:**
  - Implement G4 Sensitive Detector and Hit object for tracker detector.
  - Implement a monitor tool to save user output.




```
CEPCTest/
├── CMakeLists.txt
├── include
│   └── CEPCTest
│       ├── DDG4SensitiveDetector.h
│       ├── Geant4Hits.h
│       ├── GenericTrackerSensDetTool.h
│       └── GenericTrackerSensitiveDetector.h
├── options
│   └── cepc_gaussino.py
└── src
    ├── Components
    │   ├── GenericTrackerMonTool.cpp
    │   ├── GenericTrackerMonTool.hh
    │   └── GenericTrackerSensDetToolComponent.cpp
    └── Lib
        ├── DDG4SensitiveDetector.cpp
        ├── Geant4Hits.cpp
        ├── GenericTrackerSensDetTool.cpp
        └── GenericTrackerSensitiveDetector.cpp
```

CEPC-on-Gaussino prototype (4)

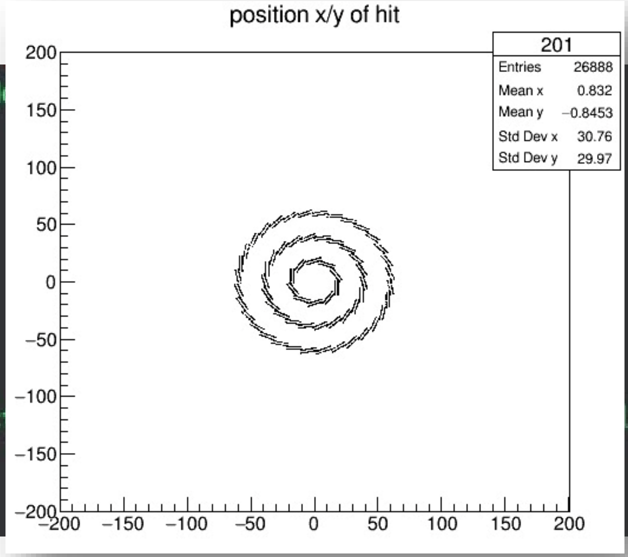
- **Job option**
  - CEPC-on-Gaussino is configured in the Python script.
  - Register the sensitive detector factory and associate it with DD4hep detector name.
  - Register the monitor tool for testing only.
  - The output in EDM4hep is not implemented yet.

```
montool = giga.addTool(GenericTrackerMonTool, name="GenericTrackerM
montool.OutputLevel = DEBUG
montool.CollectionName = "VXDCollection"

giga.MonitorTools = ["GenericTrackerMonTool"]

GaussinoGeometry().GeometryService = "DD4hepCnvSvc"

DD4hepCnvSvc().DescriptionLocation = "CEPCSW/Detector/DetCRD/compac
DD4hepCnvSvc().SensDetMappings = {"VXD": "VXD"}
# DD4hepCnvSvc().OutputLevel = DEBUG
```



https://gitlab.cern.ch/talin/build-cepc-on-gaussino/-/blob/main/my_gaussino_dd4hep.py?ref_type=heads

CEPC-on-Gaussino prototype (5)

- The installation script is used to build LHCb, gaussinoextlibs, Gaussino, and CEPCSW with branch name cepc-on-gaussino.

```
$ git clone ssh://git@gitlab.cern.ch:7999/talin/build-cepc-on-gaussino.git
$ cd build-cepc-on-gaussino

$ bash build.sh build-all

$ source setup.sh

$ gaudirun.py my_gaussino_dd4hep.py
```

```
function env-setup() {
    # == LCG  ==
    lcg_version_lcg=LCG_103 # The version used by LCG self.
    lcg_platform=x86_64-centos7-gcc11-opt

    export PATH=/cvmfs/sft.cern.ch/lcg/contrib/git/bin:$PATH
    export LD_LIBRARY_PATH=/cvmfs/sft.cern.ch/lcg/contrib/git/lib64:$LD_LIBRARY_PATH

    source /cvmfs/sft.cern.ch/lcg/views/${lcg_version_lcg}/${lcg_platform}/setup.sh

    # == Gaussino and dependencies  ==

    export WORKTOP=$(pwd)

    for project in LHCb gaussinoextlibs Gaussino CEPCSW; do

        project_path=$WORKTOP/$project/InstallArea
        export CMAKE_PREFIX_PATH=$project_path:$CMAKE_PREFIX_PATH
        export PATH=$project_path/bin:$PATH
        export LD_LIBRARY_PATH=$project_path/lib:$LD_LIBRARY_PATH
        export PYTHONPATH=$project_path/lib:$PYTHONPATH
        export PYTHONPATH=$project_path/python:$PYTHONPATH

    done

    export GAUSSINOROOT=$WORKTOP/Gaussino/Sim/Gaussino
}
```

Project	Git repo
LHCb	https://gitlab.cern.ch/talin/LHCb/-/tree/cepc-on-gaussino?ref_type=heads
gaussinoextlibs	https://gitlab.cern.ch/talin/gaussinoextlibs/-/tree/cepc-on-gaussino?ref_type=heads
Gaussino	https://gitlab.cern.ch/talin/Gaussino/-/tree/cepc-on-gaussino?ref_type=heads
CEPCSW	https://gitlab.cern.ch/talin/CEPCSW/-/tree/cepc-on-gaussino?ref_type=heads
Installation script	https://gitlab.cern.ch/talin/build-cepc-on-gaussino

Implementation (1): Geometry

- Gaussino provides several ways to setup geometry.
- Specify the factory (**GiGaFactoryBase<G4VUserDetectorConstruction>**) in GiGaMT.DetectorConstruction and setup the corresponding properties.
  - GiGaMTDetectorConstructionFAC (default): property GiGaMTGeoSvc
  - GDMLConstructionFactory: property GDML
  - DD4hepDetectorConstructionFAC: property DescriptionLocation
- If use the default factory, the geometry could be customized using the GeoSvc (**IGiGaMTGeoSvc**).
  - DD4hepCnvSvc
    - Property DescriptionLocation

```cpp
class DD4hepCnvSvc : public extends<Service, IGiGaMTGeoSvc> {
public:
  using extends::extends;
  virtual ~DD4hepCnvSvc() = default;

  // Service pure member functions
  StatusCode initialize() override;
  StatusCode finalize() override;

  // Pointer to the root of G4 geometry tree
  G4VPhysicalVolume* constructWorld() override;
  void               constructSDandField() override;

protected:
  // Function to load the geometry into DD4hep. Virtual
  // to allow being replaced in derived classes if needed.
  virtual const dd4hep::Detector& getDetector() const;
```

```python
giga = GiGaMT()
dettool = giga.addTool(
    GiGaMTDetectorConstructionFAC(),
    name="DetConst",
)
giga.DetectorConstruction = getattr(giga, "DetConst")

dettool.GiGaMTGeoSvc = self.getProp("GeometryService")
dettool.SensDetVolumeMap = self.getProp("SensDetMap")
extra_tools = self.getProp("ExtraGeoTools")
dettool.AfterGeoConstructionTools = extra_tools
add_constructors_with_names(dettool, extra_tools)

algs = []
algs += self._set_external_detector(dettool)
algs += self._set_parallel_geometry(dettool)
self._set_custom_simulation_regions(dettool)
self._set_gdml_import(dettool)
self._set_gdml_export(dettool)
```

python/Gaussino/Geometry.py

Implementation (2): Sensitive Detector

- DDG4 classes are reused with minor changes:
  - A hit type: **Geant4Hits**. This is from DDG4.
  - A DDG4 based SD: **DDG4SensitiveDetector**. Also from DDG4, but adding the GiGaMessage in the base class.
  - A concrete SD: **GenericTrackerSensitiveDetector**.
- Gaussino integration part:
  - A GiGa factory **GenericTrackerSensDetTool** is used to create the SD.
  - Declare the factory component: GenericTrackerSensDetToolComponent.

GenericTrackerSensDetToolComponent.cpp 188 B

```cpp

#include "CEPCTest/GenericTrackerSensDetTool.h"

DECLARE_COMPONENT_WITH_ID(GenericTrackerSensDetTool, "GenericTrackerSensDet")
DECLARE_COMPONENT_WITH_ID(GenericTrackerSensDetTool, "VXD")
```

In order to let SD get the DD4hep instances according to the name, the component is declared with the detector name.

```cpp
DDG4SensitiveDetector::DDG4SensitiveDetector(const std::string& name)
    : G4VSensitiveDetector(name) {
    dd4hep::Detector& description = dd4hep::Detector::getInstance();

    m_detector = description.detector(name);
    m_sensitive = description.sensitiveDetector(name);
    m_readout = m_sensitive.readout();

}
```

Implementation (3): Monitoring tool

- For testing only, the histograms of positions and deposit energies are created.

```
StatusCode GenericTrackerMonTool::monitor( const G4Event& aEvent )
{
  std::lock_guard<std::mutex> guard(m_hist_lock);
  G4HCofThisEvent* collections = aEvent.GetHCofThisEvent();
  G4VHitsCollection* collect;
  dd4hep::sim::Geant4TrackerHit* hit;
  double energyTotal;
  int hitNo;
  debug() << "There are " << collections->GetNumberOfCollections() << " hit collections." << endmsg;
  for ( int iter_coll = 0; iter_coll < collections->GetNumberOfCollections(); iter_coll++ ) {
    collect = collections->GetHC( iter_coll );
    if ( collect->GetName().find( m_coll_name) != std::string::npos ) {
      size_t n_hit = collect->GetSize();
      energyTotal  = 0;
      hitNo        = 0;
      debug() << "\t" << n_hit << " hits are stored in a calorimeter collection #" << iter_coll << ": "
              << collect->GetName() << endmsg;
      for ( size_t iter_hit = 0; iter_hit < n_hit; iter_hit++ ) {
          hit = dynamic_cast<dd4hep::sim::Geant4TrackerHit*>( collect->GetHit( iter_hit ) );

          if ( hit->energyDeposit != 0 ) hitNo++;
          m_hitX->fill( hit->position.x()/CLHEP::mm );
          m_hitY->fill( hit->position.y()/CLHEP::mm );
          m_hitZ->fill( hit->position.z()/CLHEP::mm );
          m_hitEnergy->fill( hit->energyDeposit/CLHEP::GeV );
          m_hitXY->fill( hit->position.x()/CLHEP::mm, hit->position.y()/CLHEP::mm );
          energyTotal += hit->energyDeposit/CLHEP::GeV;

      }
      std::cout << "\t" << hitNo << " hits are non-zero in collection #" << iter_coll << ": " << collect->GetName()
              << std::endl;
      std::cout << "\t" << energyTotal << " MeV = total energy stored" << std::endl;
      m_totHitEnergy->fill( energyTotal );
    }
  }
  return StatusCode::SUCCESS;
}
```

Summary

- **The CEPCSW is built upon the Gaudi and Key4hep**
  - reflecting the project's objective to leverage established HEP software, while developing innovative solutions tailored to the experiment's specific needs

- **Simulation have been developed and being used to**
  - validate the detector design and explore experiment's physics potentials

- **CEPC-on-Gaussino prototype**
  - integrates the Gaussino by only building the core part

Thank you for your attention