CEPC-on-Gaussino: an application of Gaussino simulation framework for CEPC experiment

Tao Lin,[a,*] **Weidong Li,**[a,b] **Xingtao Huang,**[c] **Teng Li,**[c] **Ziyan Deng,**[a] **Chengdong Fu**[a] **and Jiaheng Zou**[a]

[a] *Institute of High Energy Physics, Chinese Academy of Sciences, 19B Yuquan Road, Shijingshan District, Beijing, China*
[b] *University of Chinese Academy of Sciences, 19A Yuquan Road, Shijingshan District, Beijing, China*
[c] *Institute of Frontier and Interdisciplinary Science, Shandong University, Qingdao, Shandong, China*

E-mail: lintao@ihep.ac.cn

The Circular Electron Positron Collider (CEPC) is a future Higgs factory to measure the Higgs boson properties. Like the other future experiments, the simulation software plays a crucial role in CEPC for detector designs, algorithm optimization and physics studies. Due to similar requirements, the software stack from the Key4hep project has been adopted by CEPC. As the initial application of Key4hep, a simulation framework has been developed for CEPC based on DD4hep, EDM4hep and k4FWCore since 2020. However, the current simulation framework for CEPC lacks support for the parallel computing. To benefit from the multi-threading techniques, the Gaussino project from the LHCb experiment has been chosen as the next simulation framework in Key4hep. This contribution presents the application of Gaussino for CEPC. The development of the CEPC-on-Gaussino prototype will be shown and the simulation of a tracker detector will be demonstrated.

42nd International Conference on High Energy Physics (ICHEP2024)
18-24 July 2024
Prague, Czech Republic

*Speaker

1. Introduction

The Circular Electron Positron Collider (CEPC) is an e^+e^- Higgs factory designed to produce Higgs, W, Z and top quarks, with the goal of discovering new physics beyond the Standard Model. In November 2018, the CEPC Conceptual Design Report (CDR) was released [1, 2]. Following this, the CEPC Accelerator Technical Design Report (TDR) was published in in December 2023 [3]. The CEPC team is currently working on the Reference Detector TDR, which is scheduled for released in June 2025. To support the detector design and physics performance studies, the CEPC software (CEPCSW) plays a crucial role by providing simulation and reconstruction tools to the end users.

The CEPCSW is developed based on a common turnkey software stack called Key4hep [4]. The primary goal of Key4hep is to maximize the sharing of software components among different experiments. Currently, Key4hep is utilized by CEPC, CLIC [5, 6], FCC [7–9], ILC [10, 11] and other future experiments. By adopting the Key4hep software stack, CEPCSW is composed of three main parts: the underlying external libraries, the core software and the CEPC-specific applications.

The core software provides the key functionalites to the applications. The Gaudi framework [12] defines the interfaces for all software components and control their execution. EDM4hep [13] defines a generic event data model, while K4FWCore manages the event data. DD4hep [14] provides the geometry description. In addition to these software components, there are CEPC-specific software elements, such as the simulation framework. This framework includes the physics generator interface, Geant4 simulation and beam background mixing.

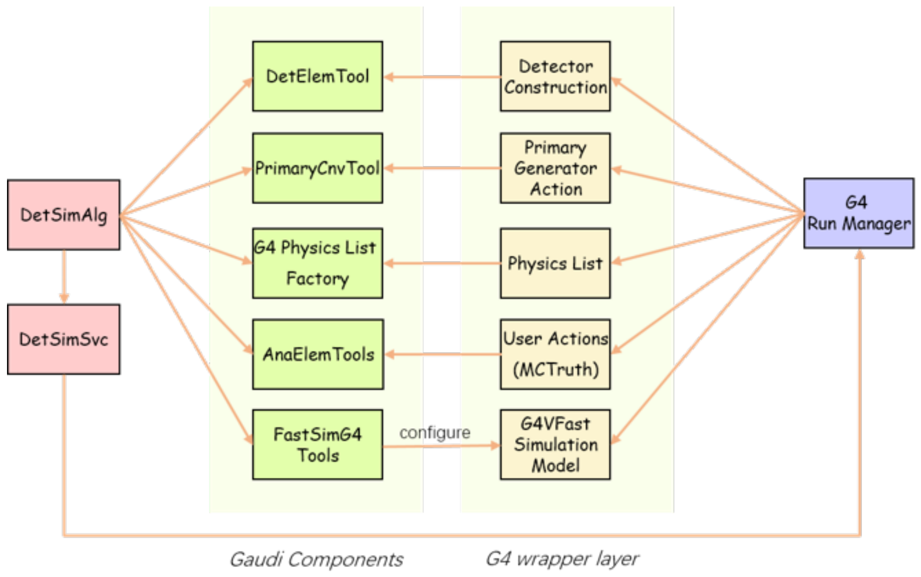
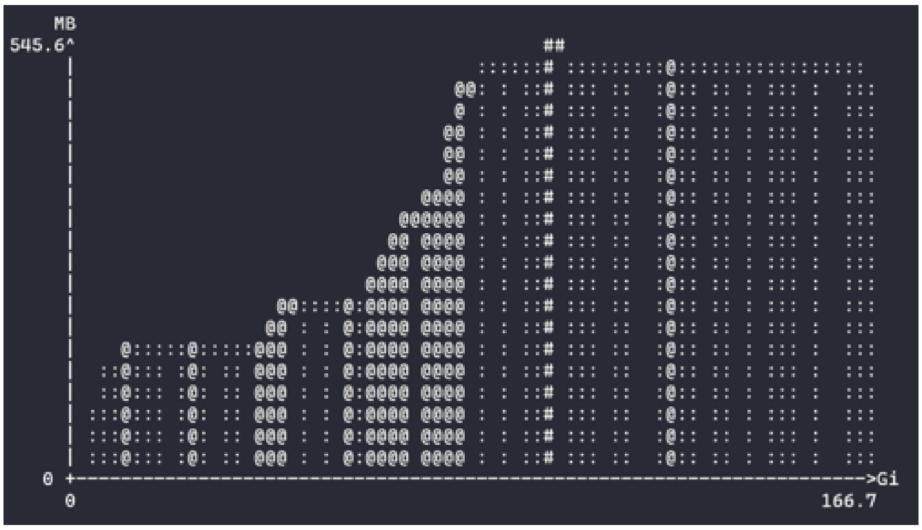


Figure 1: Design of simulation framework

Figure 1 illustrates the design of the simulation framework. The entry point is a Gaudi algorithm called DetSimAlg, which comprises a list of Gaudi tools responsible for detector construction, primary particle generation, physics list, user actions and fast simulation. Corresponding Geant4-derived classes are implemented to integrate the Gaudi tools with Geant4. These Geant4-derived classes are registered with the Geant4 Run Manager during the initialization stage. The detector construction tool and physics list tool are invoked at beginning of the event loop. During the event loop, DetSimAlg calls the Geant4 Run Manager to simulate an event, which in turn invokes the appropriate primary generator action tools and user action tools.

2. Moving to a new simulation framework

The current simulation framework does not support multi-threading. Multi-threading simulation could significantly reduce the memory usage by sharing geometries and the physics list. The memory usage of the current simulation was measured using the CEPC detector option TDR_o1_v01 and the Geant4 physics list QGSP_BERT. To measure the memory usage at initialization, 100 single muon events were simulated to minimize memory usage during the event loop. The resident set size (RSS) memory is approximately 950 MB at the initialization stage. Figure 2 shows the heap memory usage, measured with the profiler tool Valgrind Massif. According to the Massif report, most of the memory is used by geometry related components, such as the ROOT TGeo [17] geometries used by DD4hep, the Geant4 geometry converted by DDG4, and Geant4 geometry voxelization during G4GeometryManager::BuildOptimisations. Therefore, implementing multi-threaded simulation could reduce memory usage by sharing these components.



```
    MB
545.6^                                          ##
     |                                    ::::::# :::::::::@:::::::::::::::::
     |                                  @@: : ::# ::: ::  :@:: :: : ::: :  :::
     |                                  @ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                                 @@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                                 @@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                                 @@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                               @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                             @@@@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                            @@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                           @@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                          @@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                  @@::::@:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |                 @@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |    @:::::@:::::@@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |  ::@::: :@: :: @@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     |  ::@::: :@: :: @@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     | :::@::: :@: :: @@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     | :::@::: :@: :: @@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
     | :::@::: :@: :: @@@ : : @:@@@@ @@@@ : : ::# ::: ::  :@:: :: : ::: :  :::
   0 +------------------------------------------------------------------------>Gi
     0                                                                   166.7
```

Figure 2: Heap memory usage of simulation in CEPCSW

There are two options for a multi-threaded simulation framework. One option is to develop a new framework based on the current serial version. The other option is to adopt an existing framework. Gaussino [15] is a potential solution for simulation within Key4hep project [16]. Gaussino is a simulation framework from LHCb. As shown in Figure 3, the LHCb simulation framework Gauss is divided into the common parts, named Gaussino, and the LHCb dedicated parts, named the Gauss-on-Gaussino. The underlying framework is based Gaudi Functional and Gaudi Hive, which provide better support for multi-threading. Therefore, Gaussino has been chosen as the next simulation framework in CEPCSW, and the CEPC-on-Gaussino prototype has been proposed.

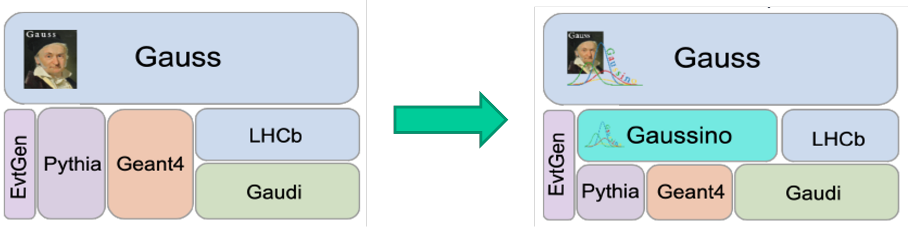


Figure 3: Evolution of LHCb simulation framework

3. CEPC-on-Gaussino prototype

One of the challenges in adopting Gaussino in CEPCSW is the existing dependencies on LHCb software, and the decoupling of these dependencies is still on going. Figure 4 shows the current software dependencies. To develop the CEPC-on-Gaussino prototype, there are three steps:

1. Using Gaussino with the full LHCb software;
2. Creating a modified version with fewer LHCb dependency;
3. Directly using the Key4hep version without LHCb dependency, which is not available at the moment.

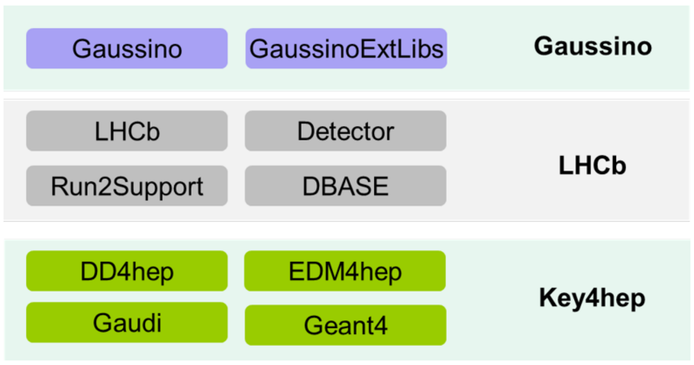
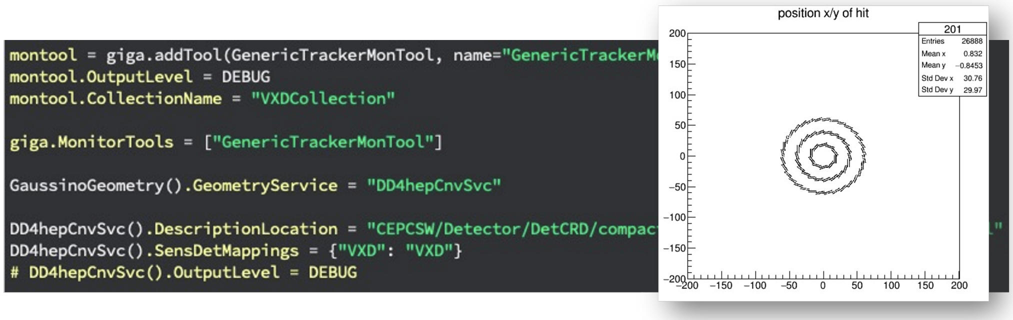


Figure 4: The dependencies of Gaussino

The first step involes using Gaussino with LHCb software dependencies. Since the external libraries are deployed at /cvmfs, building Gaussino with LHCb software installation script is relatively straightforward. During this step, only the geometries from CEPCSW are used in Gaussino, while all the underlying software is based on LHCb software. To load the CEPC tracker detector, DD4hepCnvSvc is adopted as the geometry service instead of the LHCb detector description library. As there is no detector response implemented, the Geant4 stepping verbose information are printed to verify whether the simulation works. Multi-threading is also tested with multiple cores.

The second step involves creating a modified version with fewer LHCb dependencies. During this step, all the underlying software is based on CEPC software. Gaussino and its dependencies are built as external libraries of CEPC software. Special branches in the repositories are created, and the CMakeLists.txt files are modified to build Gaussino. Only the necessary packages from LHCb software are built, such as the Event Data Model GenEvent and MCEvent. After optimization, approximately 20 shared libraries are built in total.

The detector responses are implemented for the tracker detector. Classes from DDG4 are reused with minor modifications, such as the hit class Geant4Hit and the sensitive detector base class DDG4SensitiveDetector. A concrete sensitive detector class GenericTrackerSensitiveDetector is then implemented. To integrate the sensitive detector into Gaussino, an additional Gaudi factory named GenericTrackerSensDetTool is used to create the object. A monitoring tool is also implemented to save the user output, including histograms of positions and deposit energies. In the monitoring tool, an instance of G4Event is provided at the end of each event, the hit collections are retrieved and converted to the Geant4Hit. Then these hit information are filled into the histograms. Firgue 5 shows the job options and the corresponding histogram.



```
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
```

Figure 5: The monitoring tool in job options and the histogram of positions

4. Summary

The simulation framework in CEPCSW has been developed to support the CEPC Reference Detecotor TDR. To benefit from multi-threading techniques and reduce memory usage, it is necessary to transition from a serial version to a multi-threaded version. The Gaussino project from LHCb has been chosen as the simulation framework within Key4hep. Consequently, the CEPC-on-Gaussino prototype has been proposed and implemented without dependencies on the whole LHCb software stack.

Acknowledgments

This work is supported by the NSFC Basic Science Center Program for “Joint Research on High Energy Frontier Particle Physics” (Grant No. 12188102), the Youth Innovation Promotion Association of CAS (No. 2022011), and the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No 101004761.

References

[1] CEPC Study Group collaboration, *CEPC Conceptual Design Report: Volume 1 - Accelerator*, 1809.00285.

[2] CEPC Study Group collaboration, *CEPC Conceptual Design Report: Volume 2 - Physics & Detector*, 1811.10545.

[3] CEPC Study Group collaboration, *CEPC Technical Design Report: Accelerator*, *Radiat. Detect. Technol. Methods* **8** (2024) 1 [2312.14363].

[4] Key4hep collaboration, *Key4hep: Status and Plans*, *EPJ Web Conf.* **251** (2021) 03025.

[5] M. Aicheler, P. Burrows, M. Draper, T. Garvey, P. Lebrun, K. Peach et al., eds., *A Multi-TeV Linear Collider Based on CLIC Technology: CLIC Conceptual Design Report*, .

[6] L. Linssen, A. Miyamoto, M. Stanitzki and H. Weerts, eds., *Physics and Detectors at CLIC: CLIC Conceptual Design Report*, 1202.5940.

[7] FCC collaboration, *FCC Physics Opportunities: Future Circular Collider Conceptual Design Report Volume 1*, *Eur. Phys. J. C* **79** (2019) 474.

[8] FCC collaboration, *FCC-ee: The Lepton Collider: Future Circular Collider Conceptual Design Report Volume 2*, *Eur. Phys. J. ST* **228** (2019) 261.

[9] FCC collaboration, *FCC-hh: The Hadron Collider: Future Circular Collider Conceptual Design Report Volume 3*, *Eur. Phys. J. ST* **228** (2019) 755.

[10] T. Behnke, J.E. Brau, B. Foster, J. Fuster, M. Harrison, J.M. Paterson et al., eds., *The International Linear Collider Technical Design Report - Volume 1: Executive Summary*, 1306.6327.

[11] ILC collaboration, H. Baer et al., eds., *The International Linear Collider Technical Design Report - Volume 2: Physics*, 1306.6352.

[12] G. Barrand et al., *GAUDI - A software architecture and framework for building HEP data processing applications*, *Comput. Phys. Commun.* **140** (2001) 45.

[13] F. Gaede, T. Madlener, P. Declara Fernandez, G. Ganis, B. Hegner, C. Helsens et al., *EDM4hep - a common event data model for HEP experiments*, *PoS* **ICHEP2022** (2022) 1237.

[14] M. Frank, F. Gaede, C. Grefe and P. Mato, *DD4hep: A Detector Description Toolkit for High Energy Physics Experiments*, *J. Phys. Conf. Ser.* **513** (2014) 022010.

[15] M. Mazurek, M. Clemencic and G. Corti, *Gauss and Gaussino: the LHCb simulation software and its new experiment agnostic core framework*, *PoS* **ICHEP2022** (2022) 225.

[16] Key4hep collaboration, *Key4hep: Progress Report on Integrations*, *EPJ Web Conf.* **295** (2024) 05010 [2312.08152].

[17] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, *Nucl. Instrum. Meth. A* **389** (1997) 81.