Status of CEPC Offline Software

Wenxing Fang, Xingtao Huang, Teng Li, Weidong Li,
Tao Lin, Jiaheng Zou

Feb. 13, 2023

IAS Program on High Energy Physics (HEP 2023)

Contents

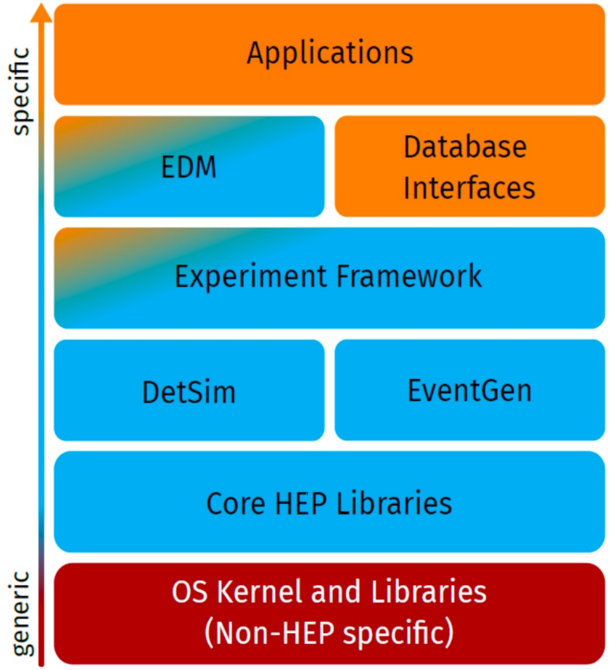
- Introduction
- Overview of CEPCSW
- Status and new development of CEPCSW
- Summary

Introduction

- The CEPC software development first started with the iLCSoft
  - Reused most software modules: Marlin, LCIO, MokkaC, Gear
  - Developed its own software components for simulation and reconstruction
  - Massive M.C. data produced for detector and physics potential studies
  - CDR was released in Nov, 2018, based on results from the iLCSoft
- A new CEPC software (CEPCSW) prototype was proposed at the Oxford workshop in April 2019
- The consensus among CEPC, CLIC, FCC, ILC and other future experiments was reached at the Bologna workshop in June 2019
  - Develop a Common Turnkey Software Stack (Key4hep) for future collider experiments
  - Maximize the sharing of software components among different experiments
- Further discussion on development of Key4hep at the 'Future Software Implementations' session of the IAS program in Jan. 2020

Key4hep

- HEP software usually consist of lots of applications
  - Application layer of modules/algorithms /processors performing physics task (*PandoraPFA, FastJet, ACTS,...*)
  - Data access and representation layer including EDM
  - Experiment core orchestration layer (*Gaudi, Marlin, ...*)
  - Specific components reused by many experiments (*DD4hep, Delphes, Pythia,...*)
  - Commonly used HEP core libraries (*ROOT, Geant4, CLHEP, ...*)
  - Commonly used tools and libraries (*Python, CMake, boost,...)*



Thomas Madlener, Epiphany Conference 2021

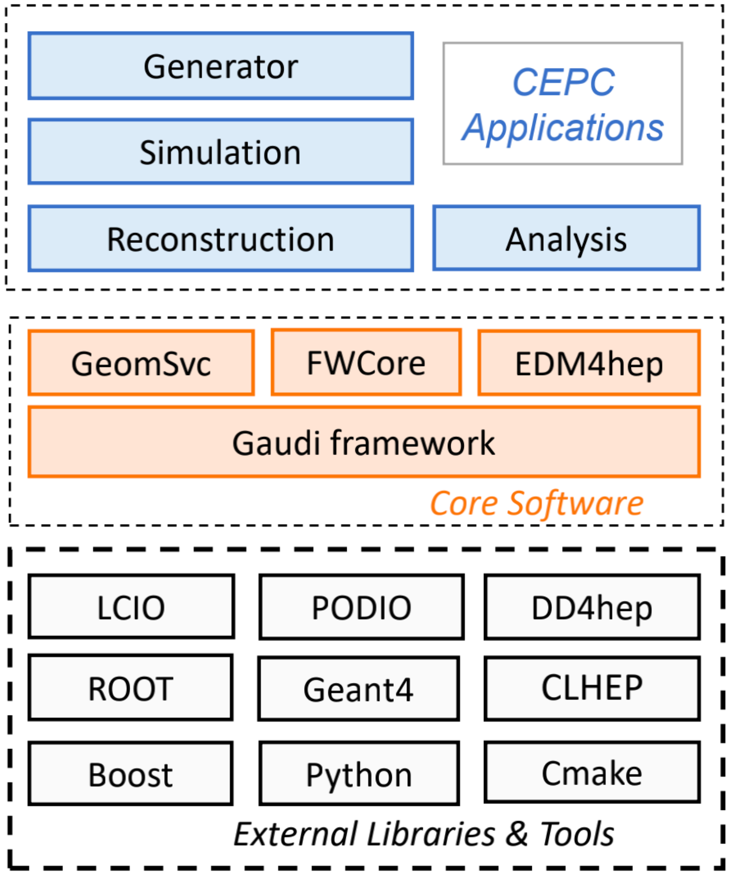
- CEPCSW is being fully integrated with Key4hep to share software with other future experiments
- IHEP and SDU are also involved in Key4hep development as non-EU members

Overview of CEPCSW

❖ CEPCSW software structure

- Core software
- Applications: simulation, reconstruction and analysis (see talks given by Weidong and Shengshen)
- External libraries

https://github.com/cepc/CEPCSW



❖ Core software

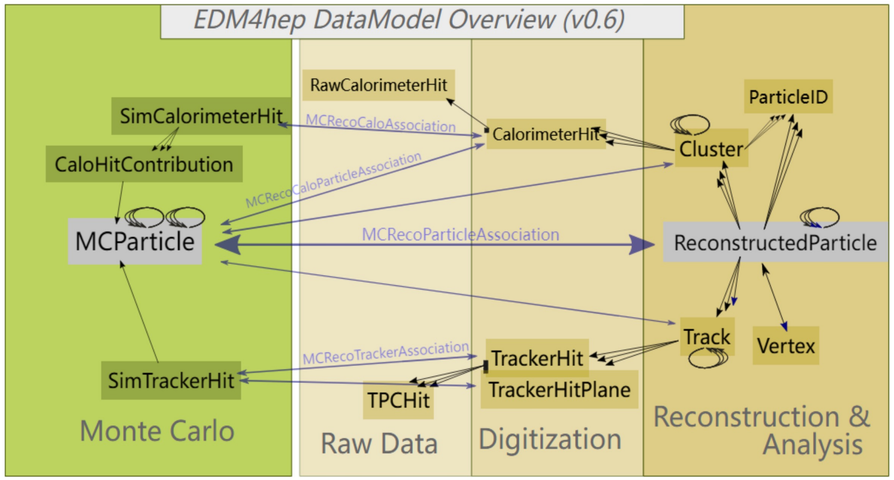
- Gaudi/Gaudi Hive: defines interfaces to all software components and controls their execution.
- EDM4hep: generic event data model
- K4FWCore: manages the event data
- DD4hep: geometry description
- CEPC-specific framework software: generator, Geant4 simulation, beam background mixing, fast simulation, machine learning interface, etc.

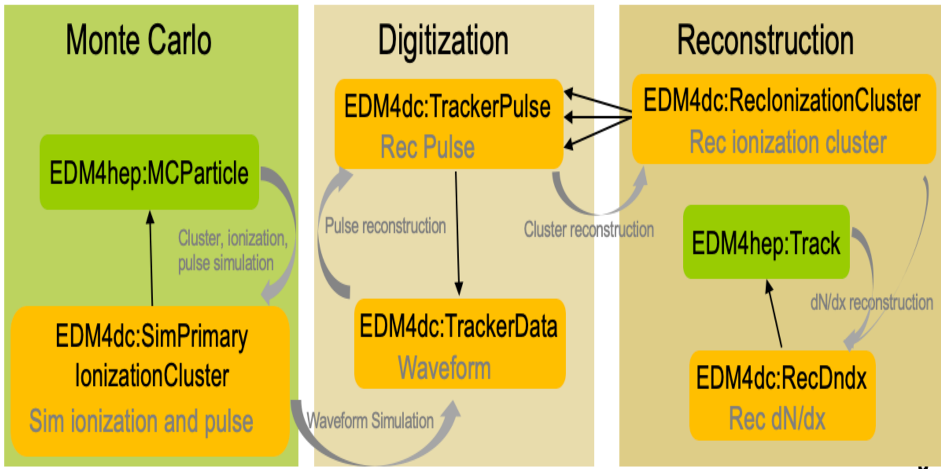
Status of CEPCSW

- CEPCSW is under rapid development, and its latest version is v0.2.6
  - Well supported detector simulation and reconstruction studies on the 4th conceptual detector
- Lots of progress has been made on core software of CEPCSW since last IAS meeting
  - Optimizations on key components according to application requirements
    - Event Data Model
    - Detector Description
    - Simulation Framework
  - Developments on adopting new technologies to boost CEPCSW performance
    - Multi-threaded Detector simulation
    - Heterogeneous Computing
    - Machine Learning Integration based on ONNX
    - Analysis framework based on RDataframe
    - Automated Validation System

Event Data Model

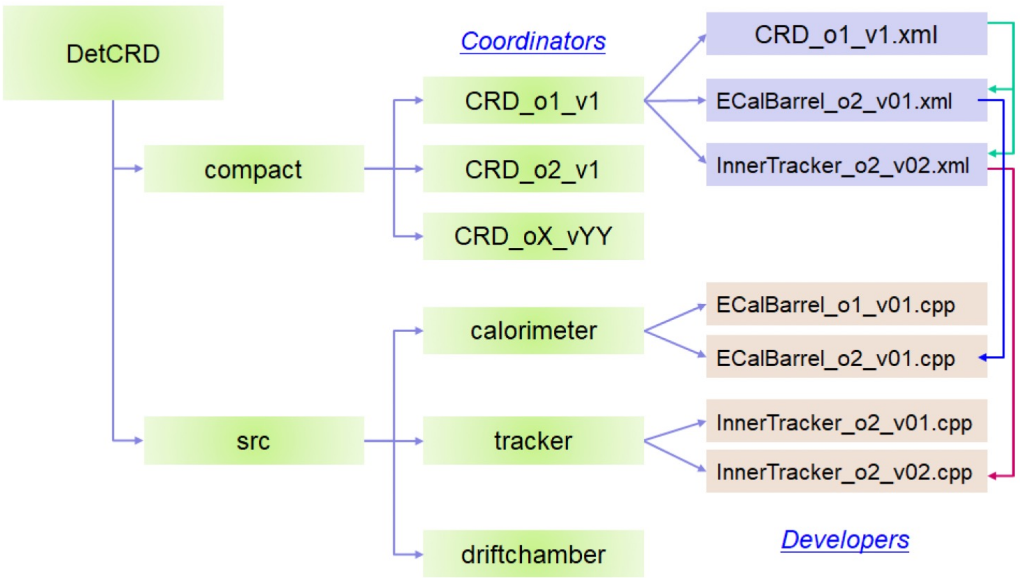
- EDM of CEPCSW is adopted from EDM4hep
  - In different data processing stages
  - For different sub-detectors
- Extension of EDM4hep is developed to accommodate the drift chamber dN/dx study
  - Also can be used for TPC detector
  - Will be merged into EDM4hep soon ([PR]())

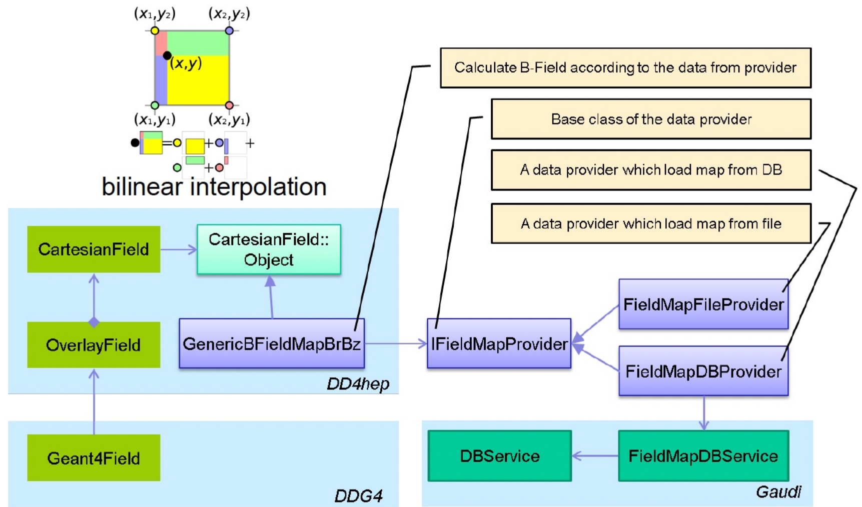




Detector Description

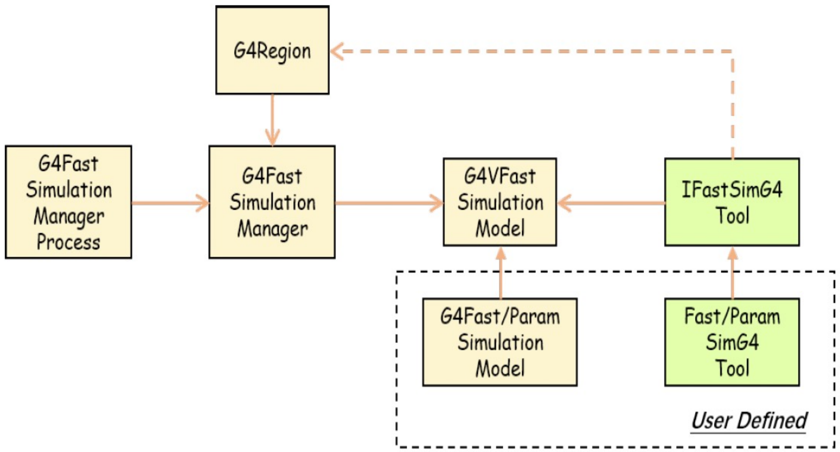
- DD4hep is adopted to provide a full detector description with a single source of information
- Different detector design options are managed in git repository and easily to be changed in CEPCSW
- The non-uniform magnetic field has also been implemented in CEPCSW

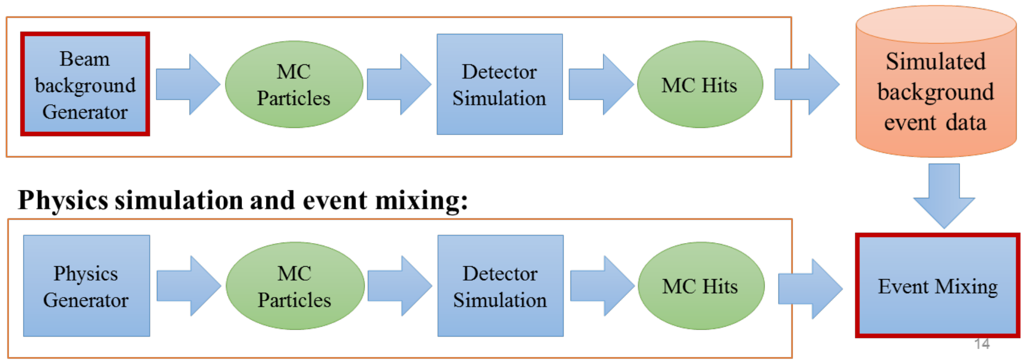




Detector Simulation

- The Geant4-based full detector simulation framework has been developed in CEPCSW and supported sub-detectors simulations and their performances study
  - including silicon detectors, time projection chamber, drift chamber and calorimeters.
- The region-based fast simulation interface is also developed to integrate different fast simulation models into Geant4.
- CEPCSW provides an unified solution for different backgrounds' simulation and event mixing at the hit level

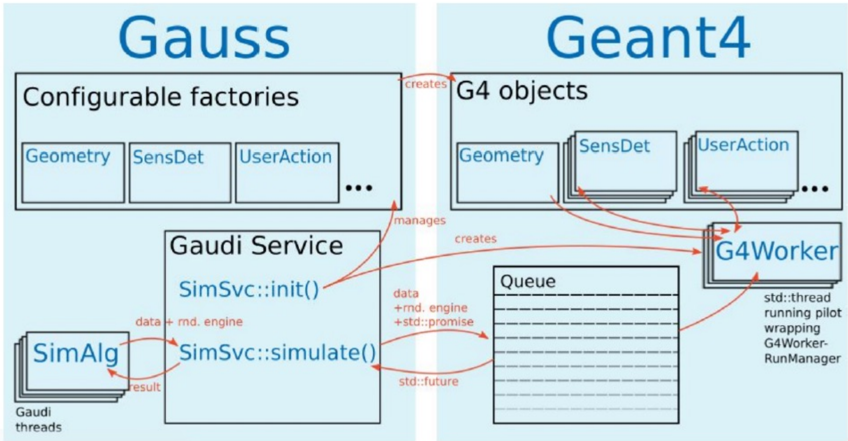




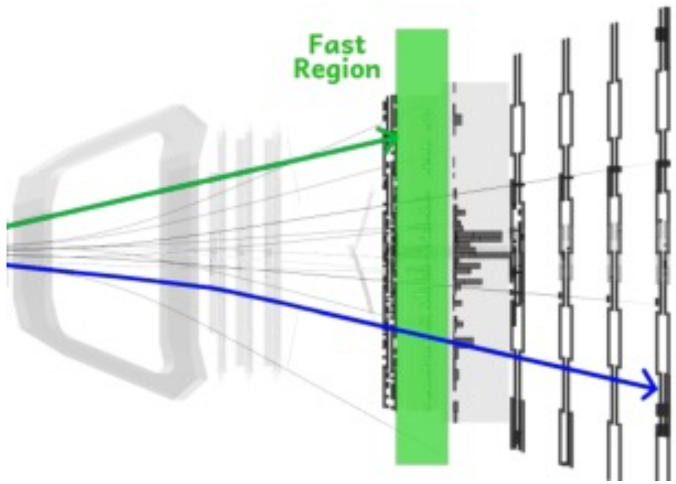
Gaussino: a new experiment-independent simulation framework

Ref: [See Talk given by Michał Mazurek, Gauss and Gaussino, ICHEP 2022]

- Developed based on the LHCb simulation framework (Gauss)
  - Well Integrated multi-threaded functions of Gaudi and Geant4
  - Achieved good scalability

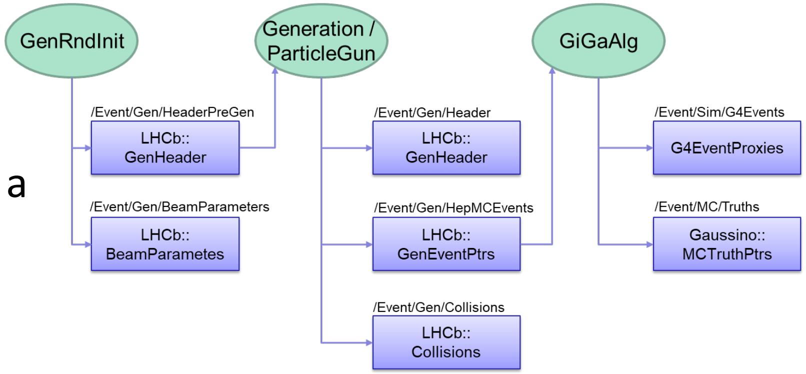


- Key Features:
  - Multi-threaded interfaces to Event generator (Pythia8) and detector simulation (Geant4MT)
    - Supports parallel execution of multiple events at the same time as well as for parallelism within a single event.
  - Dedicated fast simulation Interface to invoke fast simulations for a given detector



CEPC detector simulation based on Gaussino

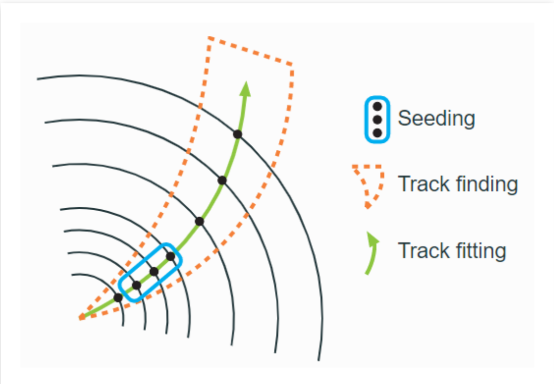
- Application of Gaussino to CEPC detector simulation is under going
  - Idea: reuse the existing code and use Gaussino as a black box.
  - Challenge: how to integrate the existing CEPC detector geometry.

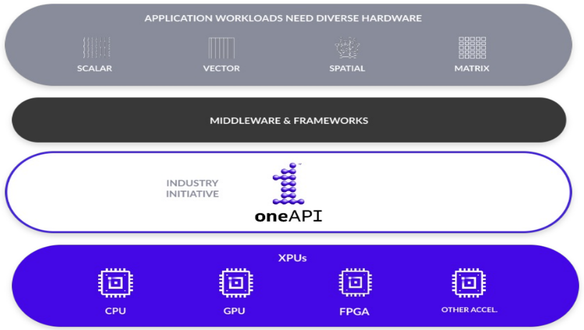


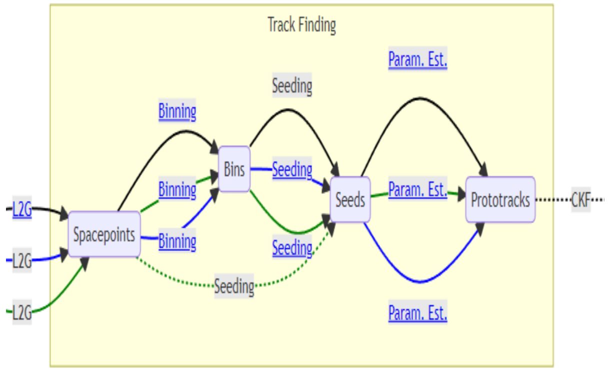
  - Two possible methods:
    - Method 1:
      - DD4hepDetectorConstructionFAC decides which object will be created for detector construction
      - DD4hepDetectorConstruction is responsible to construct CEPC detector with DD4hep
    - Method 2:
      - GiGaMTDetectorConstructionFAC creates geometry by invoking GeoSvc
      - DD4hepCnvSvc will be used to create geometry with DD4hep
  - A prototype for CEPC detector simulation with the method 1 is under developing

Heterogeneous Computing

- Utilizing heterogeneous resource is one of possible ways to copy with the increasing HEP data processing/analysis.
- Lots of efforts has been devoted on heterogeneous computing, for example
  - TRACCC is a project below ACTS to demonstrate tracking chain on different kinds of computing hardware (CPU/GPU/FPGA).
- The strategies:
  - SYCL enables the definition of data parallel functions by providing required APIs and runtime libraries
  - OneAPI can provide a unified programming model and enables code reuse across heterogeneous devices (CPU/GPU/FPGA)







https://github.com/acts-project/traccc

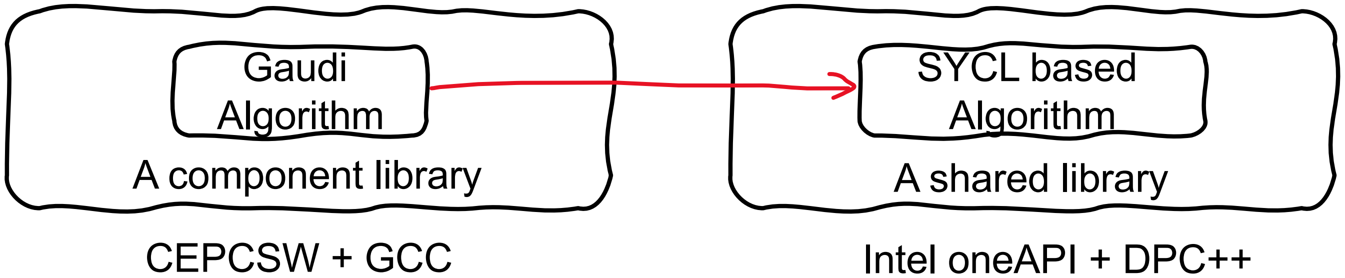
Heterogeneous Computing

Activities in CEPCSW

- We are able to run TRACCC in a standalone environment and managed to build/run TRACCC on both CPU/GPU.

Config	Hardware	OS	Compiler	SYCL backend	Bulid traccc	Run traccc
1	Intel CPU (IHEP login node)	CentOS 7.8	LCG 101 (GCC 10.3 + clang 12) + oneAPI DPC++	CPU	OK	OK
2	Intel CPU + NVIDIA RTX 8000 (workstation)	CentOS 7.9	LCG 101 (GCC 11.1) + intel/llvm (2021-12)	CUDA 11.2	OK	OK

- Now the TRACCC seeding algorithm has been integrated within CEPCSW by developing middleware between Gaudi algorithm and SYCL based algorithm.



Machine Learning Integration

- Machine Learning becomes more and more important in HEP data processing
  - Different tasks may use different Machine learning libraries and produce different models
  - We need an unified way to integrate different models in CEPCSW and run inference easily
- ONNX is an open format built to represent machine learning models.
  - Support to convert from other models to ONNX, such as Tensorflow, PyTorch etc.
  - Easy to run inference on different platforms, such as ONNX Runtime, ONNX MLIR etc.
  - Some applications of ONNX in HEP
    - Fast simulation in Geant4 using ONNX inference interface [1]
    - Fast Inference for Machine Learning in ROOT TMVA [2]
- ONNX Runtime is a cross-platform inference and training accelerator
  - Accelerate inference on different hardware platform (CPUs/GPU/FPGA)

[1] Anna Zaborowska *et al.*, Fast Simulation : from Classical to Machine Learning Models
[2] Sitong An et al., Fast Inference for Machine Learning in ROOT/TMVA

Machine Learning Integration

- ONNX/ONNX Runtime have been integrated with CEPCSW
- Provided an example, OrtInferenceAlg,
  - In initialize()
    - Create a session object of ONNX runtime
    - Load and run an ONNX model
  - In execute()
    - Compute output for an input data

```
Ort::MemoryInfo info("Cpu", OrtDeviceAllocator, 0, OrtMemTypeDefault);

auto input_tensor = Ort::Value::CreateTensor(info,
                                             inputs.data(),
                                             inputs.size(),
                                             dims.data(),
                                             dims.size());
std::vector<Ort::Value> input_tensors;
input_tensors.push_back(std::move(input_tensor));

auto output_tensors = m_session->Run(Ort::RunOptions{ nullptr },
                                     m_input_node_names.data(),
                                     input_tensors.data(),
                                     input_tensors.size(),
                                     m_output_node_names.data(),
                                     m_output_node_names.size());

for (int i = 0; i < output_tensors.size(); ++i) {
    LogInfo << "[" << i << "]"
            << " output name: " << m_output_node_names[i]
            << " results (first 10 elements): "
            << std::endl;
    const auto& output_tensor = output_tensors[i];
    const float* v_output = output_tensor.GetTensorData<float>();

    for (int j = 0; j < 10; ++j) {
        LogInfo << "[" << i << "]" << "[" << j << "] "
                << v_output[j]
                << std::endl;
    }
}
```

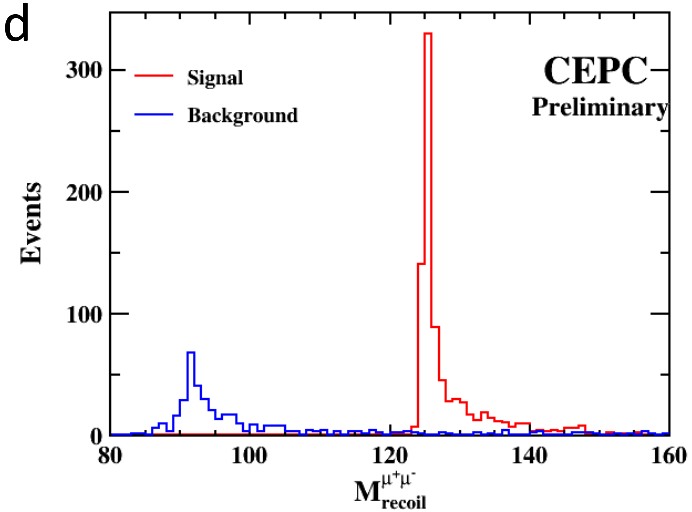
```
bool OrtInferenceAlg::initialize() {

   m_env = std::make_shared<Ort::Env>(ORT_LOGGING_LEVEL_WARNING, "ENV");
   m_seesion_options = std::make_shared<Ort::SessionOptions>();
   m_seesion_options->SetIntraOpNumThreads(m_intra_op_nthreads);
   m_seesion_options->SetInterOpNumThreads(m_inter_op_nthreads);

   m_session = std::make_shared<Ort::Session>(*m_env, m_model_file.c_str(), *m_seesion_options);
```

Analysis toolkit based on RDataFrame

- Developing a new toolkit based on new technologies of software and hardware is very crucial to rapidly analyze drastically increasing data
- RDataFrame provides powerful and flexible way analyzing data
  - Support declarative programming and parallel workflow
  - Support analysis in both Python and C++
  - Already support reading EDM4hep root files
  - Actively used by FCC-ee for flavour, higgs and top physics
- Development of analysis tool for CEPC
  - Large data samples have been produced with Marlin for CDR in LCIO format
  - Use K4LCIOReader to generate EDM4hep data from LCIO data
  - Developed common components (functions) for analyzing EDM4hep data
    - Analysis functions in C++: event selection, filtering, producing ROOT n-tuples, etc.
    - Python for configuration: define analysis functions, input samples, output variables, etc.

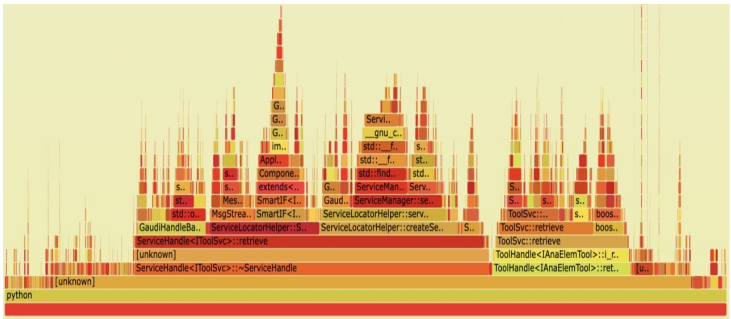
Analysis toolkit based on RDataFrame

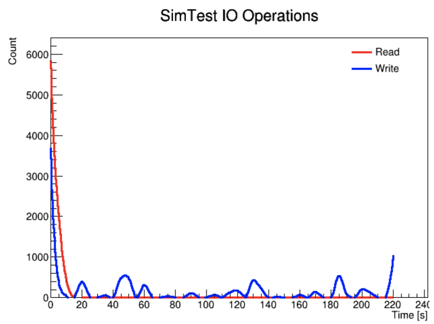
- Started with Higgs recoil analysis in e+e- -> Z H and Z-> **μμ**
  - Basic functionalities are tested: same results obtained from Marlin and RDataframe
  - Multi-threading Performance testing shows that RDataframe has good scalability
- Key analysis functions are being implemented
  - PID
  - Kinematic fitting
  - Vertexing

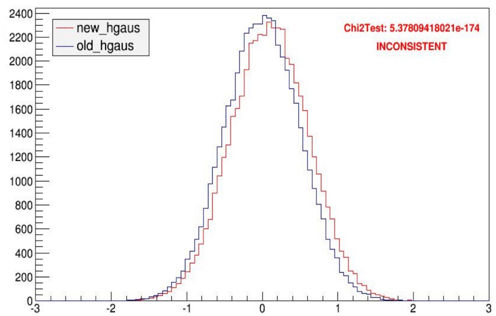


Automated Validation System

- An automated validation system is developed for software validation at different levels
  - Unit test, integrated test, performance profiling, physical validation etc.
- A toolkit is developed for building software validation workflow
  - Provide interfaces to define and run unit tests
  - Provide toolkit for performance profiling
  - Support results validation based on statistical methods
- Automated physical validation system based on massive data production (run via DIRAC resource) is being developed

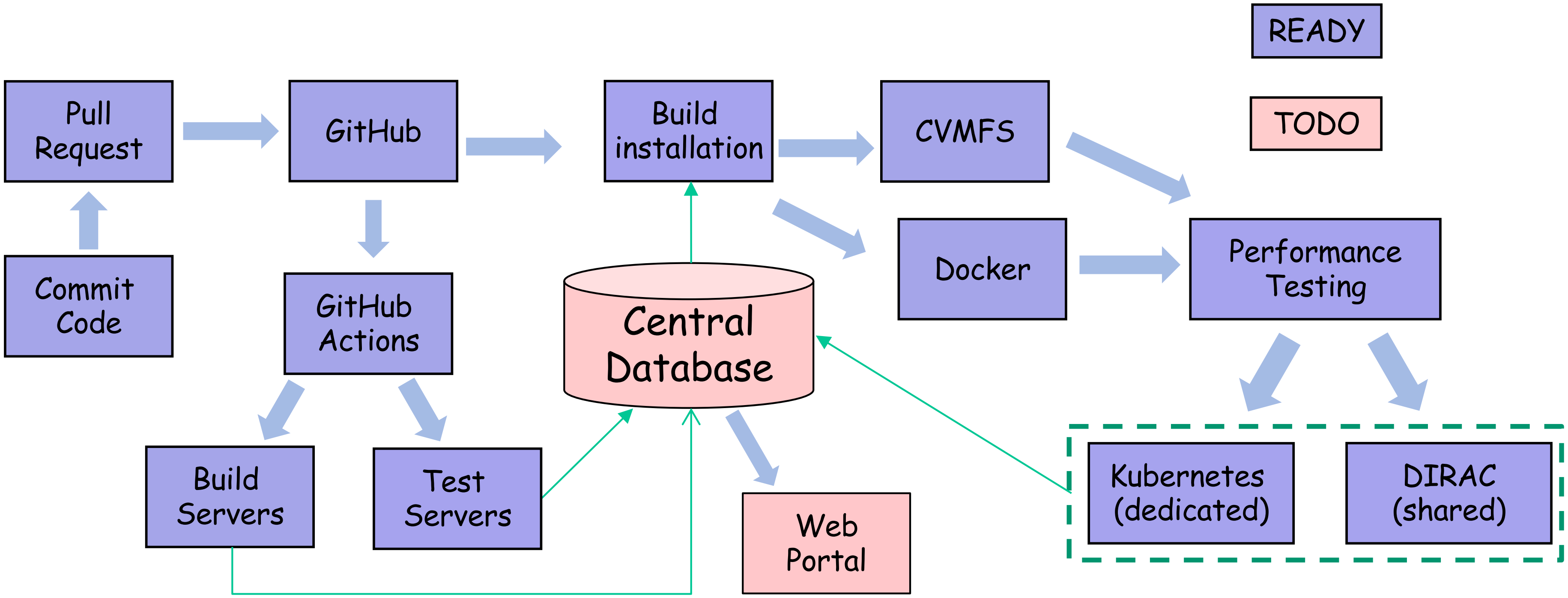






Automated Validation System

- The validation system is integrated with the Github Action system
  - Full validation workflow can be triggered by commit/merge-request
  - A web-based monitoring dashboard is also being developed
- ~ O(200) cores are now available for running validation jobs



Summary

- CEPCSW is being developed in collaboration with the Key4hep project
- Key components of the CEPCSW core software are in place and keeps optimized to well support detector simulation and reconstruction studies
- Lots of efforts are devoted to adopt new technologies to boost CEPCSW performance
  - Multi-threaded Detector simulation based on Gaussino
  - Heterogeneous computing
  - Integration of Machine Learning
  - Parallel Analysis framework based RDataFrame
  - Automated validation system

Thanks for your attention!

Welcomed to joining CEPCSW and working together!

https://github.com/cepc/cepcsw