Status of CEPCSW

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

July 5, 2023

The 2023 International Workshop on CEPC (EU Edition)

Contents

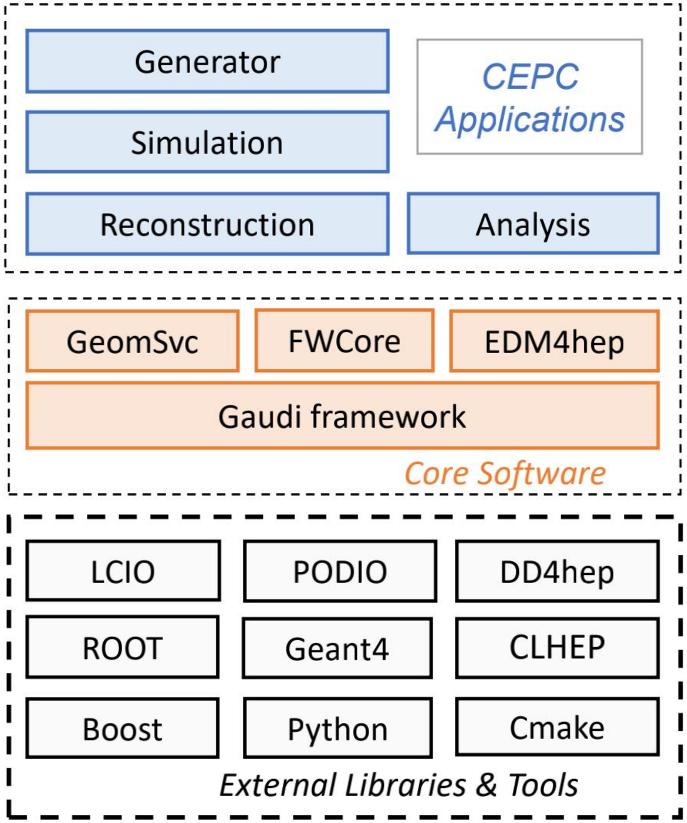
Introduction

- The CEPC software development first started with the iLCSoft
  - Reused most software modules: Marlin, LCIO, MokkaC, Gear
  - Developed CEPC's 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
- 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

Overview of CEPCSW

- CEPCSW software structure
  - Core software
  - Applications: simulation, reconstruction and analysis
  - External libraries

https://github.com/cepc/CEPCSW



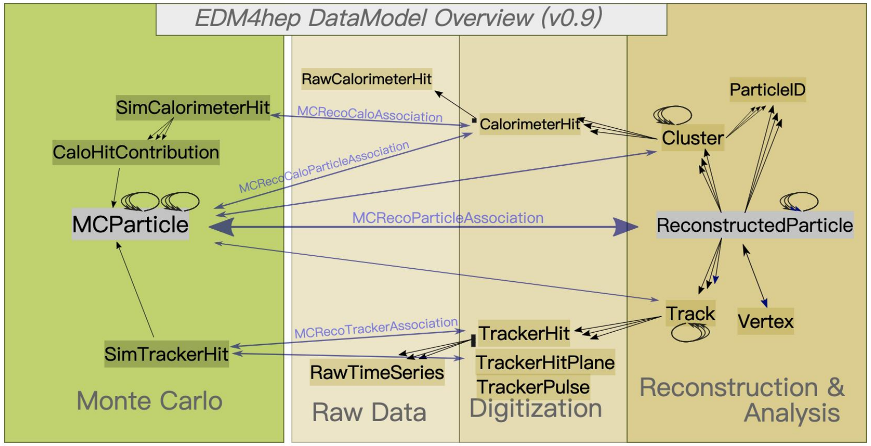
- Core software
  - Gaudi/Gaudi Hive: defines interfaces to all software components and controls their execution
  - EDM4hep: generic event data model
  - K4FWCore: manages the event data
  - DD4hep: geometry description
  - CEPC-specific framework software: generator, Geant4 simulation, beam background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

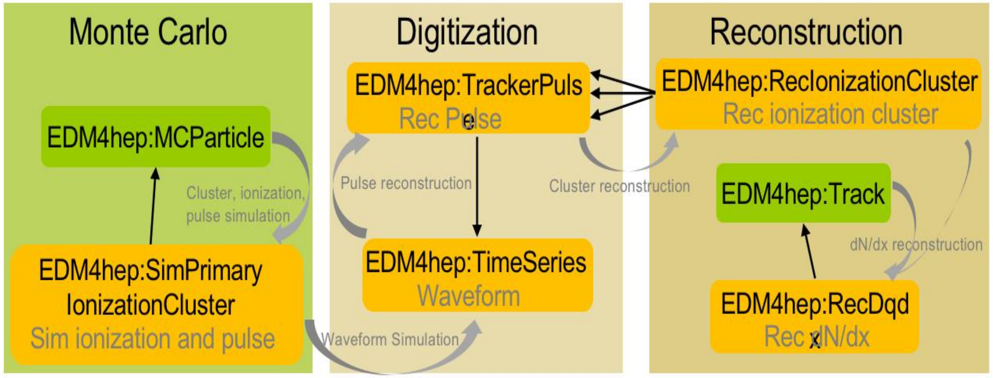
Status of CEPCSW

- 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 workshop
  - Optimization 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

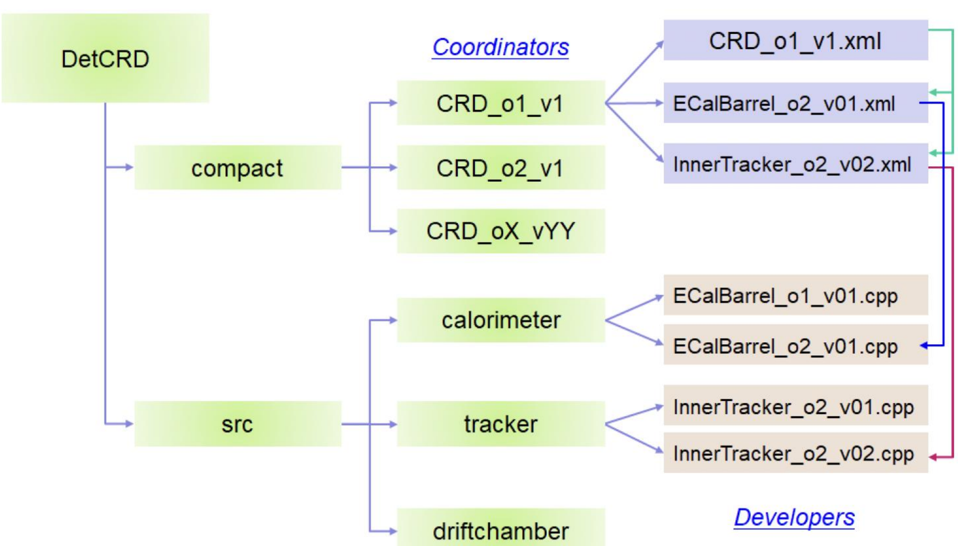
- EDM of CEPCSW is adopted from EDM4hep
  - In different data processing stages and for different sub-detectors
- Extension of EDM4hep is developed to accommodate the drift chamber dN/dx study
  - Based on the upstream mechanism of podio
  - Can also be used for TPC detector
  - Adopted by EDM4hep (PR)

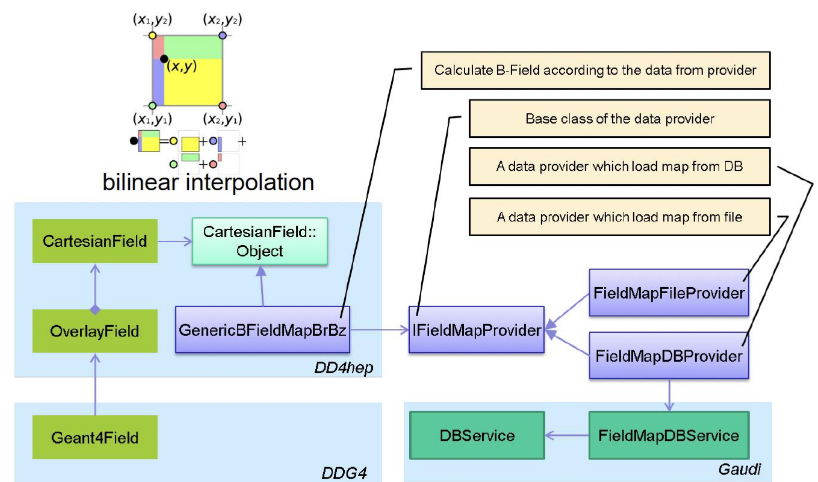




Detector Description

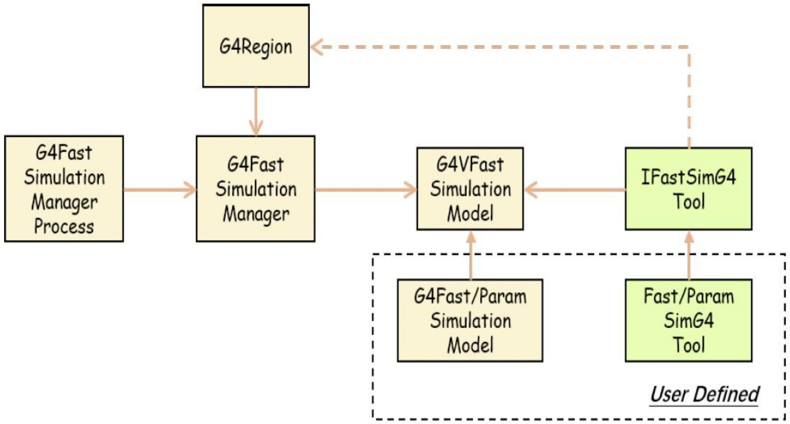
- DD4hep is adopted to provide a full detector description with a single source of information
- Different detector design options are managed in git repository and easily to be changed in CEPCSW
- The non-uniform magnetic field has been implemented

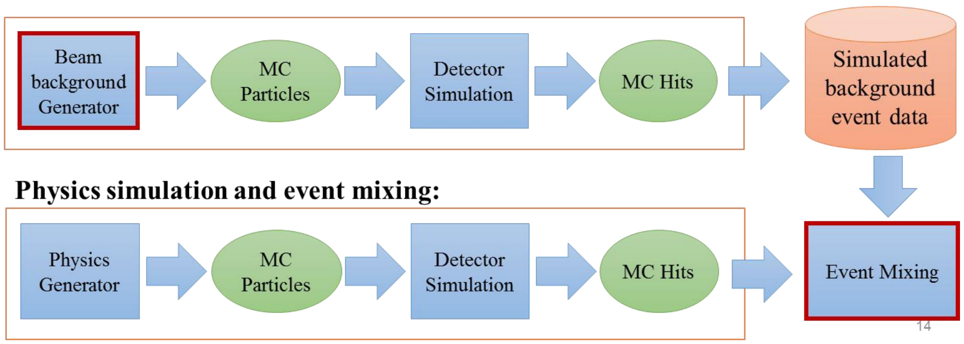




Detector Simulation (1)

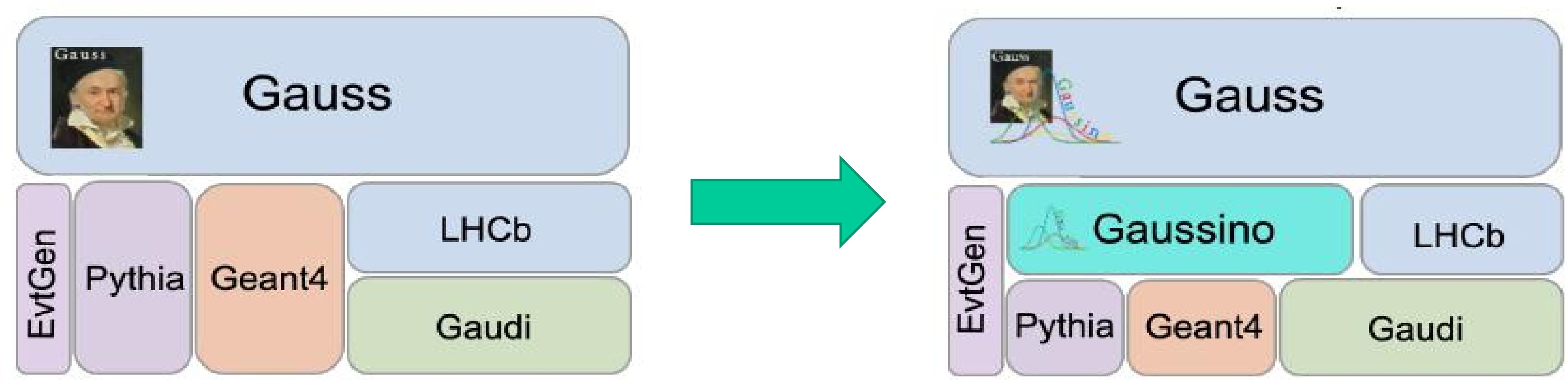
- The Geant4-based full detector simulation framework has been developed in CEPCSW and supported sub-detectors simulations and their performances study
  - Silicon detector, time projection chamber, drift chamber and calorimeters
- The region-based fast simulation interface is also developed to integrate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





Detector Simulation (2)

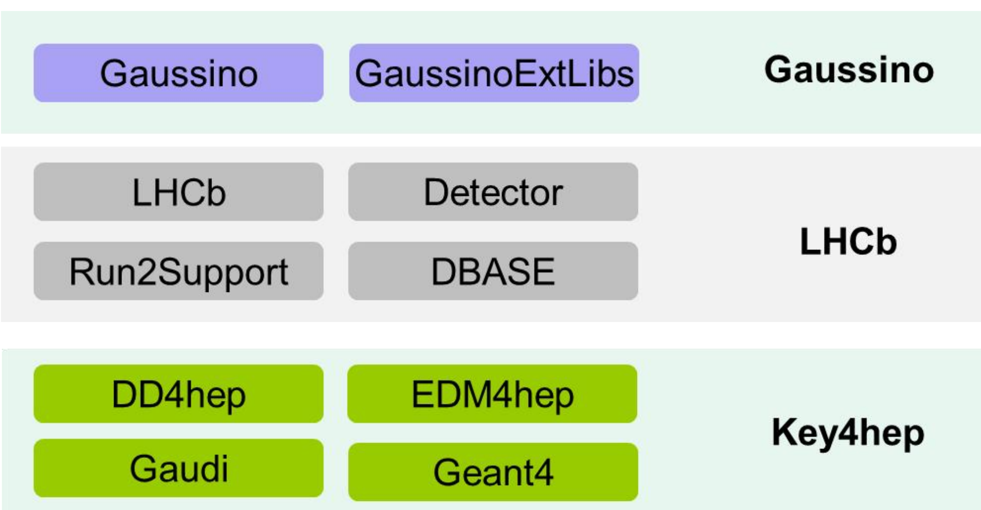
- CEPC is working with Key4hep project members re-implementing the detector simulation software based on Gaussino
- Gauss->Gaussino: evolution of the simulation framework from LHCb
  - Better support for multi-threading, machine learning and fast simulation models
  - Gauss-on-Gaussino is a new version of LHCb simulation framework

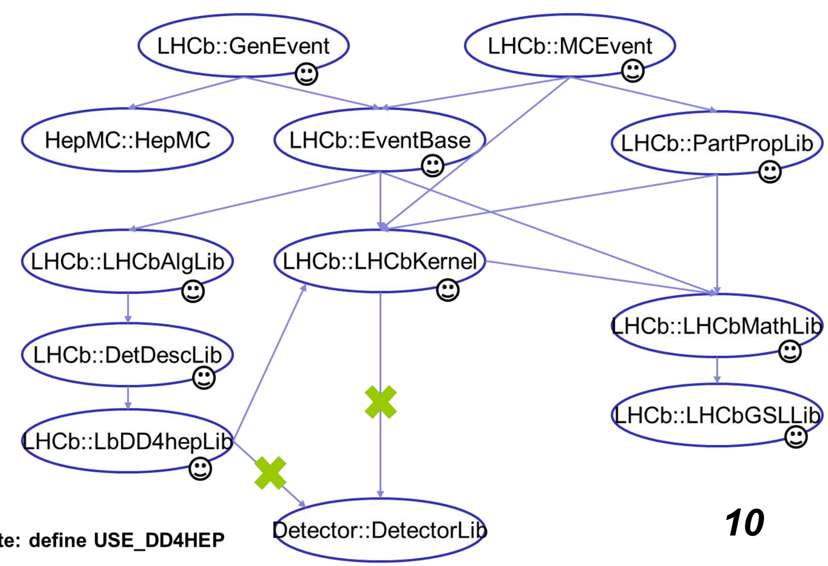


- Gaussino is being added to Key4hep by extracting experiment-independent parts from Gauss

Detector Simulation (3)

- Now Gaussino still depends on LHCb software and can not be used by other experiments directly
- Development of CEPC-on-Gaussino was planned with the following three steps
  - Using the original version having the dependency on the LHCb software
  - Creating the modified version in which the LHCb dependency is removed
  - Directly using the Key4hep version (not available at the moment)





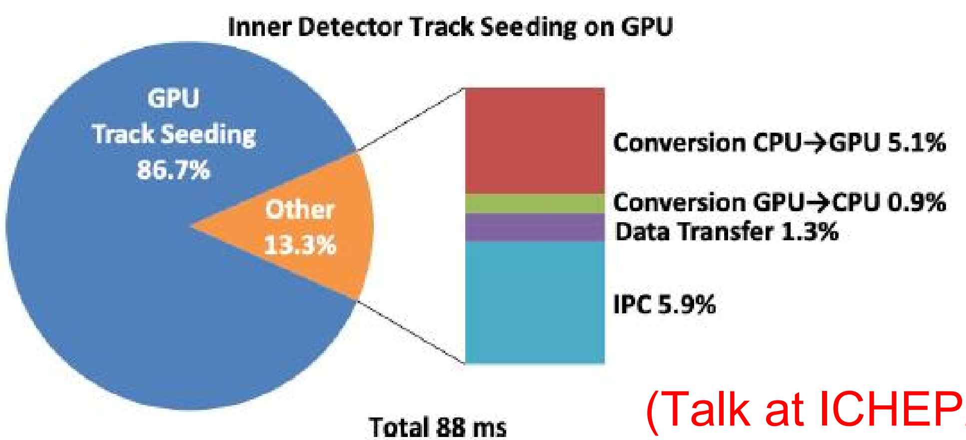
Note: define USE_DD4HEP

Heterogeneous Computing (1)

GPU-based demonstrator

- Provided a **summary of GPU results** in TDAQ TDR that demonstrate the potential of GPUs
  - Uses current ID system and μ=46 samples
- The most computationally intensive data preparation and track-seeding stages
- Overheads for data conversion, communication between processes and not having every stage moved to GPU limits potential gains.
- TDR found that using GPUs would provide the same cost/benefit as adding more CPUs, but this is already a **demonstration of feasibility**

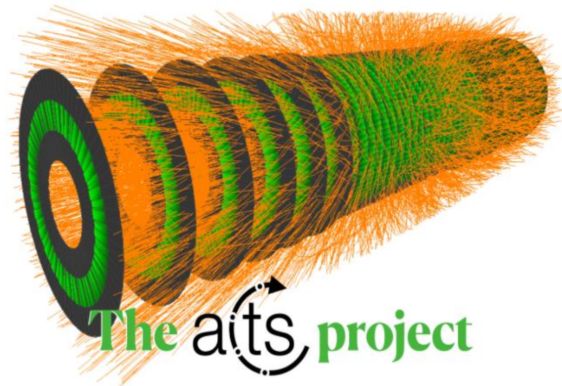
Minimizing data format conversions critical

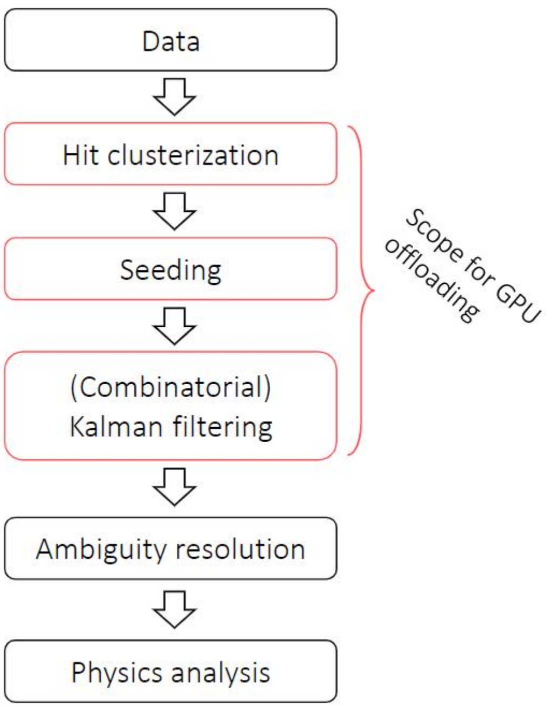


(Talk at ICHEP2022)

Heterogeneous Computing (2)

- TRACCC: one of ACTS R&D projects
  - Full chain demonstrator for track reconstruction on CPU/GPU





Category	Algorithms	CPU	CUDA	SYCL	Futhark
Clusterization	CCL	✓	✓	✓	✓
	Measurement creation	✓	✓	✓	✓
	Spacepoint formation	✓	✓	✓	○
Track finding	Spacepoint binning	✓	✓	✓	○
	Seed finding	✓	✓	✓	○
	Track param estimation	✓	✓	✓	○
	Combinatorial KF	●	●	○	○
Track fitting	KF	✓	✓	✓	○

✓: exists, ●: work started, ○: work not started yet

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

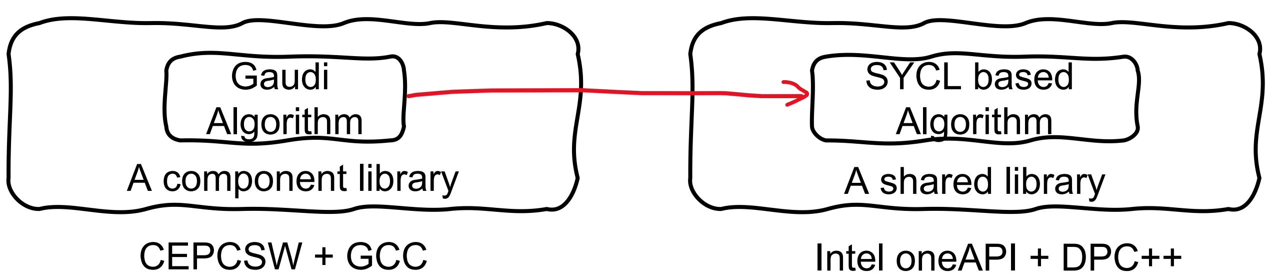
Heterogeneous Computing (3)

❖ Activities in CEPCSW

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

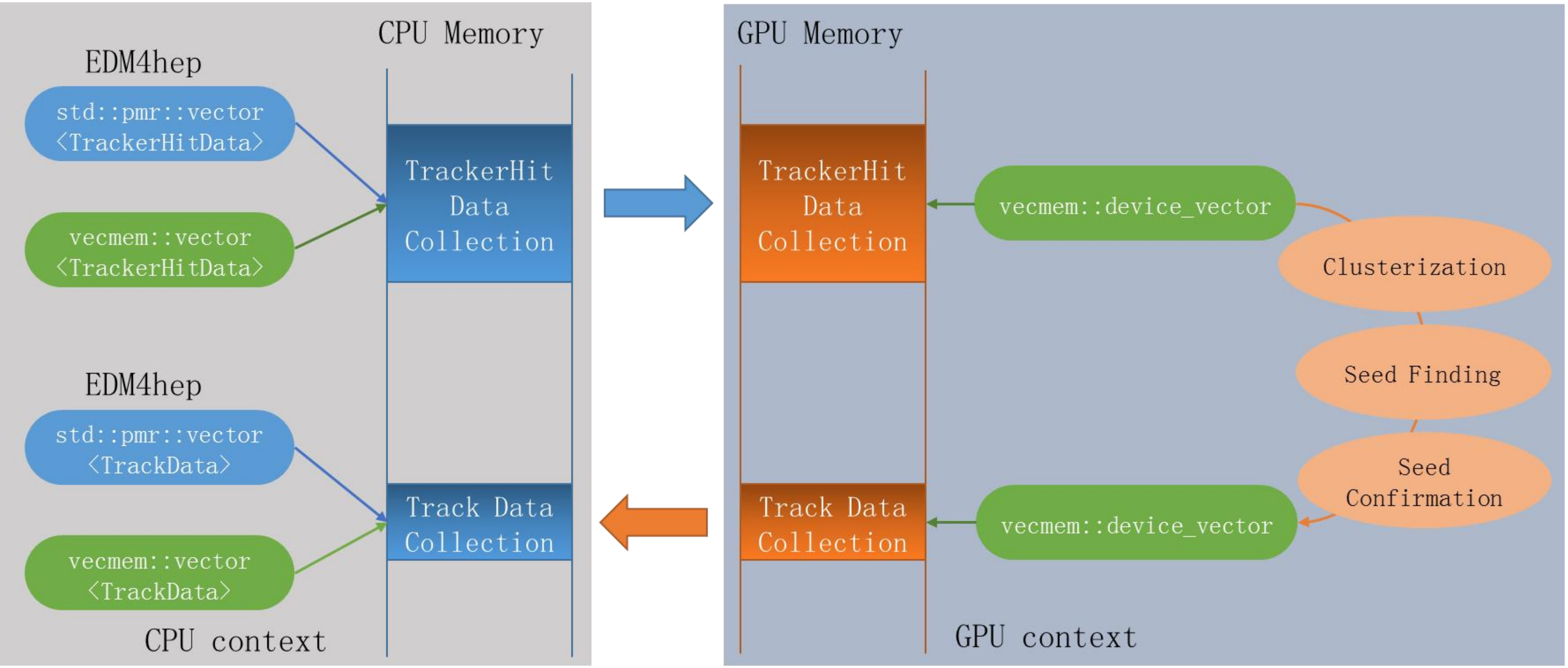
Config	Hardware	OS	Compiler	SYCL backend	Bulid traccc	Run traccc
1	Intel CPU (IHEP login node)	CentOS 7.8	LCG 101 (GCC 10.3 + clang 12) + oneAPI DPC++	CPU	OK	OK
2	Intel CPU + NVIDIA RTX 8000 (workstation)	CentOS 7.9	LCG 101 (GCC 11.1) + intel/llvm (2021-12)	CUDA 11.2	OK	OK

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



Heterogeneous Computing (4)

- Building a bridge between EDM4hep and TRACCC
  - Common memory for both EDM4hep and TRACCC
  - No data conversion is needed between them



Machine Learning Integration

- ONNX/ONNX Runtime have been integrated with CEPCSW
- Provided an example, OrtInferenceAlg,
  - During initialization
    - Create a session object of ONNX runtime
    - Load and run an ONNX model
  - During execution
    - Compute output for an input data
- Fast pulse simulation in the drift chamber provided as an example (MLP)

```
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;
    }
}
```

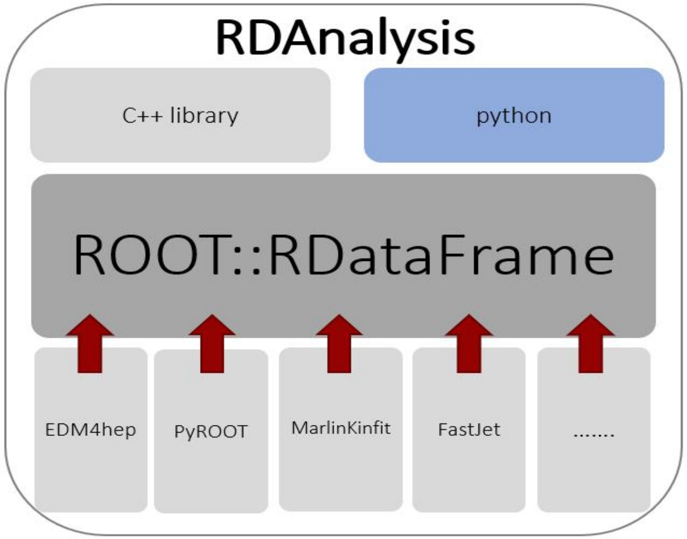
```
bool OrtInferenceAlg::initialize() {

    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);
```

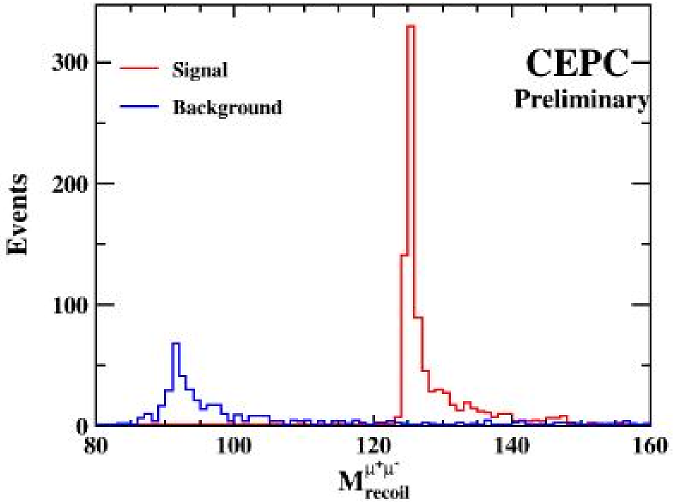
Analysis toolkit based on RDataFrame

- 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
  - Declarative programming and parallel workflow
  - Analysis in both Python and C++
  - Already support reading EDM4hep format
  - Actively used by FCC-ee for flavour, higgs and top physics
- Development and test of analysis tool for CEPCSW
  - Develop and common components (functions) for analyzing EDM4hep data
    - Analysis functions in C++: event selection, filtering, Jet clustering, vertex fitting ..
    - Python for configuration: define analysis functions, input samples, output variables ..
  - Test multithreading performance using analysis within CEPCSW

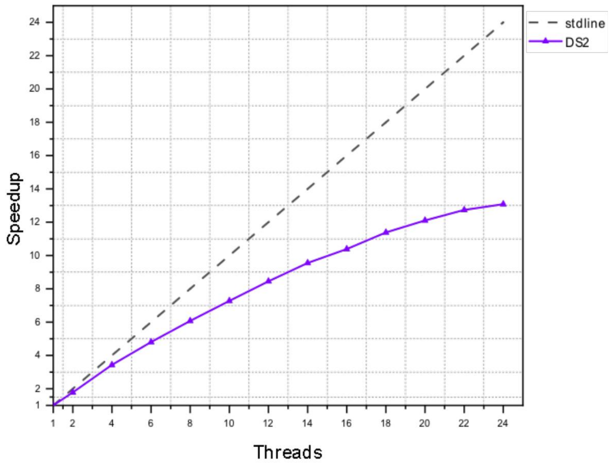
Analysis toolkit based on RDataFrame

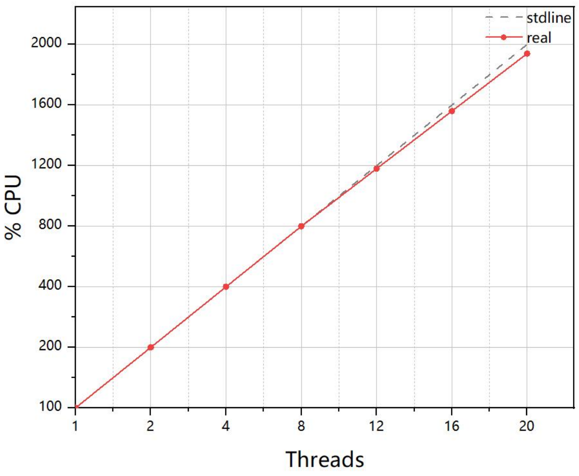
- Several packages are ported from FCC analysis, more are being implemented
  - FastJet, MarlinKinfit
  - Vertex fit, jet tag, PID etc.
- Functionalities and performance test performed on two analysis channels
  - e+e- -> Z(mumu)H
  - e+e- ->H(2jet) mumu





Identical results with Marlin

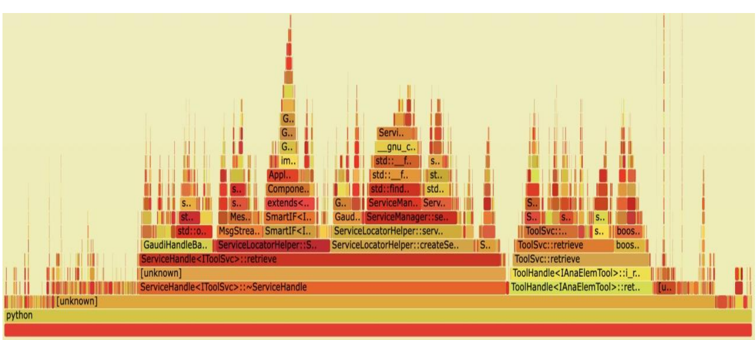


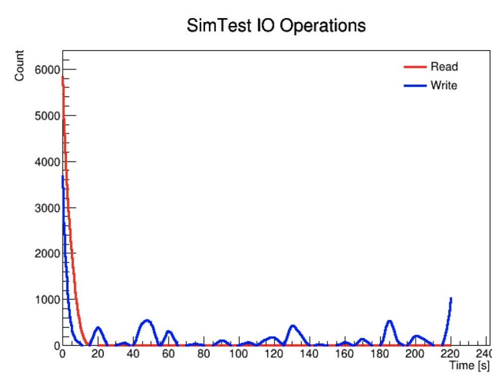


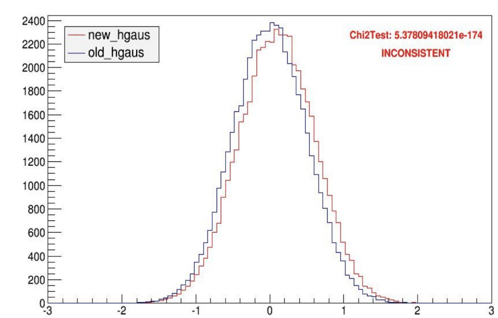
performance test

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

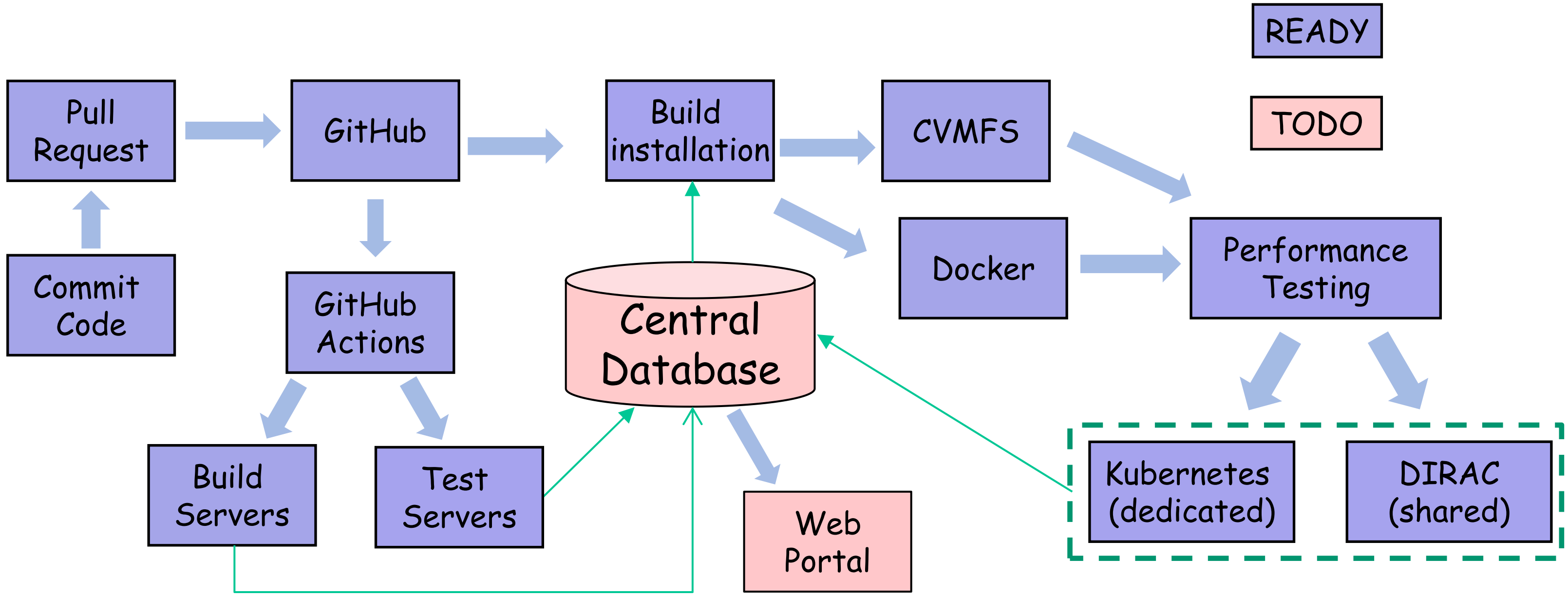






Automated Validation System

- The validation system is integrated with the Github Action system
  - Full validation workflow can be triggered by commit/merge-request
  - Developing running validation jobs on the grid
- ~ 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
  - Track reconstruction using heterogeneous resources
  - Integration of ML models
  - Parallel analysis framework based RDataFrame
  - Automated validation system

Thanks for your attention!

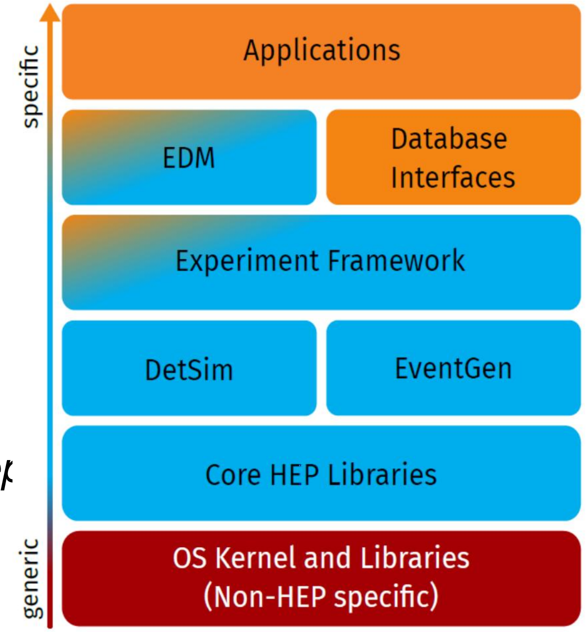
Welcomed to joining CEPCSW and working together!

https://github.com/cepc/cepcsw

Backup

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

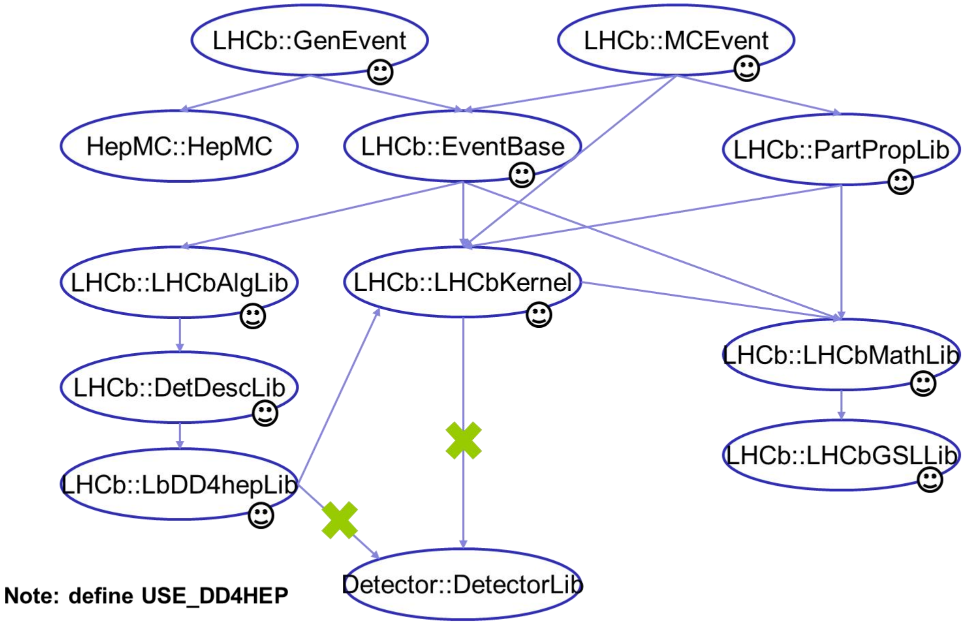
ONNX Introduction

- 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

Detector Simulation

- Reusing GenEvent and MCEvent from the LHCb project
  - Minimum number of packages are selected
  - Non-required dependencies were removed



- Source code of CEPC-on-Gaussino
  - LHCb: https://gitlab.cern.ch/talin/LHCb/-/tree/cepc-on-gaussino
  - gaussinoextlibs: https://gitlab.cern.ch/talin/gaussinoextlibs/-/tree/cepc-on-gaussino
  - Gaussino: https://gitlab.cern.ch/talin/Gaussino/-/tree/cepc-on-gaussino
- Building script: https://gitlab.cern.ch/talin/build-cepc-on-gaussino