CEPC Software Development

Weidong Li, IHEP/CAS

representing CEPC software and computing team

8th CEPC IAC meeting

1st November, 2022

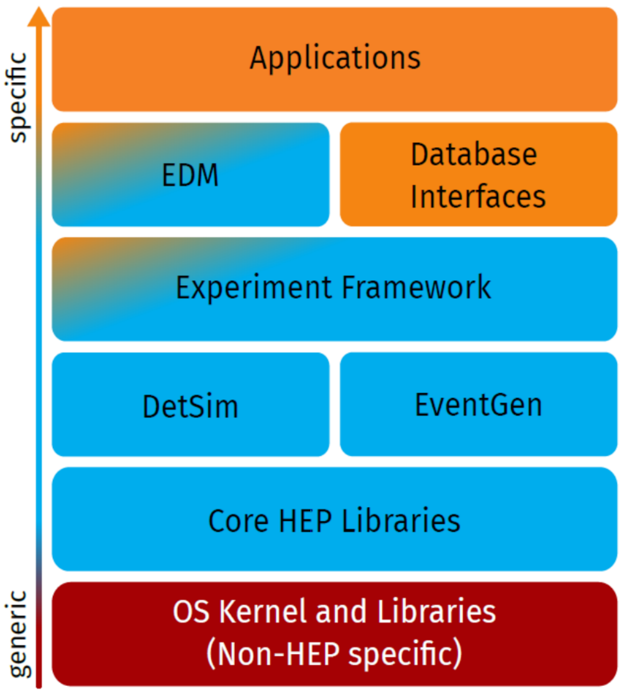
Contents

- Introduction
- Core software
- Simulation framework
- Reconstruction algorithms
- CEPC computing
- Summary

Introduction: Key4hep

❖ HEP software encompasses:

- 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 libraries (*ROOT, Geant4, CLHEP, …*)
- Commonly used standard tools and libraries (*Python, CMake, boost, …*)



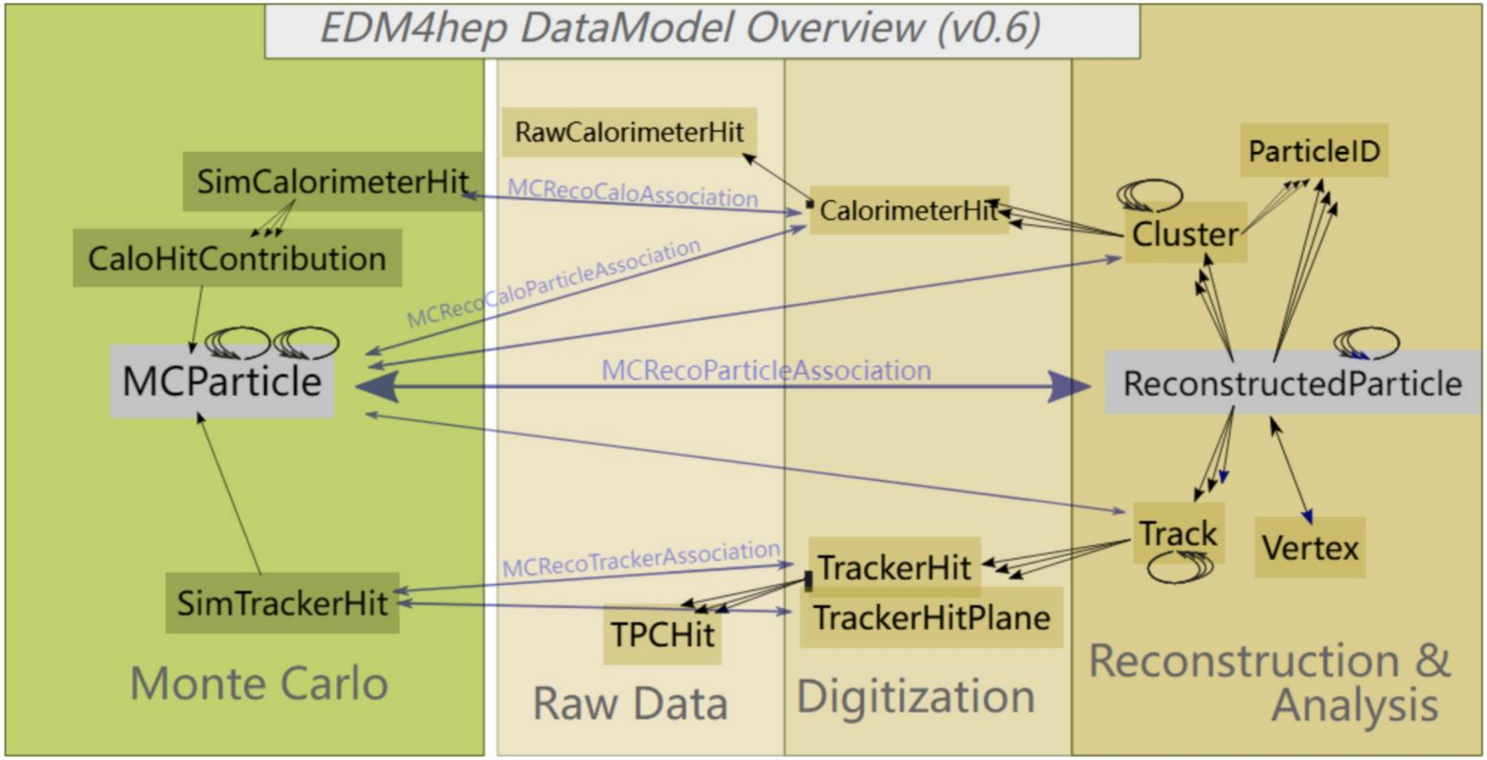
Thomas Madlener, Epiphany Conference 2021

❖ Turnkey software for future colliders

- Agreement reached at Future Collider Software Workshop (Bologna, June 2019) to develop a common software stack for future experiments
  - CEPC, CLIC, FCC, ILC, etc.

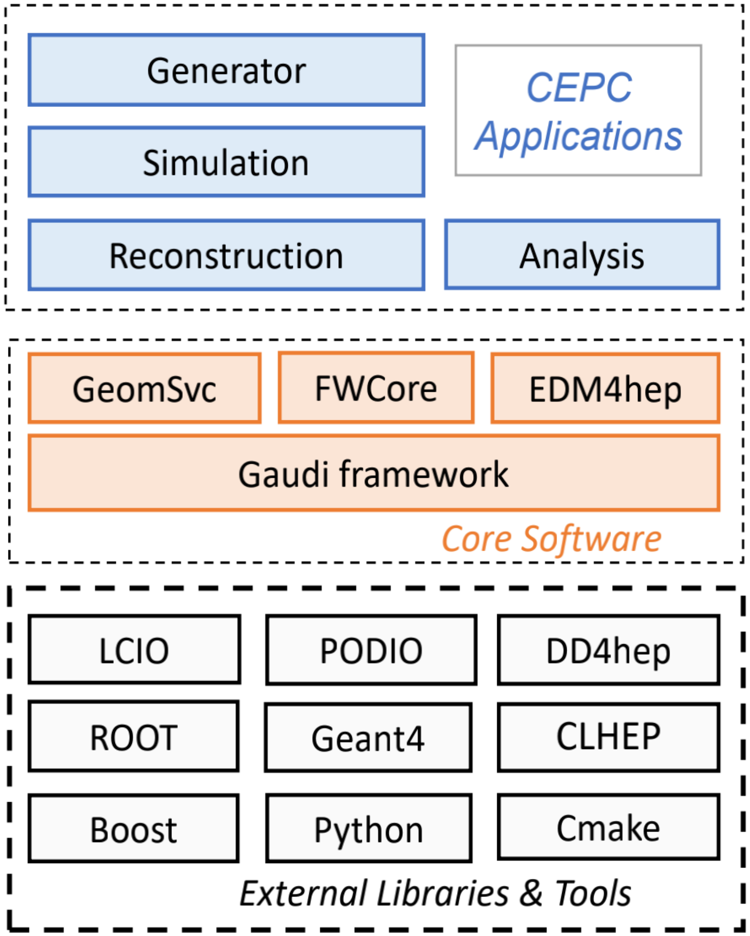
Introduction: Edm4hep

- Edm4hep is a common event data model being used by CEPC, CLIC, FCC, and ILC
- The generic event data model not only describes data objects created at different processing stages but also reflects the relationships between them.
- Automatically generated from YAML scripts



Introduction: CEPCSW

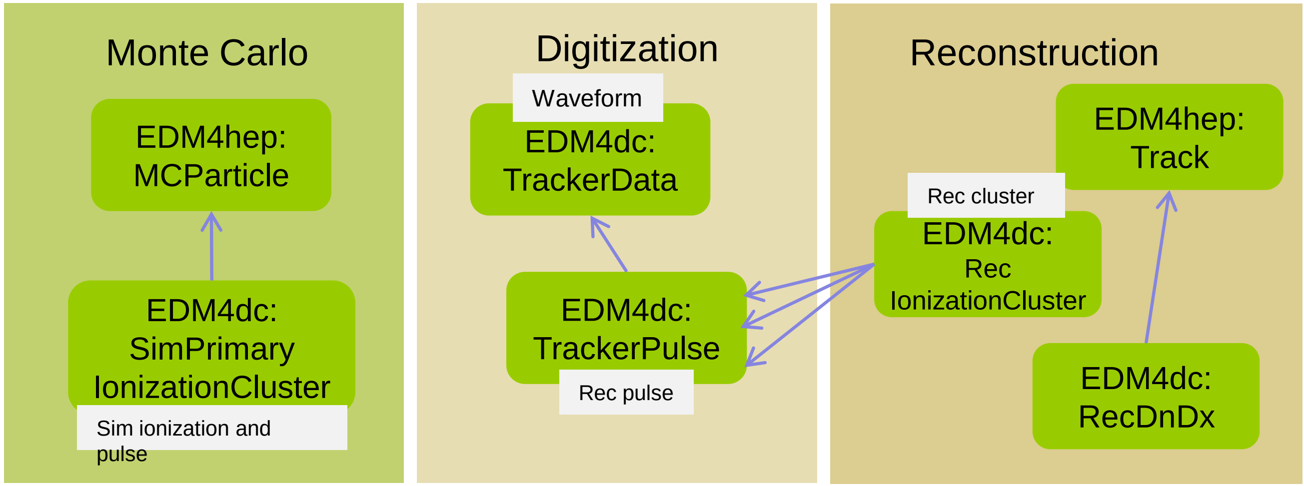
- Based on Key4hep and Edm4hep
- Providing a friendly software environment to algorithm developers and physicists
- Recent efforts on core software are focused on enhancing support to algorithm development and boosting software performance
  - Extension of Edm4hep
  - Heterogeneous computing
  - Machine learning integration based on ONNX
  - RDataFrame based analysis
  - Automated validation



https://github.com/cepc/CEPCSW

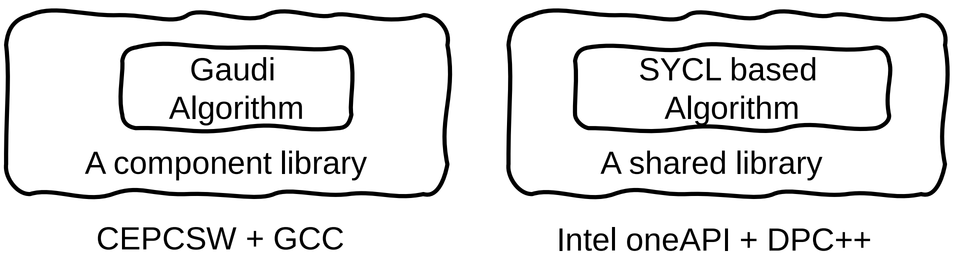
Core software: extension of EDM4hep

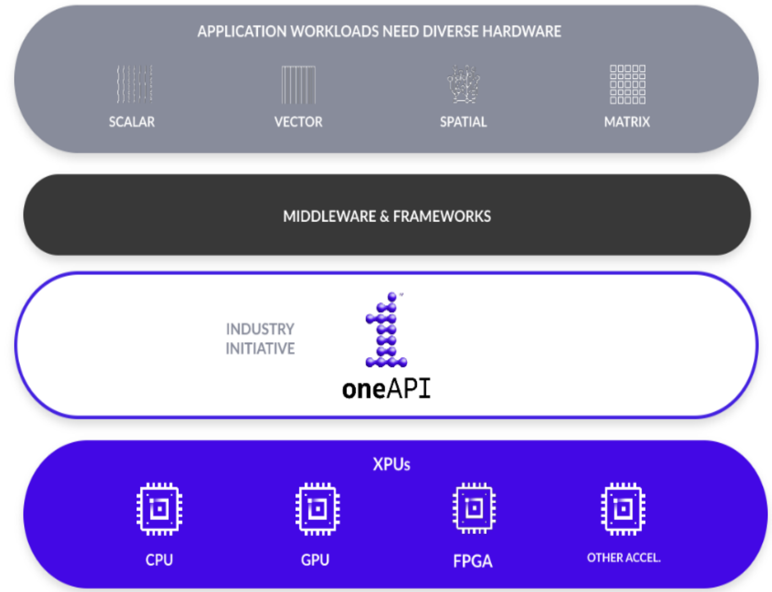
- Extending the event data model of TPC detector to accommodate the requirements from cluster counting of drift chamber
  - Meeting with the IDEA-CEPC drift chamber working group
  - Implementation details still being discussed within the EDM4hep group



Core software: heterogeneous computing

- TRACCC is a project under ACTS to demonstrate tracking chain on different kinds of computing hardware (CPU/GPU/FPGA).
- The first completed work is to port the TRACCC track seeding algorithm to CEPCSW.





- Plan to extend the algorithm to adapt to the geometry of the CEPC pixel detector and complete performance tests with EDM4hep in the heterogeneous computing environment.

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

Core software: machine learning integration

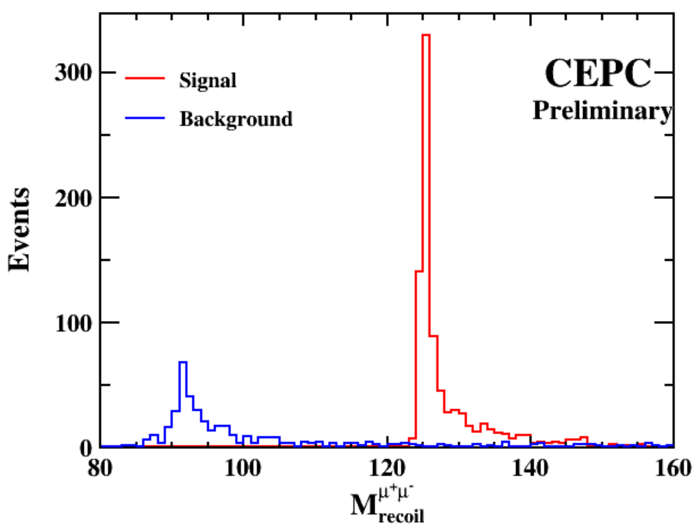
- To integrate machine learning models trained in different frameworks with CEPCSW, ONNX is used as the intermediate representation of ML models:
  - ML models are converted to the ONNX representation
  - Inference is performed in the CEPCSW
- Easy-to-follow examples are provided in CEPCSW, users only need to:
  - Create the ONNX session object at runtime
  - Specify where the model locates

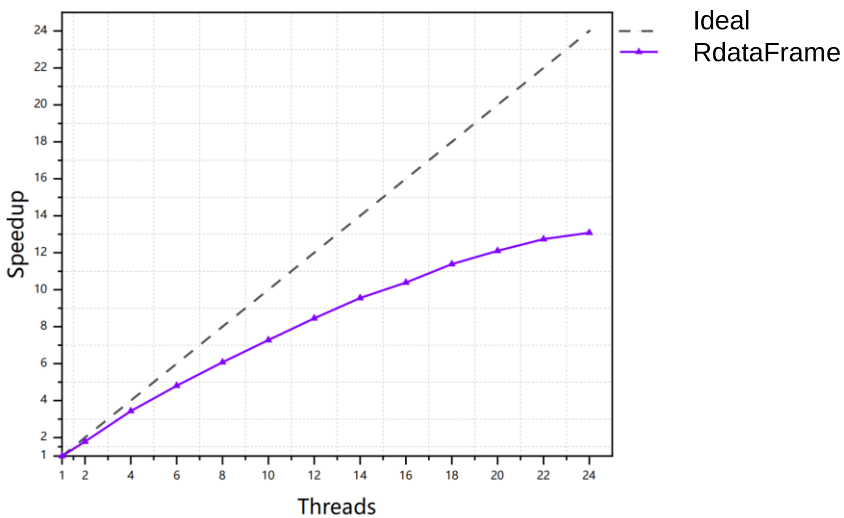
```
Ort::MemoryInfo info("Cpu", OrtDeviceAllocator, 0, OrtMemTypeDefault);

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,
    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);
```

Core software: RDataFrame based analysis

- An analysis toolkit based on RDataFrame is being developed
- We starts with an analysis example e+e- -> Z(mumu)H
  - Use CEPCSW to convert LCIO data produced with Marlin to EDM4hep data
  - Basic functionalities are tested: same results obtained from Marlin and RDataFrame
  - Performance tests are performed with multi-threading enabled
    - Performance of RDataFrame scales well with the number of threads

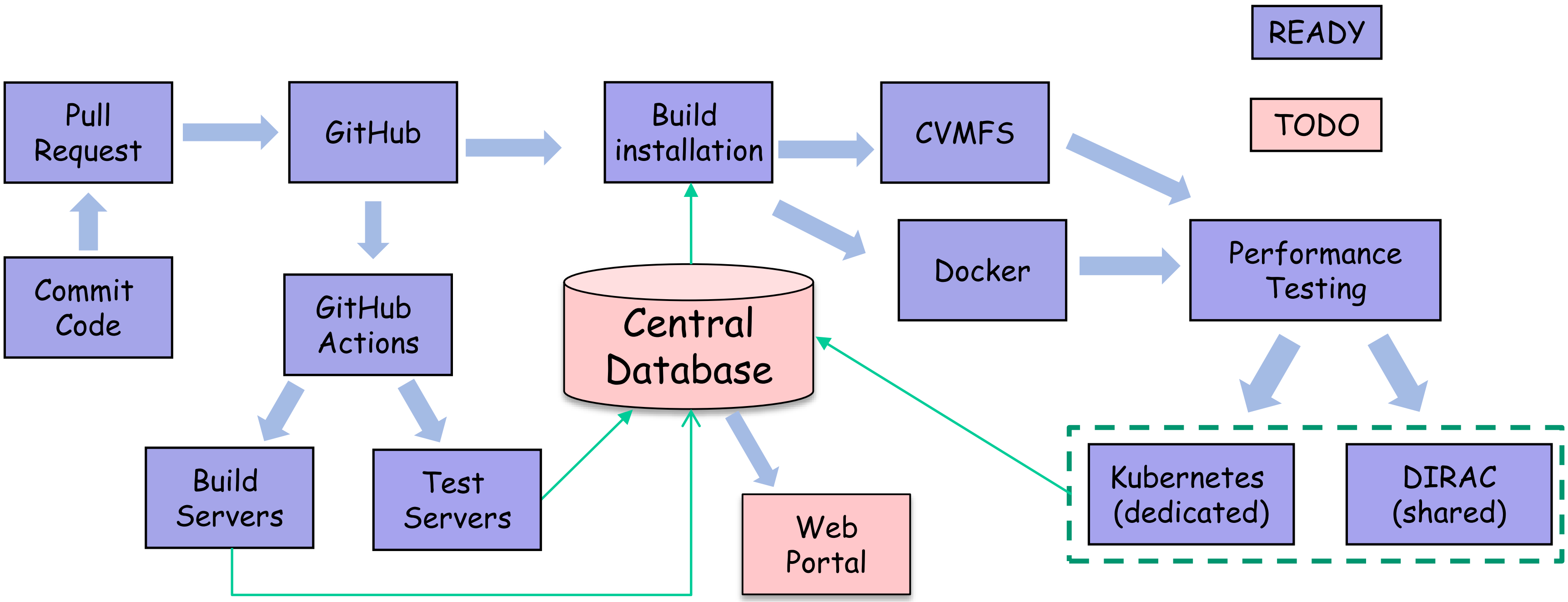




Scalability of RDataFrame with number of threads

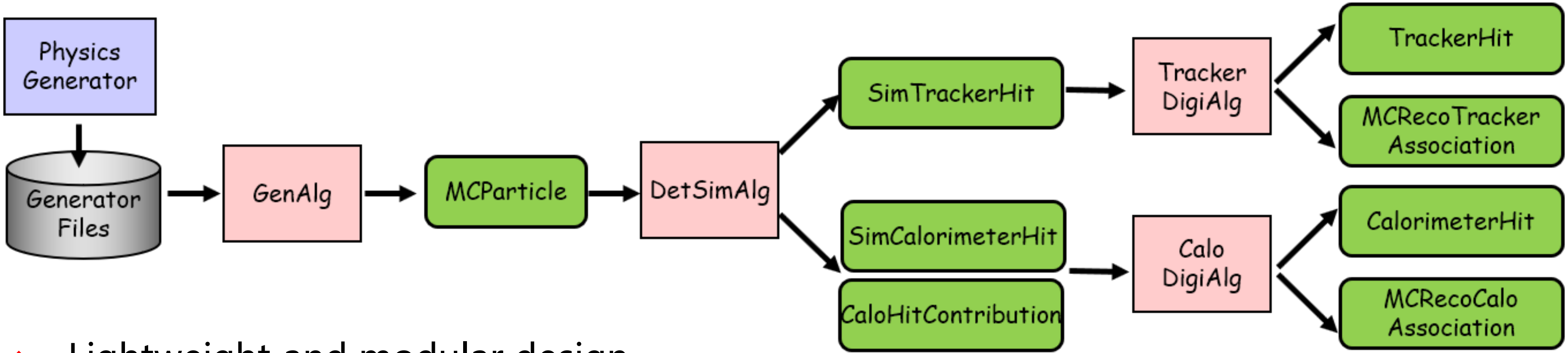
Core software: automated validation

- An automated validation system is being developed for software validation at different levels
- The validation system was 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

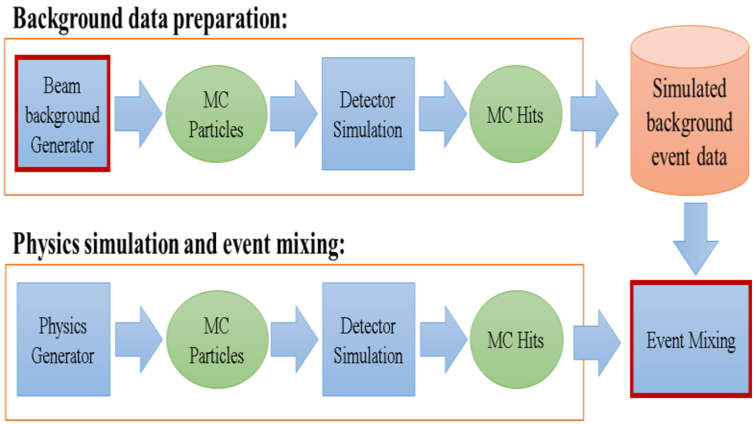


Simulation framework (1)

- A simulation framework was developed for CEPC, which provides a complete simulation chain for physics and detector performance studies.
- Well integrated with Key4hep software stack (EDM4hep, DD4hep, Gaudi)

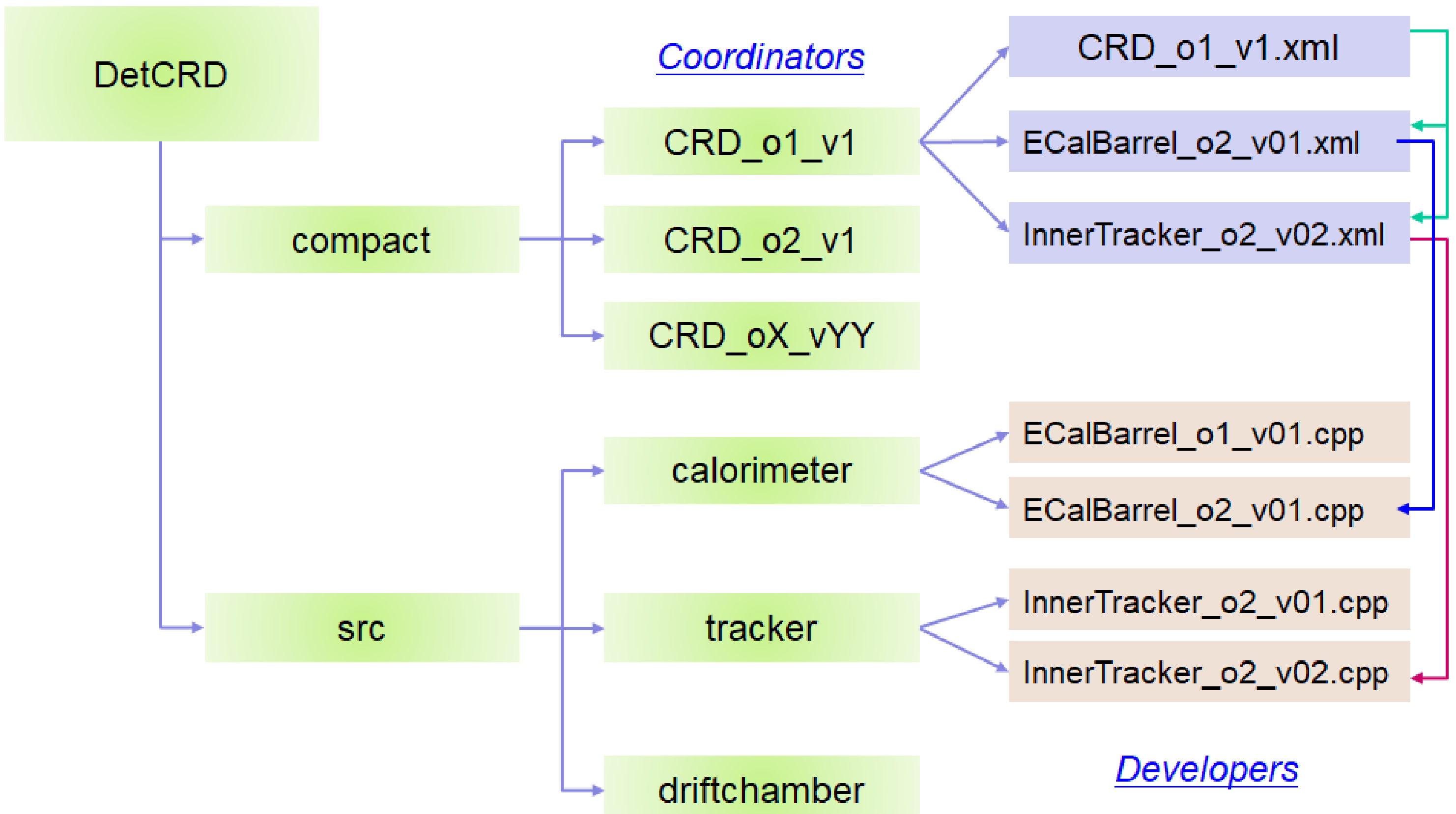


- Lightweight and modular design
  - Can properly handle detector response and MC truth
  - Support both full and fast simulation
  - Support Guinea-Pig for background simulation
  - Support hit level background mixing



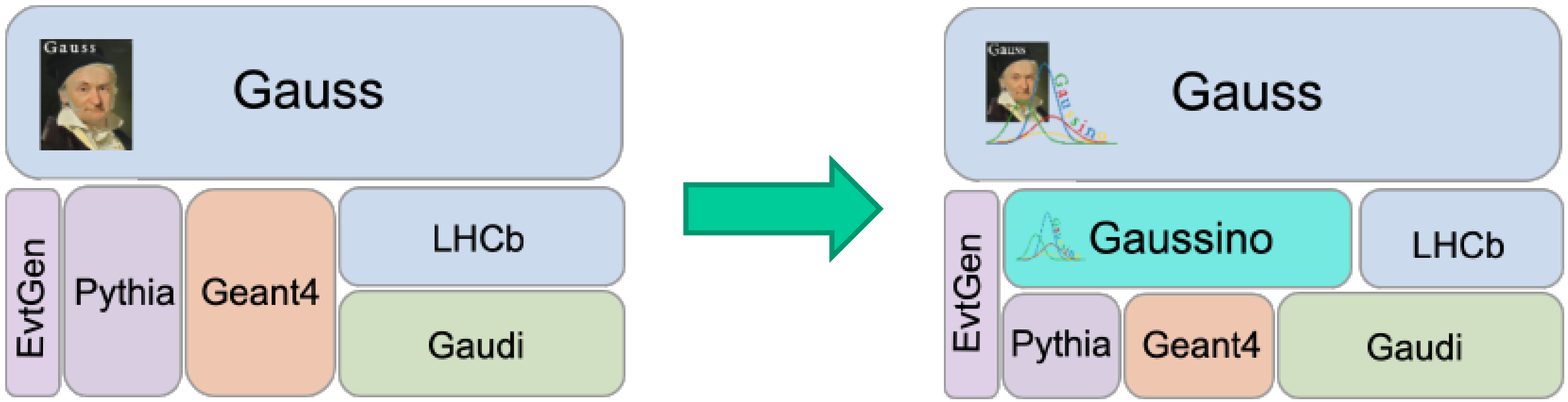
Simulation framework (2)

- Detector geometry management
  - A detector design option is defined by a compact file e.g. CEPC Reference Detector (CRD_o1-v1)
  - Details of sub-detectors are described by XML compact files and C++ constructors.



Simulation framework (3)

- Working closely with Key4hep people to evaluate the possibility of using Gaussino at CEPC.
- Gaussino is a common simulation framework, created by removing LHCb experiment-specific parts from Gauss.

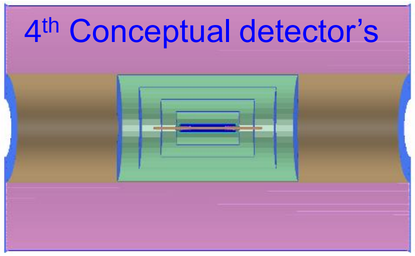


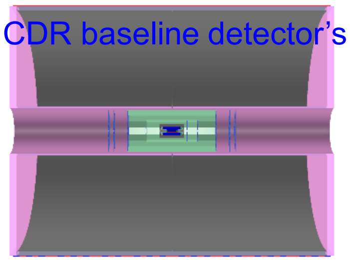
- Work has been done at CEPC
  - ✓ Build and run Gaussino in LHCb
  - ☐ Build and run Gaussino in CEPCSW
  - ☐ Integration with CEPC geometry

	Single threaded	Multi threaded
Gen only	OK	OK
Gen + Sim	OK	OK

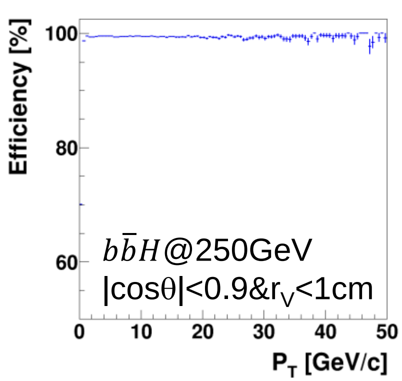
Reconstruction: Silicon Tracker (1)

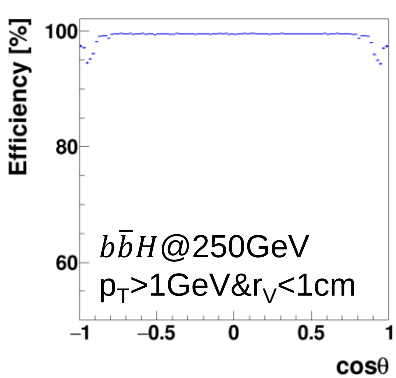
- Developing the tracking algorithm for silicon trackers of the 4th conceptual detector
  - Vertex detector (VXD) : **6** pixel layers
  - Silicon inside(DC)/internal tracker (SIT) : **4(3)** pixel/doubly-strip layers
  - Silicon outside(DC)/external tracker (SOT/SET) : **1** pixel/doubly-strip layers
  - Endcap/forward tracker (EIT&EOT/FTD) : **2** pixel + **3-5** pixel/doubly-strip layers
- Detector design can be modified easily through compact files for optimization purpose
  - Change construction of silicon modules : CDR trackers, MOST2 vertex detector, endcap tracker
  - Change layouts through parameters : layer number, layer position, layer material etc.
- Implemented two tracking chains in parallel
  - ILD tracking : used by CDR and also works for the 4th conceptual detector
  - Conformal tracking: reusing the code for digitization and track fitting

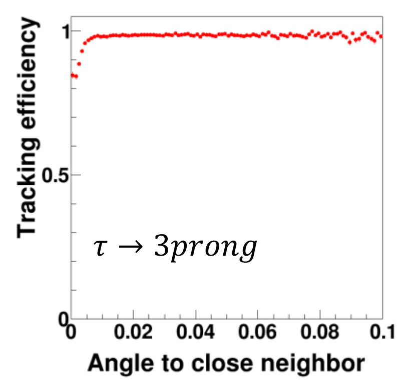




Tracking efficiency in CRD_o1_v04 preliminary:

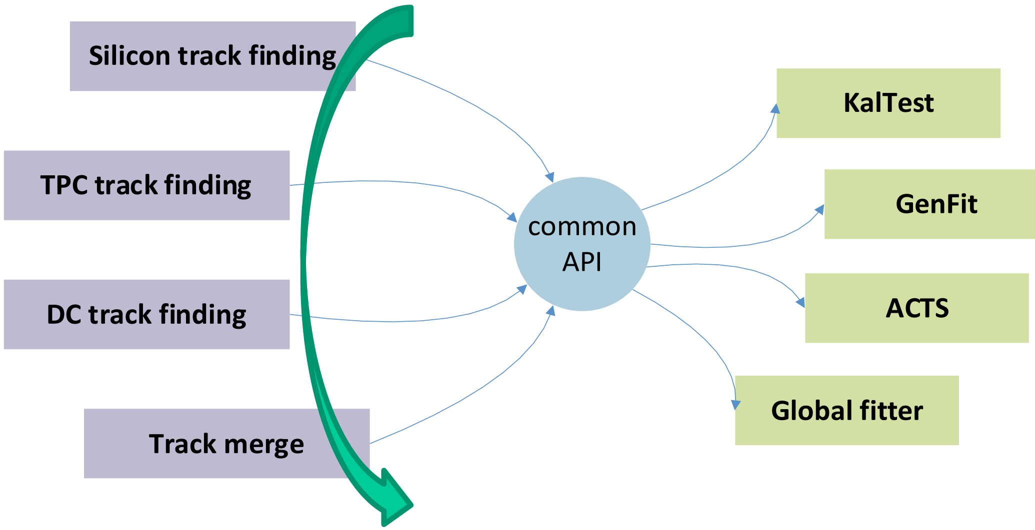




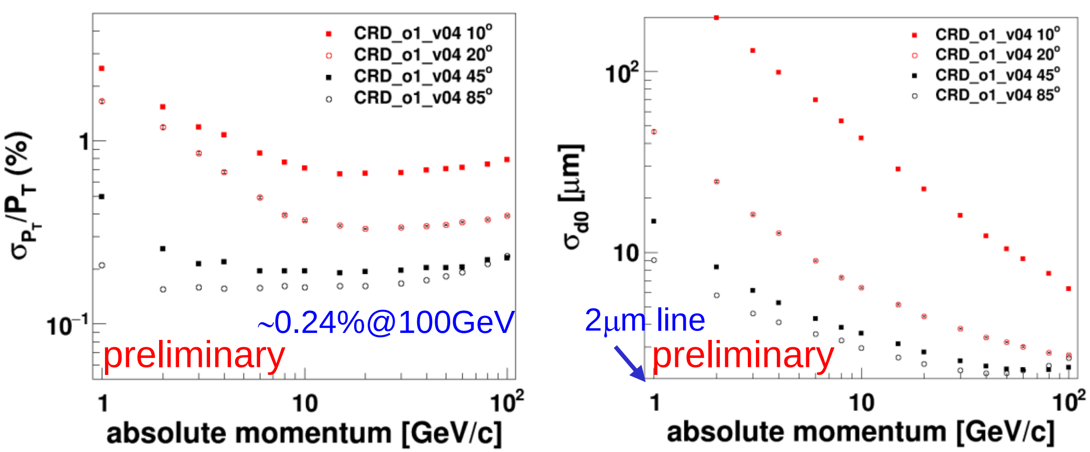


Reconstruction: Silicon Tracker (2)

- A common interface to unify different track fitting algorithm
  - After track finding, invoke and select a fitter through the API for track fitting according to the configuration
  - To choose best combination of track finding and fitting to balance the CPU usage and performance

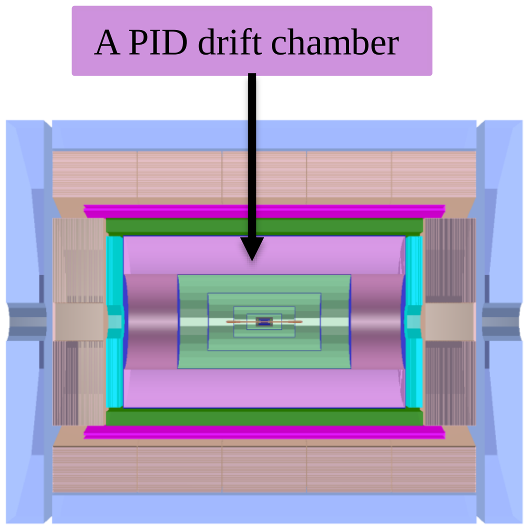


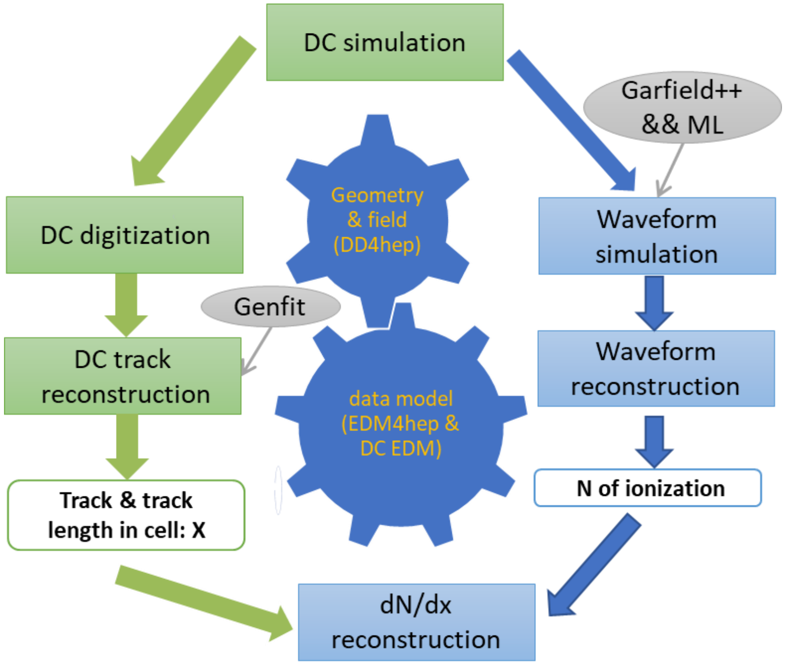
- Status
  - Common API has been created
    - KalTest, completed
    - DDKalTest, GenFit, ACTS and global fitter, ongoing
  - Silicon tracking algorithm
    - Conformal tracking : switching to the common API
  - Performance test with CRD_o1_v04
    - σ_{IP}= [15μm, 36nm, 2.8mm]
    - Radius of beam pipe: 10mm



Reconstruction: Drift Chamber (1)

- Drift chamber is the key detector in the 4th conceptual detector design to provide:
  - Good PID ability (2σ π/K separation at P < ~ 20 GeV/c)
  - Precise momentum measurement (eff. ~100%, σp<=0.1%)
- The drift chamber software was developed from scratch
  - Detector description and event model were based on DD4hep and EDM4hep, respectively.
- Algorithm chain encompasses
  - DC simulation, digitization and reconstruction
  - dN/dx simulation and reconstruction

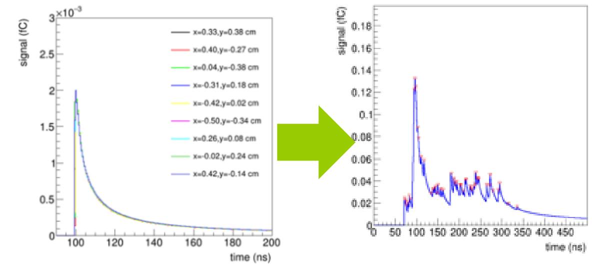




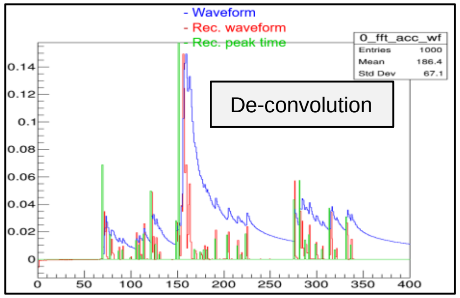
Drift chamber simulation and reconstruction flow

Reconstruction: Drift Chamber (2)

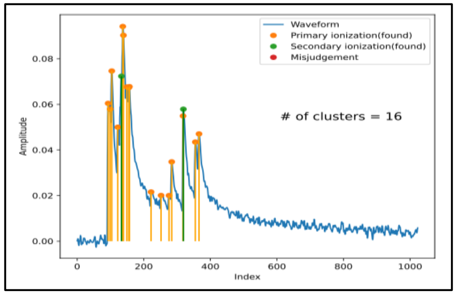
- Track finding
  - Dummy algorithm based on MC truth
  - Real algorithm using combinatorial Kalman Filter (CKF)
- Track fitting
  - Combining Silicon and DC measurements
  - Employing Genfit2 as the fitter
- dN/dx simulation
  - Primary ionization: Geant4 +TrackHeed from Garfield++
  - Neural network for simulation of other detector responses
- dN/dx reconstruction
  - De-convolution algorithm for peak finding
  - Neural network based cluster reconstruction



Waveform simulation



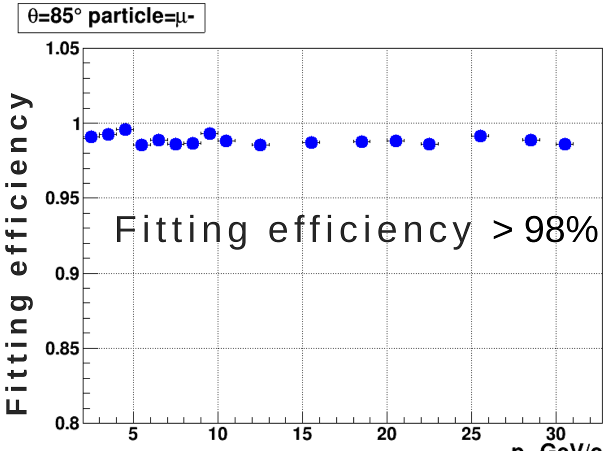
Pulse reconstruction

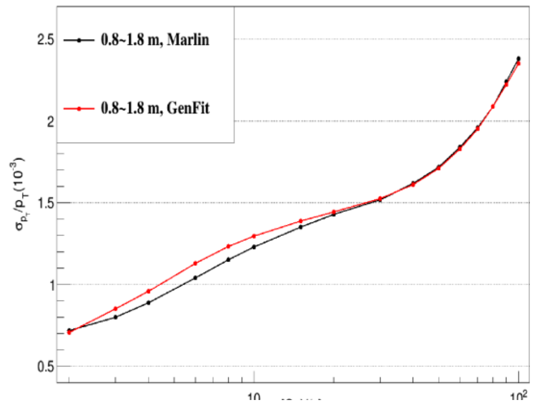


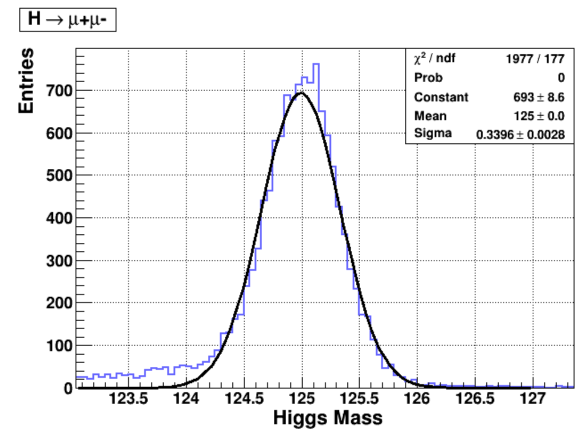
Cluster reconstruction

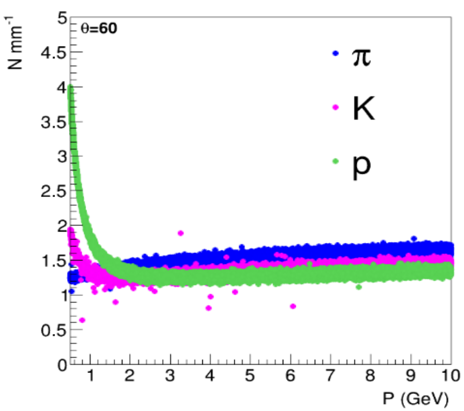
Reconstruction: Drift Chamber (3)

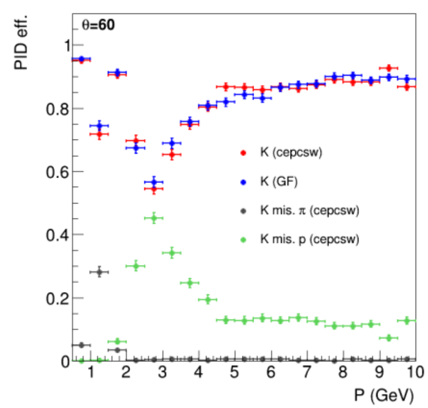
- The tracking performance of CEPC drift chamber is reasonable
- The PID performance obtained in CEPCSW agrees well with those from standalone Garfield++ simulation





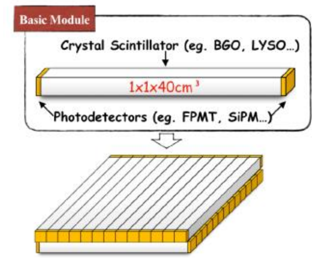


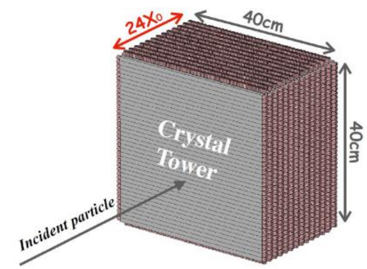


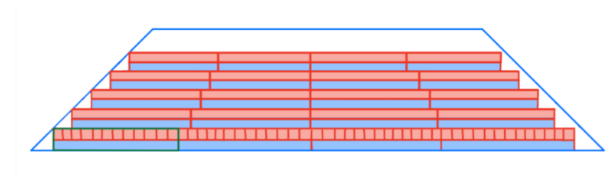


Reconstruction: Crystal ECAL (1)

- Design concept for long crystal bar ECAL
  - Homogeneous structure → Optimal energy resolution $3\%/\sqrt{E} \oplus 1\%$
  - Significant reduction of number of electronics readout channels
  - Time measurements at both ends to determination of shower position along the bar



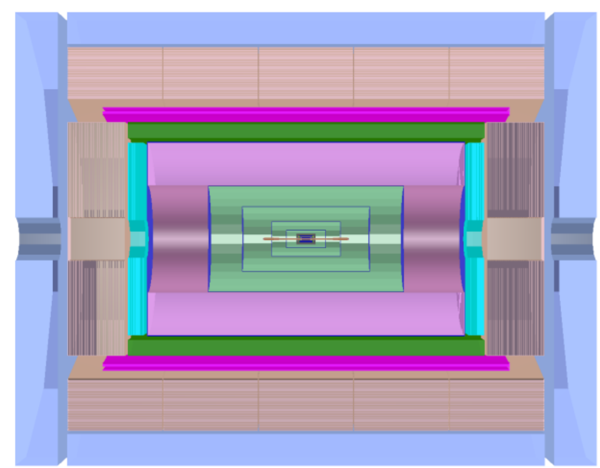




- Particle Flow Approach:

  Imaging Calorimeter + Topological Analysis

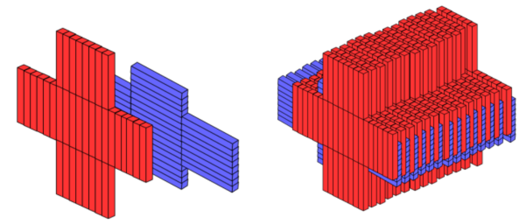
- Challenges
  - Ambiguity caused by mismatch of horizontal and vertical bars
  - Identification of energy deposits for each individual particle



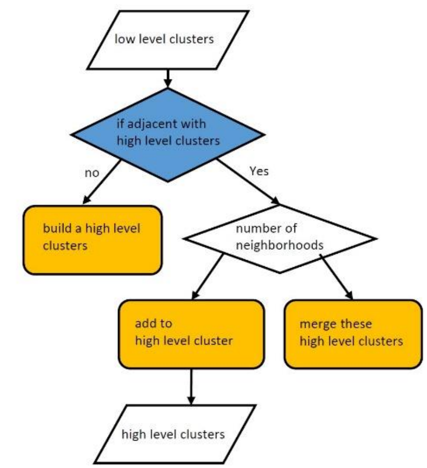
- High performance reconstruction algorithm is required!

Reconstruction: Crystal ECAL (2)

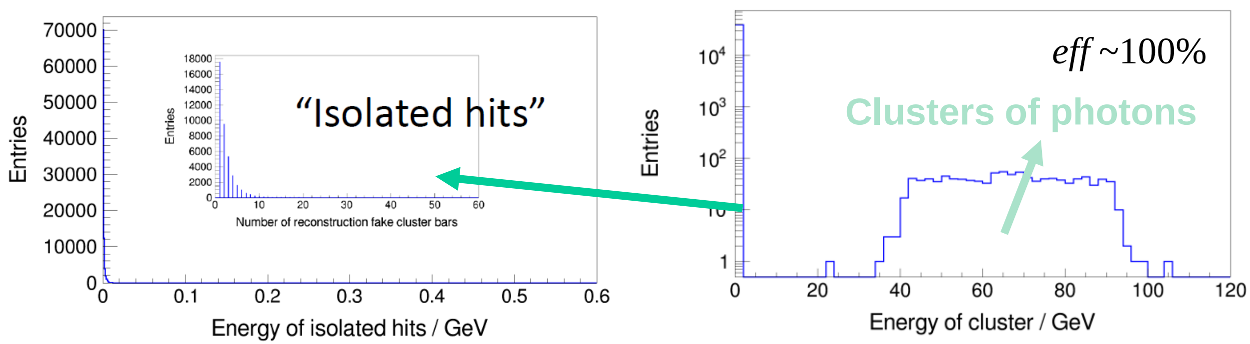
- Clustering
  - A group of adjacent fired crystals $E_i > E_{thres}$



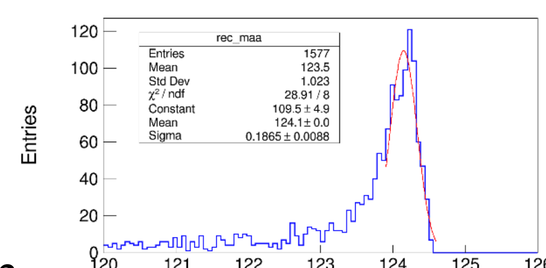
2DCluster 3DCluster



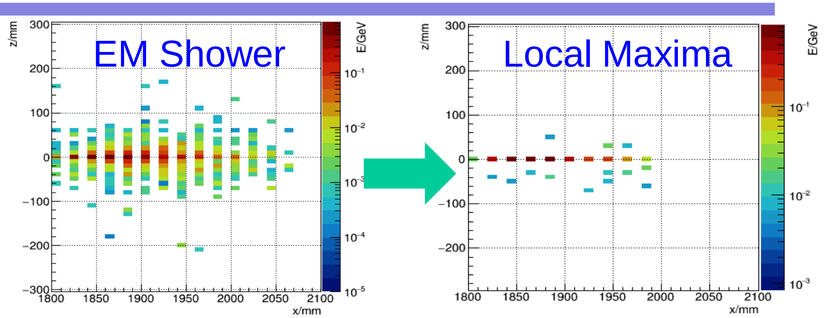
- Performance check of clustering algorithm
  - $e^+e^- \to ZH \to \nu\nu\gamma\gamma$ at $\sqrt{s} = 240$ GeV
  - 2 high energy clusters and many low energy clusters (isolated hits)



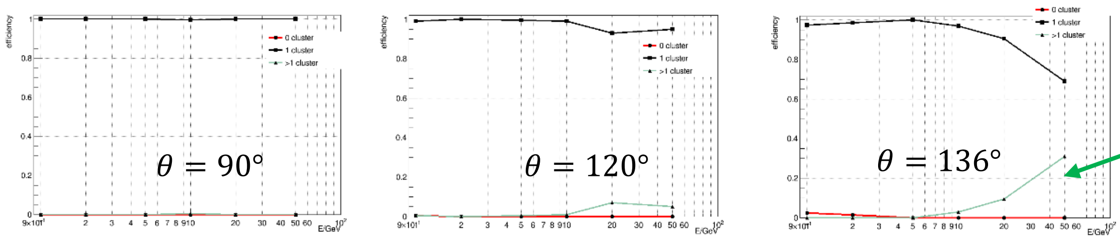
- Performance of inv. mass of $\gamma\gamma$
  - Position reconstruction alg. to calculate the direction
  - $M_{\gamma\gamma} = \sqrt{2E_{\gamma_1}E_{\gamma_2}(1-cos\theta_{\gamma_1\gamma_2})}$=124.148 $\pm$ 0.011 GeV/c^2
  - Longitudinal energy leakage will be corrected based on longitudinal profile

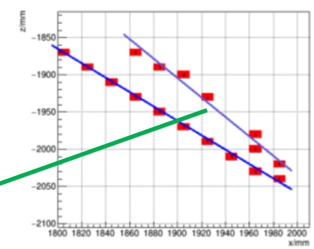


Reconstruction: Crystal ECAL (3)



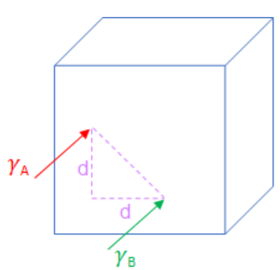
- EM shower recognition
  - Local Maxima + Hough Transformation
- Efficiency and fake rate of photon
  - Low energy or small $|\cos\theta|$: one & only one cluster

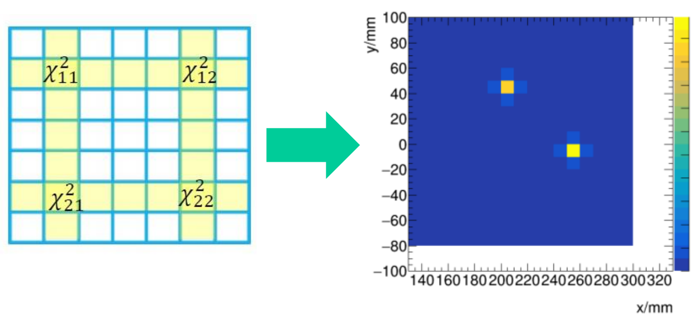


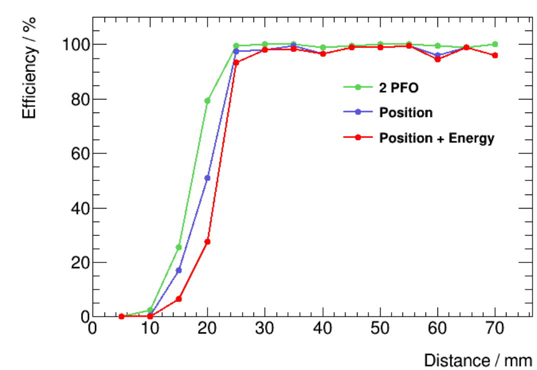


Fluctuations of energy deposits increase fake shower

- Energy splitting based on lateral distribution: $E_{i\mu} = \frac{E_{i\mu}^{exp}}{\sum_{\mu} E_{i\mu}^{exp}} \times E_{mea}^{i}$
- Ambiguity removal by exploiting energy and time correlation in longitudinal direction





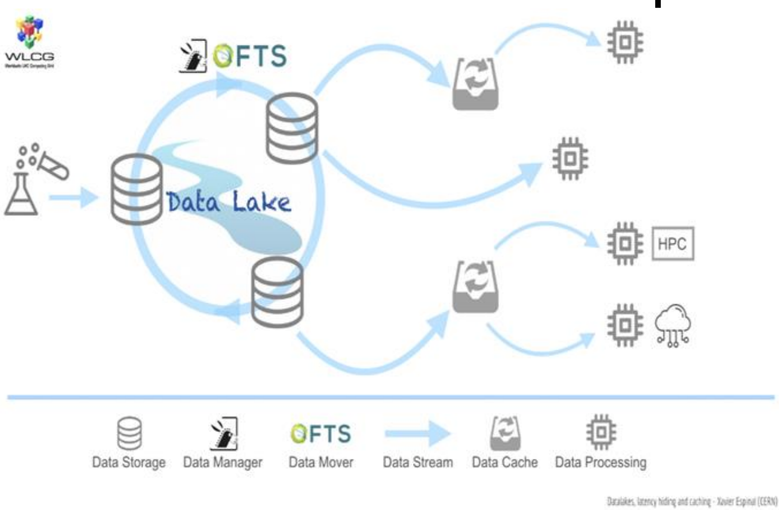


CEPC computing (1)

- CEPC has established a distributed computing platform for detector R&D
  - DIRAC as middleware , also using VOMS, FTS, CVMFS, StoRM, EOS, WLCG middleware
  - IHEPDIRAC contains tools developed for CEPC applications
- There are about 4600 CPU cores in this platform
  - IHEP holds 2000 dedicated cores including 640 cores shared with ILC
  - Other sites including UK contribute another 2600 cores
- Users can access these resources and move data through DIRAC tools to any connected site
- In recent years, following WLCG, the platform itself evolves:
  - TPC (Third Party Copy) protocol
  - AAI (Authentication Authorization and Identity)

CEPC computing (2)

- CEPC computing is expected to be faced with the same challenges and concerns on resource and sustainability with other future experiments
- To be prepared to meet these challenges, we start investigating solutions and related technologies
  - Heterogeneous resources infrastructure
  - “Data Lake” model with Rucio and XCache
- In addition, CEPC is seeking opportunities to establish collaborations on common computing infrastructure and tools with other experiments.



From S.Campana, Computing - challenges and future directions (ECFA 2021)

Summary

- CEPC Software is being developed in collaboration with Key4hep project.
- Significant progress has been made since last CEPC workshop:
  - Core software: event data model, heterogeneous computing, machine learning integration, RDataFrame based analysis, automated validation
  - Reconstruction algorithms for silicon tracker, drift chamber, and crystal ECAL
  - 2000 dedicated CPU cores added to the CEPC computing platform
- More cooperation opportunities with other future experiments are being explored.

Thank You !

谢谢