STCF core software and analysis tools

Teng LI on behalf of the STCF core software development team

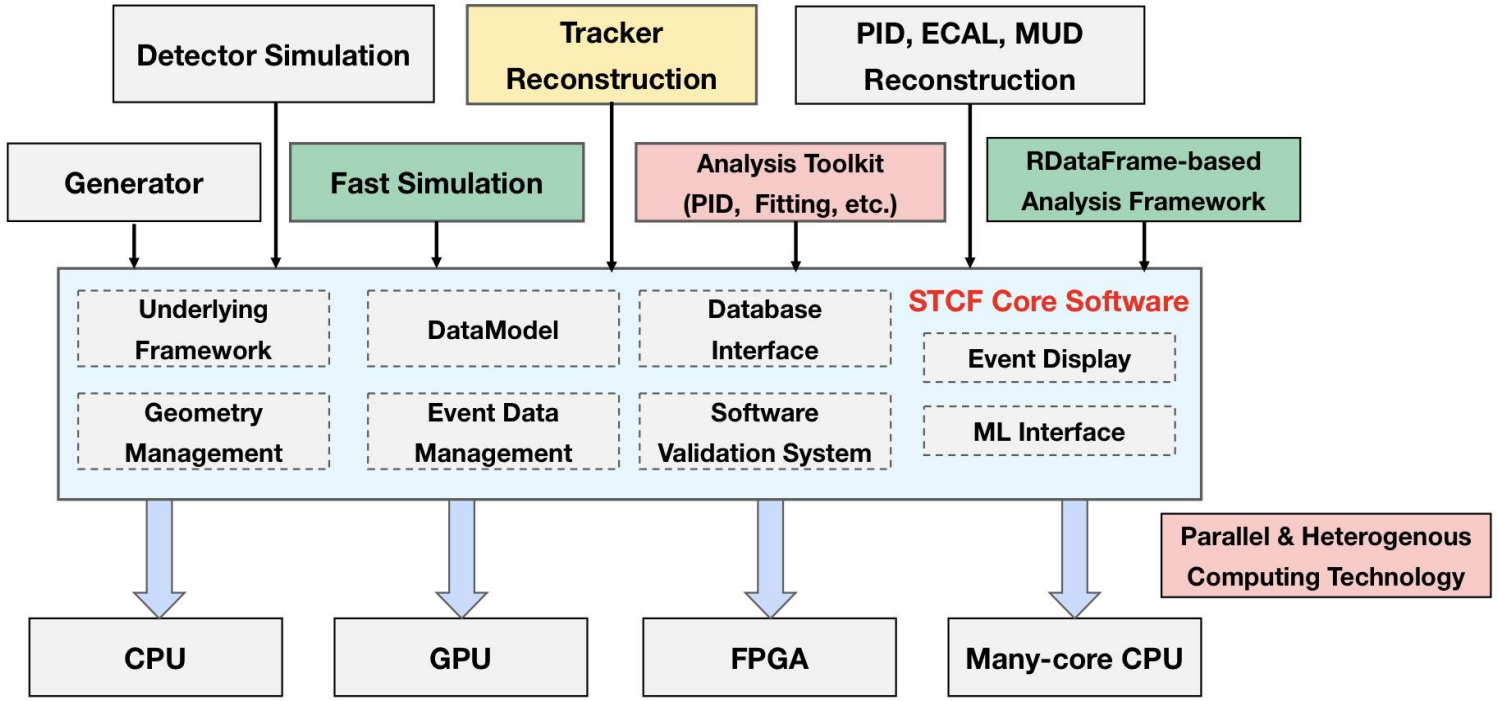
Shandong University

2024-11-21

FTCF 2025 Workshop, SYSU

Introduction

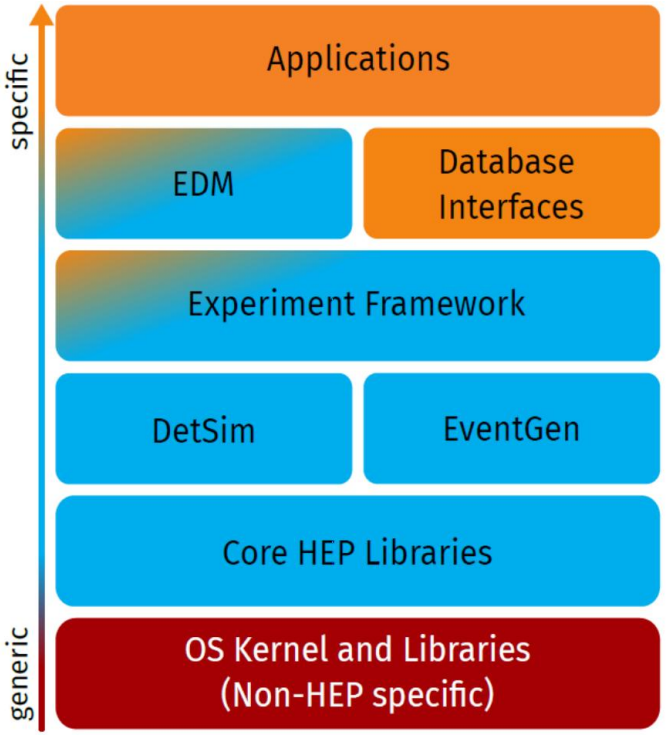
- The task of the STCF core software
  - To fulfill official offline data processing tasks, i.e. detector simulation, digitization, calibration and reconstruction
  - Provide a common platform for users to perform data analysis
- Overview of STCF core software



- The underlying framework
- Event data management
- Detector description and conditions data management
- Event display
- Support of ML, parallel computing, and heterogeneous computing
- Software and physics validation
- Software build, installation and distribution

Main R&D Challenges

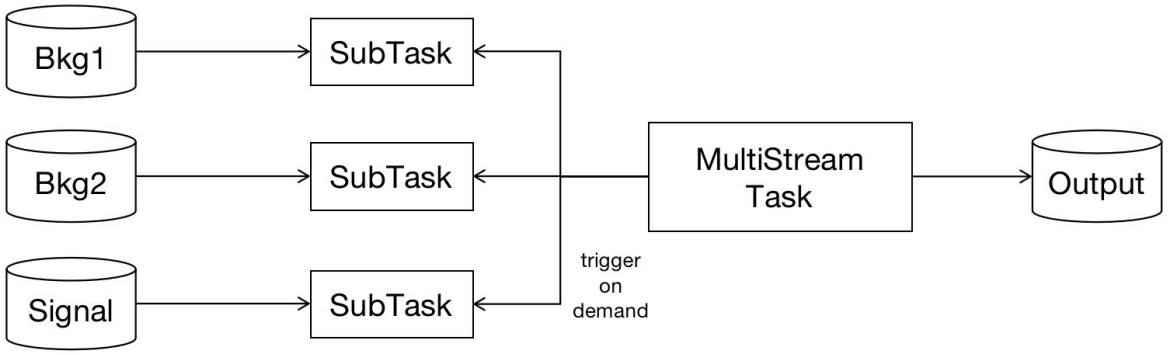
- Main R&D challenges and innovations for STCF core software
  - The huge data volume requires much more advanced performance
    - Relying on pure CPU resource to process **exabytes** of data is hardly realistic under previous cost-model
    - Parallel computing, and heterogeneous resources, like GPUs, or event FPGAs need to be considered to overcome the challenges
    - The core software needs to provide ready-to-use development and run time environment for these techniques
    - Support of flexible ML inference is also nesessary
  - Adoption of common software developed for future colliders
    - OSCAR is developed partially based on Key4hep, including EDM based on podio, geometry based on DD4hep etc.

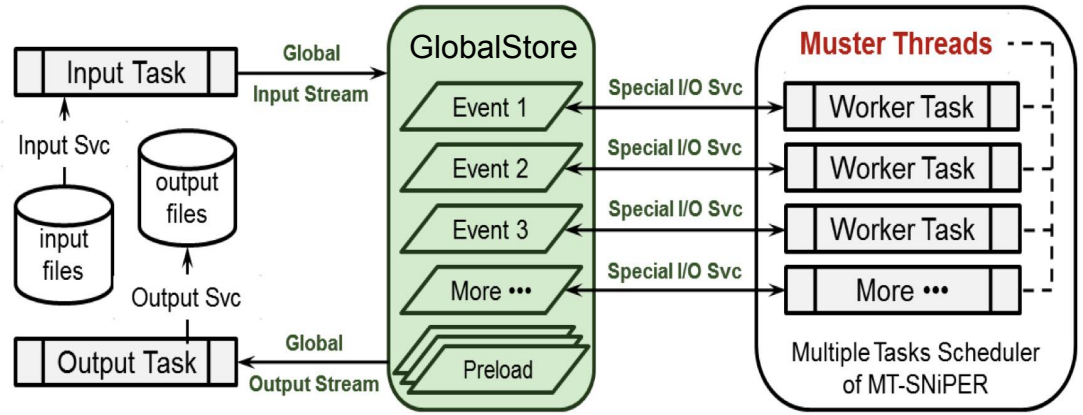


Key4hep
Thomas Madlener,
Epiphany Conference 2021

Underlying Framework: SNiPER

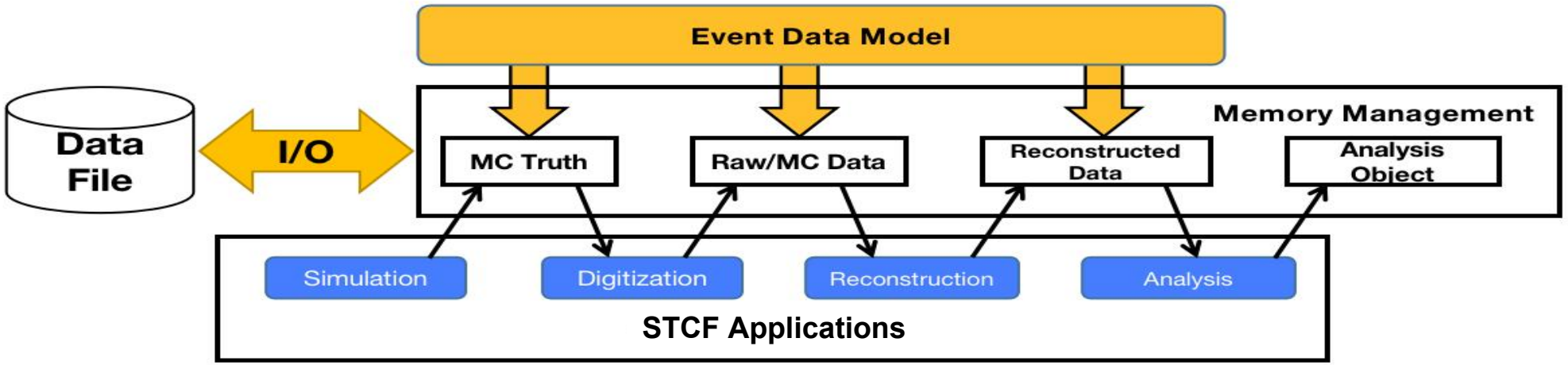
- The underlying framework builds the skeleton of OSCAR
  - Provide basic functionalities of **event loop control, algorithm scheduling, thread management, user interface, job configuration, logging** etc.
- OSCAR adopts SNiPER as the underlying framework
  - Developed since 2012, maintained by 10+ developers from IHEP, SDU, etc.
  - Adopted by JUNO (neutrino), LHAASO (cosmic ray), nEXO (neutrinoless double beta decay) and HERD (dark matter)
- Advantages of SNiPER
  - Lightweighted, efficient and highly extendable. Support of flexible data processing chain.
  - Efficient multithreading. C++/Python hybrid programing.



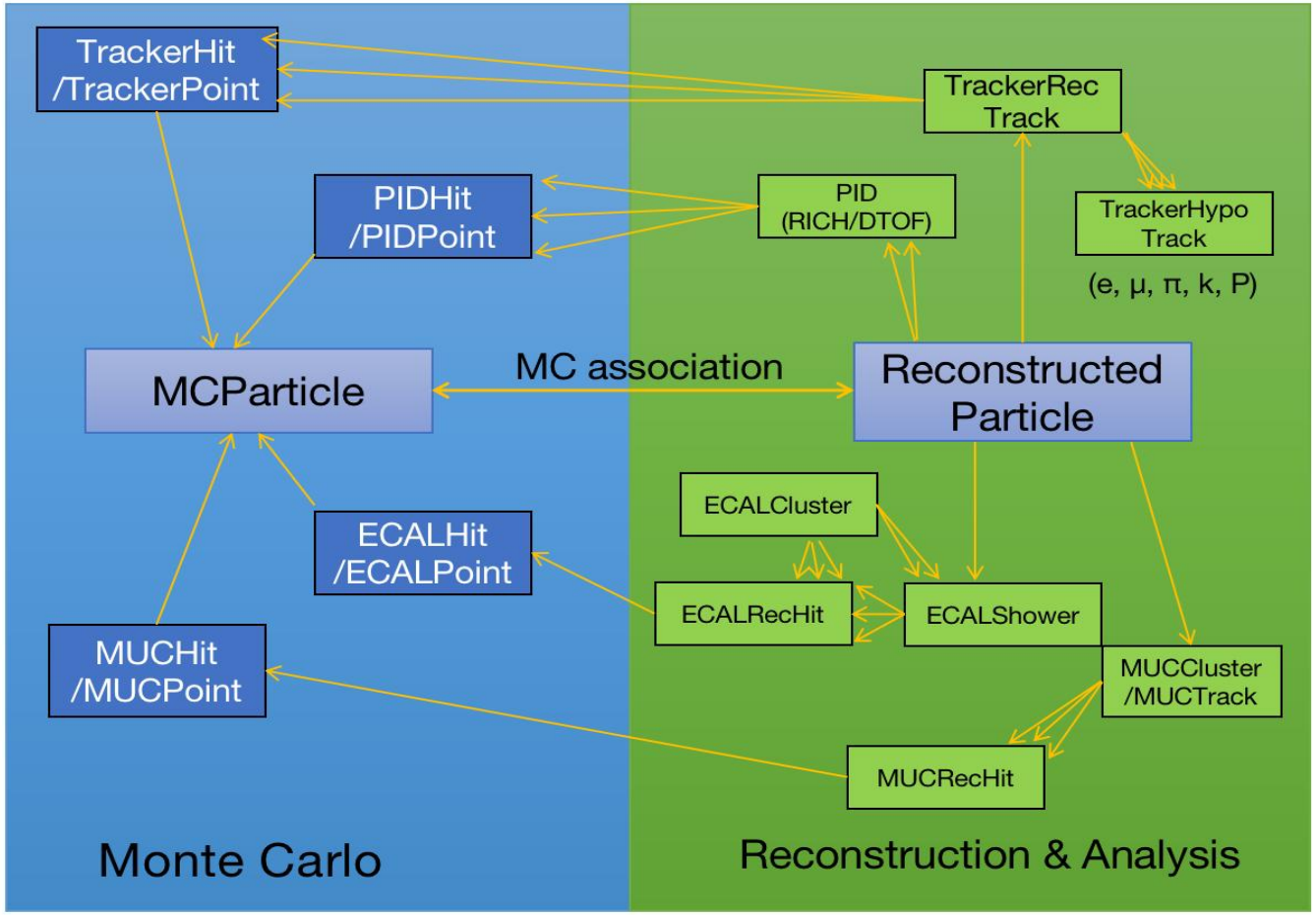


Event Data Management: **Requirements**

- Event data management is the most crucial part of the framework
  - **Provide tools to define the Event Data Model (EDM)**
    - The definition of physics event data (MC particles, hits, readouts, tracks, clusters, reconstructed particles),
    - Construct relationship between data objects (e.g. which particle makes these hits? Which hists are used to fit a track, etc.)
  - **Provide automated memory management and data I/O functionalities**
  - **Provide backward and forward compatibility, very important for the long running of STCF.**
  - **Guarantee thread-safety, and provide high performance for MT applications**



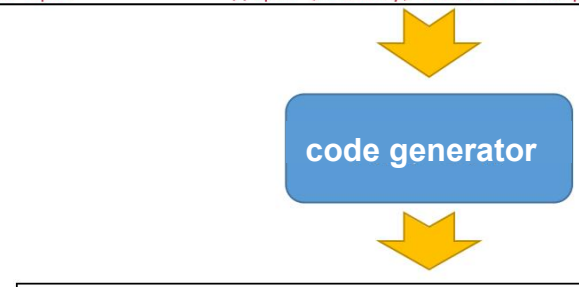
Event Data Model and of OSCAR



EDM classes defined in OSCAR

YAML

```yaml
MCParticle:
  Description : "Data class for storing Monte Carlo particles"
  Author : "SDU"
  Members :
    - int trackID                // track ID
    - int PDG                    //PDG code of the particle
    - int generatorStatus        //status of the particle as defined by the generator
    - int simulatorStatus        //status of the particle from the simulation program - use BIT constants
    - int type                   //particle type. only generatorStatus== 1 or 2 has the type
    - float charge               //particle charge
    - float time                 //creation time of the particle in [ns] wrt. the event, e.g. for preassig
    - double mass                //mass of the particle in [MeV]
    - Vector3d vertex            //production vertex of the particle in [mm].
    - Vector3d endpoint          //endpoint of the particle in [mm]
    - Vector3f momentum          //particle 3-momentum at the production vertex in [GeV]
    - Vector3f momentumAtEndpoint //particle 3-momentum at the endpoint in [GeV]
    - Vector3f spin              //spin (helicity) vector of the particle.
```

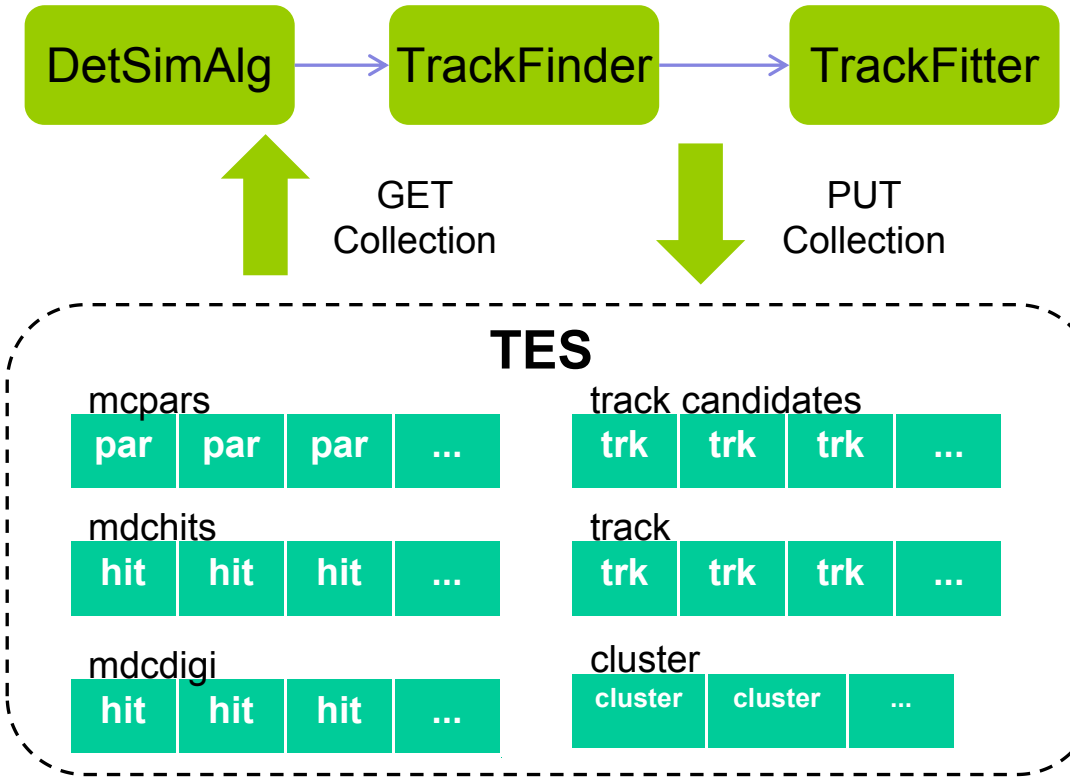


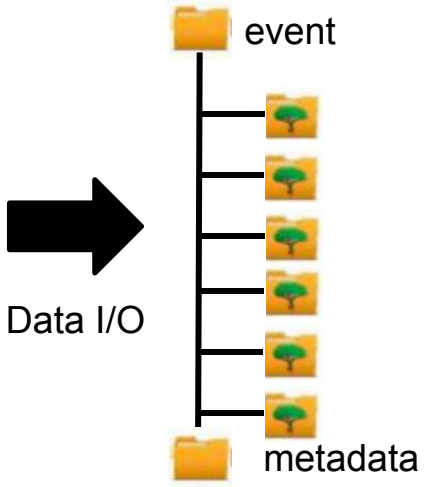
MCParticle.h	MCParticle.cpp	C++
MutableMCParticle.h	MutableMCParticle.cpp	
MCParticleObj.h	MCParticleObj.cpp	
MCParticleData.h	MCParticleData.cpp	
MCParticleCollection.h	MCParticleCollection.cpp	
MCParticleCollectionData.h	MCParticleCollectionData.cpp	

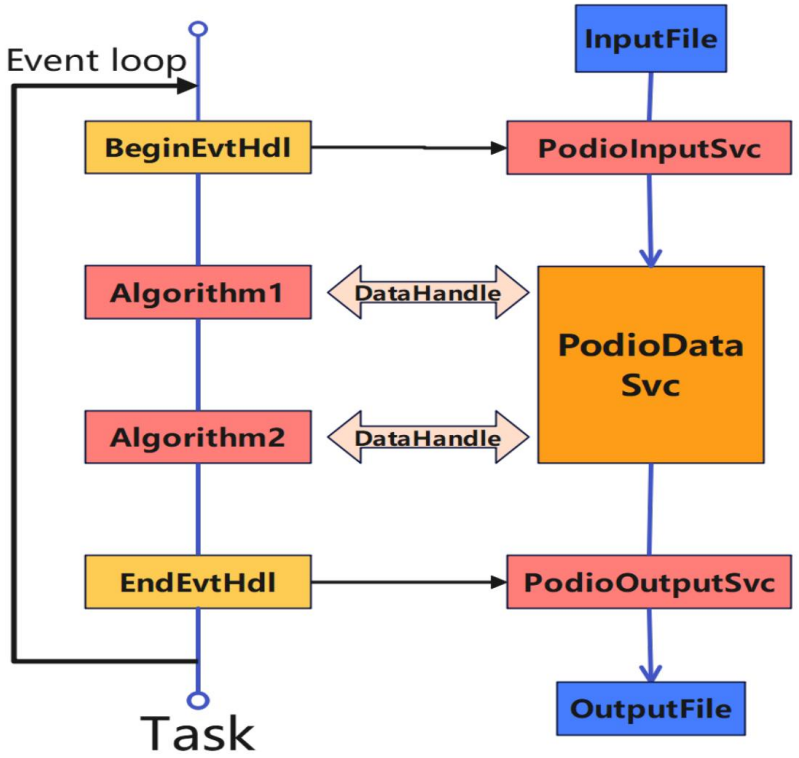
Based on YAML definition, generate EDM C++ code accordingly

Transient Event Store and Data I/O

- **Transient Event Store** (TES) is where EDM objects are stored in memory
  - TES in OSCAR is developed based on podio::EventStore
  - Now being migrated to podio::Frame for more flexible usage



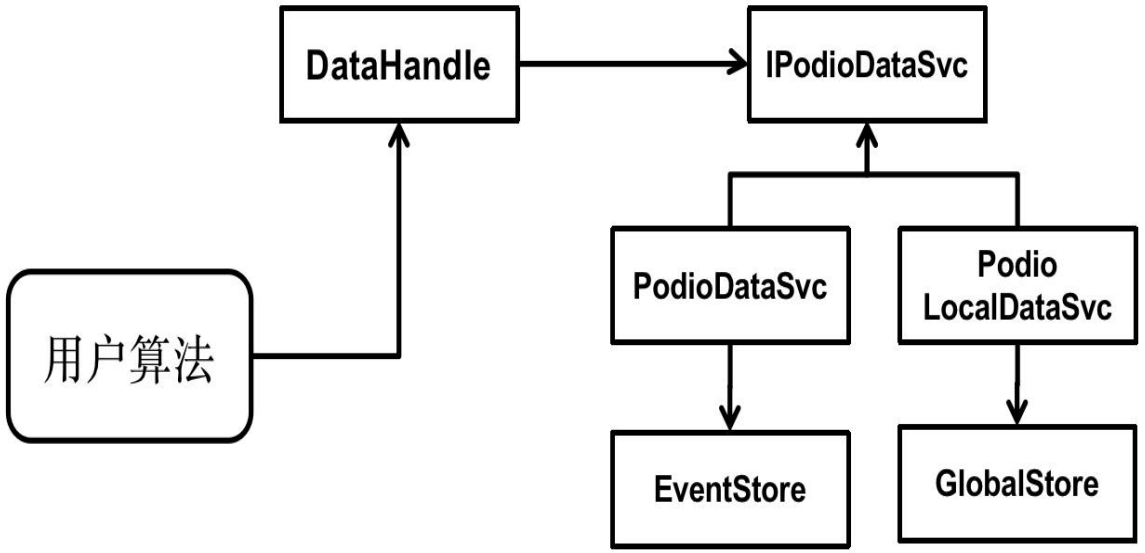


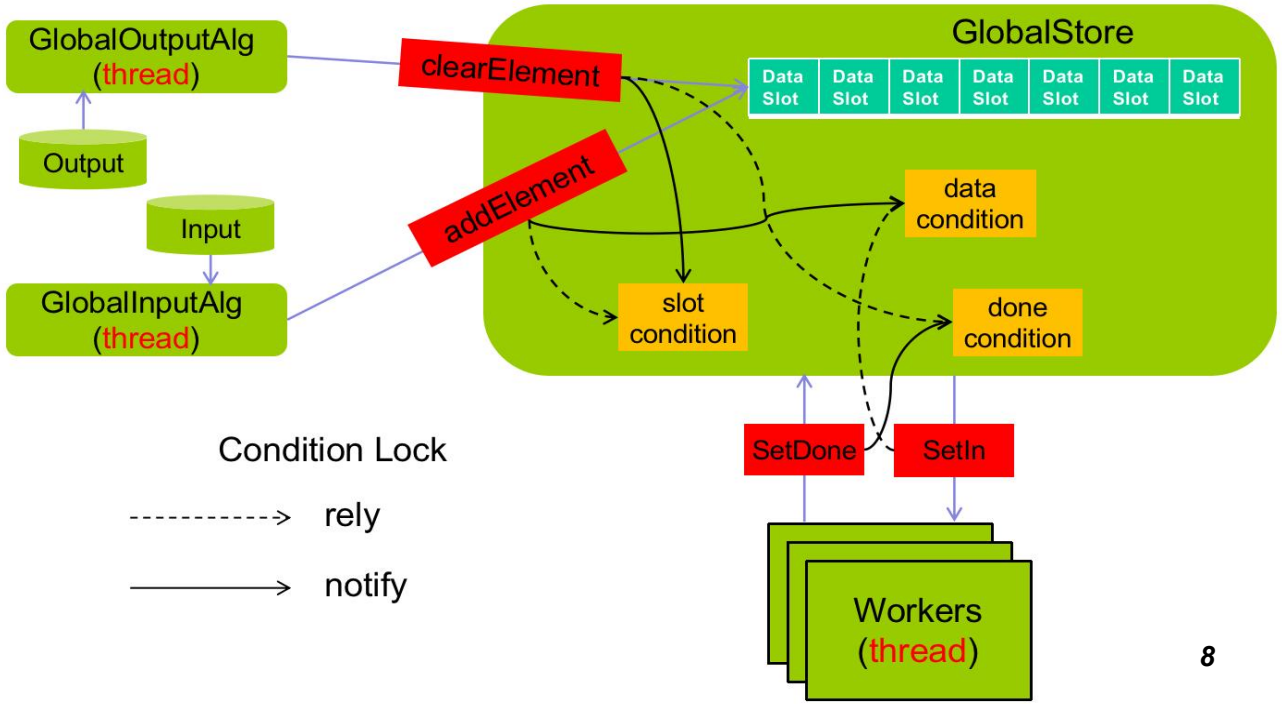


Implementation of TES and data I/O
- PodioDataSvc
- PodioInputSvc
- PodioOutputSvc

Parallelized Event Data Management

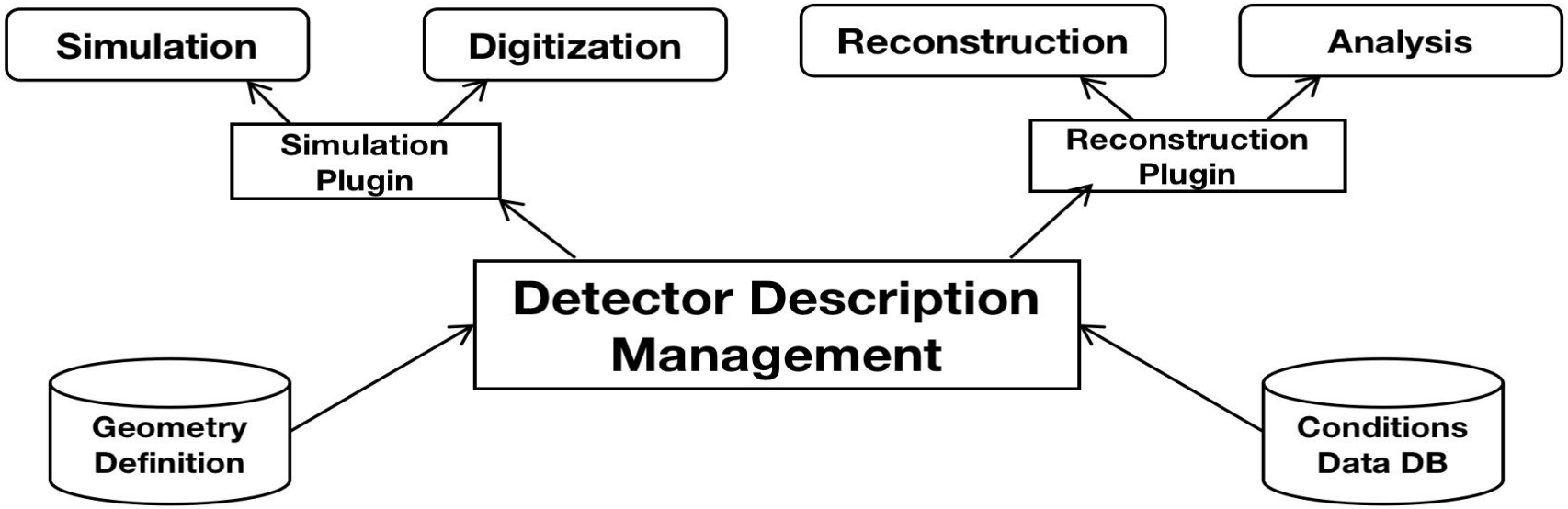
- To enable parallelized data processing, a GlobalStore is developed based on podio
  - Re-implement podio::EventStore to cache multiple events (each within one data slot)
  - Use several condition lock to enable safety exchanging data between threads
  - I/O services are binded to dedicated I/O threads, to ensure performance and flexible post- or pre-processing
- Users could switch serial/parallel by just changing job configuration





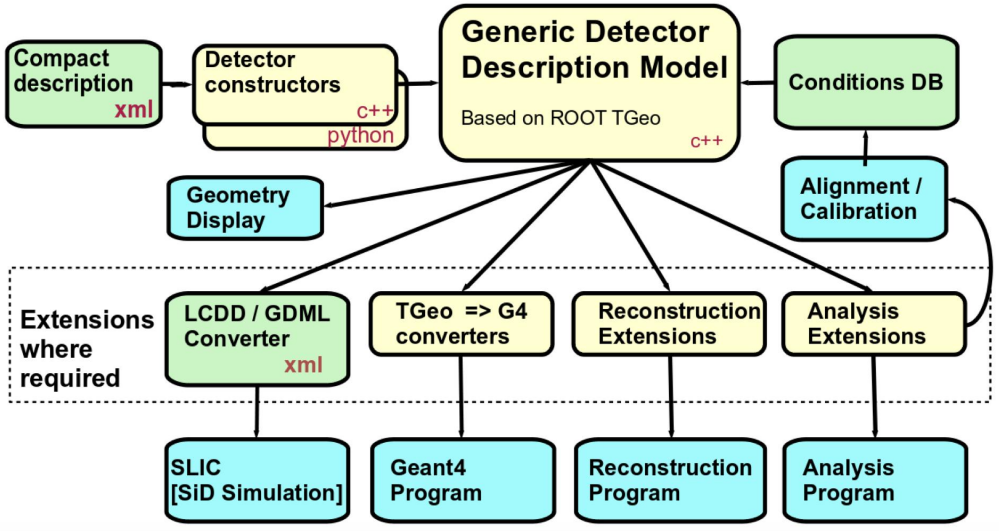
Detector Description Management: **Requirements**

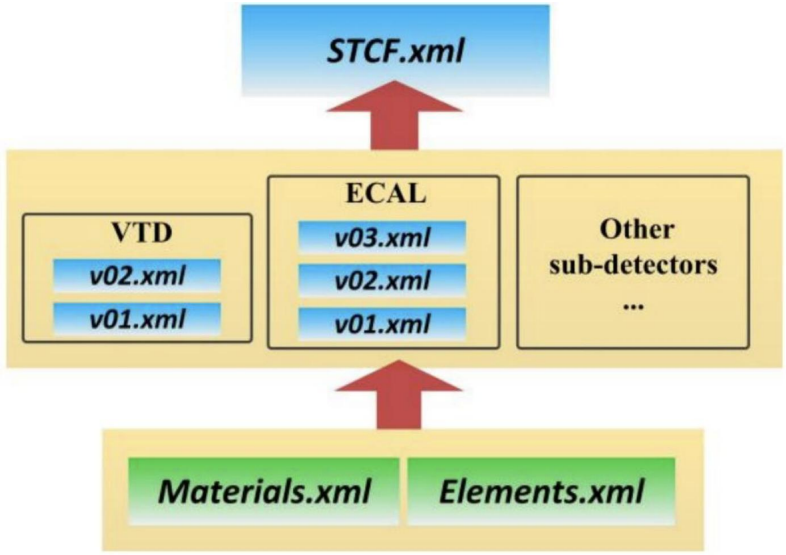
- A powerful detector description management system is necessary across the full offline data processing workflow
  - Provide simple method for **geometry description definition**
  - Provide consistent detector description for all applications
  - Provide **geometry conversion** for different applications, and versioning management
  - Provide interface for conditions data and detector alignment
  - Provide simple and **ready-to-use interfaces** for applications

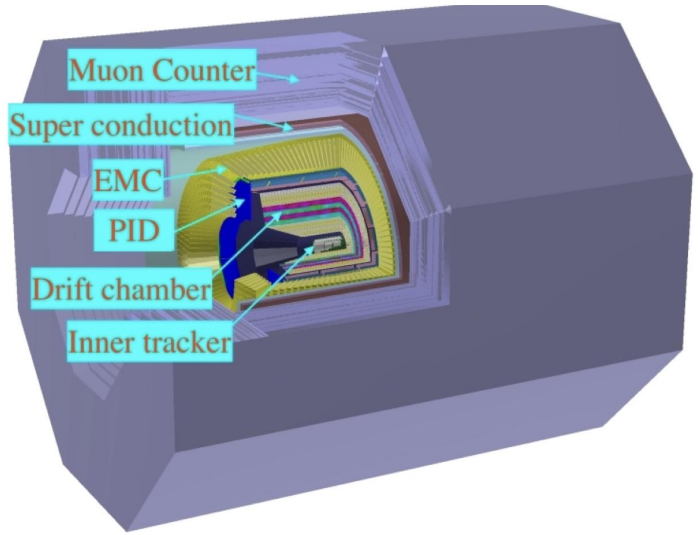


Geometry Management System

- Geometry Management System (GMS) in OSCAR is based on DD4hep
- Single source of detector information for detector description, simulation reconstruction and event display
  - Complete geometry defined with XML files and C++ parser
  - Various plugins for applications
  - Interface for alighment and conditions data

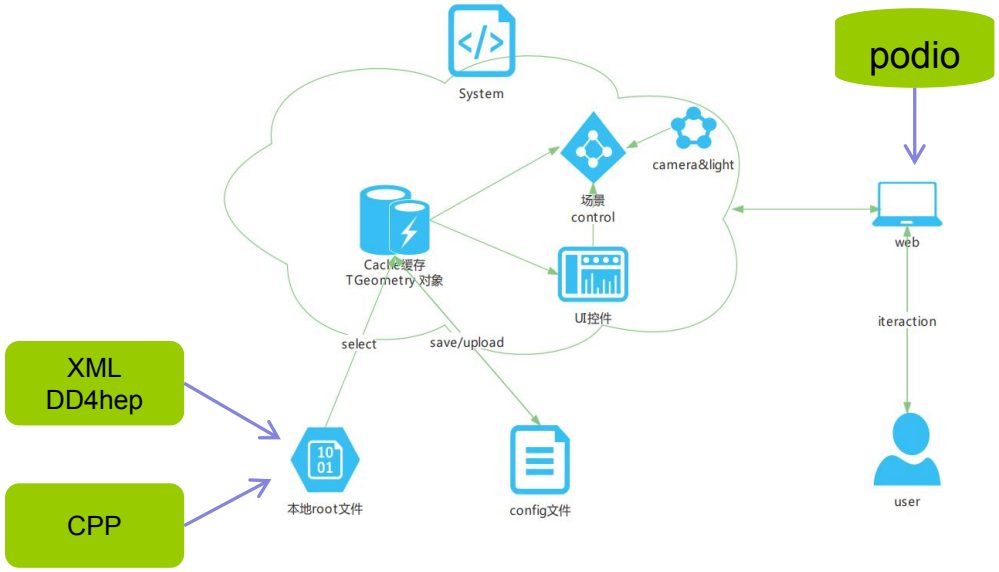


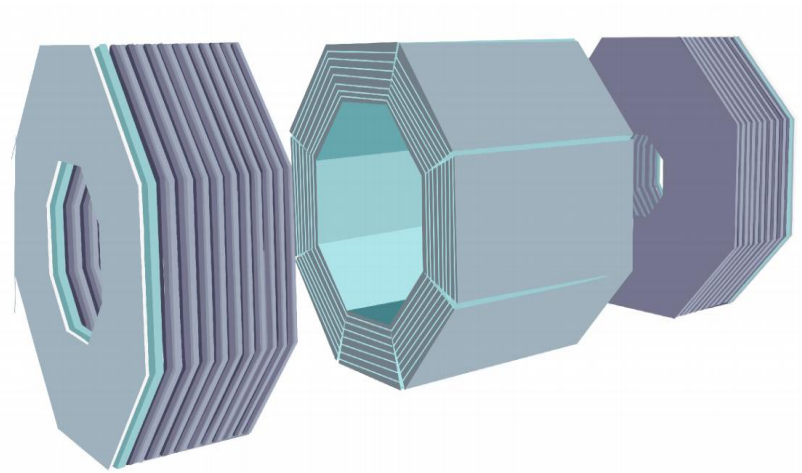


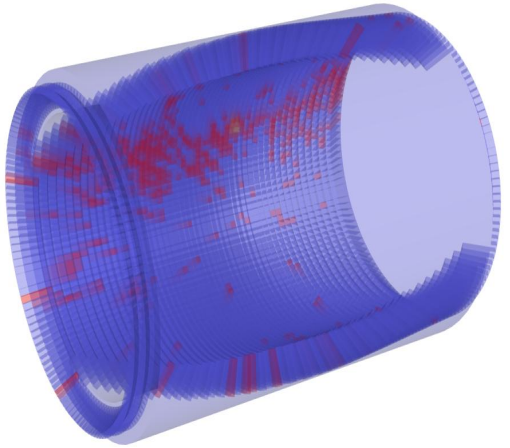


Detector and Event Display

- A common geometry and event display system is being developed
  - Based on Web3D technology and the open-source JSRoot framework
  - 3D engine and graphic libbrary based on Three.JS
  - Using the Vue.js HTML5 development framework to implement the Web interface
  - Reducing 3D motion lag by the multi-threading capabilities of Web Worker framework
  - Geometry information from detector description from DD4hep (XML), and event data read from podio

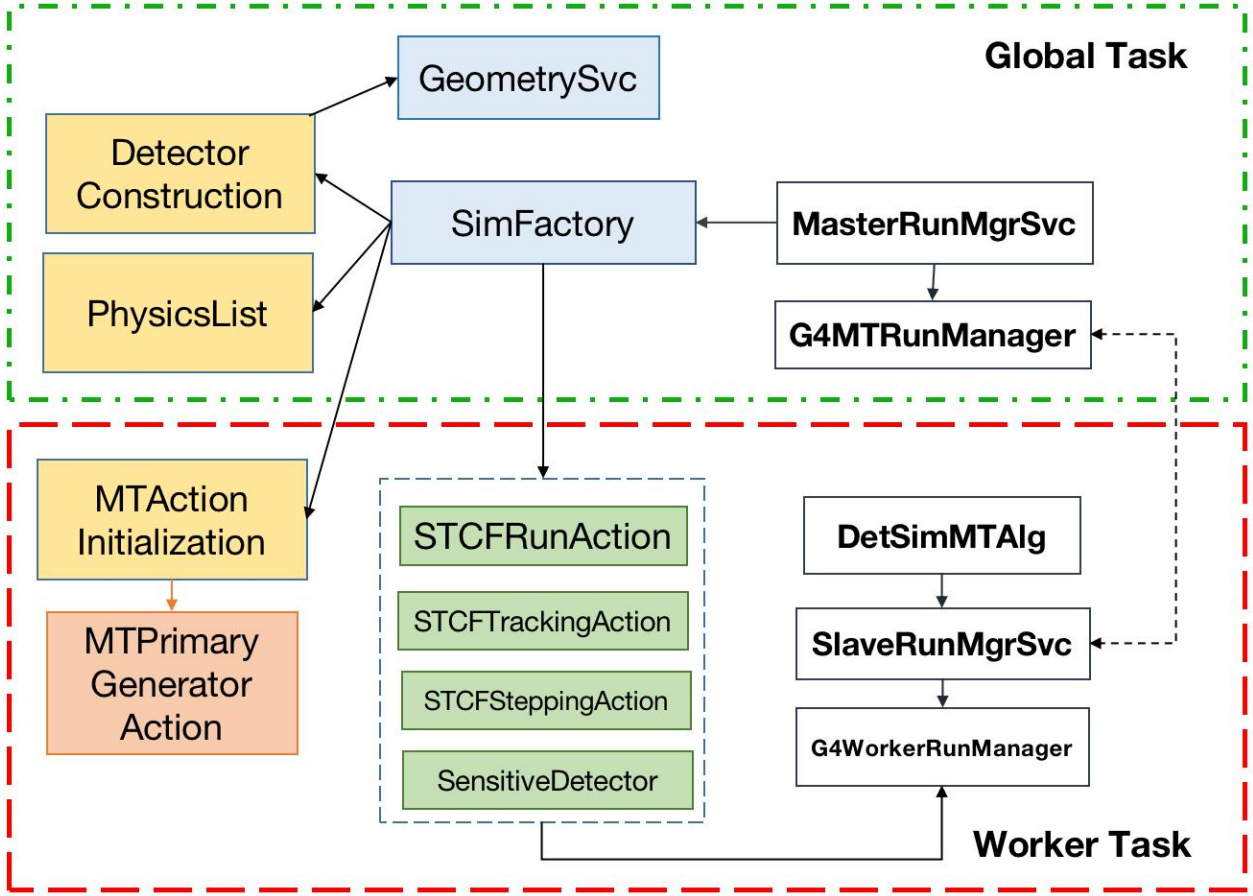


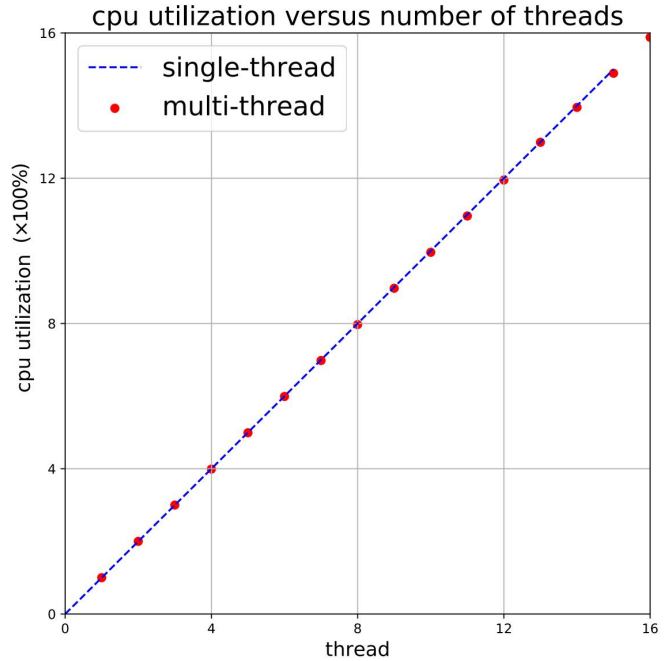


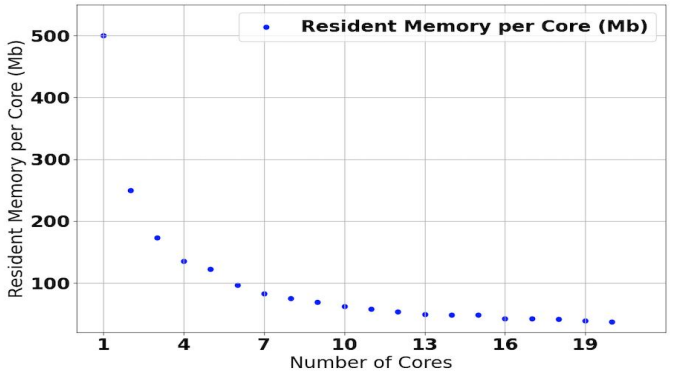


Parallelized Data Processing

- Based on the MT-SNiPER and parallelized DM system, parallelized detector simulation and reconstruction applications are implemented
  - Basic performance tests show promising scalability

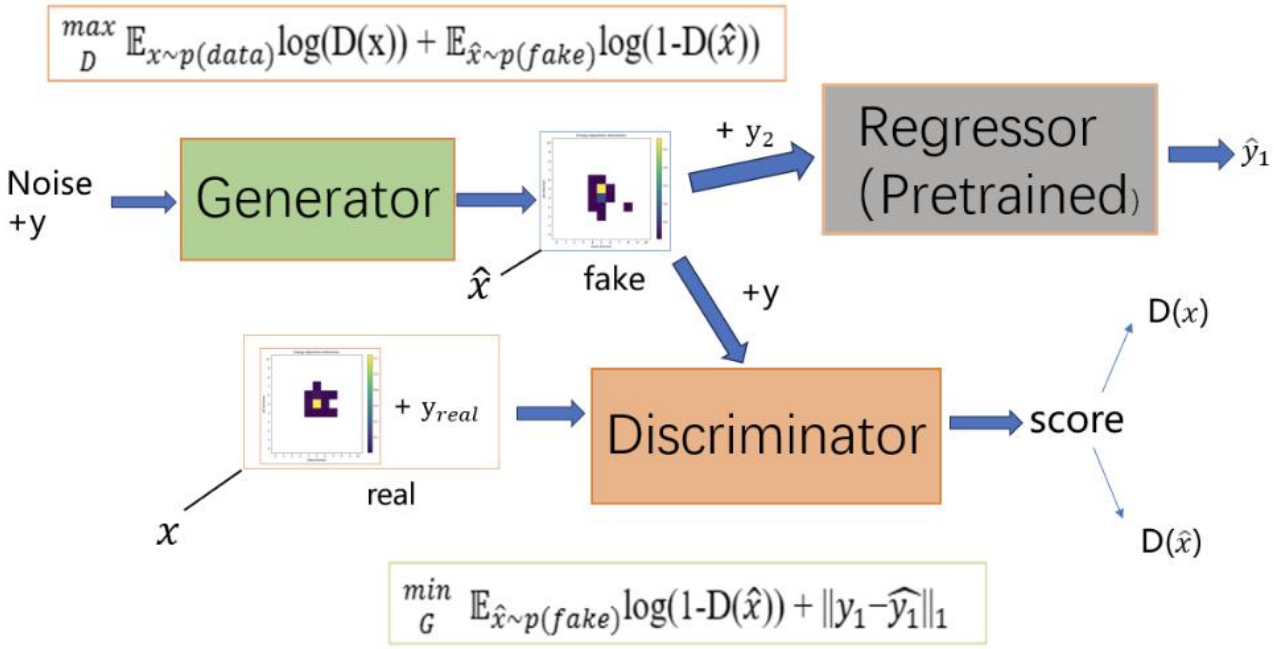


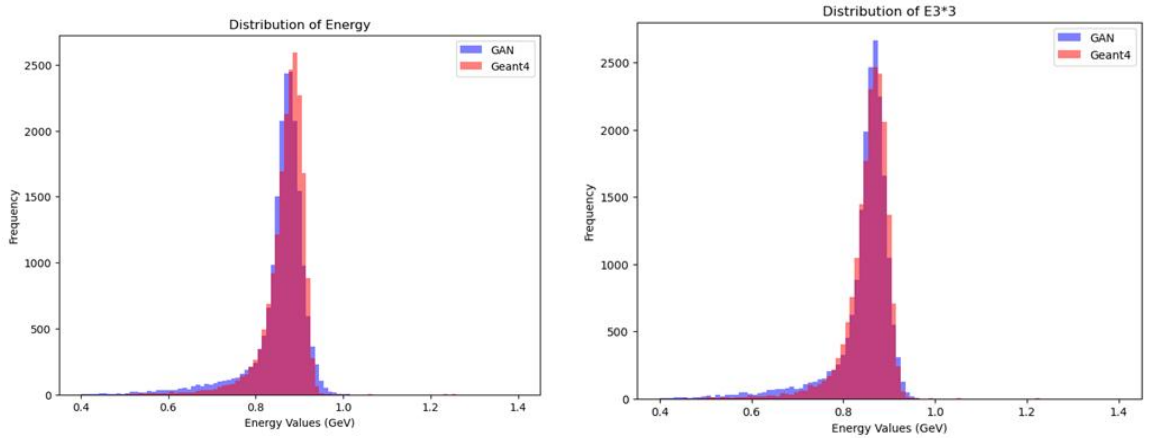


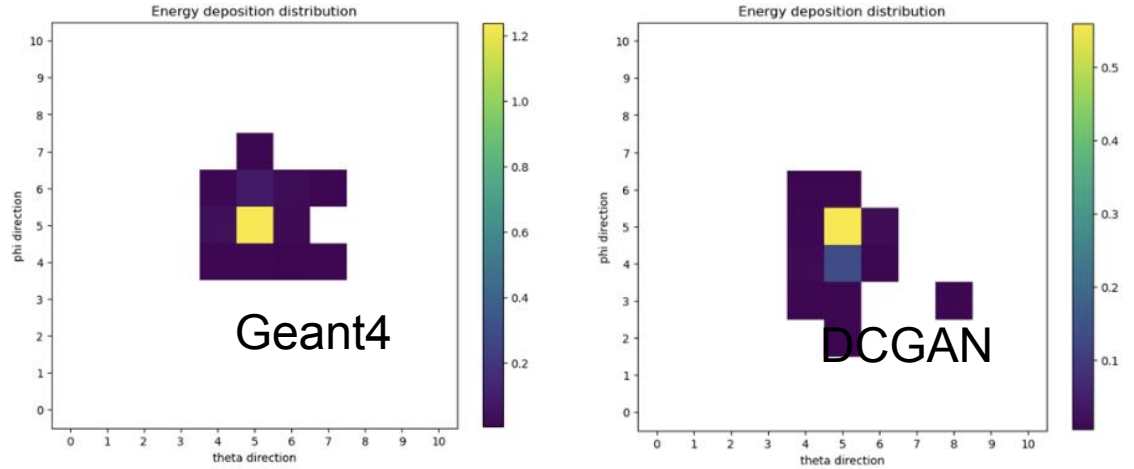


Fast ECAL Simulation based on GAN

- A ECAL fast simulation software based on DCGAN is being developed and optimized
  - Integrate fast simulation and Geant4-based full simulation
  - Expect to speed up the ECAL simulation by 1-2 orders of magnitude



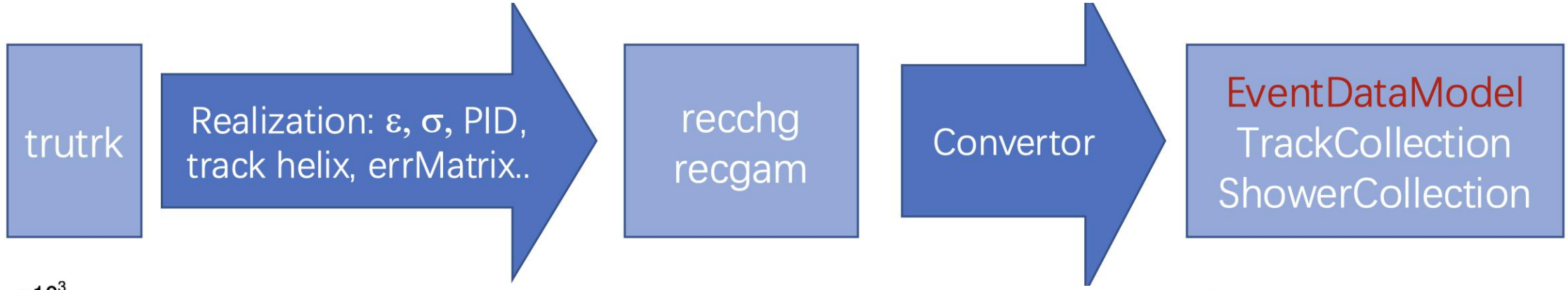


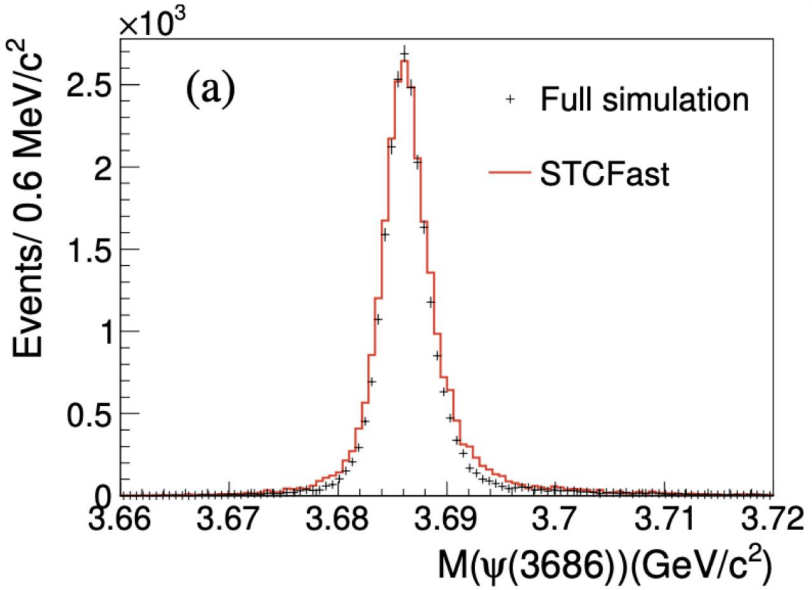


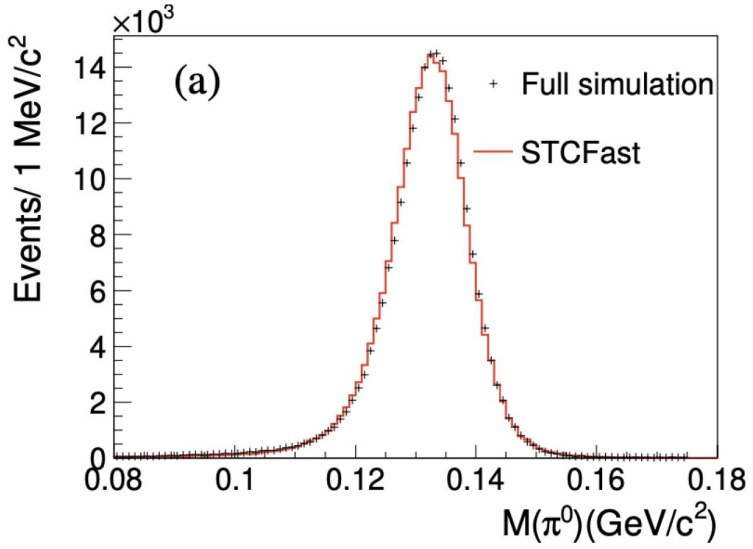
See Yujie's talk later

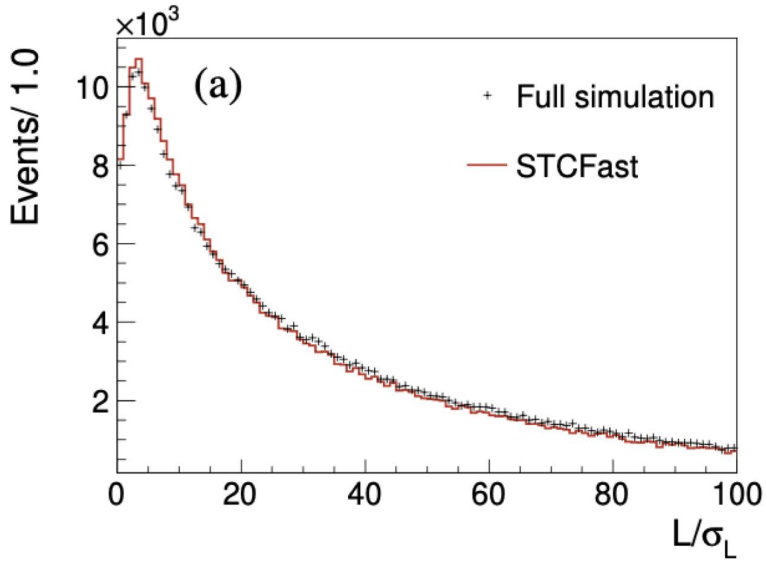
Fast Simulation Framework

- The fast simulation framework is now integrated with OSCAR (more features being developed)
- Flexible for different detecting response and friendly for physics sensitivity study



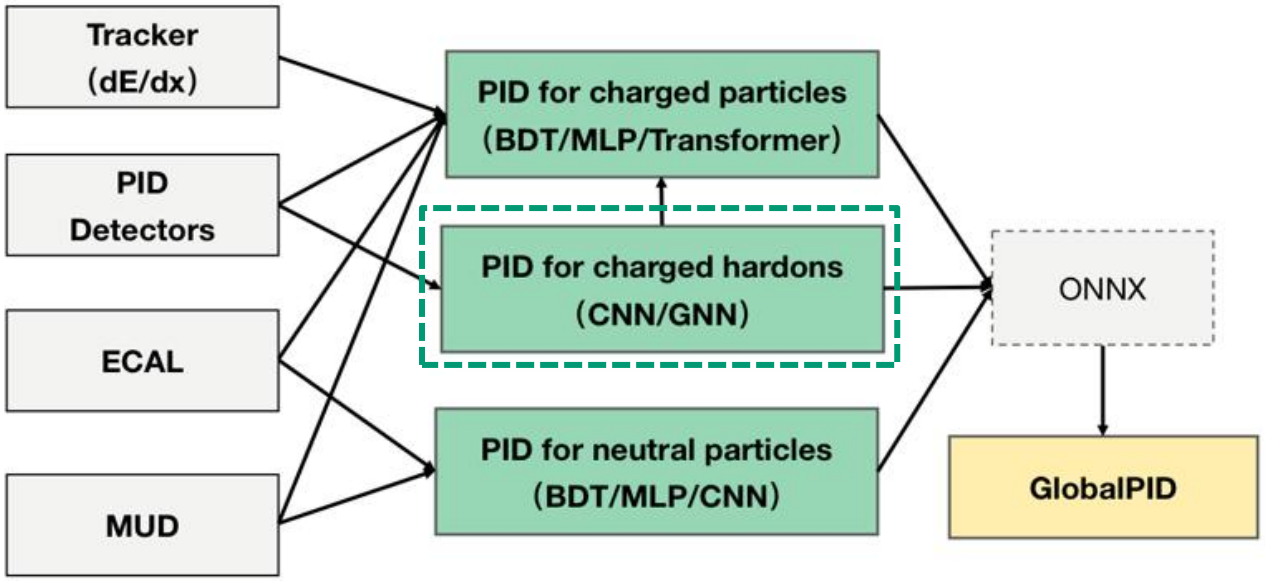




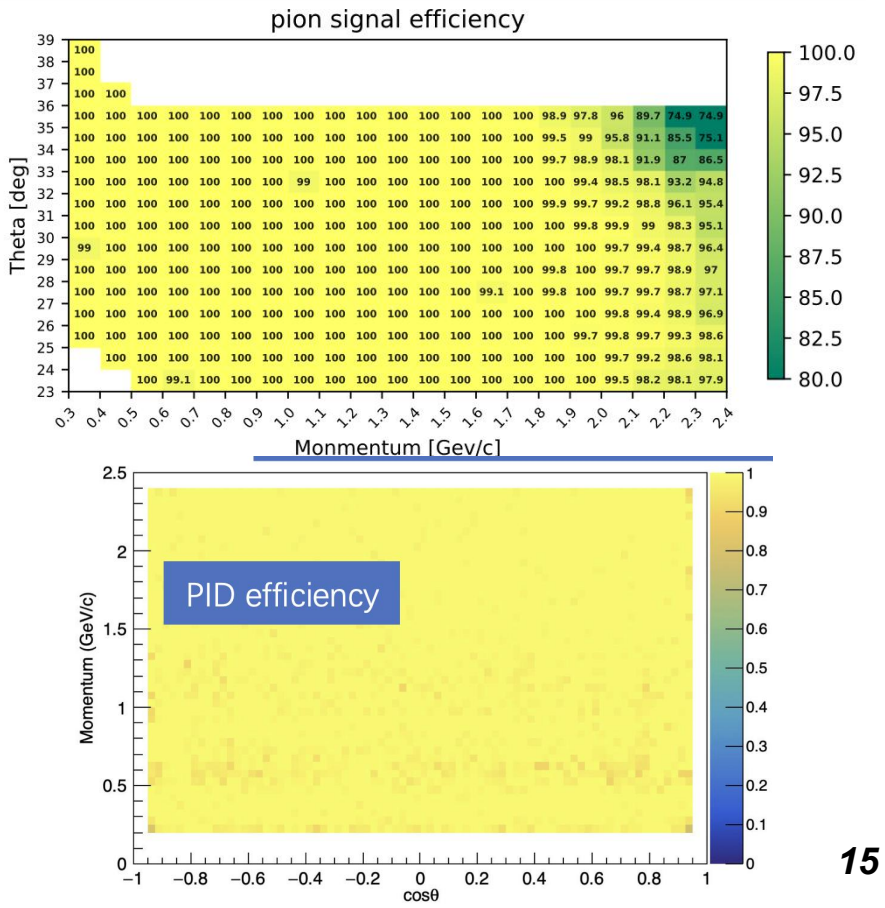


Particle Identification Based on ML

- A global PID software is being developed to enhance STCF PID capabilities
  - Combining information from multiple sub detectors using ML technique
  - Make use of low level detector response data (such as DTOF and RICH readouts)



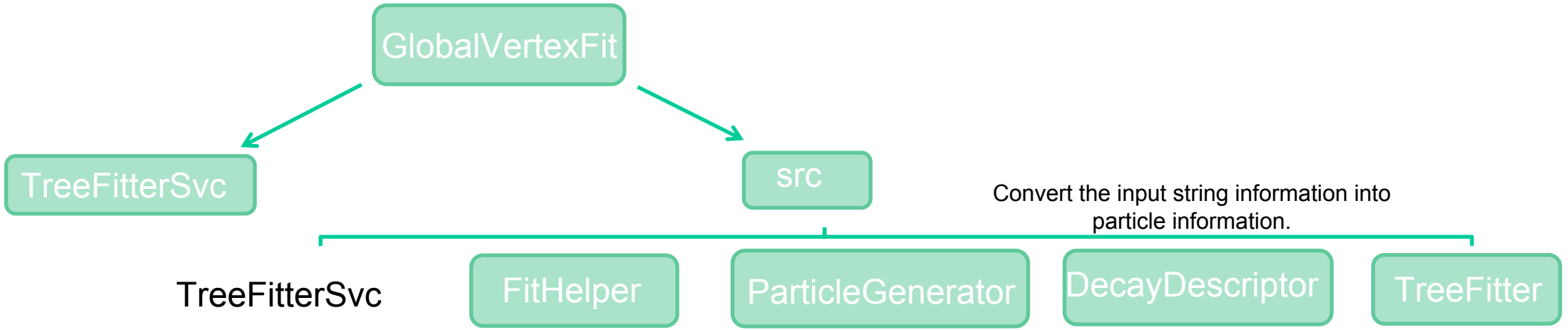
PID algorithms based on machine learning for STCF

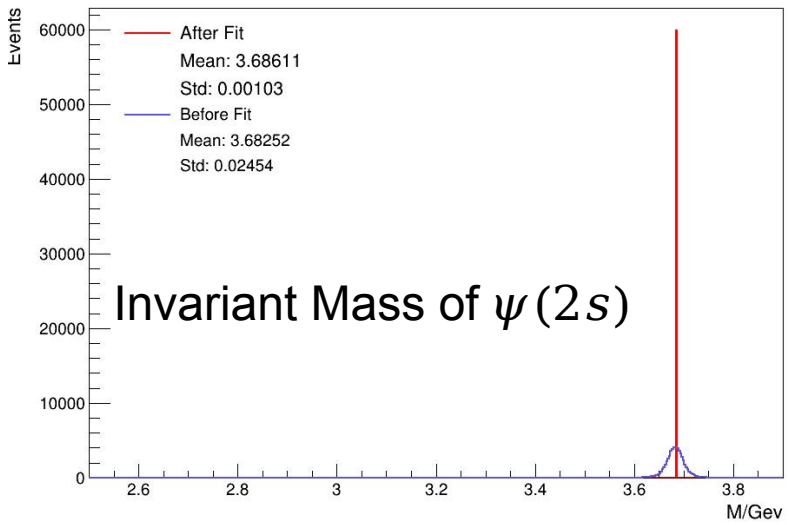


Integrated with OSCAR via ONNX and C APIs
More information in Zhipeng's and Yuncong's talks

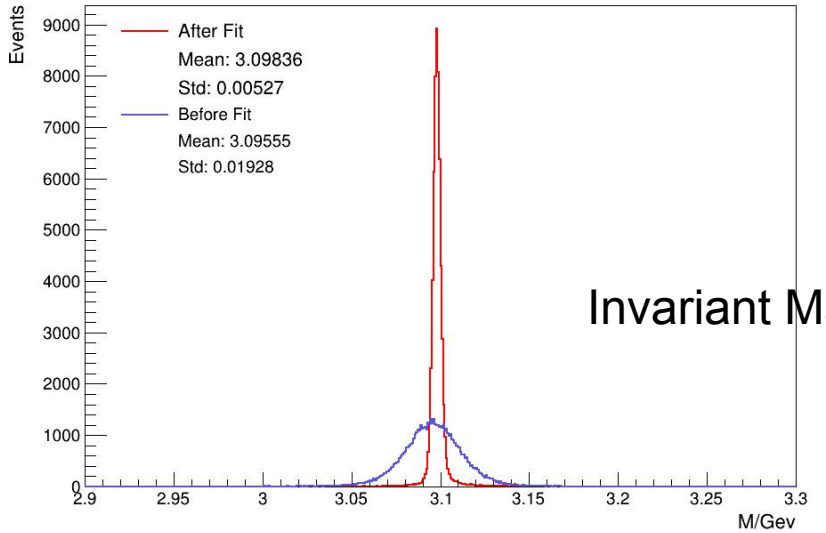
Kinematic and Vertex Fit Toolkit

- Two sets of kinematic and vertex fitting toolkits are ported from BESIII and Belle II





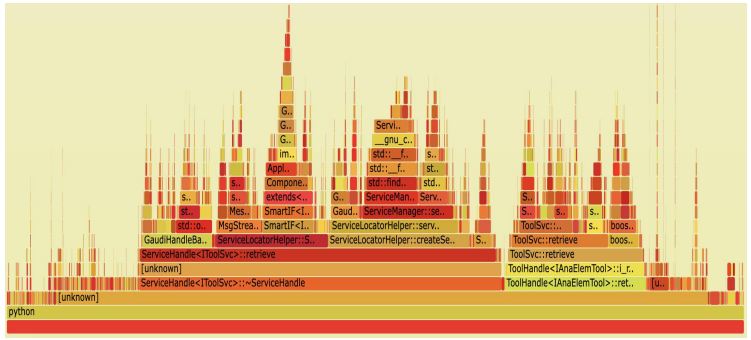
Invariant Mass of $\psi(2s)$

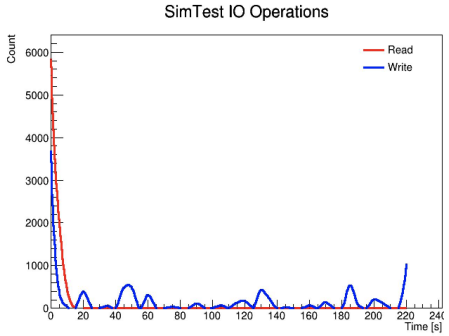


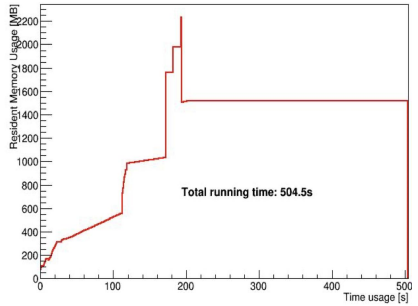
Invariant Mass of J/ψ

Automated Software Validation

- Software validation system is developed, to support building software validation on different levels
  - Unit test, integrated test, software performance profiling and physics result validation
- To be integrated with Gitlab Action for automated validation
  - Used CTest to integrate validation cases integrated with CI system
  - Trigger validation jobs on different levels on schedule/commits








```
-bash-4.2$ ctest
Test project /home/tli/2.5.0_test/offline/build
      Start  1: test_random_svc.py
 1/16 Test  #1: test_random_svc.py ..............   Passed    1.04 sec
      Start  2: test_demoSim.py
 2/16 Test  #2: test_demoSim.py .................   Passed    2.89 sec
      Start  3: test_read_update.sh
 3/16 Test  #3: test_read_update.sh .............   Passed    8.52 sec
      Start  4: test_simple_rw.sh
```

Summary

- We introduced the basic design and functionalities of STCF core software
  - Developed partially based on Key4hep
  - Many components are extended specificlly for STCF, but are also re-usable by other experiments
- Based on the core components, many STCF applications are (being) developed
  - Detector simulation, reconstruction algorithms, event display, analysis toolkit including particle ID, Vertex/KineticFit, RDataframe based analysis framework etc.
  - On-going physics analysis studies with MC data are in progress (see Yuncong's talk)
- We have been continuously improving the core software
  - Software and physics performance has been continuously improved
  - Many applications are being developed based on concurrent/heterogeneous computing and machine learning