STCF离线数据处理软件

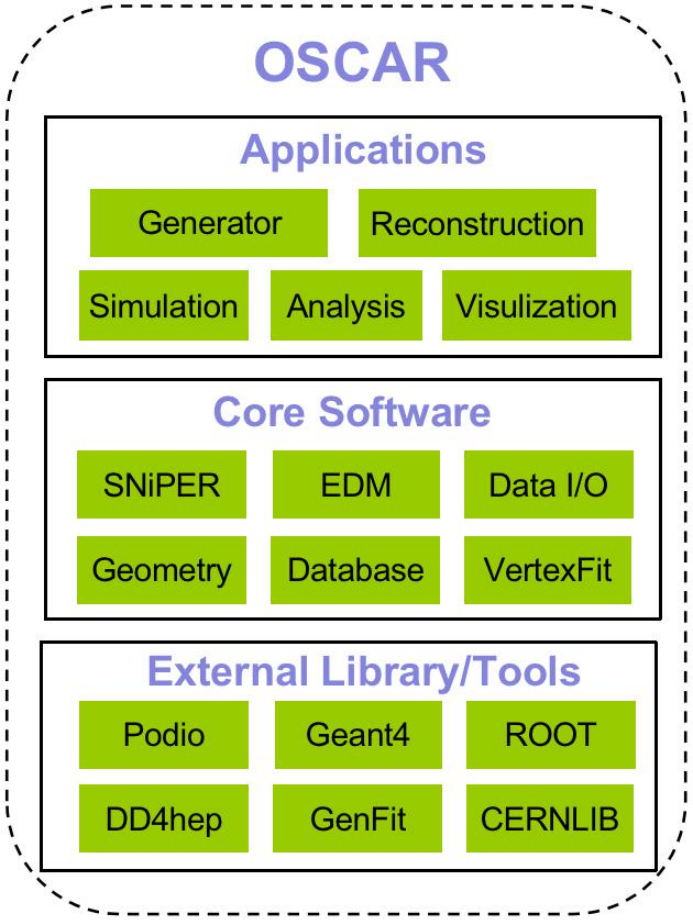
Teng LI

on behalf of the STCF offline software team

2023-06-10

Overview of STCF Offline Software System

- The Offline Software of Super Tau-Charm Facility (OSCAR) is designed for detector design, MC data production and physics analysis
- OSCAR is partially based on Key4hep
  - Reuse some components. Extend others for STCF
- Core software are developed for common functionalities
  - Event loop control (sequently or concurrently)
  - Detector data and event data management
  - Common tools for data analysis
  - Other common services
- Some applications are migrated from BESIII



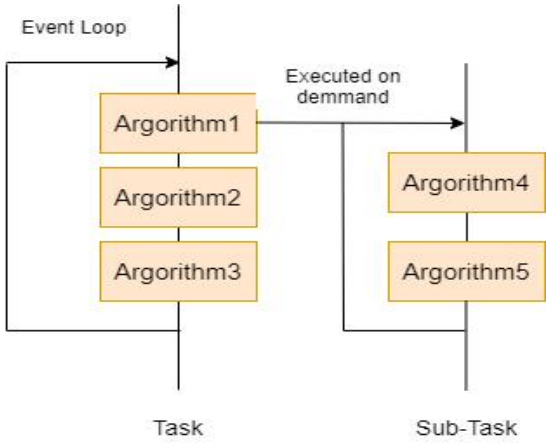
W.H. Huang et al 2023 JINST 18 P03004

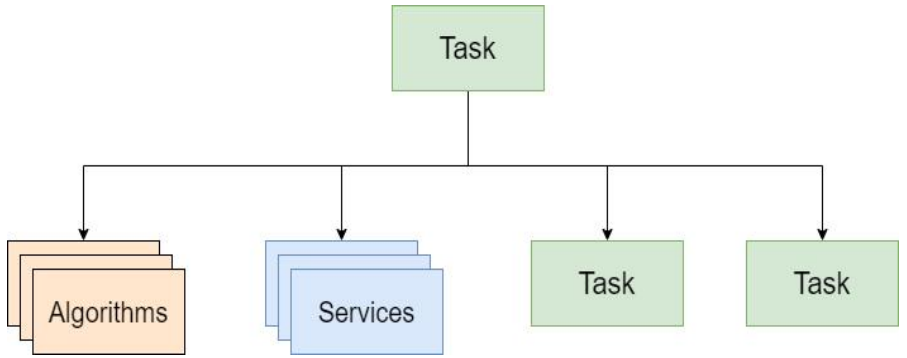
Development Environment

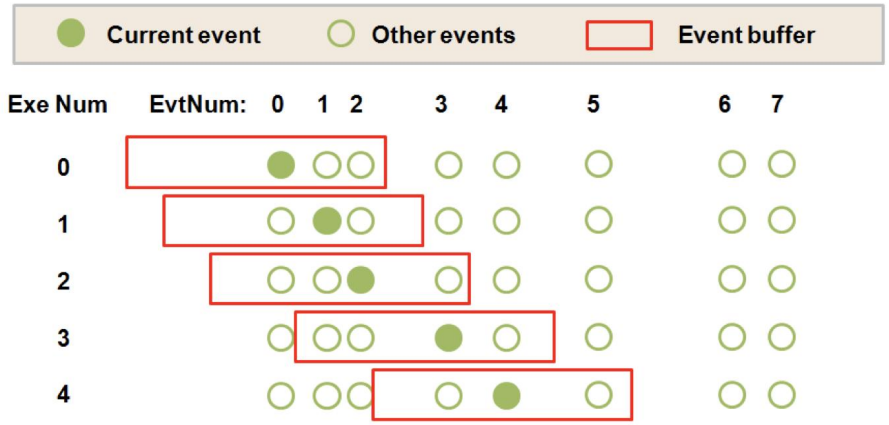
- Supported Operating System: SLC 7 and CentOS 7
- Programming Language: C++ 14, Python 3.8
- Job configuration： Python and Json
- Software management ： CMake
- Version Control Tool： Gitlab
  - URL: http://202.141.163.203:8009/oscar
  - Fork-merge-request, issue tracker, wiki, CI/CD
- Users manual: http://202.141.163.203:8008/oscar_manual

Underlying Framework: SNiPER

- Lightweighted, precisely aimed at small-scaled HEP experiments
- Adopted by JUNO (neutrino), LHAASO (cosmic ray), nEXO (neutrinoless double beta decay) and HERD (dark matter)
  - Provide basic functionalities of event loop control, application interface, job configuration, logging etc.
- Advantages of SNiPER
  - Lightweighted, efficient, highly extendable. Flexible event loop control. Flexible to be integrated with other software, e. g. podio, ROOT, ...
  - C++/Python hybrid programing, highly configurable. Efficient multithreading.



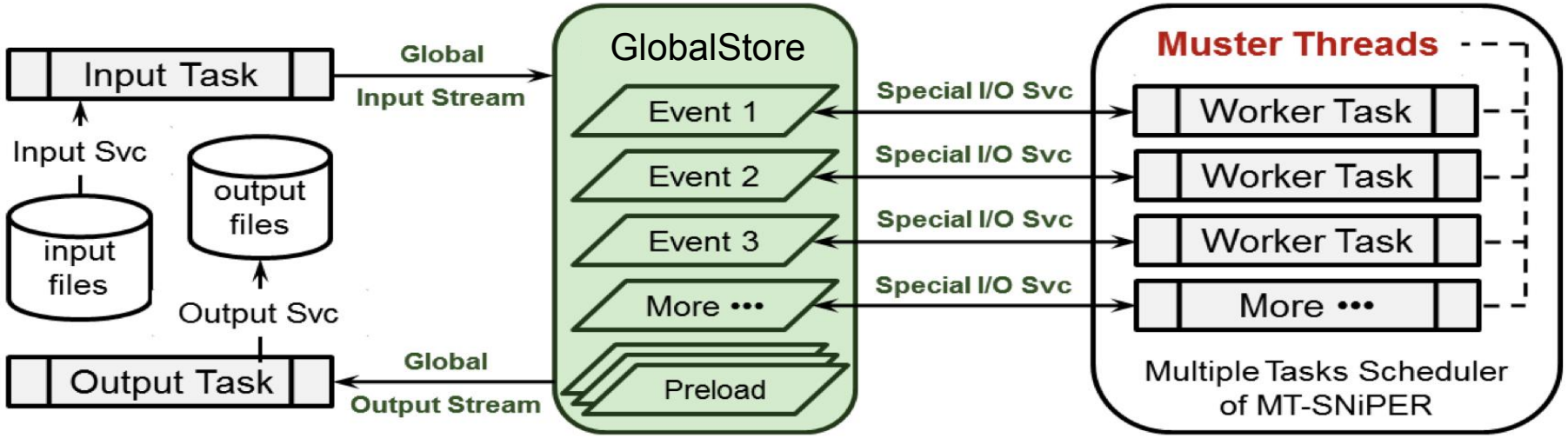




reference: J. H. Zou et al J. Phys.: Conf. Ser. 664 (2015) 072053
J. H. Zou et al EPJ Web Conf. 214 (2019) 05026

Parallelism in MT-SNiPER

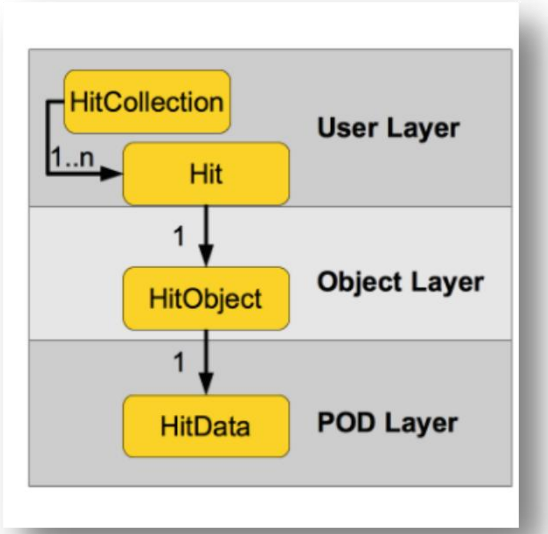
- SNiPER provides simple interfaces for building multithreaded applications
  - Based on Intel TBB
  - SNiPER Muster (Multiple SNiPER Task Scheduler) works as a thread pool/scheduler
  - Data I/O is binded to dedicated I/O thread for flexibility
  - A Global Store is developed to support multithreaded event data management
  - Application code is mostly consistent for serially and parallelly execution



Event Data Model Based on Podio

- Event Data Model (EDM) lies at the heart of OSCAR
  - Define the structure of event data in memory and in data files
  - Implement relationship between data objects (hit-track-MC particle)
  - Handle schema evolution
- EDM is defined based on podio (Key4hep, adopted by FCC CEPC, ILC, ...)
  - Generate C++ code based on YAML definition
  - Support both C++ and Python
  - Good multithreading support
  - Powerful and flexible relationshop between data objects
  - Support multiple data file format

F. Gaede, etc. , CHEP2019

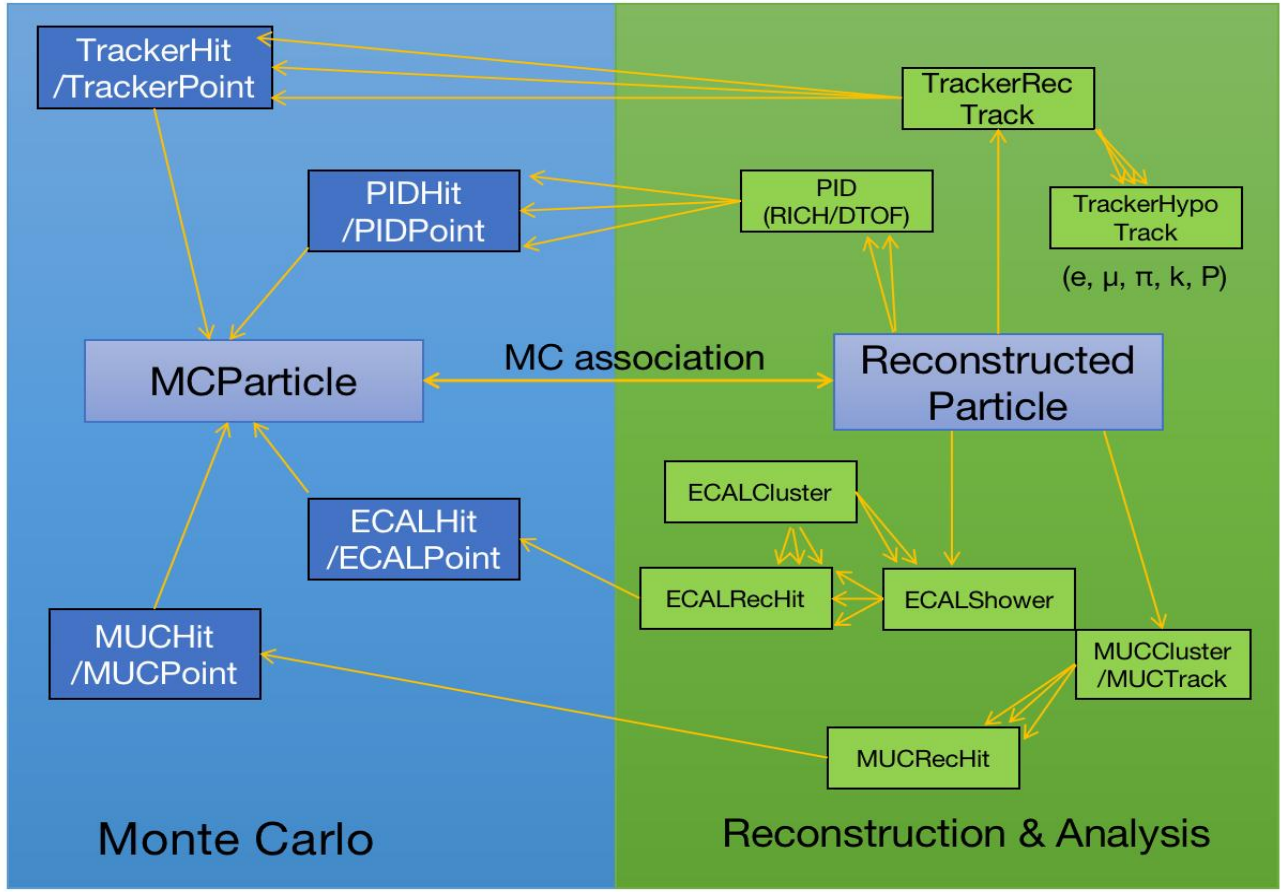


https://github.com/AIDASoft/podio

Event Data Model Based on Podio

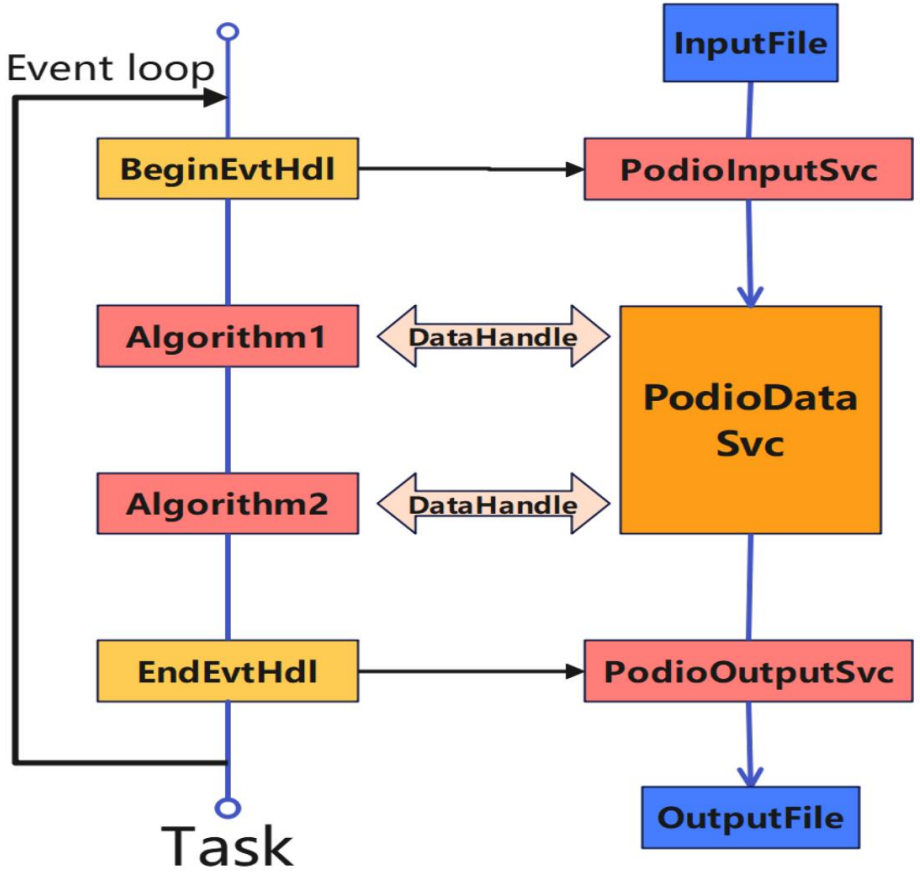
- Due to the specific requirements of STCF, EDM4hep is not directly used
- Design EDM classes based on Podio and reuse some EDM4hep classes

- Re-use MCParticle and ReconstructedParticle in EDM4hep as the core index
- Design EDM classes specificly for STCF simulation and reconstruction (for the PID system, and contains more information for detector optimization and physics analysis)
- MCParticle and ReconstructedParticle are correlated based on track matching algorithm, bridging MC and reconstructed data



Event Data Management

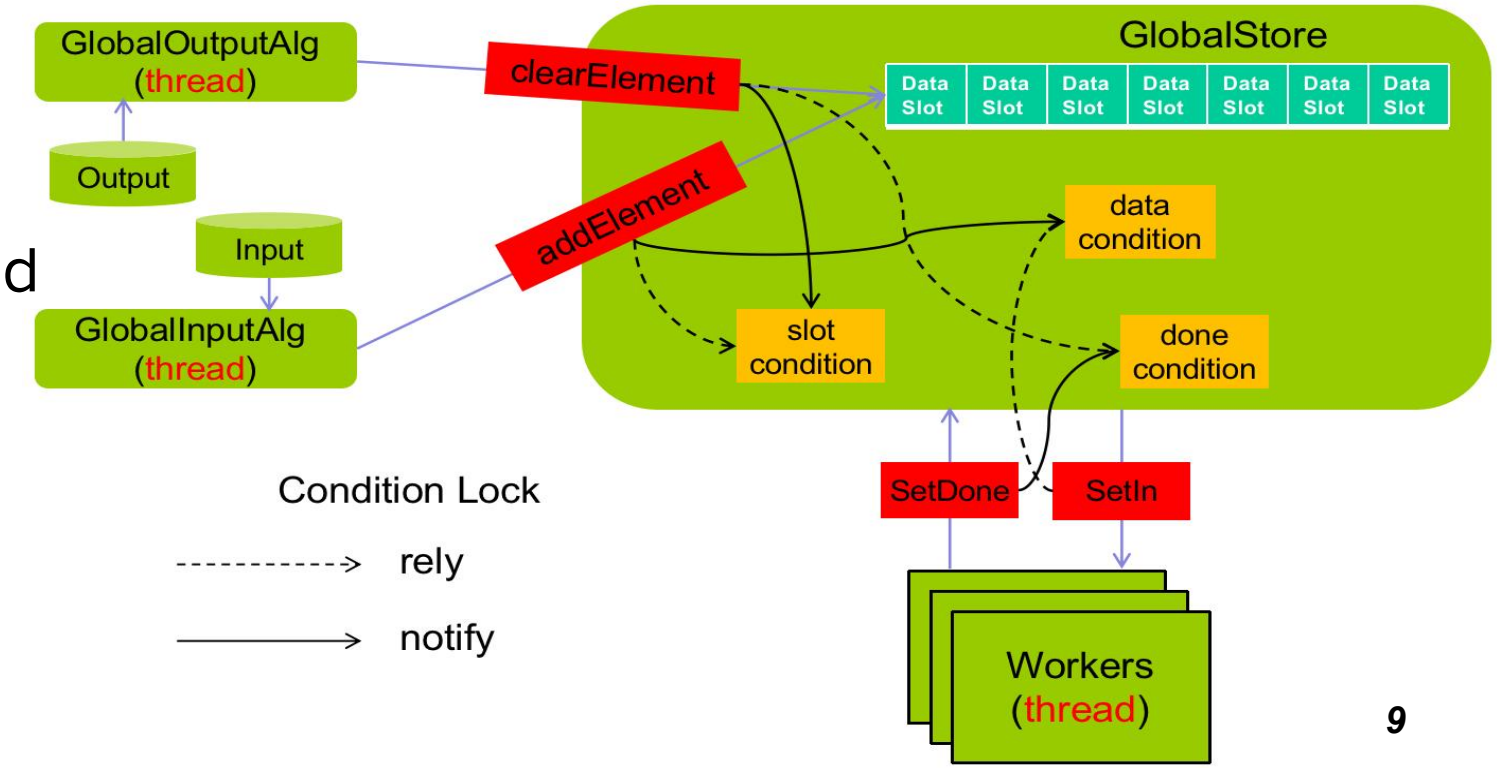
- Event data management system manages event data in memory, provides interfaces for user applications and handles data I/O
- Extend SNiPER DM system based on Podio
  - PodioDataSvc: memory management
  - PodioInputSvc: data input
  - PodioOutputSvc: data output
  - DataHandle: interface
- Event data and user application are completely decoupled



W.H. Huang et al 2023 JINST 18 P03004

Parallelized Event Data Management

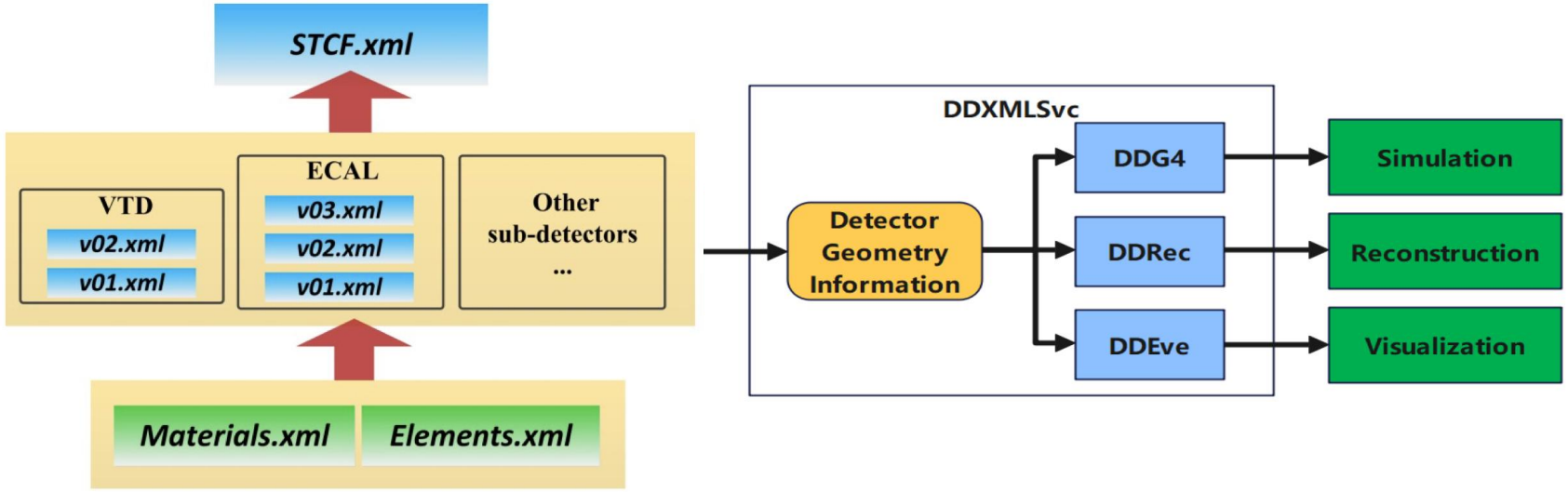
- To enable parallelized data processing, a GlobalStore is developed based on Podio
  - Re-implement podio::EventStore to cache multiple events (each within one data slot)
  - Use several condition lock to enable safety exchanging data between threads
  - I/O services are binded to dedicated I/O threads,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



Geometry Management System

- Detector description in OSCAR is based on DD4hep
- Single source of detector information for detector description, simulation reconstruction and event display
  - DDG4 for delivering detector geometry to Geant4
  - DDRec for delivering detector geometry to reconstruction algorithms
  - DDXMLSvc: the unified interface to DD4hep, including DDG4 and DDRec

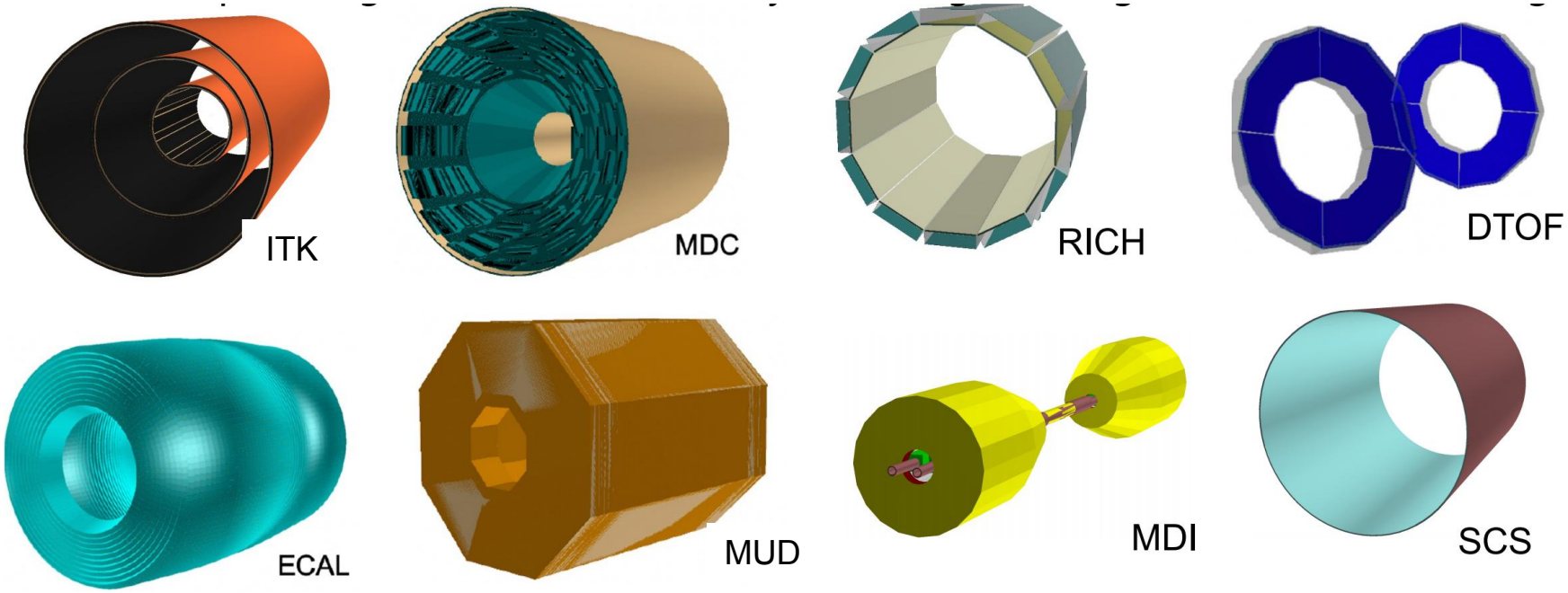
Flexible combinations of different versions of detector design, and combinations of sub-systems



H. Li et al 2021 JINST 16 T04004

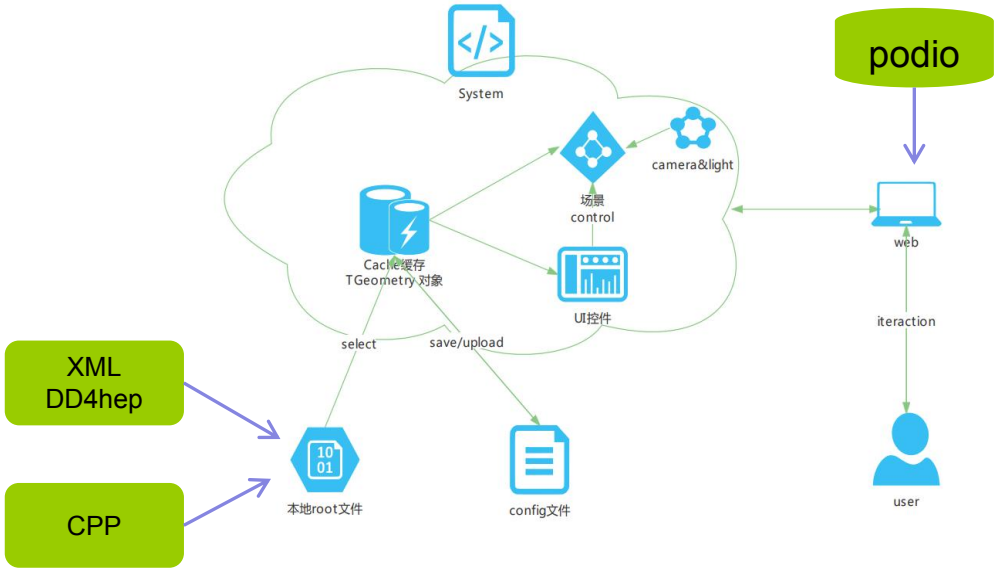
Detector Geometry Description

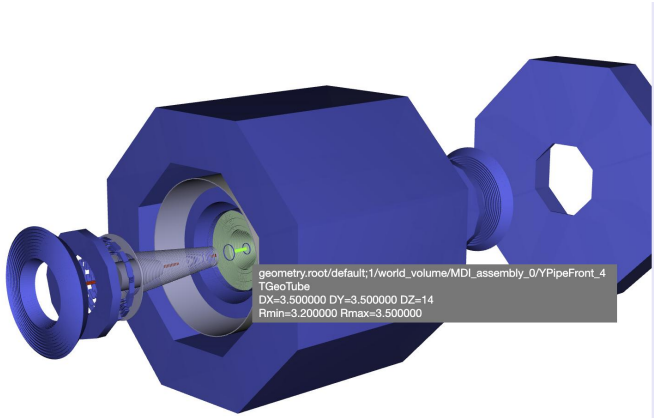
- The Full STCF Detector is described with DD4hep
- Each sub-detector is implemented with a single compact file
- The version number is used for different design options
- Optimizing the detector geometry according to changes of the detector design

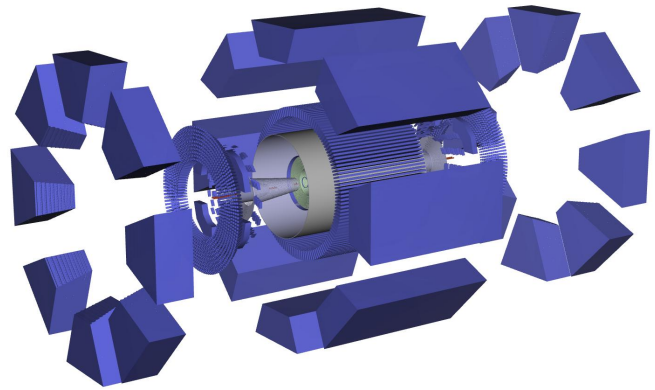


Geometry and Event Display

- A common geometry and event display system is being developed
  - User interface and 3D display based on WebGL
  - 3D engine and graphic libbrary based on Three.JS
  - Read geometry information from detector description based DD4hep (XML)
  - Event data read from Podio

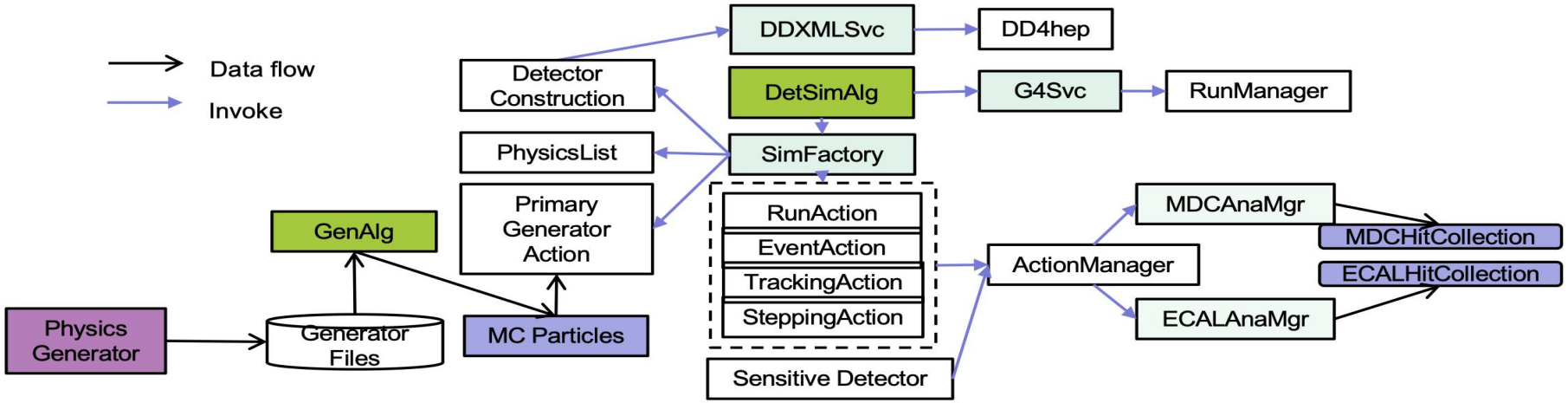




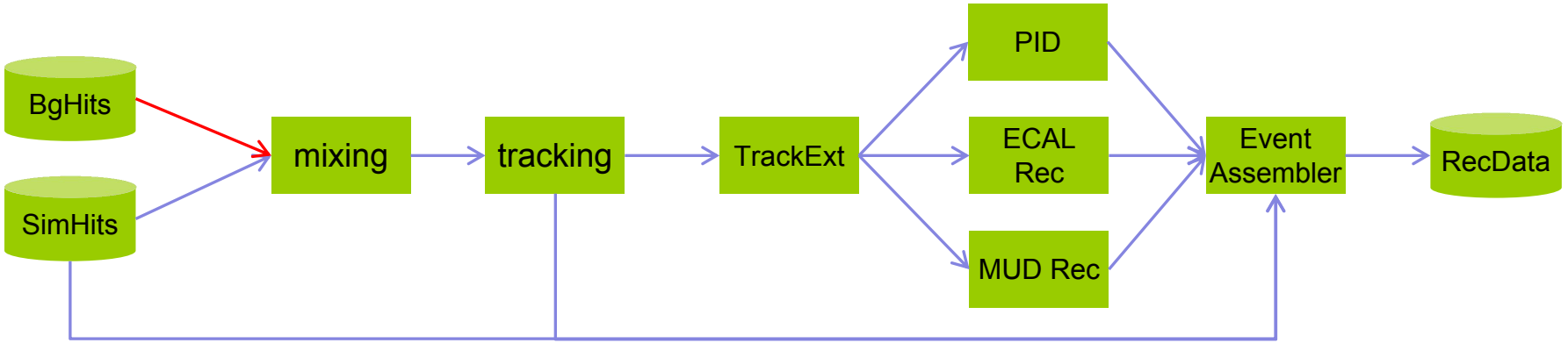


Detector Simulation and Reconstruction

- Full chain of detector simulation has been built
  - Flexible configuration of generator, geometry, user actions

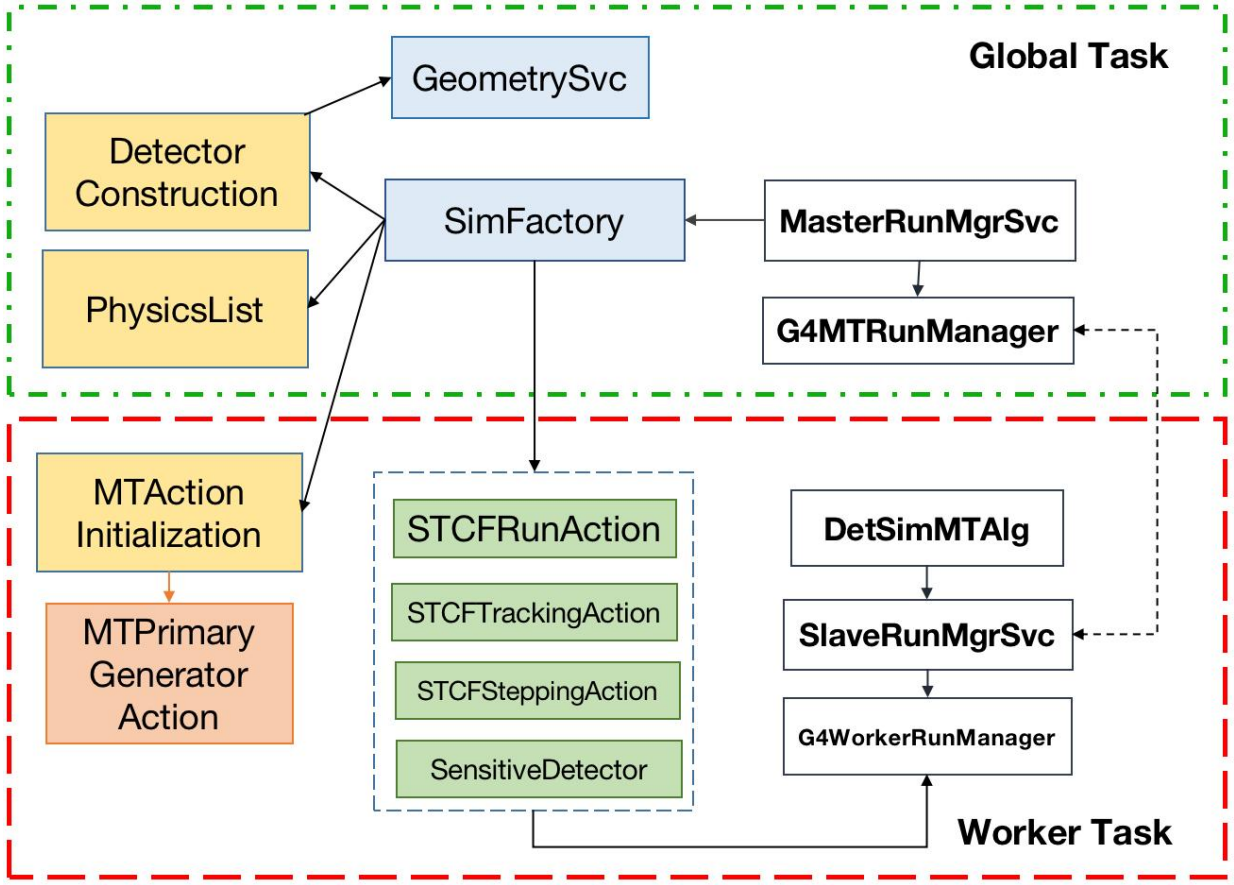


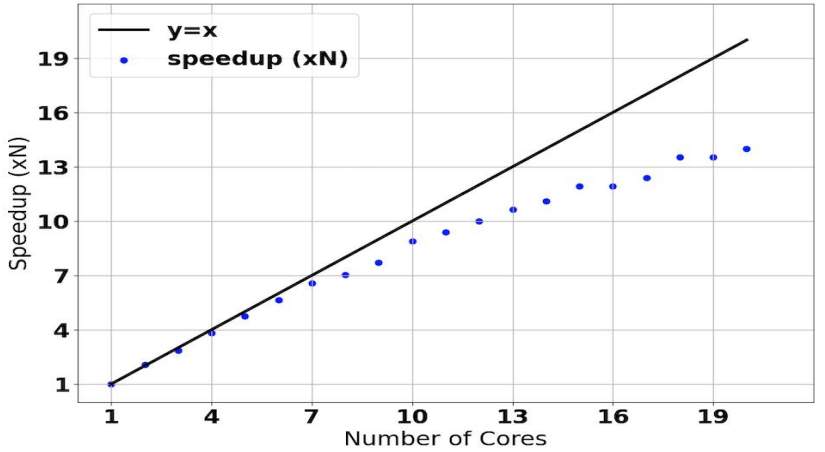
- Reconstruction chain

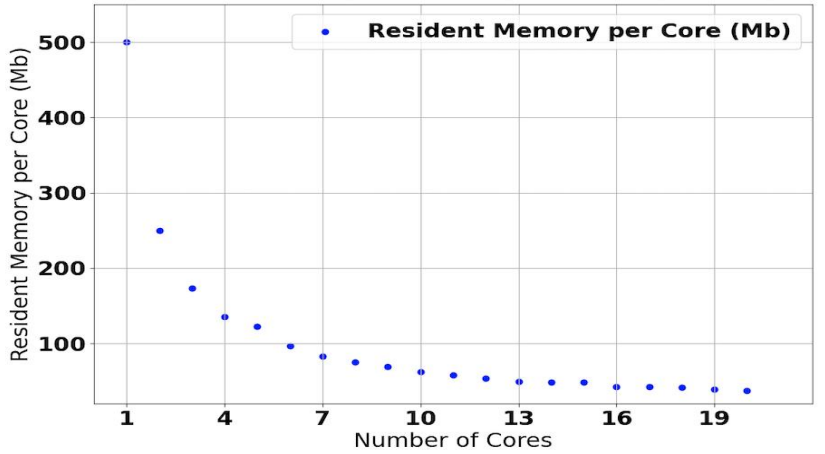


Parallelized Detector Simulation

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

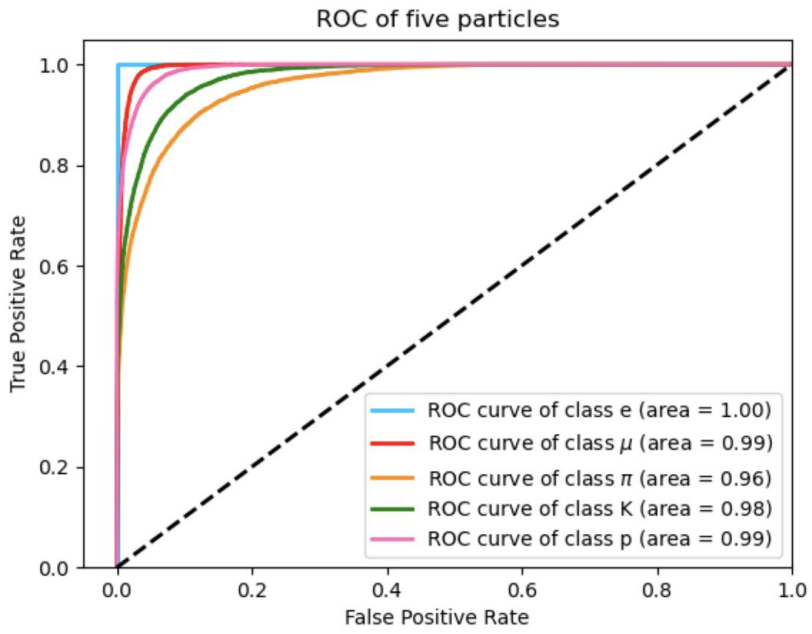


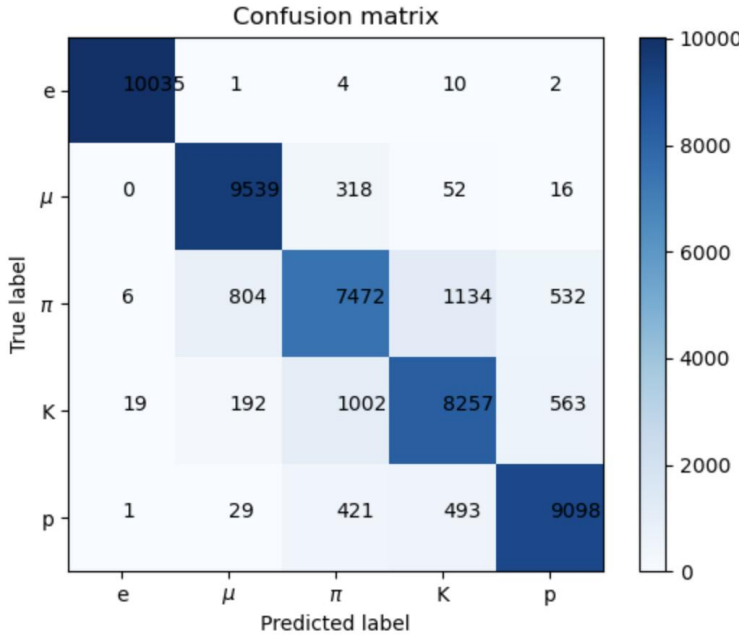


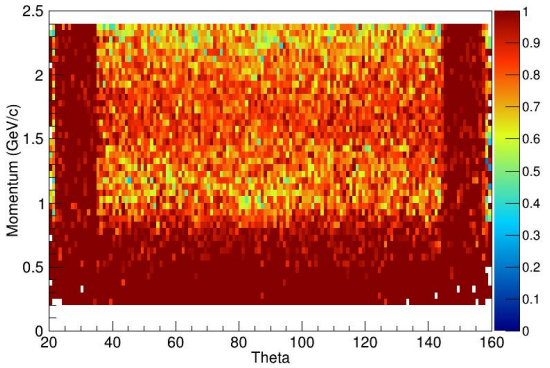


Global PID Software

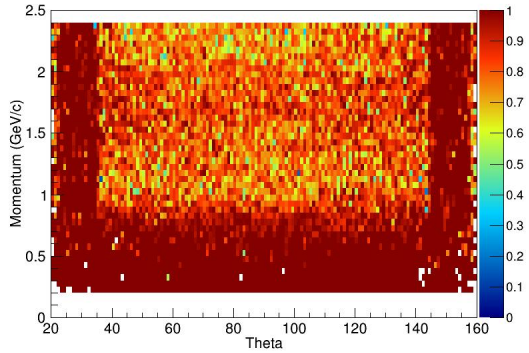
- To expoilt the detector sufficiently, a global PID software that takes all information from sub-detectors is developed based on ML
  - Based on data-driven method, extract features from many correlated variables and perform PID for charged particles (e/μ/π/K/P)
  - Based on XGBoost C API, integrated into OSCAR





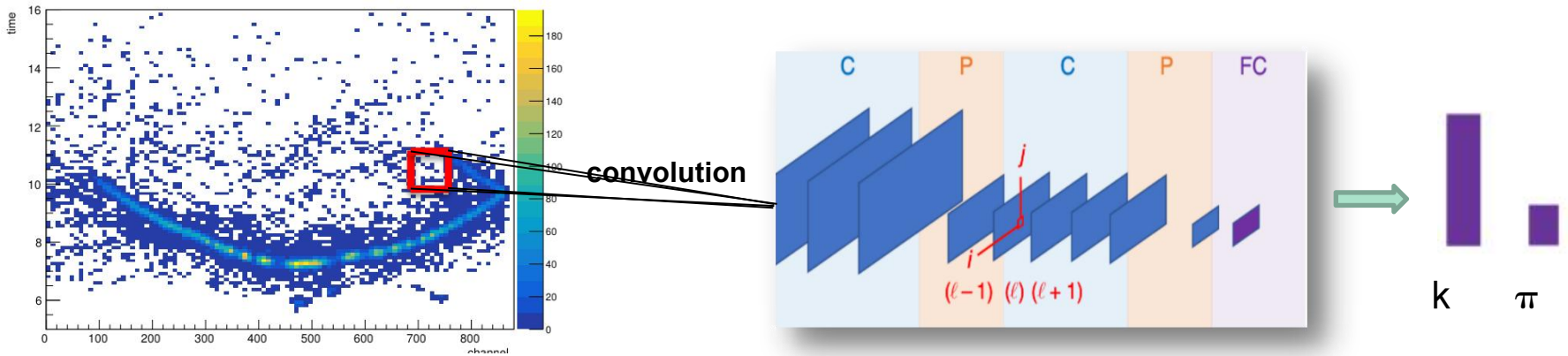


K+/K- efficiencies

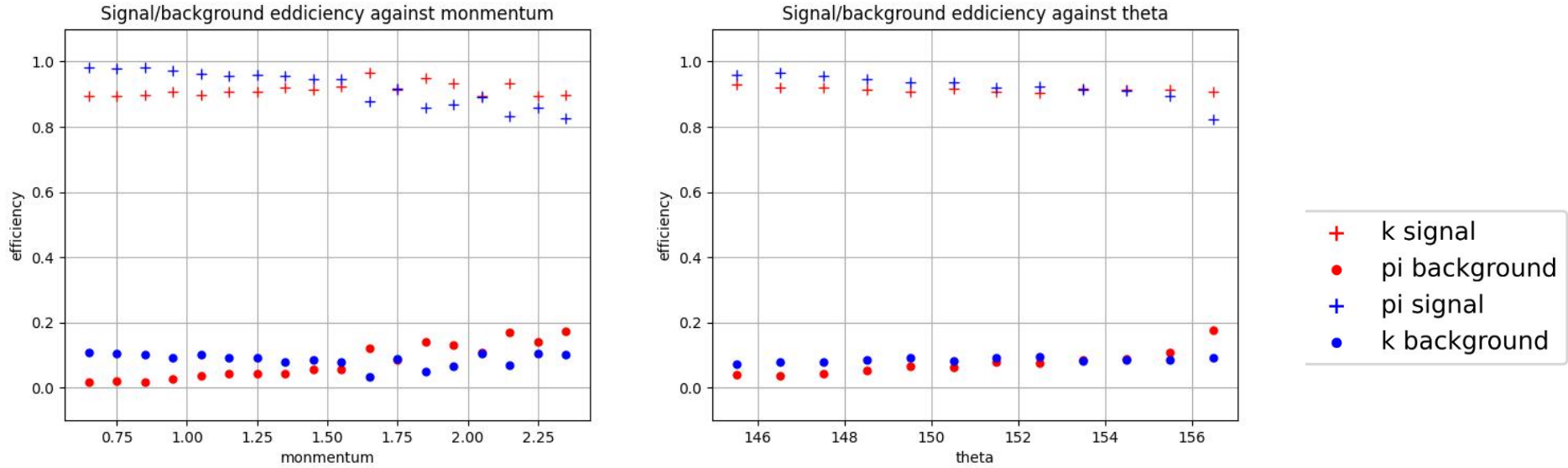


PID Software Based on CNN

- Construct pixel-map according to hit-time and -position of Cherenkov photons, as input of convolutional neural network for PID

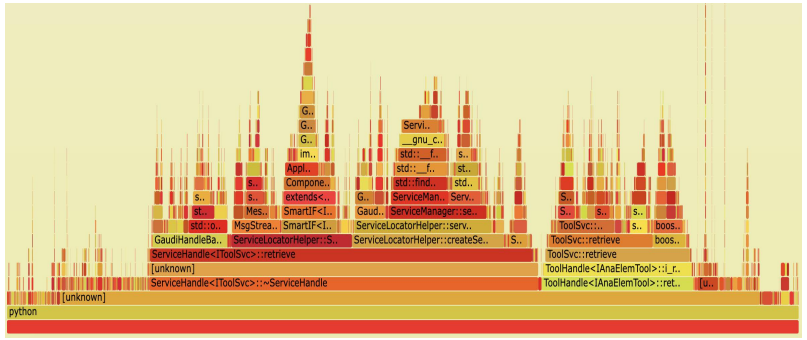


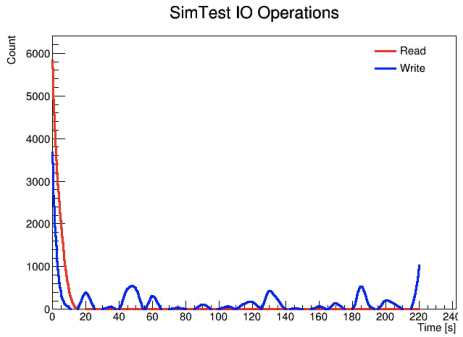
- Applied to RICH and DTOF. The output PID likelihood could be further fed into global PID software

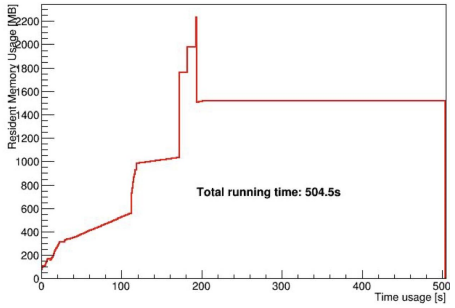


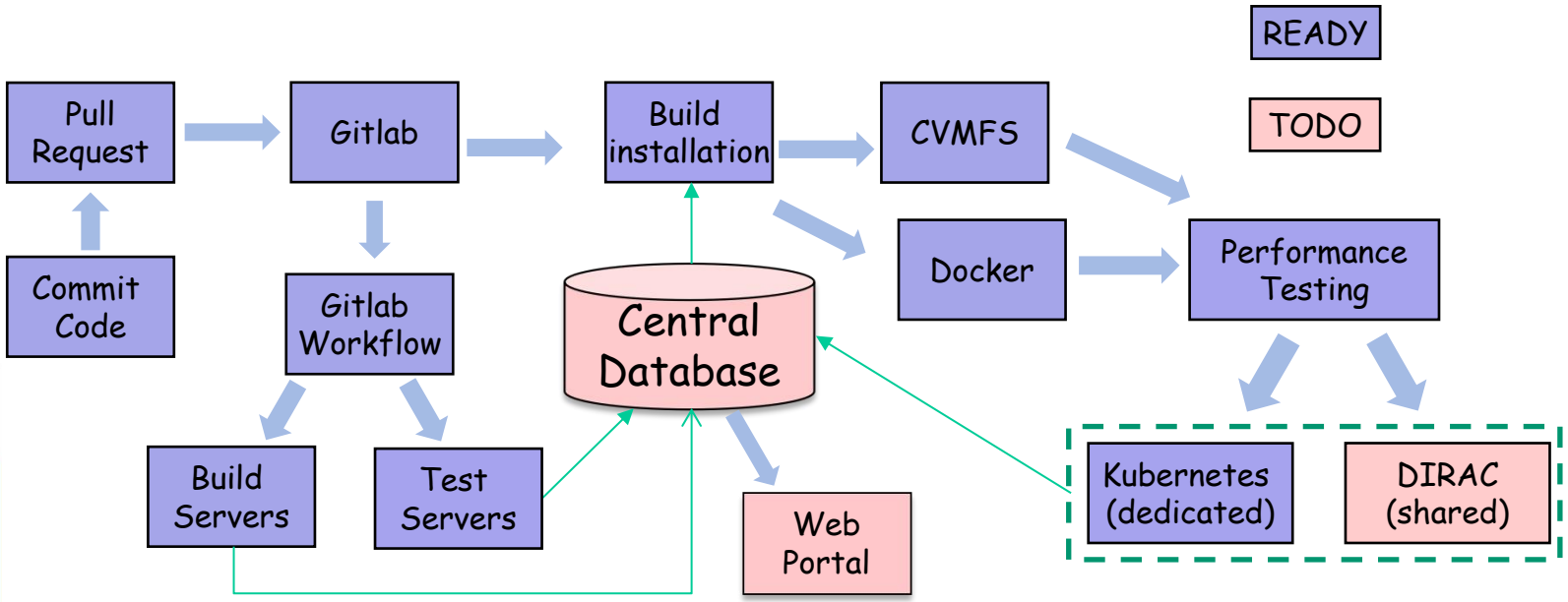
Automated Software Validation

- A software validation toolkit is developed, to support building software validation on different levels
  - Unit test, integrated test, software performance profiling and physics result validation
- Integrated with Gitlab Action system for automated validation
  - Trigger validation jobs on different levels on schedule/commits
  - Same system is being adopted by CEPC and Key4hep as well









Summary

- We introduced the basic design and functionalities of STCF offline software system (OSCAR), developed since a few years ago
  - 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
  - Some algorithms ported from BESIII
  - Detector simulation, reconstruction algorithms, event display, analysis toolkit such as particle ID, Vertex/KineticFit, RDataframe based framework etc.
  - Now support preliminary physics analysis with MC data
- We have been continuously improving OSCAR based on new technologies
  - Physics performance of reconstruction algorithm has been continuously improved
  - Many applications are being developed based on concurrent/heterogeneous computing, machine learning and quantum computing (see talks in the following sessions)