Core Software of STCF

Teng LI on behalf of the STCF core software development team

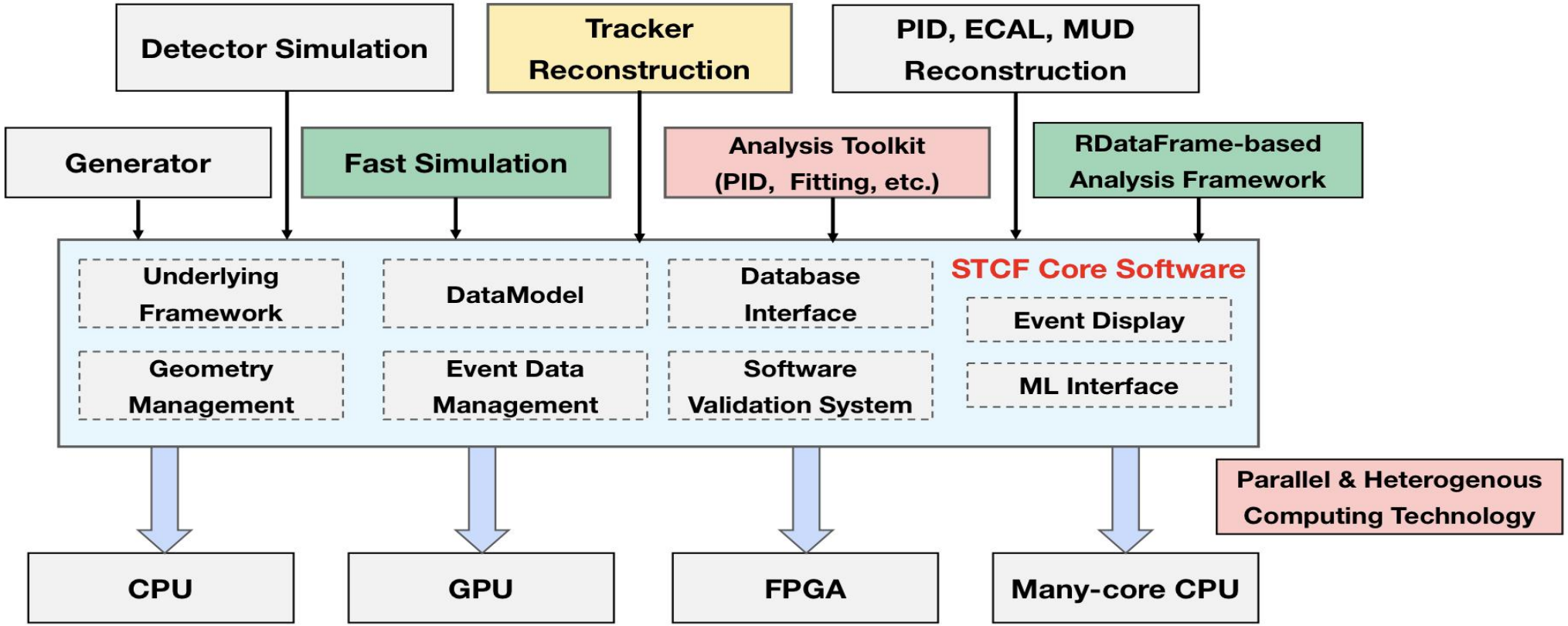
Shandong University

The 2024 International Workshop on Future Tau Charm Facilities

2024-1-17, Hefei

Introduction

- The task of the STCF core software
  - To provide the common software platform for the entire offline data processing
  - To support detector simulation, calibration, reconstruction and data analysis
- The scope of the STCF core software

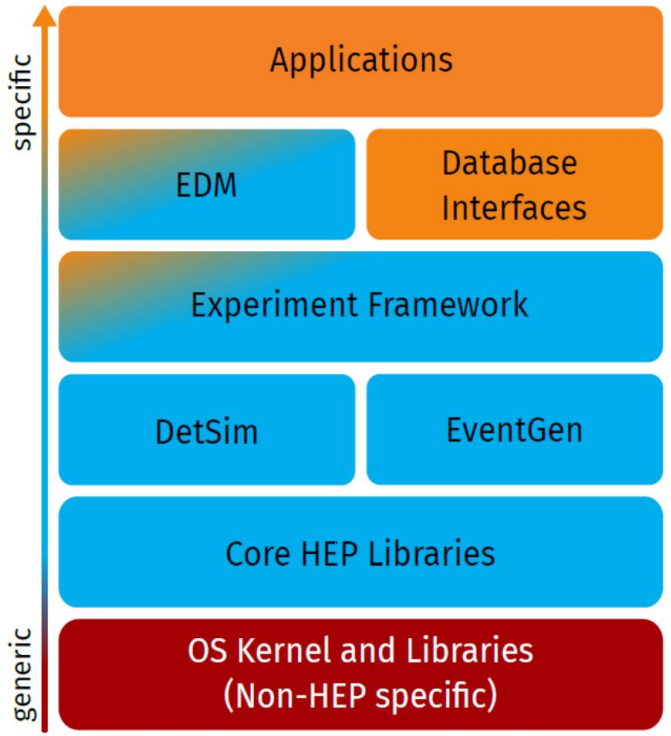


Introduction

- The task of the STCF core software
  - To provide the common software platform for the entire offline data processing
  - To support detector simulation, calibration, reconstruction and data analysis
- The scope of the STCF core software
  - The underlying framework
  - Event data model and event data management
  - Detector data management (geometry, material, field, alignment, etc.)
  - Database management
  - Detector and event display
  - Support of ML, parallel computing, and heterogeneous computing
  - Software and physics validation system
  - Software build, installation and distribution

Introduction

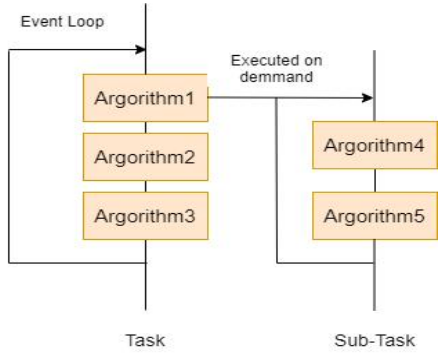
- Main R&D innovations and challenges for the core software compared to BESIII
  - The huge data volume (~100 times of BESIII) requires much more advanced performance
    - Relying on pure CPU resource to process **exabytes** of data is hardly realistic, after the end of Moore's law
    - Heterogeneous resources, like many-core CPUs, GPUs, even FPGAs need to be supported to overcome the challenge
    - The core software needs to provide ready-to-use development and run time environment for heterogeneous processers
    - Good support of 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.
    - Core developers are involved in Key4hep development

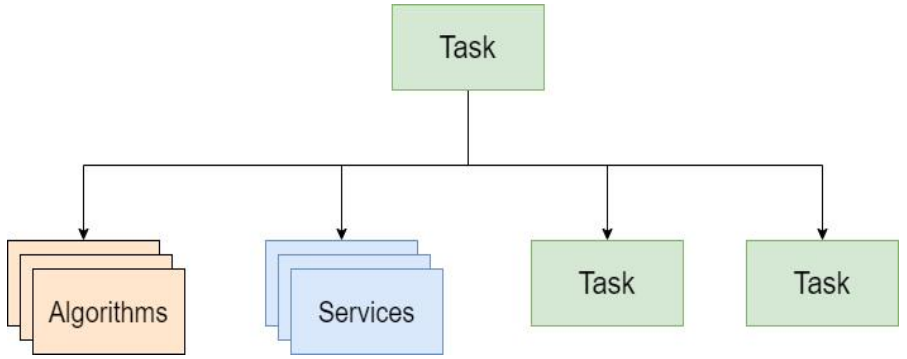


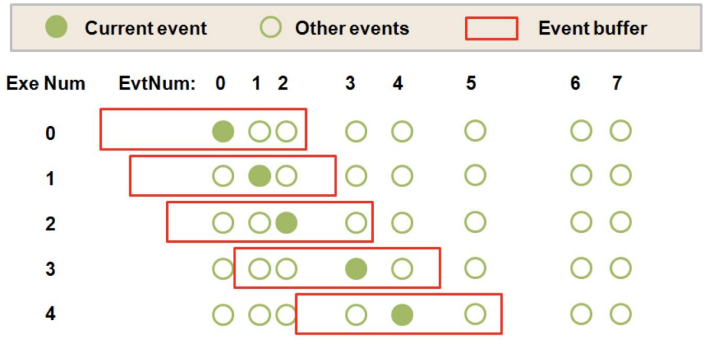
Key4hep
Thomas Madlener,
Epiphany Conference 2021

Underlying Framework: SNiPER

- The underlying framework builds the skeleton of OSCAR
  - Such as Gaudi, Marlin etc. Provide basic functionalities of event loop control, user interface, job configuration, logging etc.
- OSCAR adopts SNiPER as the underlying framework
  - Lightweighted, supporting both non-collider and collider HEP experiments
  - Adopted by JUNO (neutrino), LHAASO (cosmic ray), nEXO (neutrinoless double beta decay) and HERD (dark matter)
- Advantages of SNiPER
  - Lightweighted, efficient, highly extendable. Flexible event loop control. Flexible to be integrated with other software, e. g. podio, DD4hep, Geant, ...
  - C++/Python hybrid programing, highly configurable. Efficient multithreading.

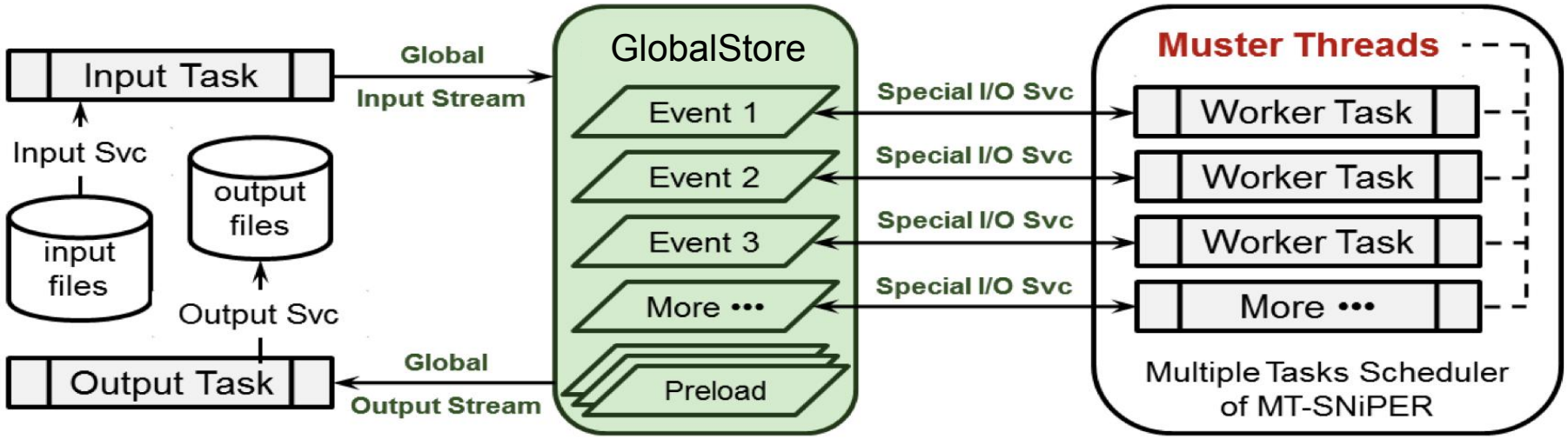






Parallelism in MT-SNiPER

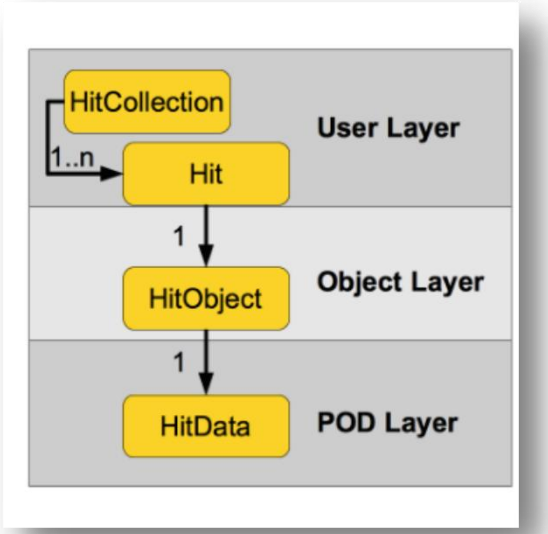
- SNiPER provides simple interfaces for building multithreaded applications
  - SNiPER Muster (Multiple SNiPER Task Scheduler) works as a thread pool/scheduler based on TBB
  - Data I/O is bound to dedicated I/O thread for flexibility
  - A GlobalStore is developed to support parallel event data management
  - Application code is mostly consistent for serially and parallelly execution
  - Track-level and Algorithm-level parallelism are under R&D



Event Data Model Based on Podio

- Event Data Model (EDM) lies at the heart of OSCAR
  - Define event data in memory and in data files (transient and persistent event store)
  - 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
  - Both C++ and Python are supported
  - Multithreading support
  - Powerful and flexible relationshop between data objects
  - Multiple storage backends (data file formats)

F. Gaede, etc. , CHEP2019

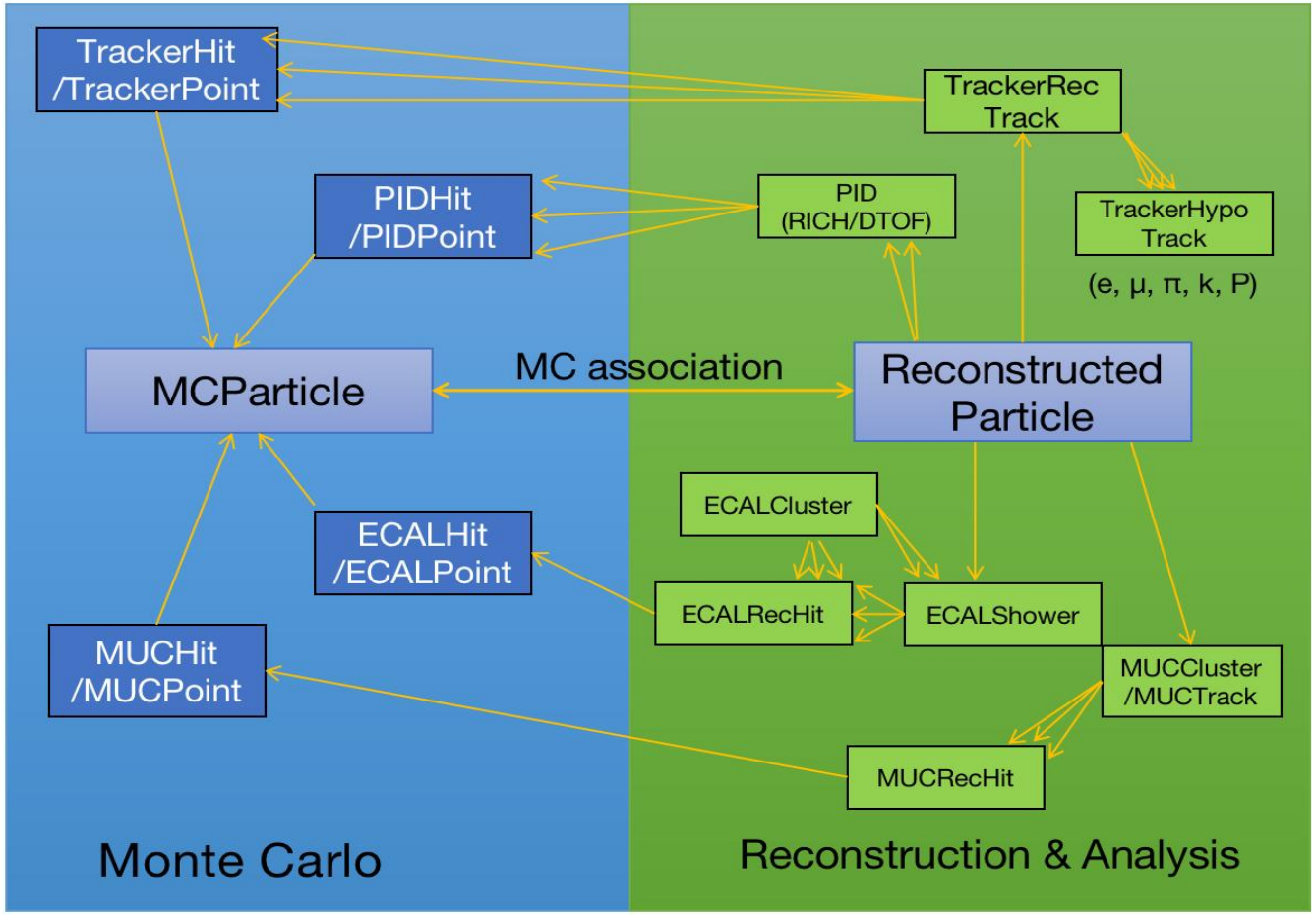


https://github.com/AIDASoft/podio

Event Data Model Based on Podio

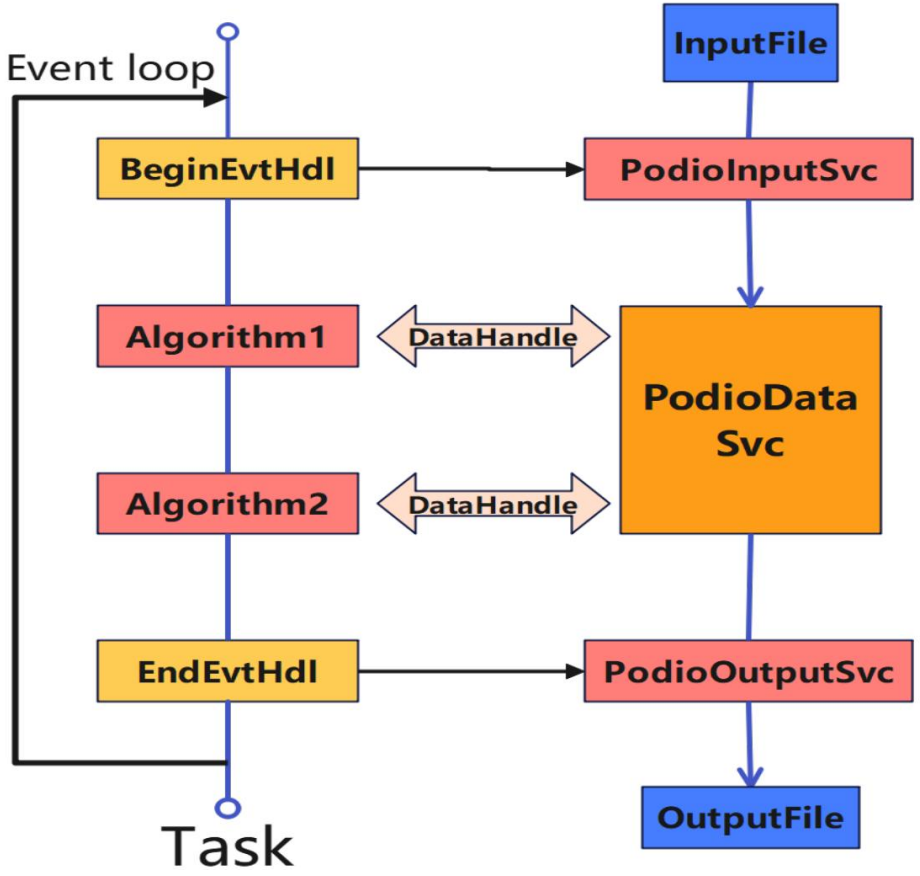
- Due to the specific requirements of STCF, EDM4hep is partially adopted
- Design EDM classes based on podio and reuse some EDM4hep classes

- Use **MCParticle** and **ReconstructedParticle** in EDM4hep as the core index
- EDM classes specificly for STCF **simulation and reconstruction**
- MCParticle and ReconstructedParticle are correlated based on track matching algorithm, **bridging MC and reconstructed data**



Event Data Management

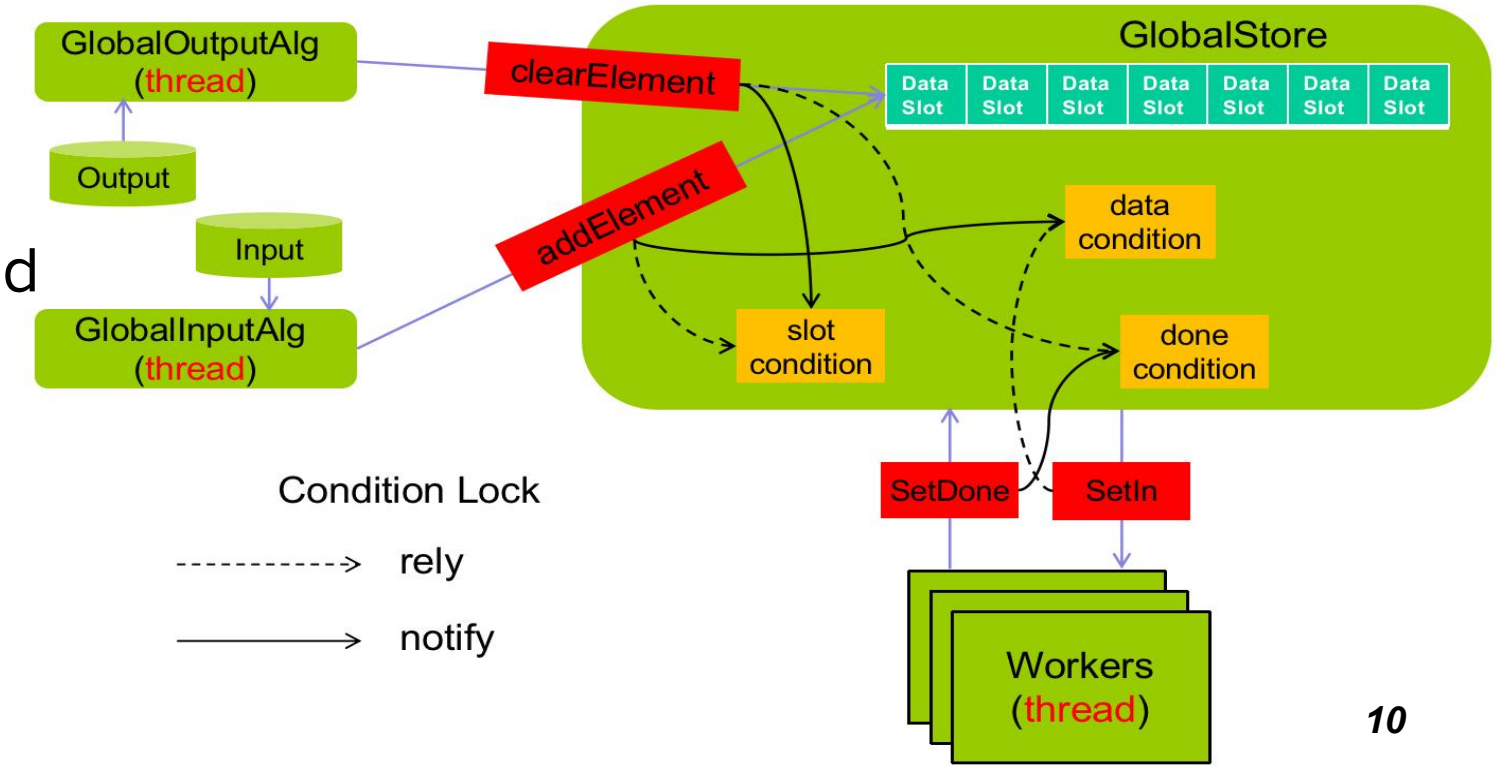
- Event data management system manages event data in memory, provides interfaces for user applications and handles data I/O
- Extend SNiPER DM system based on Podio
  - PodioDataSvc: transient event store (TES)
  - PodioInputSvc: data input
  - PodioOutputSvc: data output
  - DataHandle: interface
-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



W.H. Huang et al 2023 JINST 18 P03004

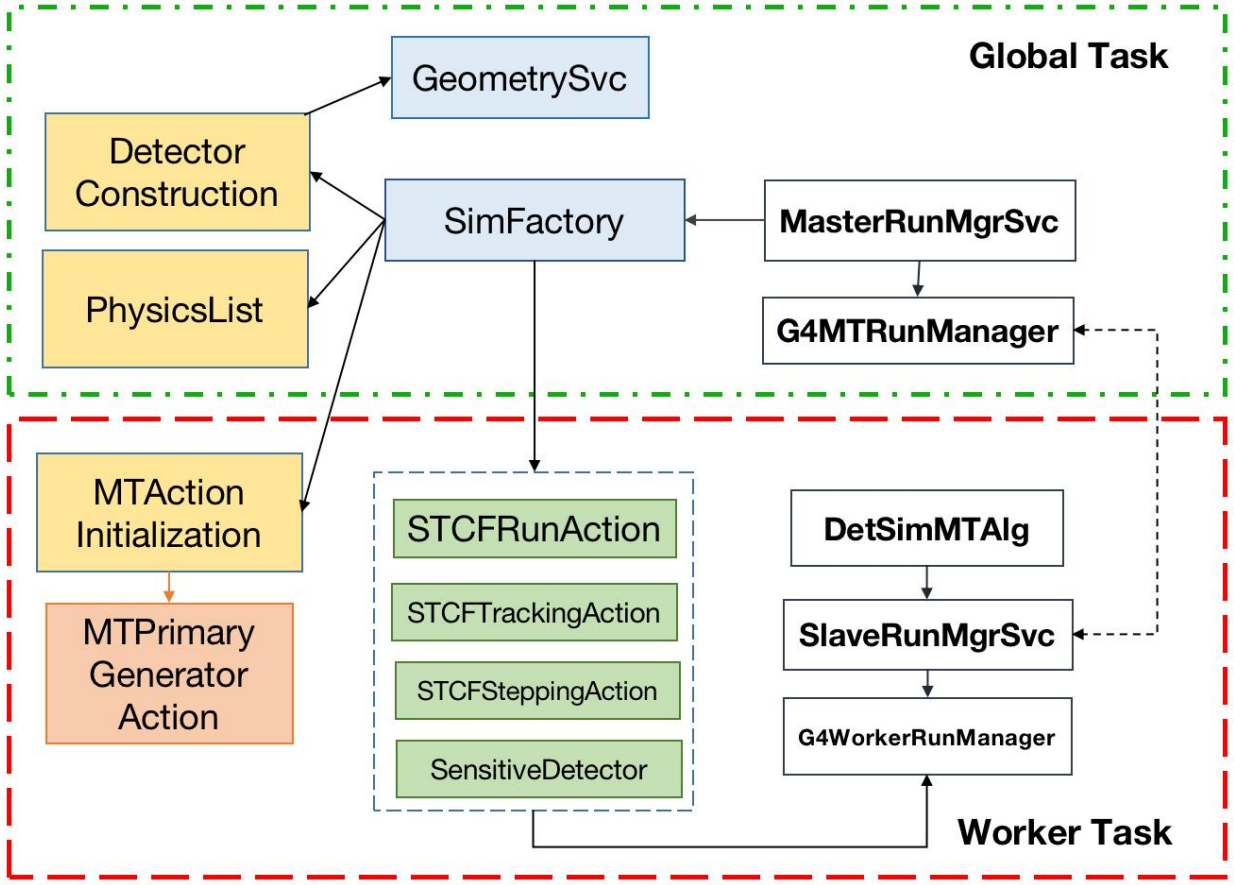
Parallelized Event Data Management

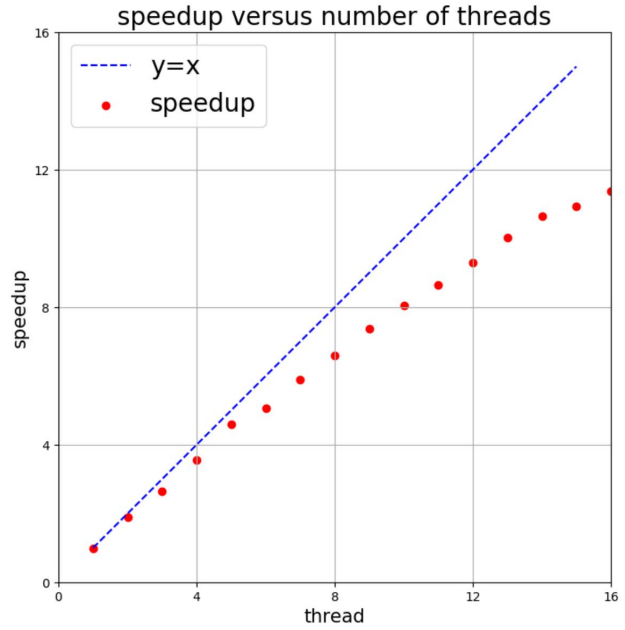
- To enable parallelized data processing, a GlobalStore is developed based on podio
  - Re-implement podio::EventStore to cache multiple events (each within one data slot)
  - Use several condition lock to enable safety exchanging data between threads
  - I/O services are binded to dedicated I/O threads, 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

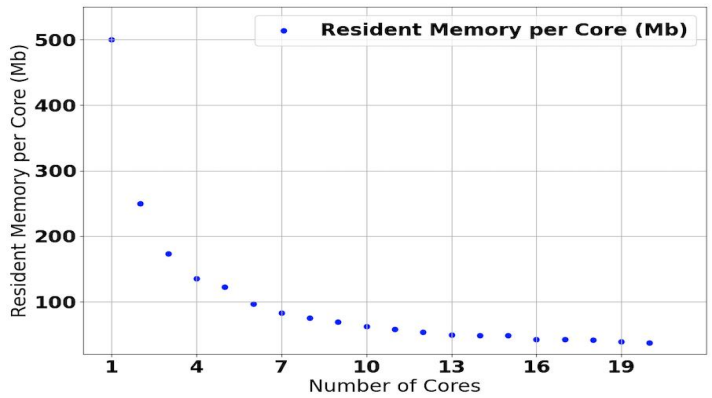


Parallelized Detector Simulation

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



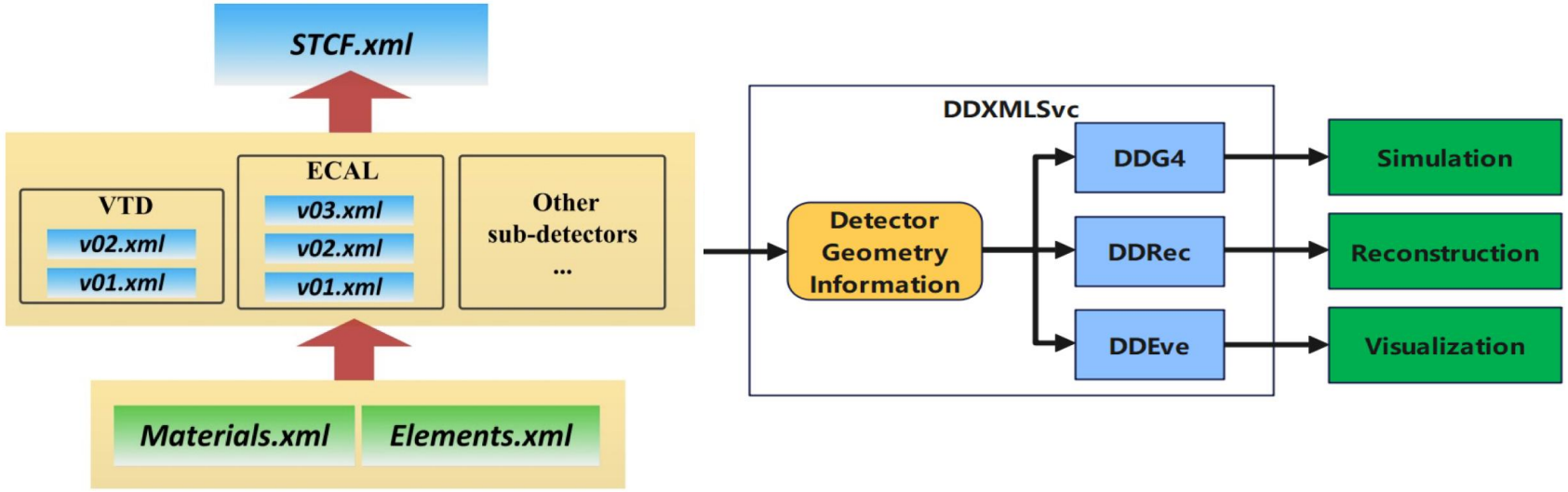




Geometry Management System

- 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

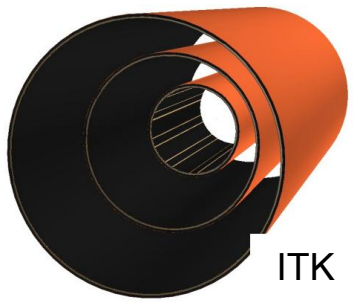
Flexible combinations of different versions of detector design, and combinations of sub-systems

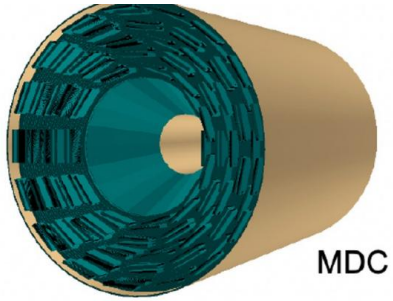


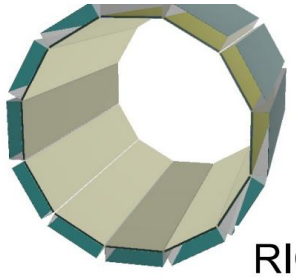
H. Li et al 2021 JINST 16 T04004

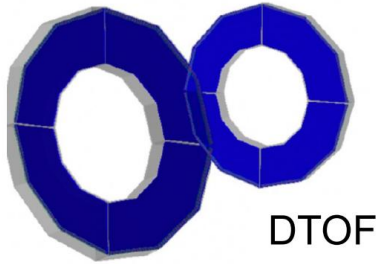
Detector Geometry Description

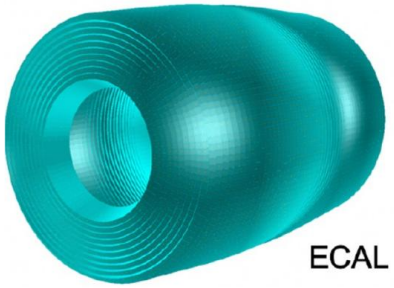
- The Full STCF Detector is described with DD4hep
- Each sub-detector is implemented with a single compact file
- The version number is used for different design options
- Optimizing the detector geometry according to changes of the detector design

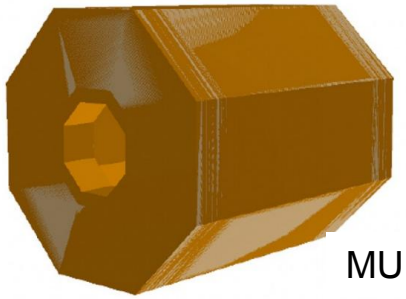


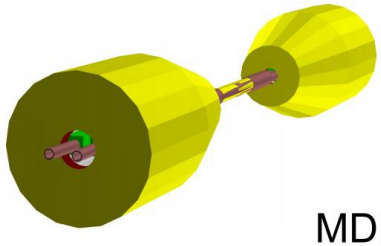


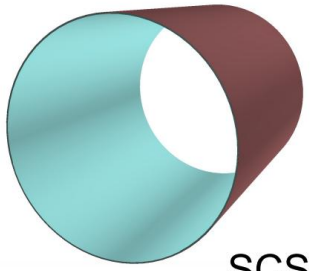






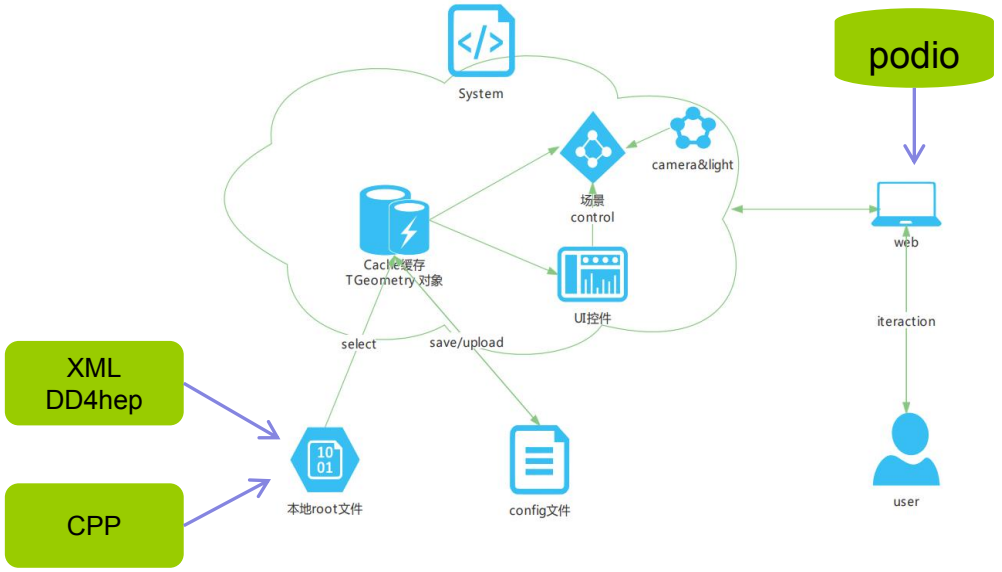


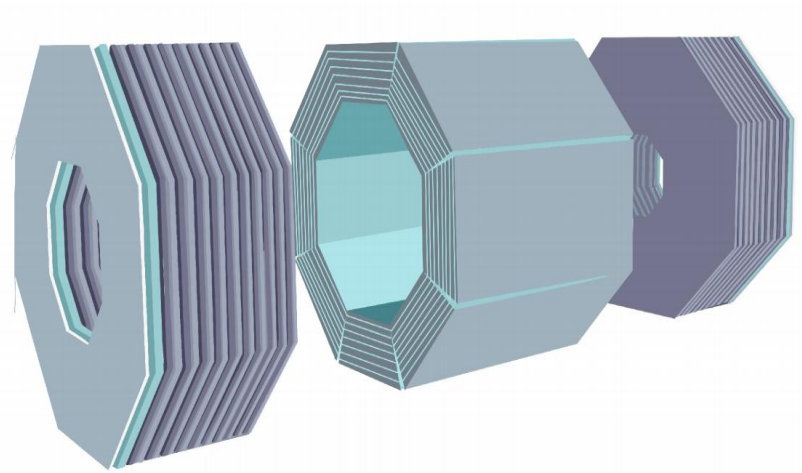


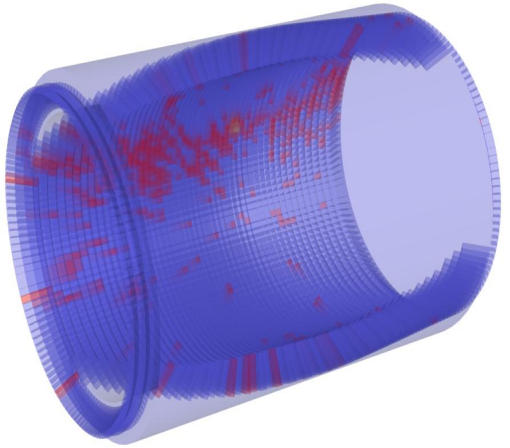


Detector and Event Display

- 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







Machine Learning Model Integration

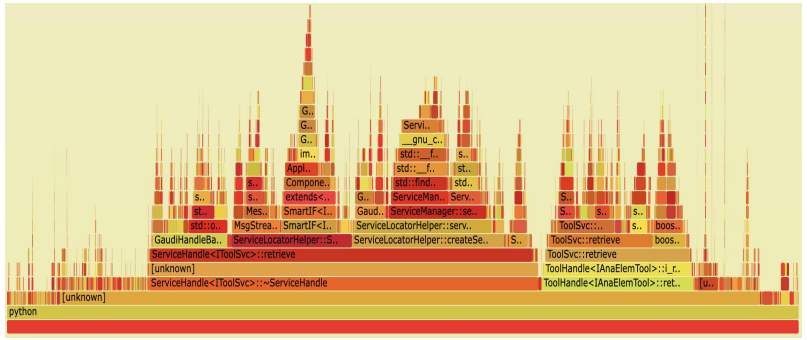
- ONNX Runtime has been integrated with OSCAR to support runtime inference
  - Lot's of applications in OSCAR are based on ML models, such as fast simulation, reconstruction, PID etc.
  - As an easy and unified way to integrate different models in OSCAR and run inference easily
  - Convert from other models to ONNX, such as Tensorflow, PyTorch etc.
  - Potentially to accelerate inference of larger model on different hardware platform (CPU/GPU)

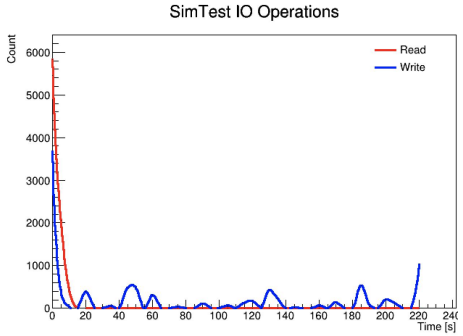
```
Ort::MemoryInfo info("Cpu", OrtDeviceAllocator, 0, OrtMemTypeDefault);

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;
```

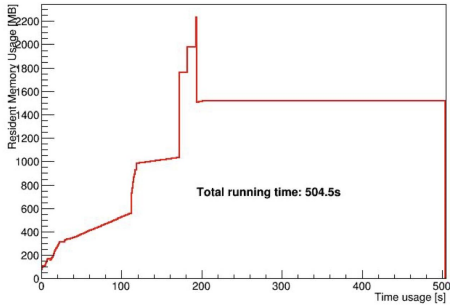
```
bool OrtInferenceAlg::initialize() {

    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);
```

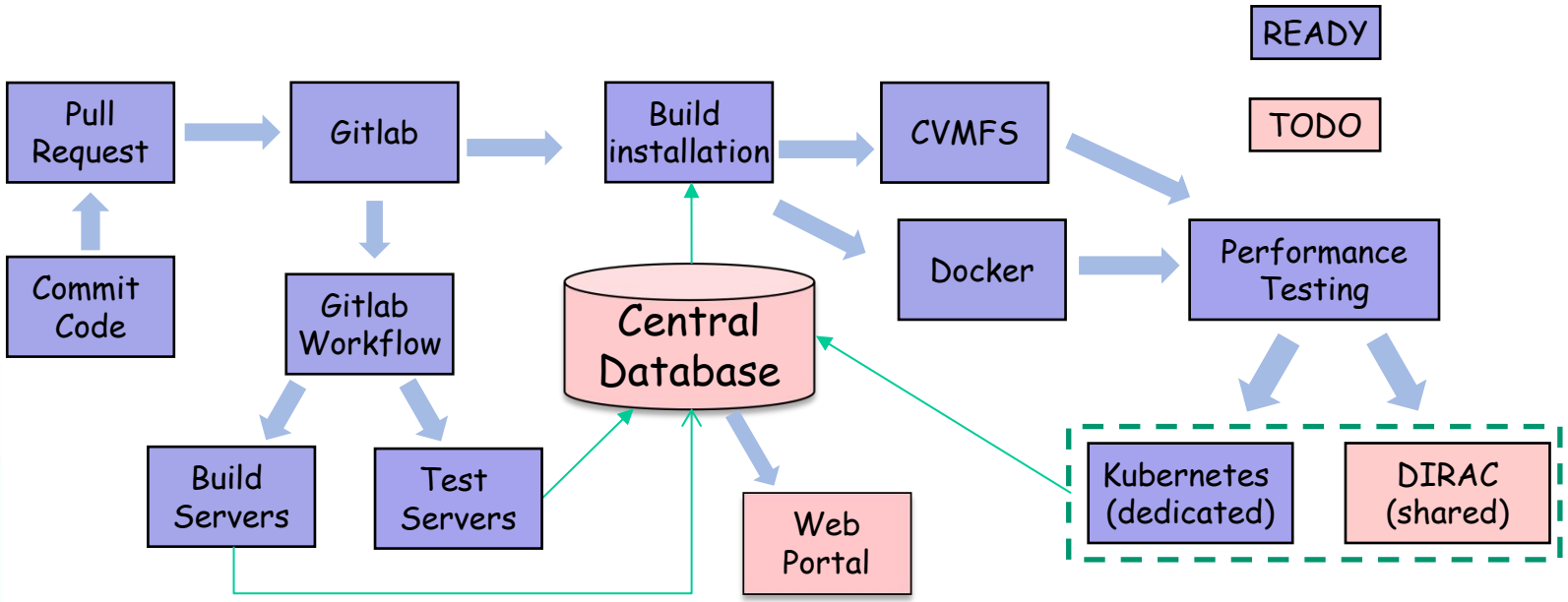
Automated Software Validation

- Software validation system is developed, to support building software validation on different levels
  - Unit test, integrated test, software performance profiling and physics result validation
- 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 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.
  - Physics analysis studies with MC data can be performed in OSCAR now
- 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