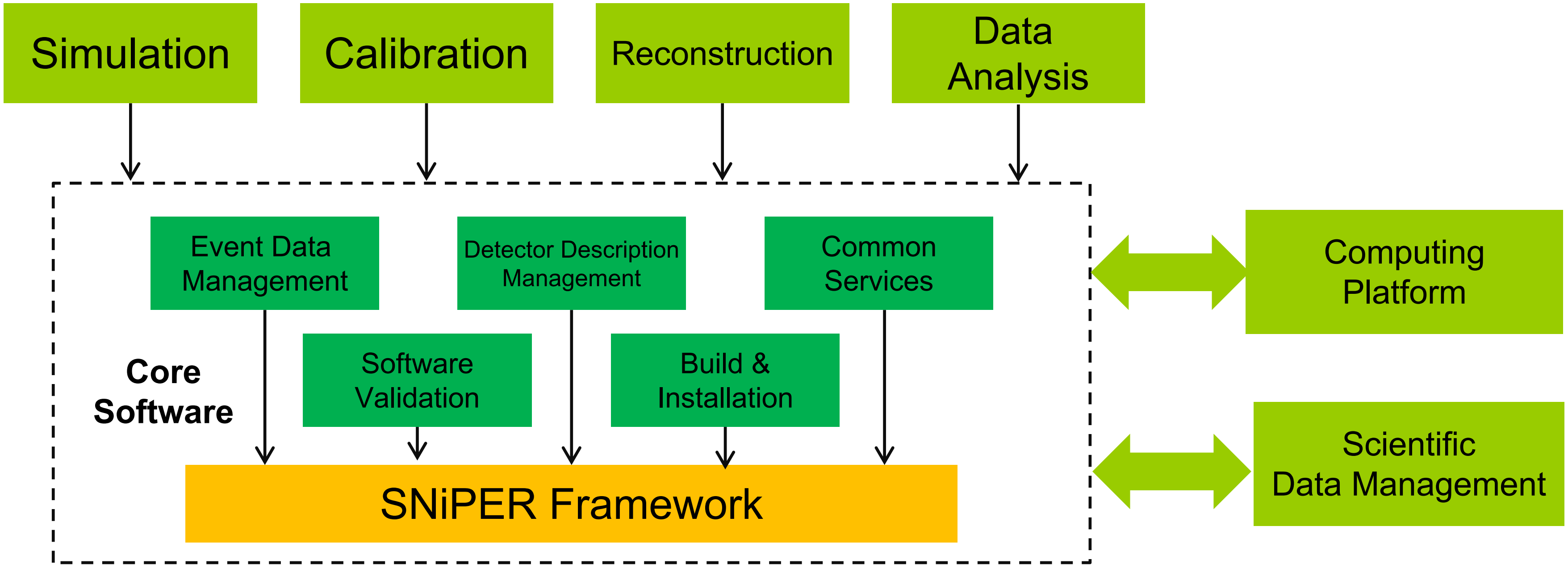
Introduction of the HERD Offline Software

Teng LI on behalf of HERDOS developers
紫金山天文台
2025.5.9

Overview of HERDOS

- The Task of HERDOS
  - To fulfill official offline data processing tasks, i.e. detector simulation, digitization, calibration and reconstruction
  - Provide a common platform for users to develop and embed analysis code



Motivation of the framework

- The **motivation** of developing the framework
  - To serve as the common software platform for the entire offline data processing
    - Provide common functionalities for data processing and make developers/users focusing on their applications and analysis
    - Improve development efficiency, reduce development difficulty and improve the software quality
    - Improve the reliability of all physical results
  - Fulfill the specific requirements from HERD
    - The software should be light-weighted, yet complete in every part and of excellent performance
    - Multi-threading is vital (for the simulation of PeV heavy nucleons)
    - Flexible and consistent detector description
    - Capable of supporting complex reconstruction and calibration jobs
    - Should take the long life-cycle into account

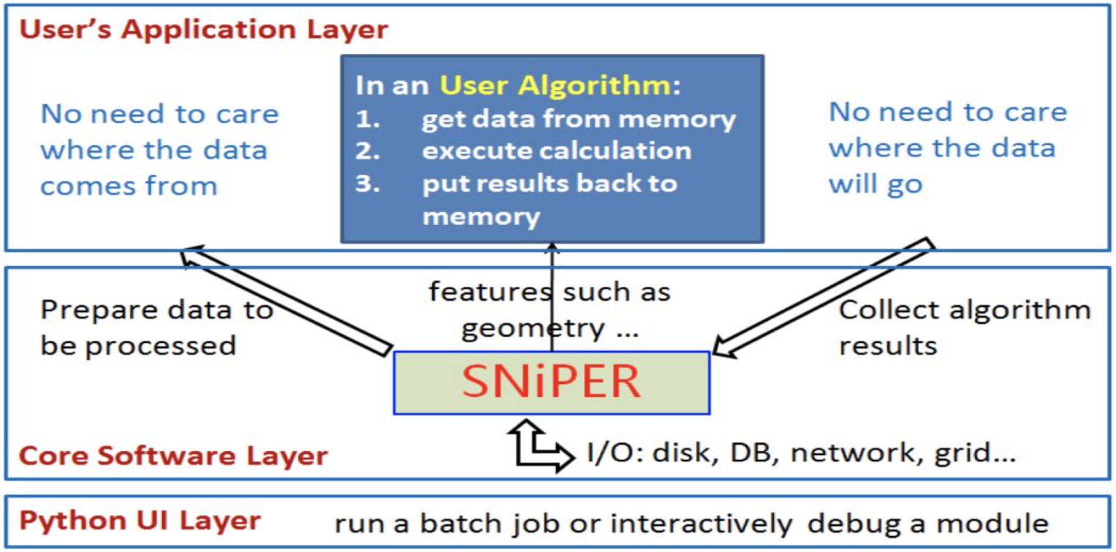
The Underlying Framework SNiPER

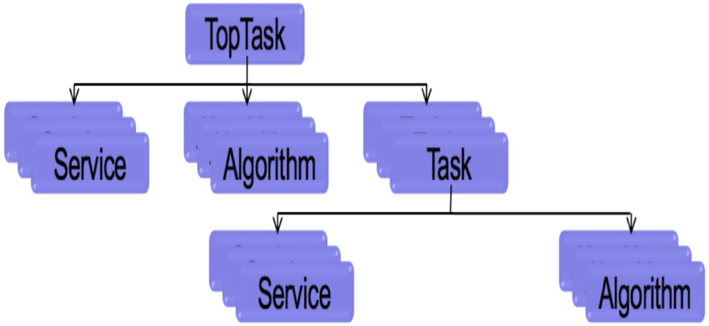
Underlying Framework: SNiPER

- The SNiPER Framework
  - Designed and developed as common framework for HEP experiments (since 2012)
  - Maintained by **10+ developers from IHEP, SDU, SYSU etc.**
  - Adopted as underlying framework for JUNO, LHAASO, nEXO, STCF etc.
- Provide common functionalities from HEP data processing tasks
- Key features of SNiPER
  - Light-weighted, with minimal dependencies of external libraries
  - High cohesion & low coupling design
  - Flexible user interface based on Python binding
  - Flexible data processing chain
  - Multi-threading support

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

Underlying Framework: SNiPER



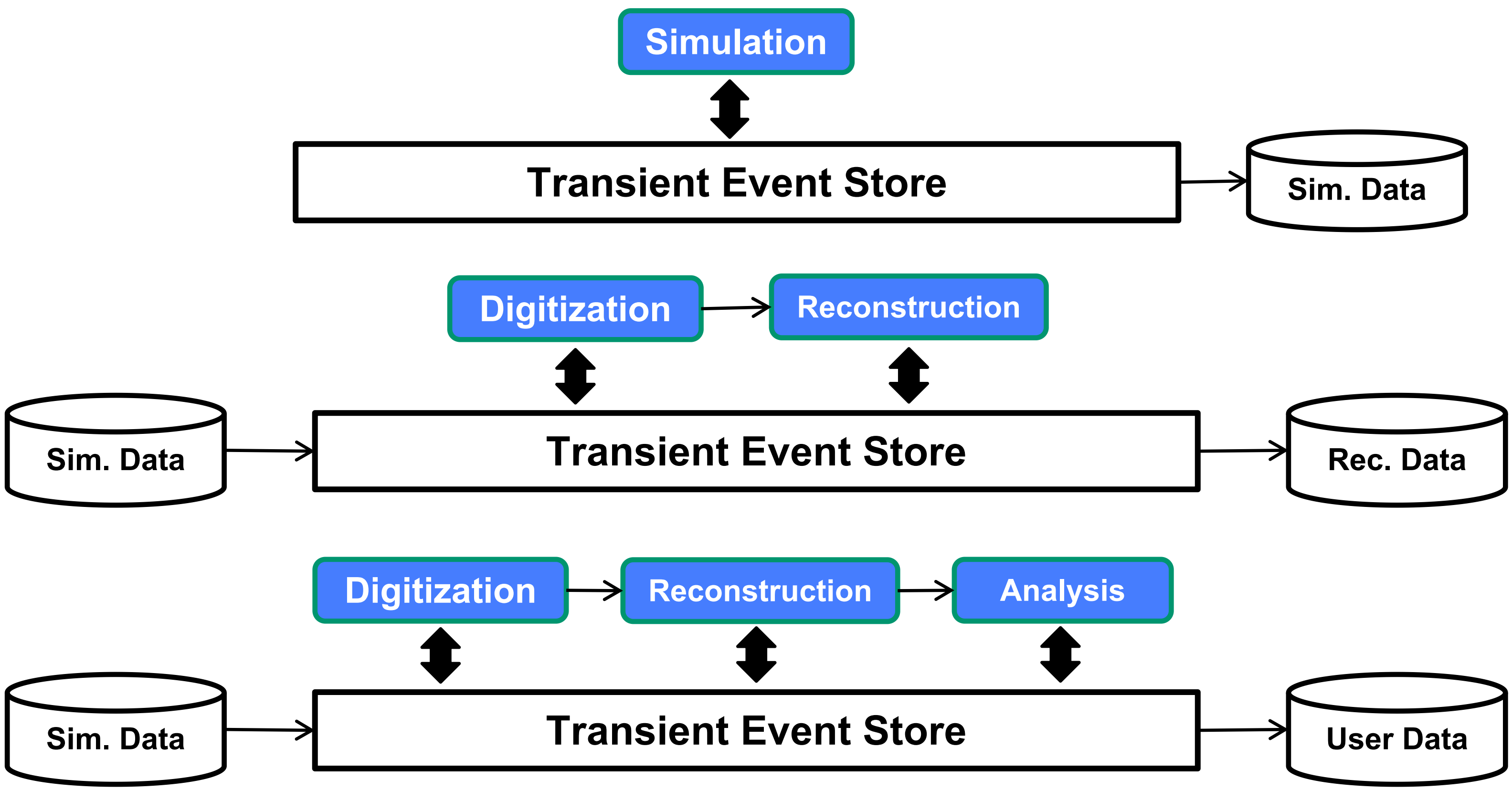


- Application layer:
  - Developer can develop specific algorithms and tasks for specific requirements
- Core layer:
  - Core functionalities such as event data management, detector description management ...
- UI layer:
  - Built based on python-binding

Data Processing Procedure with Decoupled Data

Tasks can be configured to run specific processing procedures, e.g.

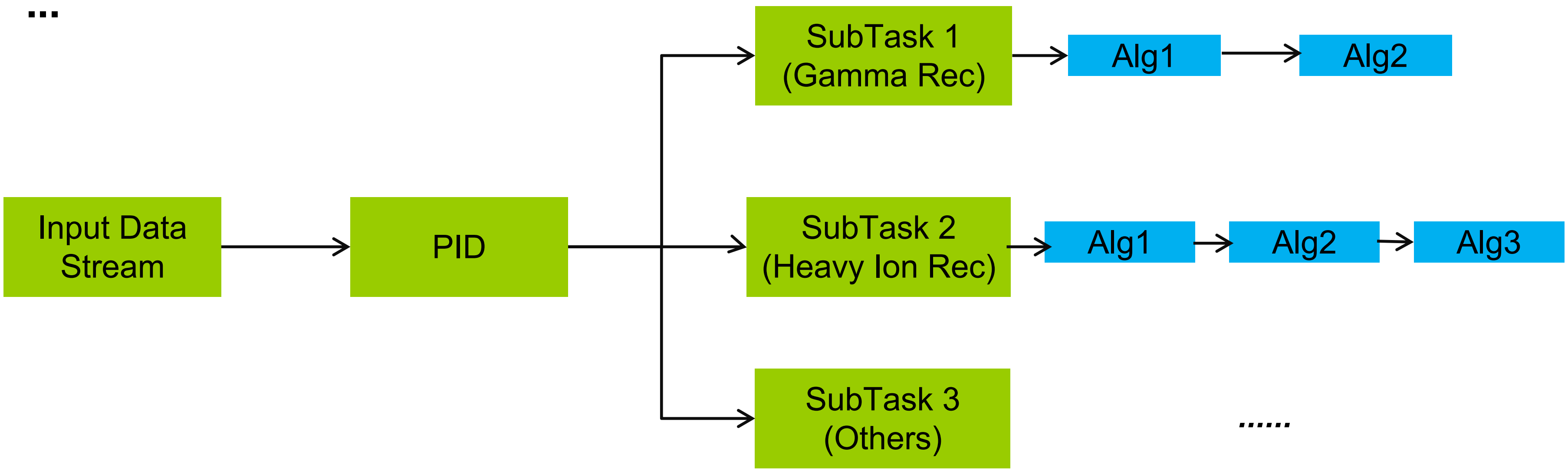
- Simulation only
- Digitization only
- Reconstruction only
- Simulation + digitization
- Simulation + digitization + reconstruction
- Anything + customized analysis code



Configure Flexible Data Processing Procedure

Other than sequential workflow, complex processing procedure is also possible (branching/ jumping/ concurrent)

- 'Steering' of reconstruction algorithms
- Analysis with multiple input streams
- Mixing of multiple MC samples
- ...



Flexible processing chain can be built on demand

User Interface

❖ High level Python module is developed to better steer HERDOS job

```python
#!/usr/bin/env python
#-*- coding: utf-8 -*-
# Author: Teng LI <tengli@sdu.edu.cn>

from HERDOSModule import *
from GeometrySvc import GeometryModule
from RandomSvc import RandomModule

app = HERDOSApplication()

# Random engine
app.registerModule(RandomModule())

# Geometry
app.registerModule(GeometryModule())

# Detector simulation
app.registerModule(DetectorSimulation())

# Data management
app.registerModule(DataManagement())

app.run()
```

- Handle the creation of HERDOS components
- En-capsule common configurations and expose properties to command line interface
- Provide detailed helper message
- **One can quickly get started without knowledge of Python and HERDOS**

User Interface

- High level Python module is developed to better steer HERDOS job

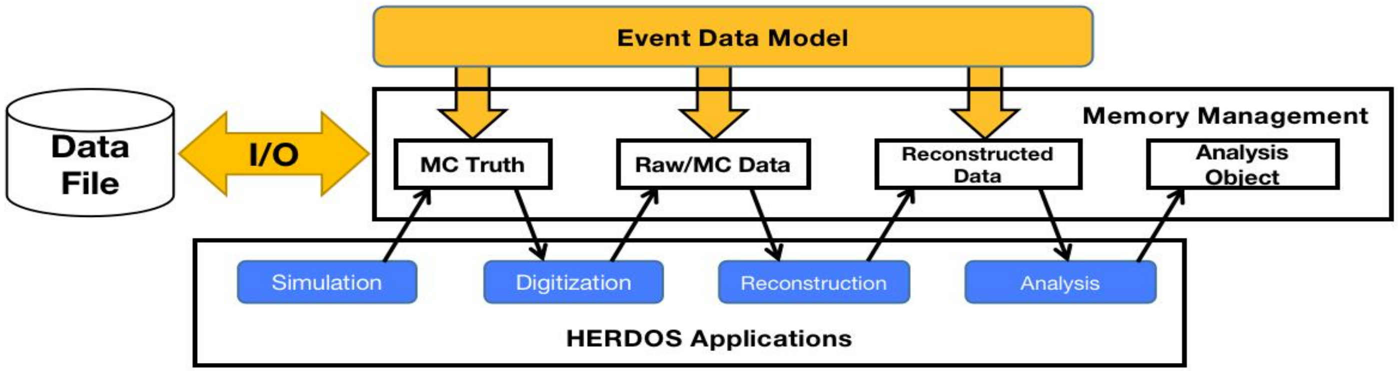
```
-bash-4.2$ python simulation.py -h
**************************************************
Welcome to SNiPER 2.1.0
Running @ lxslc713.ihep.ac.cn on Wed Feb 21 12:11:12 2024
**************************************************
usage: simulation.py [-h] [--loglevel {Test,Debug,Info,Warn,Error,Fatal}] [--dryrun] [--evtmax EVTMAX] [--user-output USER_OUTPUT] [--EnableUserOutput]
                     [--DisableUserOutput] [--profiling] [--no-profiling] [--profiling-detail] [--no-profiling-detail] [--seed SEED]
                     [--seed-status-file SEED_STATUS_FILE] [--seed-status-vector SEED_STATUS_VECTOR [SEED_STATUS_VECTOR ...]]
                     [--geometry-compact-file GEOMETRY_COMPACT_FILE] [--enable-base-box] [--disable-base-box] [--sim-random-seed SIM_RANDOM_SEED]
                     [--g4-run-mac G4_RUN_MAC [G4_RUN_MAC ...]] [--g4-commands G4_COMMANDS [G4_COMMANDS ...]] [--g4-vis-mac G4_VIS_MAC]
                     [--enable-space-lab] [--disable-space-lab] [--run-id RUN_ID]
                     [--solar-panel-param SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL_PARAM SOLAR_PANEL
PARAM]
                     [--physics-list {FTFP,QGSP,CRMC,ADPM}] [--gps-energy GPS_ENERGY [GPS_ENERGY ...]] [--gps-energy-kn GPS_ENERGY_KN]
                     [--gps-particle GPS_PARTICLE] [--input INPUT [INPUT ...]] [--output OUTPUT] [--output-colls OUTPUT_COLLS [OUTPUT_COLLS ...]]
                     [--transfer-colls TRANSFER_COLLS [TRANSFER_COLLS ...]] [--transfer-colls-exclude TRANSFER_COLLS_EXCLUDE [TRANSFER_COLLS_EXCLUDE ...]]
                     [--transfer-all]

optional arguments:
  -h, --help            show this help message and exit
  --loglevel {Test,Debug,Info,Warn,Error,Fatal}
                        Log level of the job
  --dryrun              only show the job, without running
  --evtmax EVTMAX       number of events to be processed
  --user-output USER_OUTPUT
                        output user data file name
  --EnableUserOutput    Enable User Output
  --DisableUserOutput   Disable User Output
  --profiling           enable profiling
  --no-profiling        disable profiling
  --profiling-detail    enable saving profiling details
  --no-profiling-detail
                        disable saving profiling details
  --seed SEED           common random seed (for both CLHEP and ROOT engines)
```

Properties can be set with command line interface

Event Data Management

Event Data Management: **Requirements**

- Event data management is the most crucial part of the framework
  - **Provide tools to define the Event Data Model (EDM, Data Object classes)**
    - The definition of physics event data (MC particles, hits, readouts, tracks, clusters, reconstructed particles)
    - Construct relationship between EDM objects
  - **Provide automatic memory management mechanism**
  - **Provide persistent and transient EDM conversion**
  - **Provide backward and forward compatibility，very important for long time running of HERD.**
  - **Guarantee thread-safety**



Event Data Model (EDM) Base on Podio

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

```yaml
edm::CaloDigiCell:
   Description: "calo cell digitization"
   Author: "Z.Tang Z.Quan"
   Members:
      - unsigned int cellCode        // encoded cell code
      - std::array<int, 3> grayHR    // total grayscale value in CMOS for high range, 3 radius
      - std::array<int, 3> grayLR    // total grayscale value in CMOS for low range, 3 radius
   OneToOneRelations:
      - edm::CaloSimCell siminfo    // if there is any...
```

↓

Auto Code Generation
During
Compilation

↓

```
CaloDigiCell.cc                    CaloDigiCellCollectionData.h    CaloDigiCell.h
CaloDigiCellCollection.cc          CaloDigiCellCollection.h        CaloDigiCellObj.cc
CaloDigiCellCollectionData.cc      CaloDigiCellData.h              CaloDigiCellObj.h
```

```cpp
/// Access the encoded cell code
const unsigned int& getCellCode() const;

/// Access the low(high) range = 0(1)
const short& getType() const;

/// Access the summed pixel grayscale in one spot at LG, radius of spot (R
const std::array<float, 3>& getCellGrayLG() const;
/// Access item i of the summed pixel grayscale in one spot at LG, radius
const float& getCellGrayLG(size_t i) const;
/// Access the summed pixel grayscale in one spot at HG, radius of spot (R
const std::array<float, 3>& getCellGrayHG() const;
/// Access item i of the summed pixel grayscale in one spot at HG, radius
const float& getCellGrayHG(size_t i) const;
/// Access the individual pixel grayscale in one spot at LG, where R=15
const std::array<int, 716>& getPixelGrayLG() const;
/// Access item i of the individual pixel grayscale in one spot at LG, whe
const int& getPixelGrayLG(size_t i) const;
/// Access the individual pixel grayscale in one spot at HG, where R=15
const std::array<int, 716>& getPixelGrayHG() const;
/// Access item i of the individual pixel grayscale in one spot at HG, whe
const int& getPixelGrayHG(size_t i) const;
/// Access the internal time interval between two events, [ns]
const float& getDeltaT() const;

/// Access the total grayscale value in CMOS for high range, 3 radius. to
const std::array<int, 3>& getGrayHR() const;
/// Access item i of the total grayscale value in CMOS for high range, 3 r
const int& getGrayHR(size_t i) const;
/// Access the total grayscale value in CMOS for low range, 3 radius. to b
const std::array<int, 3>& getGrayLR() const;
/// Access item i of the total grayscale value in CMOS for low range, 3 ra
const int& getGrayLR(size_t i) const;

/// Access the if there is any...
const edm::CaloSimCell getSiminfo() const;
```

Developers only need to write yaml file to define data objects

Implementation details can be ingored (thread safety, garbage collection, relationship, multiple versions compatibility, etc ..)

Tedious and error-prone work can be avoided

Defined EDM

Global:

- MCEvent
- Event
- MCParticle
- TrackingSimHit
- GlobalTrack

Calorimeter:

- CaloSimCell
- CaloDigiCell
- CaloRecoCell
- CaloClusters
- CaloShowerAxis
- CaloPDDigiCell

PSD:

- PSDDigiCell
- PSDRecoCell

TRD:

- TRDDigiCell
- TRDRecoCell

Trigger:

- FastTrigger
- Trigger

SCD:

- SiliconDigiHit
- SiliconDigiCell
- SCDCluster
- SCDTrack

FIT:

- FITDigiCell
- FITRecoCell
- FITCluster
- FITTrack

All EDM classes defined in one yaml file (DataModel/EventDataModel/datalayout.yaml)

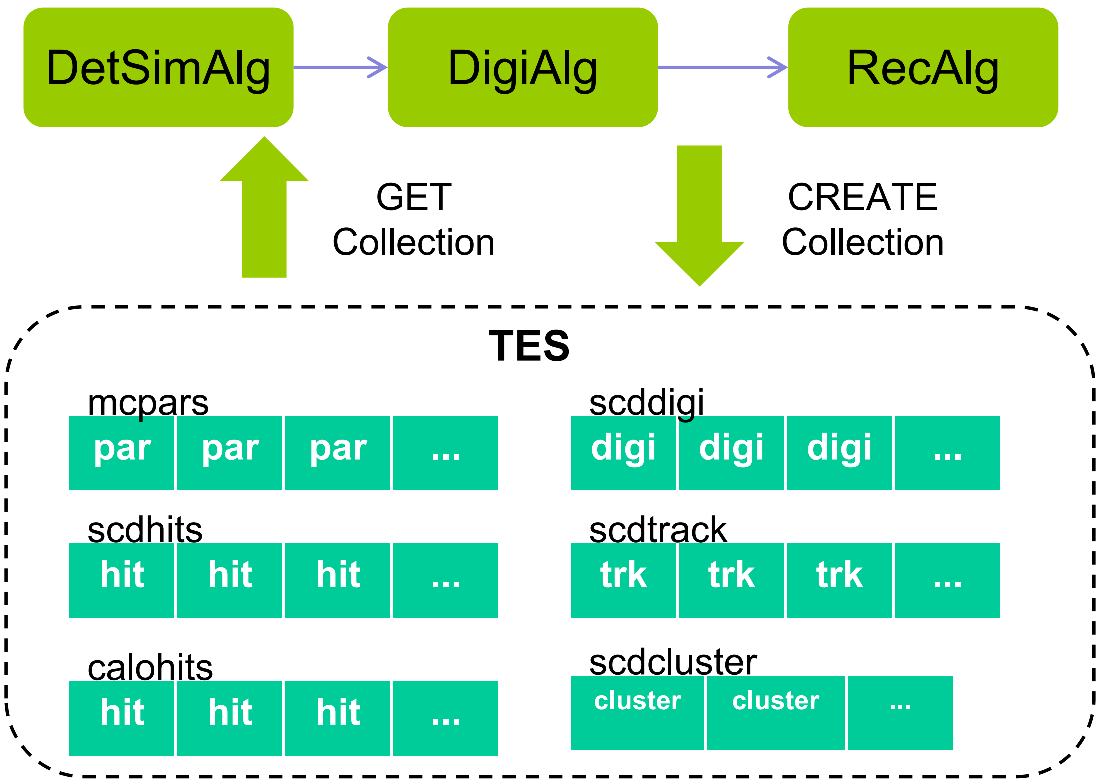
Official EDM classes can be extended on approval

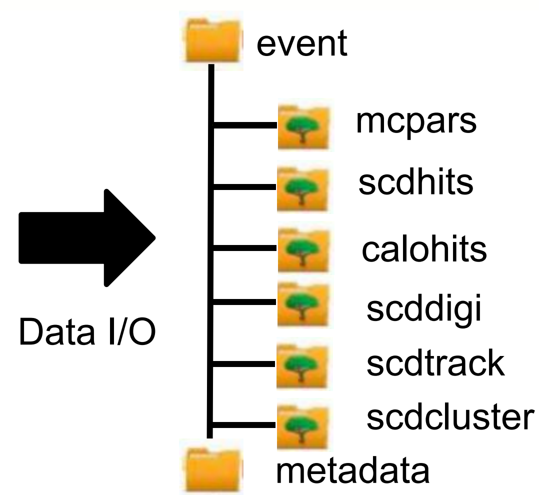
Users could define their own EDM classes just for individual-usage

Transient Event Store

- **Transient Event Store** (TES) is where EDM objects are stored in memory, shared by all user Algorithms
- User Algorithms access event data via collections, through **DataHandle** (or the **getROColl** and **getRWColl** macros)

```
kTrackSimHits = getROColl(TrackingSimHitCollection, "scdhits");
kCaloSimHits = getRWColl(CaloSimCellCollection, "calohits");

if (kTrackSimHits)
{
    // Do some analysis here.
}

if (kCaloSimHits)
{
    for (size_t i=0; i<kCaloSimHits->size(); ++i)
    {
        auto edep = kCaloSimHits->at(i).getEdep();
        mHistEdep->Fill(edep);
    }
}
```

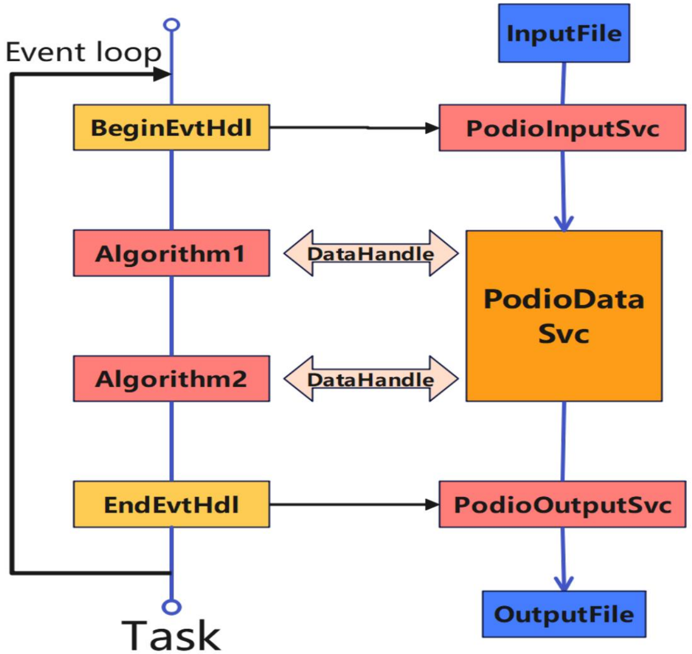




Examples of defining, accessing EDM in backup

Data Input / Output Services

- Data input/output is implemented with PodioInputSvc and PodioOutputSvc



- Options like input file, output file, collections to be wrriten, collections to be transferred, ... can be configured
- Other types of I/O services can be configured to support more file formats (like raw data)

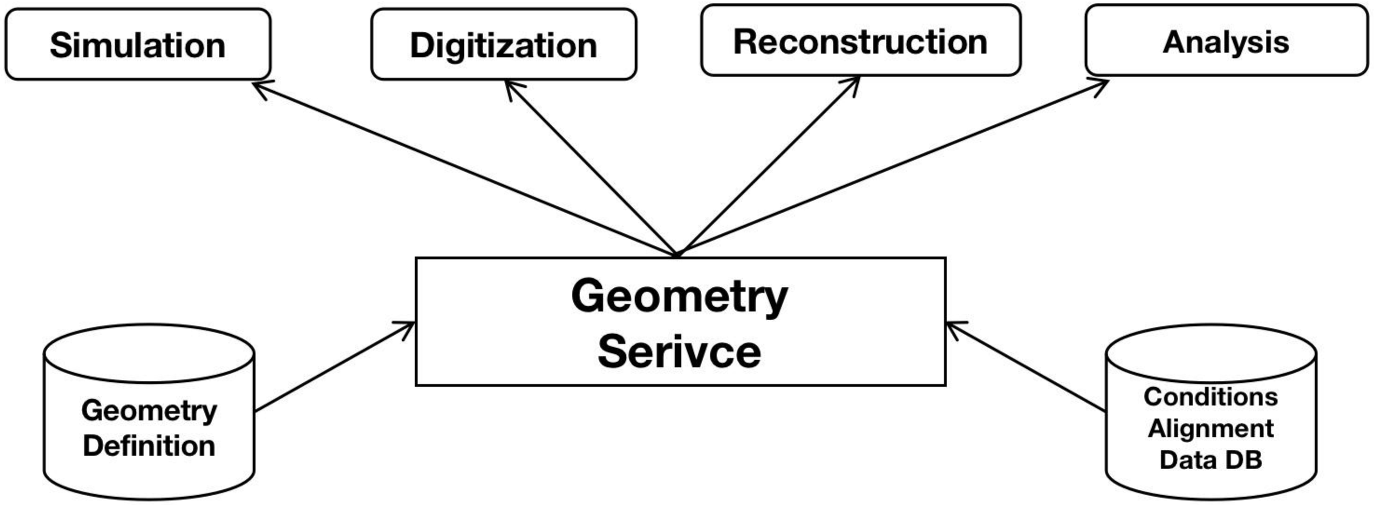
```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-

import Sniper

task = Sniper.Task("task")

import SCDDigi
task.createAlg("SCDDigi_v1")
import AnalysisExample
task.createAlg("MyAnalysis")

import PodioDataSvc
pSvc = task.createSvc("PodioDataSvc")
import PodioSvc
# read something from root file
Isvc = task.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("simhits.root")
# write something into root file
Osvc = task.createSvc("PodioOutputSvc/OutputSvc")
Osvc.property("OutputFile").set("my_track.root")
Osvc.property("OutputCollections").set(["mytrk"])

task.setEvtMax(-1)
task.show()
task.run()
```

configure data I/O in configuration file

Detector Description Management

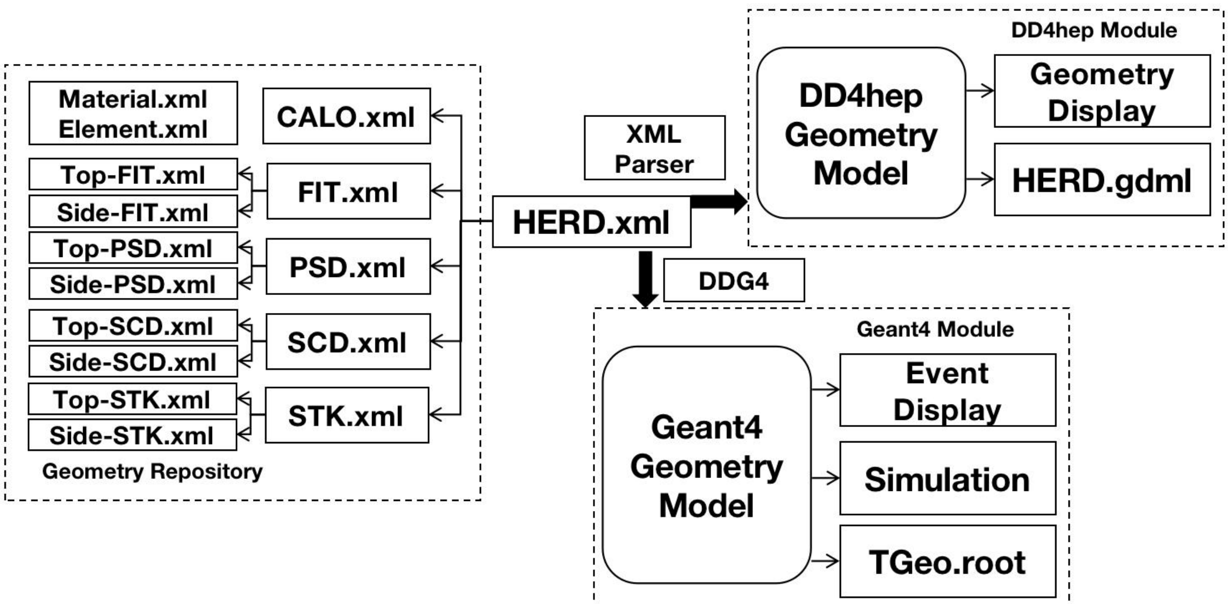
Detector Description Management: **Requirements**

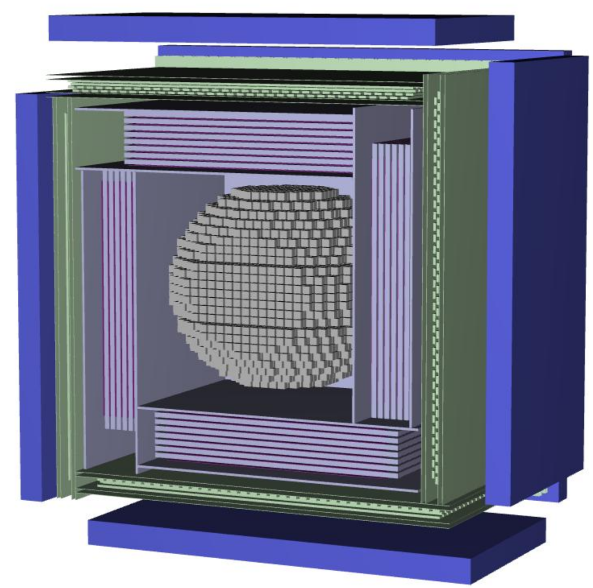
- A powerful detector description management system is necessary across the full offline data processing workflow
  - Provide **consistent detector description** for all applications
  - Provide **geometry format conversion** for different applications
  - Provide interface for **alignment / conditions data**
  - Provide **multiple version support**
  - Provide **easy-to-use interfaces, and common functionalities** for applications (such as coodinates conversion, track length calculation etc.)



Detector description Management

- Full HERD and beam test geometry are defined in XML files
  - Elements, materials defined in common files then composed together
  - Sub detectors can be defined separately with independent version
  - Different combination of detector description can be selected for each run in config file without re-compile
  - Complex geometry (including the space station) from CAD format can be included





Detector description Management

- Full HERD and beam test geometry are defined in XML files

```xml
<materials>
   <material name="LYSO2" state="solid">
      <D unit="g/cm3" value="7.1"/>
      <fraction n="0.7143" ref="Lu"/>
      <fraction n="0.0403" ref="Y"/>
      <fraction n="0.0637" ref="Si"/>
      <fraction n="0.1814" ref="O"/>
      <fraction n="0.0002" ref="Ce"/>
   </material>
   <material name="LYSO1" state="solid">
      <D unit="g/cm3" value="7.4"/>
      <fraction n="1" ref="LYSO2"/>
   </material>
   <material name="G4_AIR" state="gas">
      <MEE unit="eV" value="85.7"/>
      <D unit="g/cm3" value="0.00120479"/>
      <fraction n="0.000124000124000124" ref="C"/>
      <fraction n="0.755267755267755" ref="N"/>
      <fraction n="0.231781231781232" ref="O"/>
      <fraction n="0.0128270128270128" ref="Ar"/>
   </material>
   <material name="G4_Galactic" state="gas">
      <MEE unit="eV" value="85.7"/>
      <D unit="g/cm3" value="0.00000000001"/>
      <fraction n="1" ref="H"/>
   </material>
   <material name="Vacuum" state="gas">
      <D unit="g/cm3" value="0.00000000001"/>
      <fraction n="1" ref="G4_AIR"/>
   </material>
```

```xml
<materials>

   <isotope N="1" Z="1" name="H1">
      <atom unit="g/mole" value="1.0078250308l372"/>
   </isotope>
   <isotope N="2" Z="1" name="H2">
      <atom unit="g/mole" value="2.0141019966617"/>
   </isotope>
   <element name="H">
      <fraction n="0.999885" ref="H1"/>
      <fraction n="0.000115" ref="H2"/>
   </element>

   <isotope N="12" Z="6" name="C12">
      <atom unit="g/mole" value="12"/>
   </isotope>
   <isotope N="13" Z="6" name="C13">
      <atom unit="g/mole" value="13.0034"/>
   </isotope>
   <element name="C">
      <fraction n="0.9893" ref="C12"/>
      <fraction n="0.0107" ref="C13"/>
   </element>

   <isotope N="14" Z="7" name="N14">
      <atom unit="g/mole" value="14.0031"/>
   </isotope>
   <isotope N="15" Z="7" name="N15">
      <atom unit="g/mole" value="15.0001"/>
   </isotope>
   <element name="N">
      <fraction n="0.99632" ref="N14"/>
      <fraction n="0.00368" ref="N15"/>
   </element>
```

Detector description Management

- Full HERD and beam test geometry are defined in XML files

```xml
<detectors>
 <!--TopSCD-->
 <detector id="40" name="TopSCD" type="TopSCD_f" limits="scd_limits" vis="detector" sensitive="true" sd="SimpleTrackingSD" readout="scdhits">
  <dimensions x="scd_top_envelopX" y="scd_top_envelopY" z="scd_top_envelopZ"/>
  <position   x="0" y="0" z="scd_top_cogZ"/>

  <layer id="0" name="siplanex" x="topplane_x" y="topplane_y" z="siPlaneZ" vis="siplanex">
   <ladder name="ladderx" x="siWaferXY" y="topSiLadder" z="siWaferZ" vis="siladderx" repeat="NTopLadder">
    <sensor name="waferx" x="siWaferXY" y="siWaferXY"   z="siWaferZ" vis="siwaferx"  repeat="NtopWafer" sensitive="true" material="G4_Si"/>
   </ladder>
  </layer>
  <layer id="8" name="pcbplane" x="topplane_x" y="topplane_y" z="pcbPlaneZ" material="PCB"    vis="pcbplane"/>
  <layer id="9" name="siXYGap"  x="topplane_x" y="topplane_y" z="siXYGap"   material="Vacuum" vis="InvisibleNoDaughters"/>
  <layer id="1" name="siplaney" x="topplane_x" y="topplane_y" z="siPlaneZ"  vis="siplaney">
   <ladder name="laddery" x="siWaferXY" y="topSiLadder" z="siWaferZ" vis="siladdery" repeat="NTopLadder">
    <sensor name="wafery" x="siWaferXY" y="siWaferXY"   z="siWaferZ" vis="siwafery"  repeat="NtopWafer" sensitive="true" material="G4_Si"/>
   </ladder>
  </layer>
  <layer id="10" name="pcbplane"  x="topplane_x" y="topplane_y" z="pcbPlaneZ"  material="PCB"    vis="pcbplane"/>
  <layer id="11" name="cfhplane"  x="topplane_x" y="topplane_y" z="CFRPPlateZ" material="CarbonFibre_MJ55"  vis="cfhplane"/>
  <layer id="12" name="TrayPlate" x="toptray_x"  y="toptray_y"  z="TrayPlateZ" material="my_HC" vis="cfhplane"/>
  <layer id="13" name="cfhplane"  x="topplane_x" y="topplane_y" z="CFRPPlateZ" material="CarbonFibre_MJ55"  vis="cfhplane"/>
  <layer id="14" name="pcbplane"  x="topplane_x" y="topplane_y" z="pcbPlaneZ"  material="PCB"    vis="pcbplane"/>
  <layer id="2" name="siplaney"  x="topplane_x" y="topplane_y" z="siPlaneZ"  vis="siplaney">
   <ladder name="laddery" x="siWaferXY" y="topSiLadder" z="siWaferZ" vis="siladdery" repeat="NTopLadder">
    <sensor name="wafery" x="siWaferXY" y="siWaferXY"   z="siWaferZ" vis="siwafery"  repeat="NtopWafer" sensitive="true" material="G4_Si"/>
   </ladder>
  </layer>
  <layer id="15" name="siXYGap"  x="topplane_x" y="topplane_y" z="siXYGap"   material="Vacuum" vis="InvisibleNoDaughters"/>
  <layer id="16" name="pcbplane" x="topplane_x" y="topplane_y" z="pcbPlaneZ" material="PCB"    vis="pcbplane"/>
  <layer id="3" name="siplanex" x="topplane_x" y="topplane_y" z="siPlaneZ" vis="siplanex">
   <ladder name="ladderx" x="siWaferXY" y="topSiLadder" z="siWaferZ" vis="siladderx" repeat="NTopLadder">
    <sensor name="waferx" x="siWaferXY" y="siWaferXY"   z="siWaferZ" vis="siwaferx"  repeat="NtopWafer" sensitive="true" material="G4_Si"/>
   </ladder>
  </layer>
```

Geometry Service

- To provide an easy-to-use interface for applications, the Geometry Service is implemented to integrate and provide various detector description information:
  - Conversion between geometry description formats (XML, Geant4, ROOT, GDML, ...)
  - Global-Local coordinates conversion
  - Volume ID systems conversion
  - Calculate track length in physics volumes
  - Provide interface to get information of physical volume, placed volume and logical volume (dimention, position, ...)
- These functionalities are actively used in simulation, digitization and reconstruction algorithms

Geometry Service

- To provide an easy-to-use interface for applications, the **Geometry Service** is implemented to integrate and provide various detector description information

```
// Get geant4 geometry information
dd4hep::sim::Geant4GeometryInfo* getGeoInfo();
// Get geant4 physical Volume
G4VPhysicalVolume* getPhyVol();
// Get geant4 magnetic field
G4MagneticField* getMagField();
// Get dd4hep detector instance
dd4hep::Detector* getDetDesc();

// Get the global position of cell by its volumeid
dd4hep::Position getPosition(dd4hep::VolumeID &volId);
// Get the global position of cell by its cellcode and systemid
dd4hep::Position getPosition(SubDetector systemId, int cellcode

// Get the dimensions of cell by its volumeid
std::vector<double> dimension(dd4hep::VolumeID &volId);
// Get the dimensions of cell by its systemid and cellcode
std::vector<double> dimension(SubDetector systemId, int cellcod

// Get the physical node of cell by its volumeid
TGeoPhysicalNode *getPhyNode(dd4hep::VolumeID &volId);
```

```
// Get volume direction in the global system
// Get the x-axis direction of local coordinate system
TVector3 getMainDir(dd4hep::VolumeID &volId);
TVector3 getMainDir(SubDetector systemId, int cellcode);
// Get the direction of fiber
TVector3 getAuxDir(dd4hep::VolumeID &volId);
TVector3 getAuxDir(SubDetector systemId, int cellcode);
// Get the normal direction of the plane
TVector3 getNormDir(dd4hep::VolumeID &volId);
TVector3 getNormDir(SubDetector systemId, int cellcode);

// The local Y axis is aligned vertically or not
bool isVertAligned(dd4hep::VolumeID &volId);
bool isVertAligned(SubDetector systemId, int cellcode);
// Local Y aligned with positive direction or not, of the nearest g
bool isPosAligned(dd4hep::VolumeID &volId);
bool isPosAligned(SubDetector systemId, int cellcode);

// Get the direction information
DirectionInfo getDirectionInfo(dd4hep::VolumeID &volId);
DirectionInfo getDirectionInfo(SubDetector systemId, int cellcode);
```

```
// Transform from world coordinates to local ones at giving level
dd4hep::Position globalToLocal(const dd4hep::Position &global, int level=-1);
// Transform a point from local coordinates of a given level to global coordinates
dd4hep::Position localToGlobal(const dd4hep::Position &local, dd4hep::VolumeID &volId, int level=-1);
```

Parallelized Detector Simulation

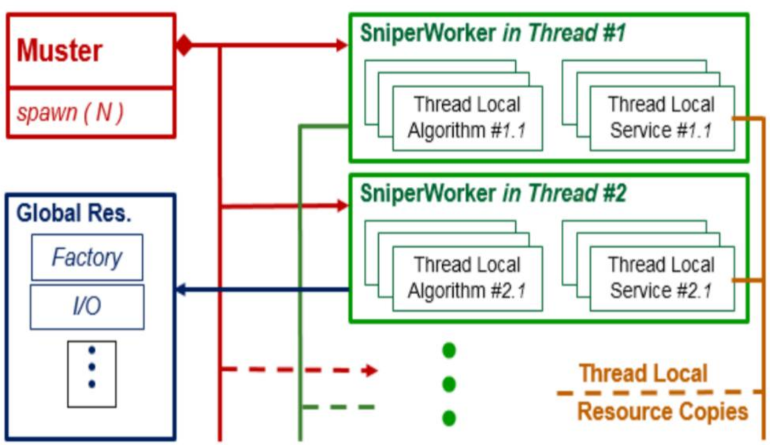
Multi-threading Support: **Motivation**

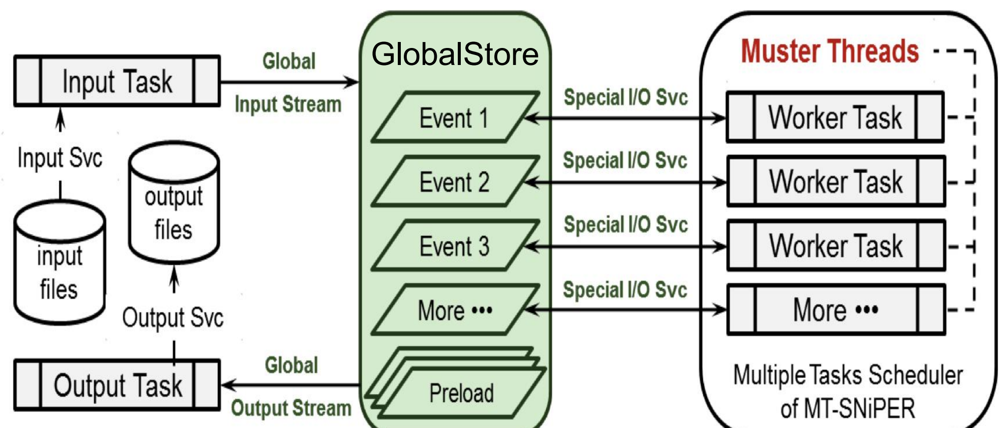
- **Motivation for HERD: full simulation of high energy (~PeV) heavy nucleons costs too much time (~day) and memory**
  - Simulating one ~1 PeV proton costs **~5h**
  - Simulating one ~3 PeV helium costs **~20h**
  - Memory consumption is too large, often causing job gets killed
- Applying concurrent simulation can:
  - **Reduce absolute time cost** of simulating heavy particles
  - **Solve the large memory consuming problem**
    - **Memory allocation is one a per-core basis on computing clusters**
    - Event level: sharing geometry, common services, I/O Buffer, physics list ...
    - Track level: largely increase memory limitation for one event
- Multi-level of multi-threading can be applied
  - **Event level** (between events): multiple events are processed concurrently
  - **Track level** (inside an event): one event is processed with multiple threads
    - Secondary tracks are simulated concurrently

Performance of multi-threading simulation shown on p26 and p28

MT SNiPER

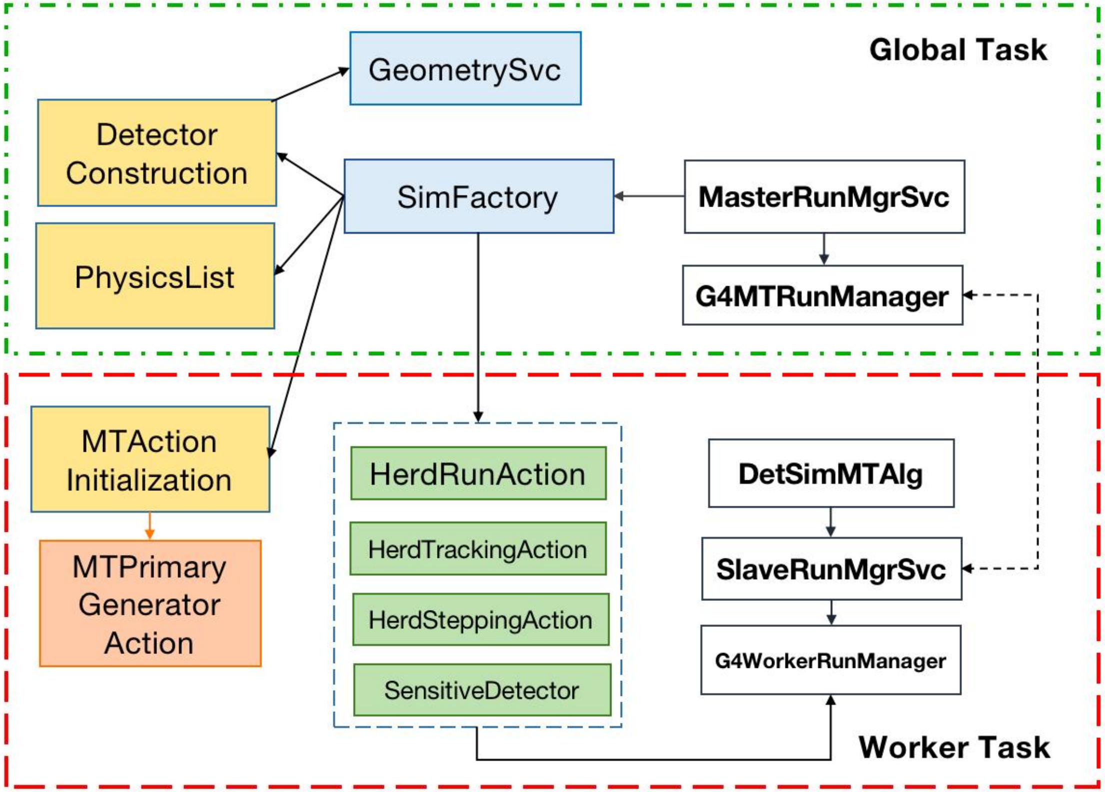
- SNiPER provides very easy-to-use interfaces for building the event-level multi-threaded applications
  - SNiPER Muster (Multiple SNiPER Task Scheduler) works as a thread pool/scheduler based on TBB
  - A GlobalStore is developed to support parallel event data management
  - Data I/O is bound to dedicated I/O thread to speed up of reading/writing data from/to files
  - Application code is mostly consistent for serially and paralleled execution



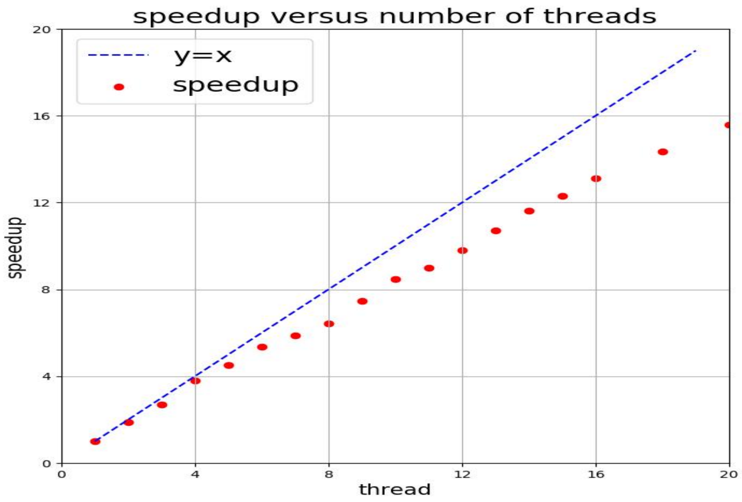


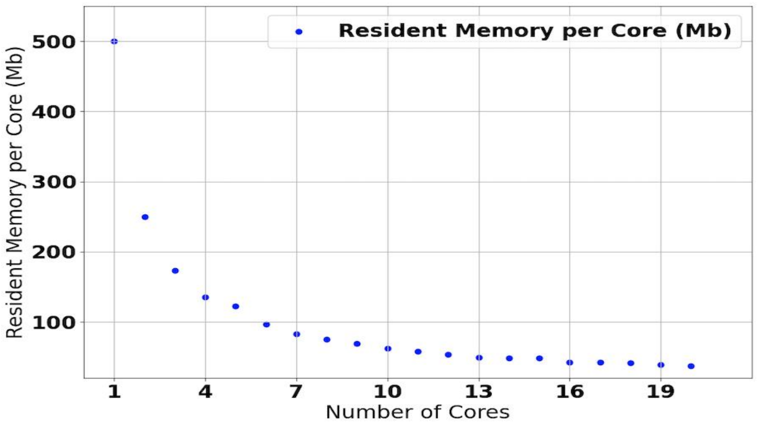
Parallelized Detector Simulation

- Based on the MT-SNiPER and parallelized DM system, the **event level** parallelized detector simulation is developed
  - Simulate events concurrently in multiple threads
  - Basic performance tests show promising scalability



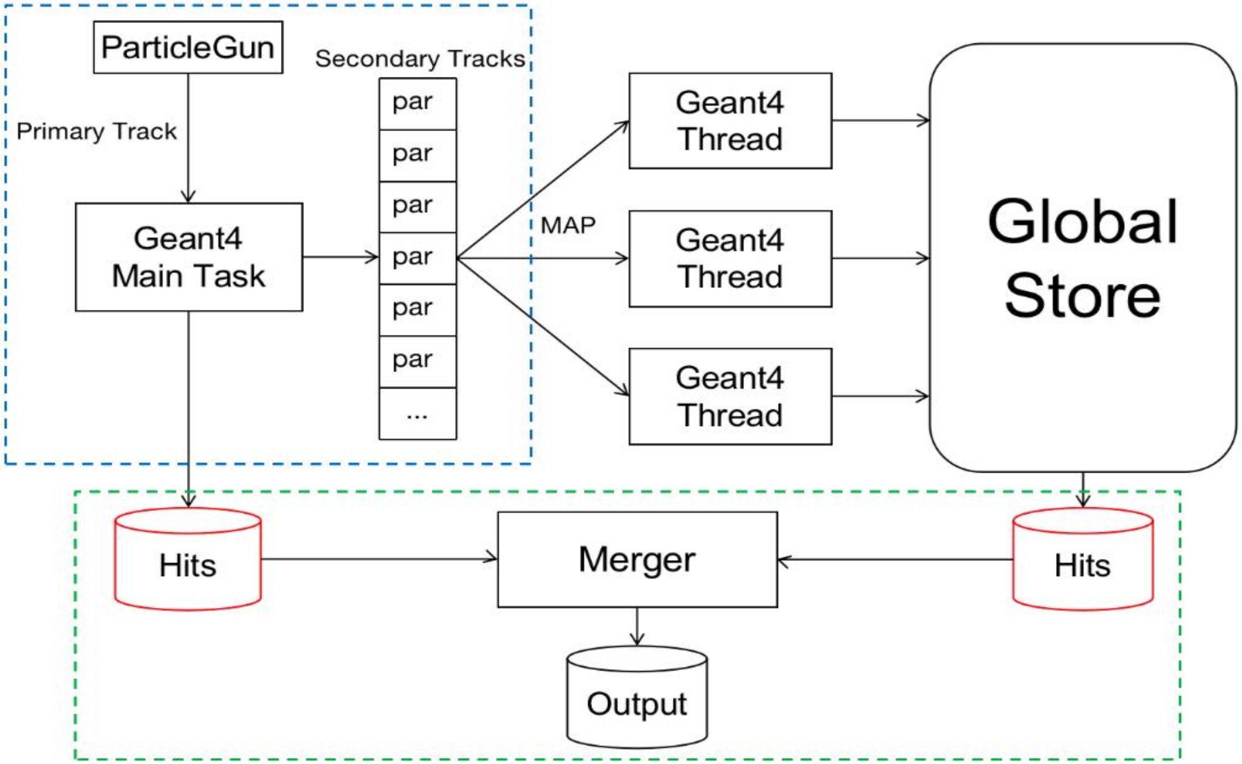
Examples of simulating 100 GeV proton





Parallelized Detector Simulation

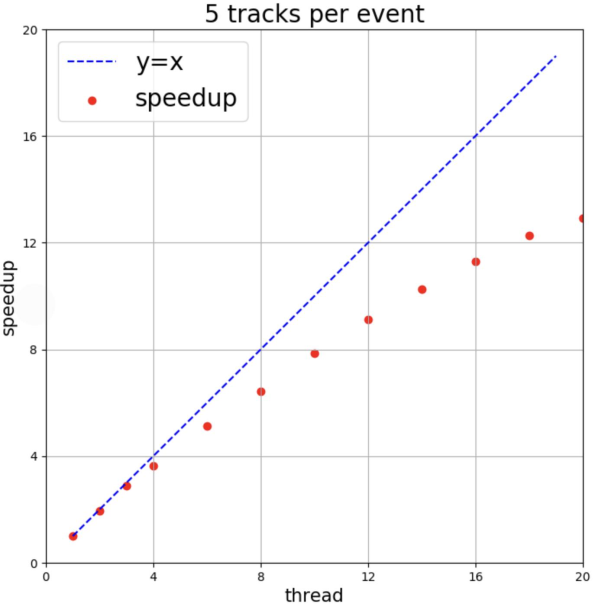
- Sub-event level detector simulation is being developed, for ultra high energy particles to reduce latency
  - Simulate the primary particle in the main Task
  - Secondary particles are dispatched to worker threads
  - Simulated hits are merged after all tracks are simulated

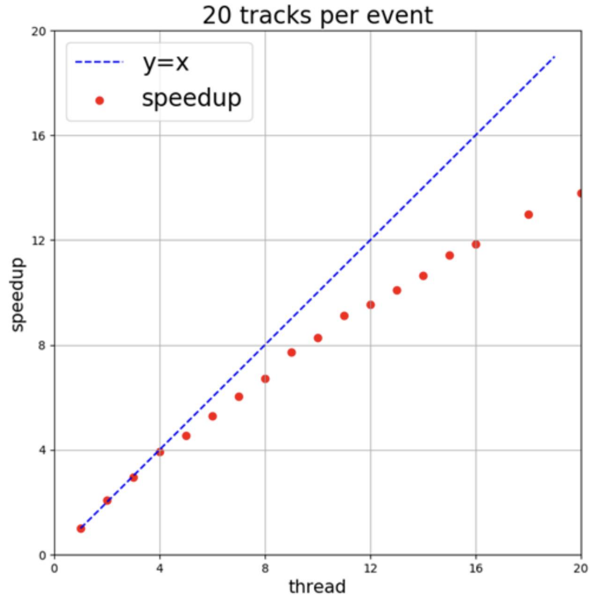


- Basic functionalities (spiting, simulating and merging) are implemented
- Results are validated
- Performance needs to be further optimized
- Separated tasks need to be merged

Parallelized Detector Simulation

- Sub-event level detector simulation is developed (being optimized), to reduce latency and memory comsuption for ultra high energy particles
  - Simulate the primary particle in the main Task
  - Secondary particles are dispatched to worker threads
  - Simulated hits are merged after all tracks are simulated





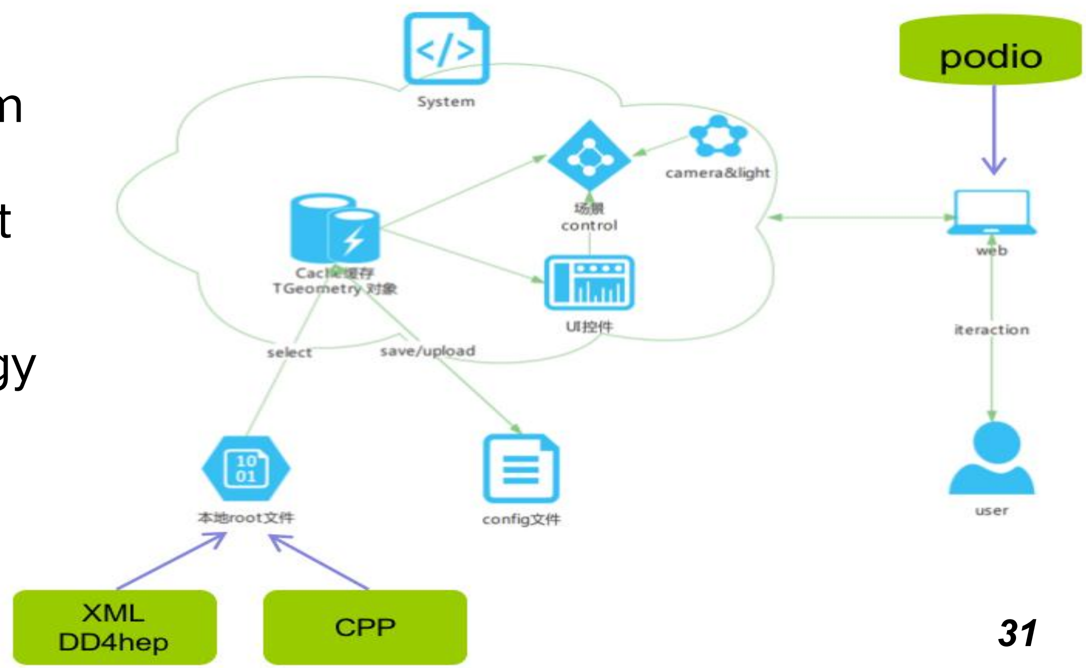
- Basic functionalities (spiting, simulating and merging) are implemented
- Results are validated
- Performance can be further optimized
- With parallelized simulation , we can smoothly simulate very high (~PeV)heavy **nucleons without limitations.**

Examples of simulating 100 GeV proton

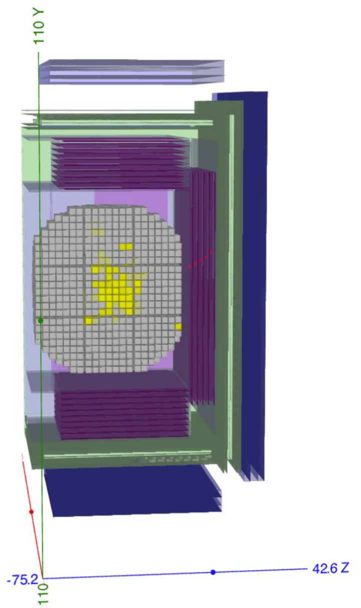
Event Display

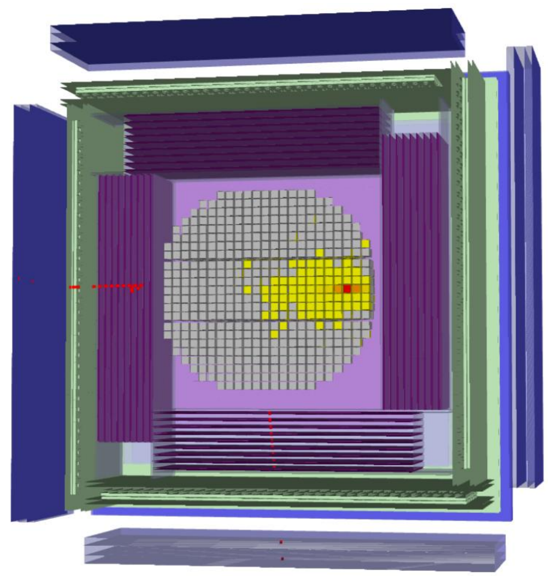
Detector and Event Display

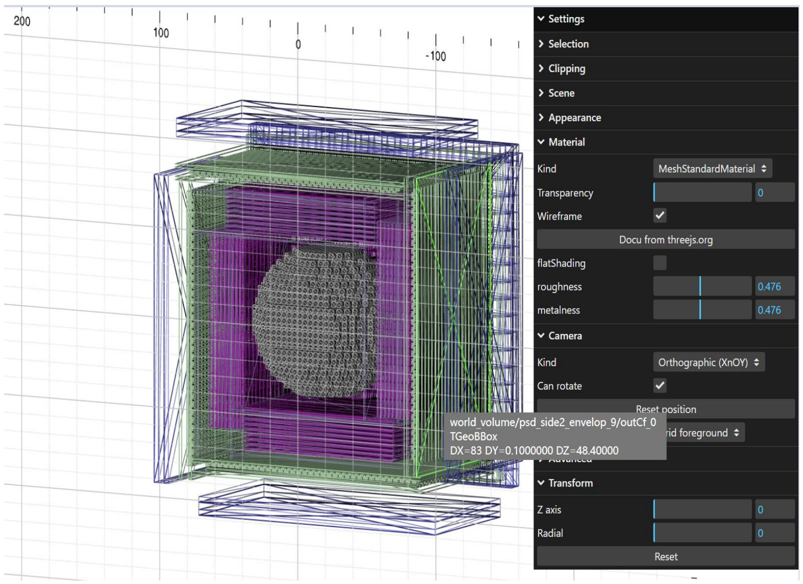
- HERD Event visualization (HERDEvE) is being developed
  - Based on Web3D technology and the open-source JSROOT
  - 3D engine and graphic library based on Three.JS
  - Using the Vue.js HTML5 development framework to implement the Web interface
  - Reducing 3D motion lag by the multi-threading capabilities of Web Worker framework
  - Geometry information from detector description from DD4hep (XML), and event data read from podio
  - State-of-the-art technology road-map is applied

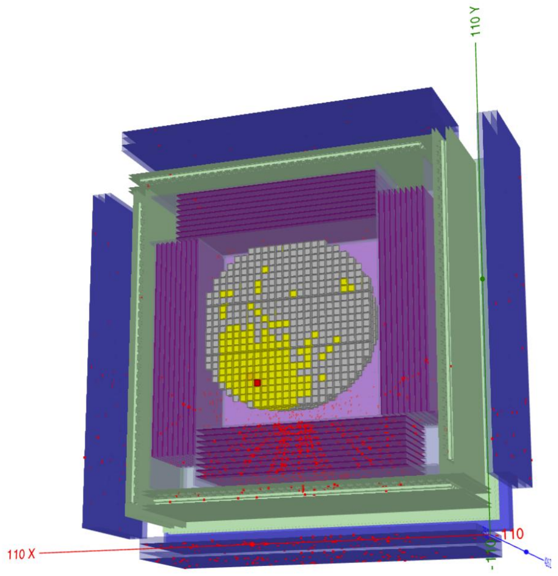


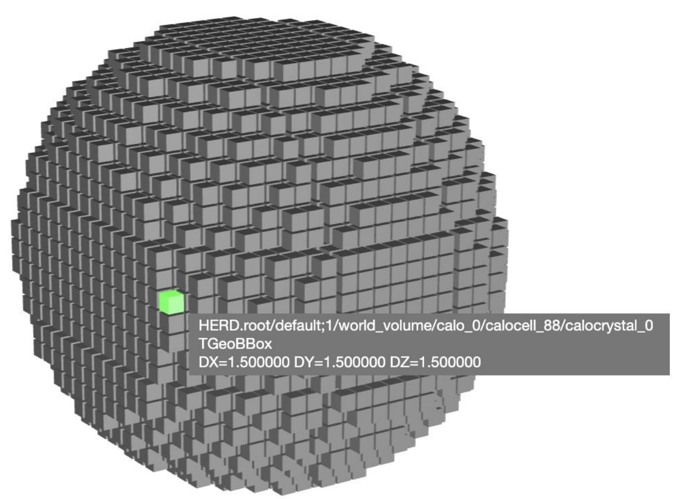
Detector and Event Display

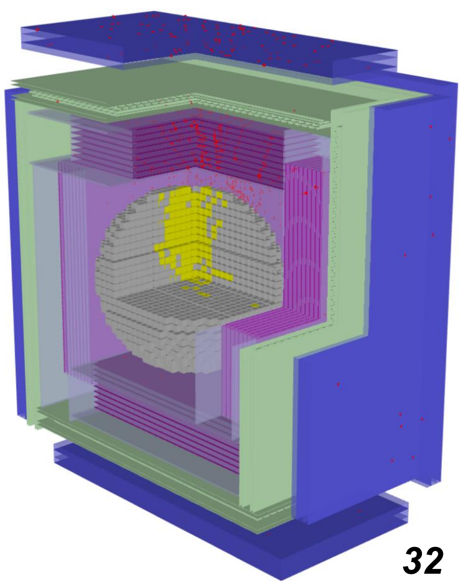










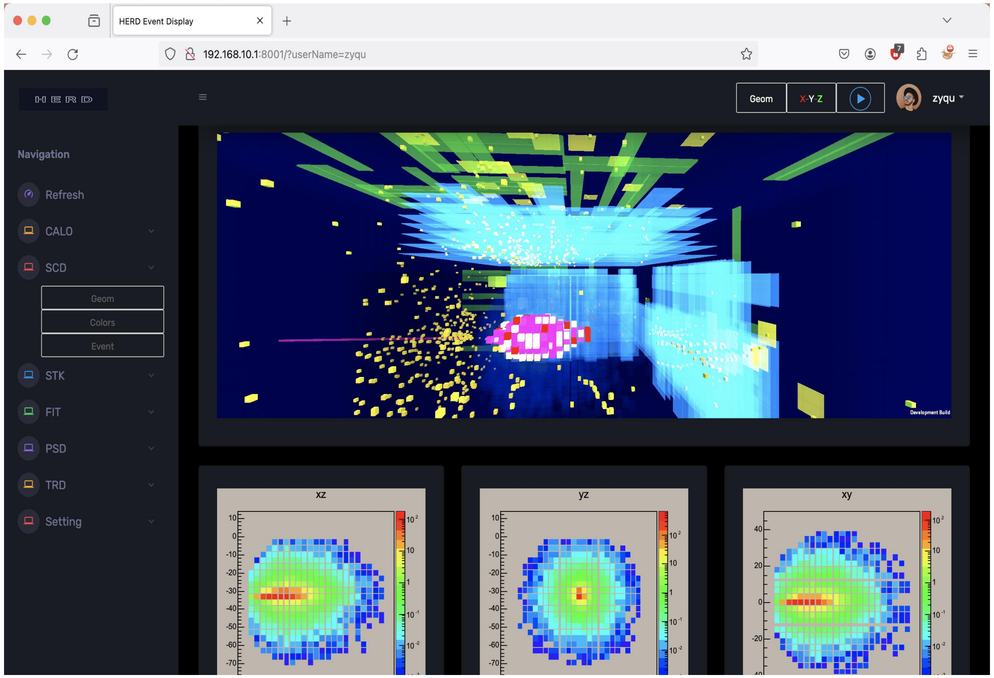


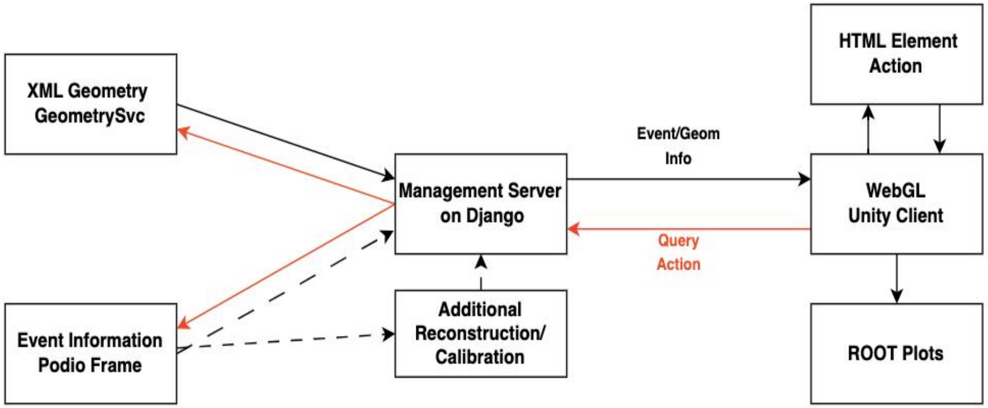
Detector and Event Display

A Unity based WebGL application, now used in beam test

Features supported:

- Environment light color/emission, rendering mode.
- Sub-detector selection/deselection,color, transparency.
- MC hit/track
- SCD/STK/FIT cluster/track
- CALO energy on each cell, color, rendering setting.
- CALO shower projection with selected cells, to ROOT plots.
- Both data stream mode and individual event display mode supported.
- Various geometry and any standard podio format supported.





Machine Learning Integration

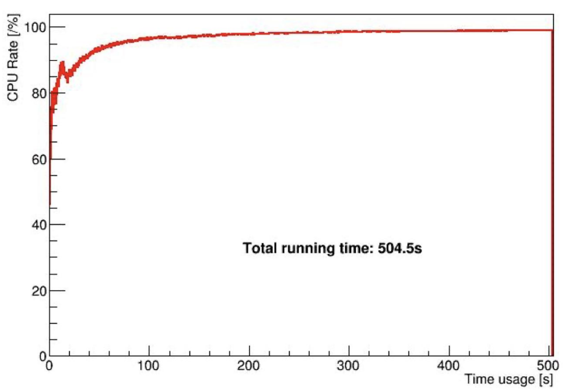
- ONNX Runtime to support machine learning runtime inference
  - Some applications in HERDOS are based on ML models developed in Python, such as particle ID, directionality reconstruction etc.
  - As an easy and unified way to integrate different models in HERDOS and run inference easily
  - Convert from other models to ONNX, such as Tensorflow, PyTorch etc.
  - Potentially to accelerate inference of larger model on different hardware platform (CPU/GPU)

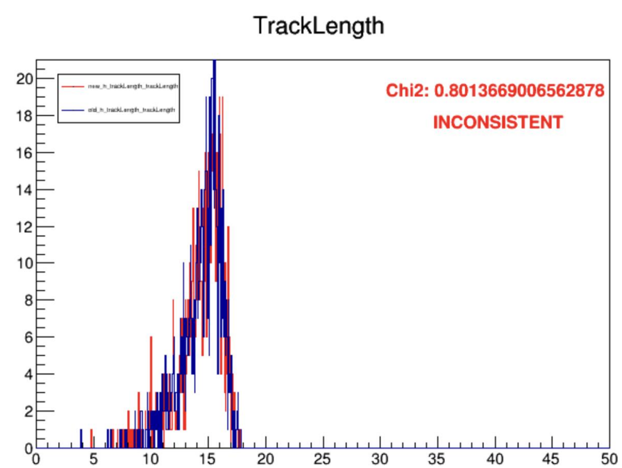
```
Ort::MemoryInfo info("Cpu", OrtDeviceAllocator, 0, OrtMemTypeDefault);

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;
}
```

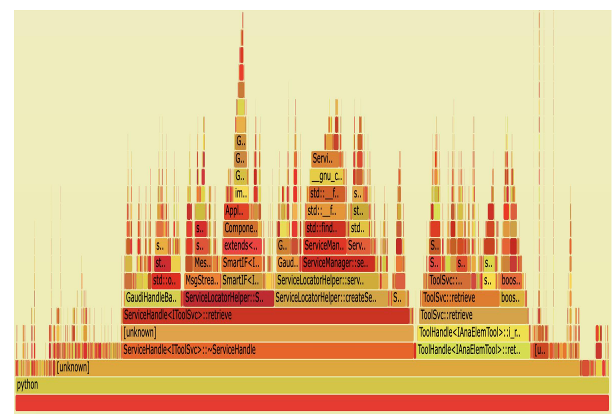
```
bool OrtInferenceAlg::initialize() {

    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);
```

Software Validation

- A validation toolkit is developed to build validation at different levels
- Support building tests of multiple levels
  - Tests based return code, logging parser (predefined log pattern)
  - Tests based on hardware limitations (wall time, memory, ...)
  - Performance (CPU, memory, disk, network, ...) profiling
  - Physics validation based on statistical tests (comparison with standard distribution)







Useful Links

- CVMFS
  - /cvmfs/herd.ihep.ac.cn/HERDOS
- HERDOS Gitlab:
  - https://code.ihep.ac.cn/herdos/offline
- HERDOS Documentation
  - https://herd.ihep.ac.cn/internal/herdos/manual
- Tutorial
  - https://indico.ihep.ac.cn/event/23203/